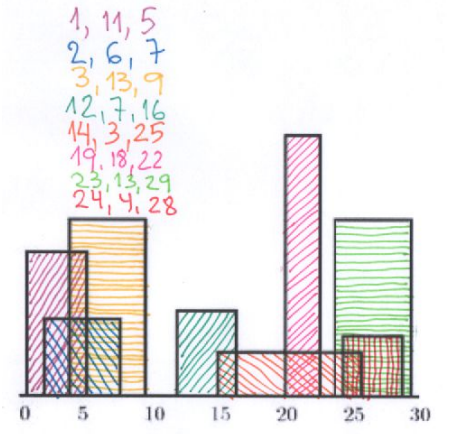
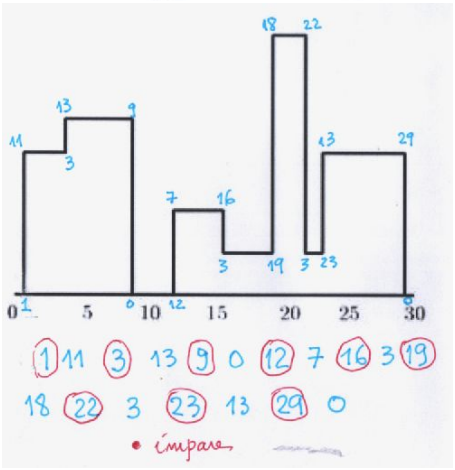
CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 21/10/2024 - 12:07:31.8
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
VIVXhb1a V: 1.02
75789 ARIEL GUSTAVO RODRIGUES DA S
24PRG507 - 1 _____ / _____ / _____

## UVA105-Perfil da skyline

Tal como esta lá no site (`http:\\onlinejudge.org`): Com o advento das estações de trabalho gráficas de alta velocidade, CAD (design auxiliado por computador) e outras áreas (design CAM, VLSI) têm feito uso cada vez mais eficaz dos computadores. Um dos problemas com desenhar imagens é a eliminação de linhas ocultas (linhas obscurecidas por outras partes de um desenho). Você deverá projetar um programa para ajudar um arquiteto a desenhar o horizonte de uma cidade, considerando os locais dos edifícios da cidade. Para tornar o problema tratável, todos os edifícios são de forma retangular e eles compartilham um fundo comum (a cidade onde estão construídos é muito plana). A cidade também é vista como bidimensional. Um edifício é especificado por uma tripla ordenada $(L_i, O_i, R_i)$ onde $L_i$ e $R_i$ são as coordenadas à esquerda e à direita, respectivamente, do edifício $_i$, $(0 < L_i < R_i)$ e $H_i$ é a altura do edifício. No diagrama abaixo, os edifícios são mostrados à esquerda com triplas $(1, 11, 5), (2, 6, 7), (3, 13, 9), (12, 7, 16), (14, 3, 25),$ $(19, 18, 22), (23, 13, 29), (24, 4, 28)$ o horizonte, mostrado à direita, é representado pela sequência: $(1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)$ Estes dados correspondem à seguinte imagem:




**Entrada** A entrada é uma sequência de construção de triplas. Todas as coordenadas dos edifícios são números inteiros menores que 10.000 e haverá pelo menos um e no máximo 5.000 edifícios no arquivo de entrada. Cada tripla (edifício) é uma linha sozinha no arquivo de entrada. Todos os inteiros em uma tripla são separados por um ou mais espaços. As triplas serão classificados por $L_i$, a coordenada $x$ esquerda do edifício, então o edifício com o menor valor à esquerda (a coordenada $x$) é o primeiro no arquivo de entrada.

**Saída** A saída deve consistir no vetor que descreve o horizonte conforme mostrado no exemplo acima. No vetor skyline $(v_1, v_2, v_3, ..., v_{n-2}, v_{n-1}, v_n)$, o $v_i$ é tal que se $i$ é um número par representa um linha horizontal (altura). Se $i$ é um número ímpar representa uma linha vertical (coordenada $x$). O vetor skyline deve representar o 'caminho' percorrido, por exemplo, por uma formiga começando no menor coordenada $x$ e viajando horizontal e verticalmente sobre todas as linhas que definem o horizonte. Por isso a última entrada em todos os vetores do horizonte será '0'.

**Exemplo** O desenho acima está assim representado:

```
Entrada
1 11 5
2 6 7
3 13 9
12 7 16
14 3 25
19 18 22
23 13 29
24 4 28
Saída
1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0
```

**Algoritmo 1** Como quase todo problema não trivial, este admite diversas estratégias de solução. Uma possível é desdobrar a cada elemento da lista de entrada em dois eventos: O primeiro determina o início de um prédio e o segundo o seu final. Os eventos terão a estrutura: (esquerda, altura, 0=início) e (direita, altura, 1=final). Para o conjunto de dados do exemplo, ficará: [(1, 11, 1), (5, 11, 0), (2, 6, 1), (7, 6, 0), (3, 13, 1), (9, 13, 0), (12, 7, 1), (16, 7, 0), (14, 3, 1), (25, 3, 0), (19, 18, 1), (22, 18, 0), (23, 13, 1), (29, 13, 0), (24, 4, 1), (28, 4, 0)] A seguir ordena-se esta lista pela primeira posição (o $x$) cria-se uma lista chamada vista vazia ([]) e outra chamada segmentos vazia ([]) e corrente valendo 0 e depois processa-se a lista:

```
se infim==0 (início)
  insere a altura na lista segmentos
  ordena segmentos em ordem decrescente
senão
  remove esta altura de segmentos
nova-altura = primeiro elemento de segmentos
          (ou 0 se vazia)
se nova-altura <> corrente
  insere altura na vista
  corrente = nova-altura
retornar vista
```

Este algoritmo implementado reproduziu os dados do exemplo corretamente, mas foi recusado pelo online judge. Certamente algum conjunto de dados (oculto) ressalvou algum bug ou imprecisão.

**Algoritmo 2** Uma saída foi buscar outro algoritmo mais simples (embora menos eficiente) e baseado numa informação perdida do enunciado (os valores são menores do que 10000). Agora define-se um vetor de 10001 elementos que representa as ordenadas $x$ de todos os envolvidos no problema. Agora a estratégia ficou mais fácil: basta examinar o que acontece em cada ponto do vetor (já que todas as medidas são inteiras).

```
final=0
altura = [0] * 10001
processando (pcom, altu, pfin)
  se pfin>final
    final=pfin (localizando o final do desenho)
  variando i de pcom até pfin
    altura[i] = max(altura[i],altu)
  cara = altura[1]
  imprima (1, cara)
  variando i de 2 até final
    se cara <> altura[i]
      imprima (i, altura[i])
      cara = altura[i]
  imprima (final, 0)
```

## Exemplos resolvidos Caso 1

```
 1 18 11
 4 25  7
 7  6 15
17 26 25
19 25 25
21 12 26
23 28 26
29 19 30
37 16 42
40 22 47
```

Deu como resultado

```
1 18 4 25 7 18 11 6 15 0 17 26 23 28
26 0 29 19 30 0 37 16 40 22 47 0
```

**Caso 2**

```
 1  3  9
12 17 21
24  4 28
24  1 29
31  9 38
39 10 45
41  7 45
42  3 46
44 23 53
45 18 51
45  7 52
47 18 55
```

Deu como resultado

```
1 3 9 0 12 17 21 0 24 4 28 1 29 0 31 9
38 0 39 10 44 23 53 18 55 0
```

## ☞ Para você fazer

Obviamente, antes de executar este caso, você está convidado (intimado !) a submeter sua solução ao online.judge da UVA. Se você obtiver o ACCEPTED no site, pode exigir um 10 nesta folha independente do que o professor (e seu corretor) acharem. Combinado ?

Para deixar as coisas mais ágeis, considere que todos os números deste caso são menores do que 100.

Eis os seus dados aqui:

```
 1  7  7
 6  9  8
10 29 13
20 23 21
23 24 29
24 20 25
24  3 32
35  7 40
44 17 53
46  5 50
```

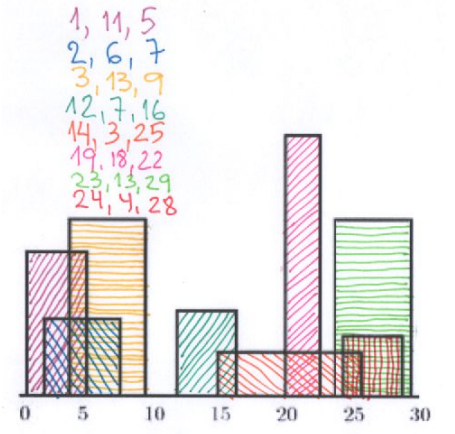
Responda aqui:

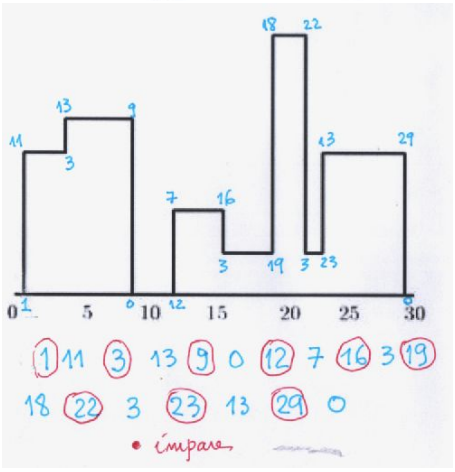
507-75789 -

CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 21/10/2024 - 12:07:31.8
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
VIVXhb1a V: 1.02
75796 CASSIANO MAGNO CHAGAS E SA
24PRG507 - 2 _____ / _____ / _____

## UVA105-Perfil da skyline

Tal como esta lá no site (`http:\\onlinejudge.org`): Com o advento das estações de trabalho gráficas de alta velocidade, CAD (design auxiliado por computador) e outras áreas (design CAM, VLSI) têm feito uso cada vez mais eficaz dos computadores. Um dos problemas com desenhar imagens é a eliminação de linhas ocultas (linhas obscurecidas por outras partes de um desenho). Você deverá projetar um programa para ajudar um arquiteto a desenhar o horizonte de uma cidade, considerando os locais dos edifícios da cidade. Para tornar o problema tratável, todos os edifícios são de forma retangular e eles compartilham um fundo comum (a cidade onde estão construídos é muito plana). A cidade também é vista como bidimensional. Um edifício é especificado por uma tripla ordenada $(L_i, O_i, R_i)$ onde $L_i$ e $R_i$ são as coordenadas à esquerda e à direita, respectivamente, do edifício $_i$, $(0 < L_i < R_i)$ e $H_i$ é a altura do edifício. No diagrama abaixo, os edifícios são mostrados à esquerda com triplas $(1, 11, 5), (2, 6, 7), (3, 13, 9), (12, 7, 16), (14, 3, 25),$ $(19, 18, 22), (23, 13, 29), (24, 4, 28)$ o horizonte, mostrado à direita, é representado pela sequência: $(1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)$ Estes dados correspondem à seguinte imagem:




**Entrada** A entrada é uma sequência de construção de triplas. Todas as coordenadas dos edifícios são números inteiros menores que 10.000 e haverá pelo menos um e no máximo 5.000 edifícios no arquivo de entrada. Cada tripla (edifício) é uma linha sozinha no arquivo de entrada. Todos os inteiros em uma tripla são separados por um ou mais espaços. As triplas serão classificados por $L_i$, a coordenada $x$ esquerda do edifício, então o edifício com o menor valor à esquerda (a coordenada $x$) é o primeiro no arquivo de entrada.

**Saída** A saída deve consistir no vetor que descreve o horizonte conforme mostrado no exemplo acima. No vetor skyline $(v_1, v_2, v_3, ..., v_{n-2}, v_{n-1}, v_n)$, o $v_i$ é tal que se $i$ é um número par representa um linha horizontal (altura). Se $i$ é um número ímpar representa uma linha vertical (coordenada $x$). O vetor skyline deve representar o 'caminho' percorrido, por exemplo, por uma formiga começando no menor coordenada $x$ e viajando horizontal e verticalmente sobre todas as linhas que definem o horizonte. Por isso a última entrada em todos os vetores do horizonte será '0'.

**Exemplo** O desenho acima está assim representado:

```
Entrada
1 11 5
2 6 7
3 13 9
12 7 16
14 3 25
19 18 22
23 13 29
24 4 28
Saída
1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0
```

**Algoritmo 1** Como quase todo problema não trivial, este admite diversas estratégias de solução. Uma possível é desdobrar a cada elemento da lista de entrada em dois eventos: O primeiro determina o início de um prédio e o segundo o seu final. Os eventos terão a estrutura: (esquerda, altura, 0=início) e (direita, altura, 1=final). Para o conjunto de dados do exemplo, ficará: [(1, 11, 1), (5, 11, 0), (2, 6, 1), (7, 6, 0), (3, 13, 1), (9, 13, 0), (12, 7, 1), (16, 7, 0), (14, 3, 1), (25, 3, 0), (19, 18, 1), (22, 18, 0), (23, 13, 1), (29, 13, 0), (24, 4, 1), (28, 4, 0)] A seguir ordena-se esta lista pela primeira posição (o $x$) cria-se uma lista chamada vista vazia ([]) e outra chamada segmentos vazia ([]) e corrente valendo 0 e depois processa-se a lista:

```
se infim==0 (início)
  insere a altura na lista segmentos
  ordena segmentos em ordem decrescente
senão
  remove esta altura de segmentos
nova-altura = primeiro elemento de segmentos
          (ou 0 se vazia)
se nova-altura <> corrente
  insere altura na vista
  corrente = nova-altura
retornar vista
```

Este algoritmo implementado reproduziu os dados do exemplo corretamente, mas foi recusado pelo online judge. Certamente algum conjunto de dados (oculto) ressalvou algum bug ou imprecisão.

**Algoritmo 2** Uma saída foi buscar outro algoritmo mais simples (embora menos eficiente) e baseado numa informação perdida do enunciado (os valores são menores do que 10000). Agora define-se um vetor de 10001 elementos que representa as ordenadas $x$ de todos os envolvidos no problema. Agora a estratégia ficou mais fácil: basta examinar o que acontece em cada ponto do vetor (já que todas as medidas são inteiras).

```
final=0
altura = [0] * 10001
processando (pcom, altu, pfin)
  se pfin>final
    final=pfin (localizando o final do desenho)
  variando i de pcom até pfin
    altura[i] = max(altura[i],altu)
  cara = altura[1]
  imprima (1, cara)
  variando i de 2 até final
    se cara <> altura[i]
      imprima (i, altura[i])
      cara = altura[i]
  imprima (final, 0)
```

## Exemplos resolvidos Caso 1

```
 1 18 11
 4 25  7
 7  6 15
17 26 25
19 25 25
21 12 26
23 28 26
29 19 30
37 16 42
40 22 47
```

Deu como resultado

```
1 18 4 25 7 18 11 6 15 0 17 26 23 28
26 0 29 19 30 0 37 16 40 22 47 0
```

**Caso 2**

```
 1  3  9
12 17 21
24  4 28
24  1 29
31  9 38
39 10 45
41  7 45
42  3 46
44 23 53
45 18 51
45  7 52
47 18 55
```

Deu como resultado

```
1 3 9 0 12 17 21 0 24 4 28 1 29 0 31 9
38 0 39 10 44 23 53 18 55 0
```

## ☞ Para você fazer

Obviamente, antes de executar este caso, você está convidado (intimado !) a submeter sua solução ao online.judge da UVA. Se você obtiver o ACCEPTED no site, pode exigir um 10 nesta folha independente do que o professor (e seu corretor) acharem. Combinado ?

Para deixar as coisas mais ágeis, considere que todos os números deste caso são menores do que 100.

Eis os seus dados aqui:

```
 1 30 11
 7  4 10
16 11 17
25 18 33
26 28 36
28  5 36
30  2 35
33  6 37
44 17 46
47 13 51
```

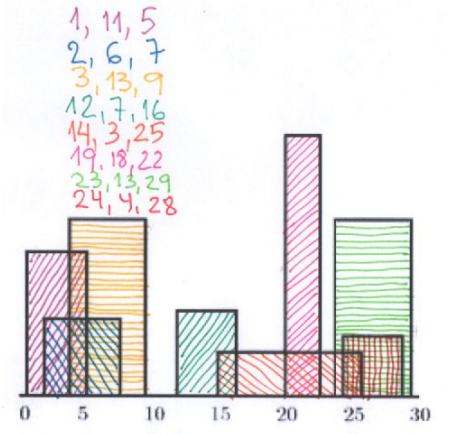
Responda aqui:

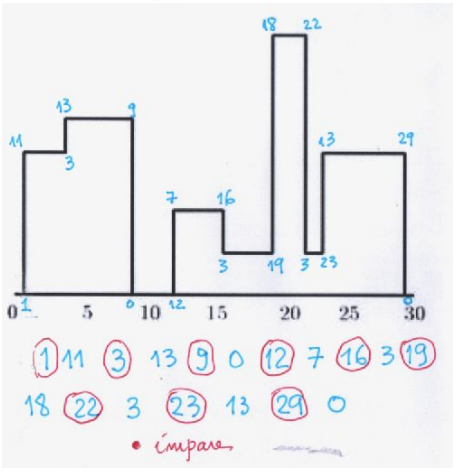
507-75796 -

CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 21/10/2024 - 12:07:31.8
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
VIVXhb1a V: 1.02
75808 CHRISTOPHER BUCH FILIPAK
24PRG507 - 3 _____ / _____ / _____

## UVA105-Perfil da skyline

Tal como esta lá no site (`http:\\onlinejudge.org`): Com o advento das estações de trabalho gráficas de alta velocidade, CAD (design auxiliado por computador) e outras áreas (design CAM, VLSI) têm feito uso cada vez mais eficaz dos computadores. Um dos problemas com desenhar imagens é a eliminação de linhas ocultas (linhas obscurecidas por outras partes de um desenho). Você deverá projetar um programa para ajudar um arquiteto a desenhar o horizonte de uma cidade, considerando os locais dos edifícios da cidade. Para tornar o problema tratável, todos os edifícios são de forma retangular e eles compartilham um fundo comum (a cidade onde estão construídos é muito plana). A cidade também é vista como bidimensional. Um edifício é especificado por uma tripla ordenada $(L_i, O_i, R_i)$ onde $L_i$ e $R_i$ são as coordenadas à esquerda e à direita, respectivamente, do edifício $_i$, $(0 < L_i < R_i)$ e $H_i$ é a altura do edifício. No diagrama abaixo, os edifícios são mostrados à esquerda com triplas $(1, 11, 5), (2, 6, 7), (3, 13, 9), (12, 7, 16), (14, 3, 25),$ $(19, 18, 22), (23, 13, 29), (24, 4, 28)$ o horizonte, mostrado à direita, é representado pela sequência: $(1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)$ Estes dados correspondem à seguinte imagem:




**Entrada** A entrada é uma sequência de construção de triplas. Todas as coordenadas dos edifícios são números inteiros menores que 10.000 e haverá pelo menos um e no máximo 5.000 edifícios no arquivo de entrada. Cada tripla (edifício) é uma linha sozinha no arquivo de entrada. Todos os inteiros em uma tripla são separados por um ou mais espaços. As triplas serão classificados por $L_i$, a coordenada $x$ esquerda do edifício, então o edifício com o menor valor à esquerda (a coordenada $x$) é o primeiro no arquivo de entrada.

**Saída** A saída deve consistir no vetor que descreve o horizonte conforme mostrado no exemplo acima. No vetor skyline $(v_1, v_2, v_3, ..., v_{n-2}, v_{n-1}, v_n)$, o $v_i$ é tal que se $i$ é um número par representa um linha horizontal (altura). Se $i$ é um número ímpar representa uma linha vertical (coordenada $x$). O vetor skyline deve representar o 'caminho' percorrido, por exemplo, por uma formiga começando na menor coordenada $x$ e viajando horizontal e verticalmente sobre todas as linhas que definem o horizonte. Por isso a última entrada em todos os vetores do horizonte será '0'.

**Exemplo** O desenho acima está assim representado:

```
Entrada
1 11 5
2 6 7
3 13 9
12 7 16
14 3 25
19 18 22
23 13 29
24 4 28
Saída
1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0
```

**Algoritmo 1** Como quase todo problema não trivial, este admite diversas estratégias de solução. Uma possível é desdobrar a cada elemento da lista de entrada em dois eventos: O primeiro determina o início de um prédio e o segundo o seu final. Os eventos terão a estrutura: (esquerda, altura, 0=início) e (direita, altura, 1=final). Para o conjunto de dados do exemplo, ficará: [(1, 11, 1), (5, 11, 0), (2, 6, 1), (7, 6, 0), (3, 13, 1), (9, 13, 0), (12, 7, 1), (16, 7, 0), (14, 3, 1), (25, 3, 0), (19, 18, 1), (22, 18, 0), (23, 13, 1), (29, 13, 0), (24, 4, 1), (28, 4, 0)] A seguir ordena-se esta lista pela primeira posição (o $x$) cria-se uma lista chamada vista vazia ([]) e outra chamada segmentos vazia ([]) e corrente valendo 0 e depois processa-se a lista:

```
se infim==0 (início)
  insere a altura na lista segmentos
  ordena segmentos em ordem decrescente
senão
  remove esta altura de segmentos
nova-altura = primeiro elemento de segmentos
          (ou 0 se vazia)
se nova-altura <> corrente
  insere altura na vista
  corrente = nova-altura
retornar vista
```

Este algoritmo implementado reproduziu os dados do exemplo corretamente, mas foi recusado pelo online judge. Certamente algum conjunto de dados (oculto) ressalvou algum bug ou imprecisão.

**Algoritmo 2** Uma saída foi buscar outro algoritmo mais simples (embora menos eficiente) e baseado numa informação perdida do enunciado (os valores são menores do que 10000). Agora define-se um vetor de 10001 elementos que representa as ordenadas $x$ de todos os envolvidos no problema. Agora a estratégia ficou mais fácil: basta examinar o que acontece em cada ponto do vetor (já que todas as medidas são inteiras).

```
final=0
altura = [0] * 10001
processando (pcom, altu, pfin)
  se pfin>final
    final=pfin (localizando o final do desenho)
  variando i de pcom até pfin
    altura[i] = max(altura[i],altu)
  cara = altura[1]
  imprima (1, cara)
  variando i de 2 até final
    se cara <> altura[i]
      imprima (i, altura[i])
      cara = altura[i]
  imprima (final, 0)
```

## Exemplos resolvidos Caso 1

```
 1 18 11
 4 25  7
 7  6 15
17 26 25
19 25 25
21 12 26
23 28 26
29 19 30
37 16 42
40 22 47
```

Deu como resultado

```
1 18 4 25 7 18 11 6 15 0 17 26 23 28
26 0 29 19 30 0 37 16 40 22 47 0
```

**Caso 2**

```
 1  3  9
12 17 21
24  4 28
24  1 29
31  9 38
39 10 45
41  7 45
42  3 46
44 23 53
45 18 51
45  7 52
47 18 55
```

Deu como resultado

```
1 3 9 0 12 17 21 0 24 4 28 1 29 0 31 9
38 0 39 10 44 23 53 18 55 0
```

## ☞ Para você fazer

Obviamente, antes de executar este caso, você está convidado (intimado !) a submeter sua solução ao online.judge da UVA. Se você obtiver o ACCEPTED no site, pode exigir um 10 nesta folha independente do que o professor (e seu corretor) acharem. Combinado ?

Para deixar as coisas mais ágeis, considere que todos os números deste caso são menores do que 100.

Eis os seus dados aqui:

```
 1 21  3
14 18 23
18 27 23
20  5 25
21 13 25
30 26 36
32 27 35
34 17 40
41 17 43
43 14 50
```

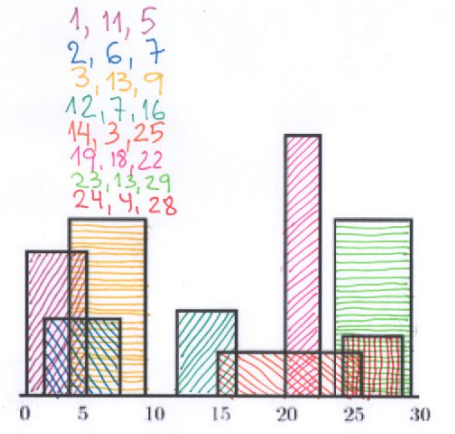
Responda aqui:

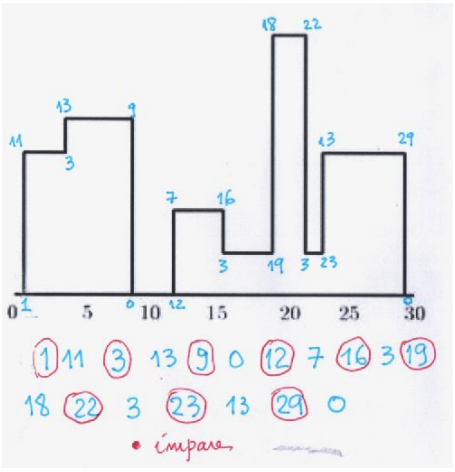
507-75808 -

CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 21/10/2024 - 12:07:31.8
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
VIVXhb1a V: 1.02
75815 ERAN MARTINEZ RAMOS
24PRG507 - 4 _____ / _____ / _____

## UVA105-Perfil da skyline

Tal como esta lá no site (`http:\\onlinejudge.org`): Com o advento das estações de trabalho gráficas de alta velocidade, CAD (design auxiliado por computador) e outras áreas (design CAM, VLSI) têm feito uso cada vez mais eficaz dos computadores. Um dos problemas com desenhar imagens é a eliminação de linhas ocultas (linhas obscurecidas por outras partes de um desenho). Você deverá projetar um programa para ajudar um arquiteto a desenhar o horizonte de uma cidade, considerando os locais dos edifícios da cidade. Para tornar o problema tratável, todos os edifícios são de forma retangular e eles compartilham um fundo comum (a cidade onde estão construídos é muito plana). A cidade também é vista como bidimensional. Um edifício é especificado por uma tripla ordenada $(L_i, O_i, R_i)$ onde $L_i$ e $R_i$ são as coordenadas à esquerda e à direita, respectivamente, do edifício $_i$, $(0 < L_i < R_i)$ e $H_i$ é a altura do edifício. No diagrama abaixo, os edifícios são mostrados à esquerda com triplas $(1, 11, 5), (2, 6, 7), (3, 13, 9), (12, 7, 16), (14, 3, 25),$ $(19, 18, 22), (23, 13, 29), (24, 4, 28)$ o horizonte, mostrado à direita, é representado pela sequência: $(1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)$ Estes dados correspondem à seguinte imagem:




**Entrada** A entrada é uma sequência de construção de triplas. Todas as coordenadas dos edifícios são números inteiros menores que 10.000 e haverá pelo menos um e no máximo 5.000 edifícios no arquivo de entrada. Cada tripla (edifício) é uma linha sozinha no arquivo de entrada. Todos os inteiros em uma tripla são separados por um ou mais espaços. As triplas serão classificados por $L_i$, a coordenada $x$ esquerda do edifício, então o edifício com o menor valor à esquerda (a coordenada $x$) é o primeiro no arquivo de entrada.

**Saída** A saída deve consistir no vetor que descreve o horizonte conforme mostrado no exemplo acima. No vetor skyline $(v_1, v_2, v_3, ..., v_{n-2}, v_{n-1}, v_n)$, o $v_i$ é tal que se $i$ é um número par representa um linha horizontal (altura). Se $i$ é um número ímpar representa uma linha vertical (coordenada $x$). O vetor skyline deve representar o 'caminho' percorrido, por exemplo, por uma formiga começando no menor coordenada $x$ e viajando horizontal e verticalmente sobre todas as linhas que definem o horizonte. Por isso a última entrada em todos os vetores do horizonte será '0'.

**Exemplo** O desenho acima está assim representado:

```
Entrada
1 11 5
2 6 7
3 13 9
12 7 16
14 3 25
19 18 22
23 13 29
24 4 28
Saída
1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0
```

**Algoritmo 1** Como quase todo problema não trivial, este admite diversas estratégias de solução. Uma possível é desdobrar a cada elemento da lista de entrada em dois eventos: O primeiro determina o início de um prédio e o segundo o seu final. Os eventos terão a estrutura: (esquerda, altura, 0=início) e (direita, altura, 1=final). Para o conjunto de dados do exemplo, ficará: [(1, 11, 1), (5, 11, 0), (2, 6, 1), (7, 6, 0), (3, 13, 1), (9, 13, 0), (12, 7, 1), (16, 7, 0), (14, 3, 1), (25, 3, 0), (19, 18, 1), (22, 18, 0), (23, 13, 1), (29, 13, 0), (24, 4, 1), (28, 4, 0)] A seguir ordena-se esta lista pela primeira posição (o $x$) cria-se uma lista chamada vista vazia ([]) e outra chamada segmentos vazia ([]) e corrente valendo 0 e depois processa-se a lista:

```
se infim==0 (início)
  insere a altura na lista segmentos
  ordena segmentos em ordem decrescente
senão
  remove esta altura de segmentos
nova-altura = primeiro elemento de segmentos
          (ou 0 se vazia)
se nova-altura <> corrente
  insere altura na vista
  corrente = nova-altura
retornar vista
```

Este algoritmo implementado reproduziu os dados do exemplo corretamente, mas foi recusado pelo online judge. Certamente algum conjunto de dados (oculto) ressalvou algum bug ou imprecisão.

**Algoritmo 2** Uma saída foi buscar outro algoritmo mais simples (embora menos eficiente) e baseado numa informação perdida do enunciado (os valores são menores do que 10000). Agora define-se um vetor de 10001 elementos que representa as ordenadas $x$ de todos os envolvidos no problema. Agora a estratégia ficou mais fácil: basta examinar o que acontece em cada ponto do vetor (já que todas as medidas são inteiras).

```
final=0
altura = [0] * 10001
processando (pcom, altu, pfin)
  se pfin>final
    final=pfin (localizando o final do desenho)
  variando i de pcom até pfin
    altura[i] = max(altura[i],altu)
  cara = altura[1]
  imprima (1, cara)
  variando i de 2 até final
    se cara <> altura[i]
      imprima (i, altura[i])
      cara = altura[i]
  imprima (final, 0)
```

## Exemplos resolvidos Caso 1

```
 1 18 11
 4 25  7
 7  6 15
17 26 25
19 25 25
21 12 26
23 28 26
29 19 30
37 16 42
40 22 47
```

Deu como resultado

```
1 18 4 25 7 18 11 6 15 0 17 26 23 28
26 0 29 19 30 0 37 16 40 22 47 0
```

**Caso 2**

```
 1  3  9
12 17 21
24  4 28
24  1 29
31  9 38
39 10 45
41  7 45
42  3 46
44 23 53
45 18 51
45  7 52
47 18 55
```

Deu como resultado

```
1 3 9 0 12 17 21 0 24 4 28 1 29 0 31 9
38 0 39 10 44 23 53 18 55 0
```

## ☞ Para você fazer

Obviamente, antes de executar este caso, você está convidado (intimado !) a submeter sua solução ao online.judge da UVA. Se você obtiver o ACCEPTED no site, pode exigir um 10 nesta folha independente do que o professor (e seu corretor) acharem. Combinado ?

Para deixar as coisas mais ágeis, considere que todos os números deste caso são menores do que 100.

Eis os seus dados aqui:

```
 1 29  8
 3 10  8
 6 25  9
 6 25 14
12 12 21
14 17 21
21 15 23
26  4 27
43 12 47
44  1 54
```

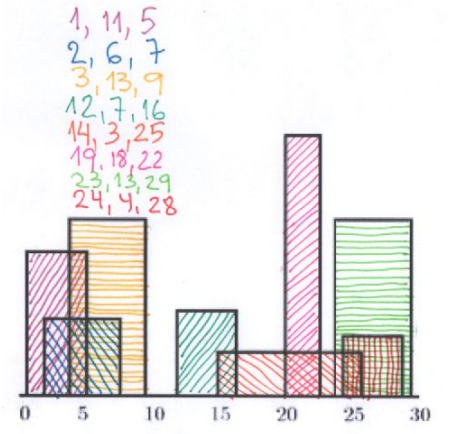
Responda aqui:

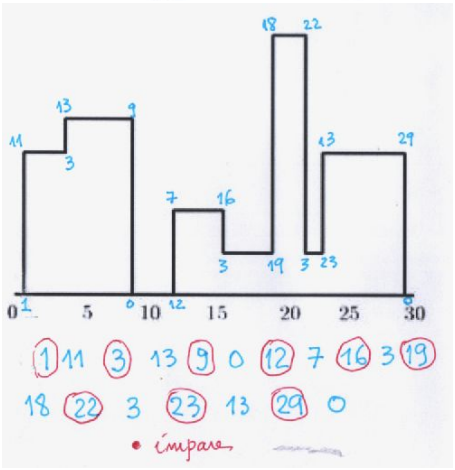
507-75815 -

CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 21/10/2024 - 12:07:31.8
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
VIVXhb1a V: 1.02
75822 HENRIQUE IVANKIO GUSSE
24PRG507 - 5 _____ / _____ / _____

# UVA105-Perfil da skyline

Tal como esta lá no site (`http:\\onlinejudge.org`): Com o advento das estações de trabalho gráficas de alta velocidade, CAD (design auxiliado por computador) e outras áreas (design CAM, VLSI) têm feito uso cada vez mais eficaz dos computadores. Um dos problemas com desenhar imagens é a eliminação de linhas ocultas (linhas obscurecidas por outras partes de um desenho). Você deverá projetar um programa para ajudar um arquiteto a desenhar o horizonte de uma cidade, considerando os locais dos edifícios da cidade. Para tornar o problema tratável, todos os edifícios são de forma retangular e eles compartilham um fundo comum (a cidade onde estão construídos é muito plana). A cidade também é vista como bidimensional. Um edifício é especificado por uma tripla ordenada $(L_i, O_i, R_i)$ onde $L_i$ e $R_i$ são as coordenadas à esquerda e à direita, respectivamente, do edifício $_i$, $(0 < L_i < R_i)$ e $H_i$ é a altura do edifício. No diagrama abaixo, os edifícios são mostrados à esquerda com triplas $(1, 11, 5), (2, 6, 7), (3, 13, 9), (12, 7, 16), (14, 3, 25),$ $(19, 18, 22), (23, 13, 29), (24, 4, 28)$ o horizonte, mostrado à direita, é representado pela sequência: $(1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)$ Estes dados correspondem à seguinte imagem:




**Entrada** A entrada é uma sequência de construção de triplas. Todas as coordenadas dos edifícios são números inteiros menores que 10.000 e haverá pelo menos um e no máximo 5.000 edifícios no arquivo de entrada. Cada tripla (edifício) é uma linha sozinha no arquivo de entrada. Todos os inteiros em uma tripla são separados por um ou mais espaços. As triplas serão classificados por $L_i$, a coordenada $x$ esquerda do edifício, então o edifício com o menor valor à esquerda (a coordenada $x$) é o primeiro no arquivo de entrada.

**Saída** A saída deve consistir no vetor que descreve o horizonte conforme mostrado no exemplo acima. No vetor skyline $(v_1, v_2, v_3, ..., v_{n-2}, v_{n-1}, v_n)$, o $v_i$ é tal que se $i$ é um número par representa um linha horizontal (altura). Se $i$ é um número ímpar representa uma linha vertical (coordenada $x$). O vetor skyline deve representar o 'caminho' percorrido, por exemplo, por uma formiga começando no menor coordenada $x$ e viajando horizontal e verticalmente sobre todas as linhas que definem o horizonte. Por isso a última entrada em todos os vetores do horizonte será '0'.

**Exemplo** O desenho acima está assim representado:

```
Entrada
1 11 5
2 6 7
3 13 9
12 7 16
14 3 25
19 18 22
23 13 29
24 4 28
Saída
1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0
```

**Algoritmo 1** Como quase todo problema não trivial, este admite diversas estratégias de solução. Uma possível é desdobrar a cada elemento da lista de entrada em dois eventos: O primeiro determina o início de um prédio e o segundo o seu final. Os eventos terão a estrutura: (esquerda, altura, 0=início) e (direita, altura, 1=final). Para o conjunto de dados do exemplo, ficará: [(1, 11, 1), (5, 11, 0), (2, 6, 1), (7, 6, 0), (3, 13, 1), (9, 13, 0), (12, 7, 1), (16, 7, 0), (14, 3, 1), (25, 3, 0), (19, 18, 1), (22, 18, 0), (23, 13, 1), (29, 13, 0), (24, 4, 1), (28, 4, 0)] A seguir ordena-se esta lista pela primeira posição (o $x$) cria-se uma lista chamada vista vazia ([]) e outra chamada segmentos vazia ([]) e corrente valendo 0 e depois processa-se a lista:

```
se infim==0 (início)
  insere a altura na lista segmentos
  ordena segmentos em ordem decrescente
senão
  remove esta altura de segmentos
nova-altura = primeiro elemento de segmentos
          (ou 0 se vazia)
se nova-altura <> corrente
  insere altura na vista
  corrente = nova-altura
retornar vista
```

Este algoritmo implementado reproduziu os dados do exemplo corretamente, mas foi recusado pelo online judge. Certamente algum conjunto de dados (oculto) ressalvou algum bug ou imprecisão.

**Algoritmo 2** Uma saída foi buscar outro algoritmo mais simples (embora menos eficiente) e baseado numa informação perdida do enunciado (os valores são menores do que 10000). Agora define-se um vetor de 10001 elementos que representa as ordenadas $x$ de todos os envolvidos no problema. Agora a estratégia ficou mais fácil: basta examinar o que acontece em cada ponto do vetor (já que todas as medidas são inteiras).

```
final=0
altura = [0] * 10001
processando (pcom, altu, pfin)
  se pfin>final
    final=pfin (localizando o final do desenho)
  variando i de pcom até pfin
    altura[i] = max(altura[i],altu)
  cara = altura[1]
  imprima (1, cara)
  variando i de 2 até final
    se cara <> altura[i]
      imprima (i, altura[i])
      cara = altura[i]
  imprima (final, 0)
```

## Exemplos resolvidos Caso 1

```
 1 18 11
 4 25  7
 7  6 15
17 26 25
19 25 25
21 12 26
23 28 26
29 19 30
37 16 42
40 22 47
```

Deu como resultado

```
1 18 4 25 7 18 11 6 15 0 17 26 23 28
26 0 29 19 30 0 37 16 40 22 47 0
```

**Caso 2**

```
 1  3  9
12 17 21
24  4 28
24  1 29
31  9 38
39 10 45
41  7 45
42  3 46
44 23 53
45 18 51
45  7 52
47 18 55
```

Deu como resultado

```
1 3 9 0 12 17 21 0 24 4 28 1 29 0 31 9
38 0 39 10 44 23 53 18 55 0
```

## ☞ Para você fazer

Obviamente, antes de executar este caso, você está convidado (intimado !) a submeter sua solução ao online.judge da UVA. Se você obtiver o ACCEPTED no site, pode exigir um 10 nesta folha independente do que o professor (e seu corretor) acharem. Combinado ?

Para deixar as coisas mais ágeis, considere que todos os números deste caso são menores do que 100.

Eis os seus dados aqui:

```
 1 16  8
 6  2  8
 7 14 11
13 17 17
31  7 33
32 10 33
36  8 42
42 30 46
43 14 46
46 23 50
```

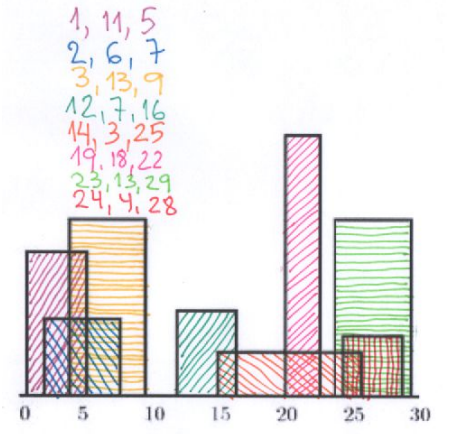
Responda aqui:

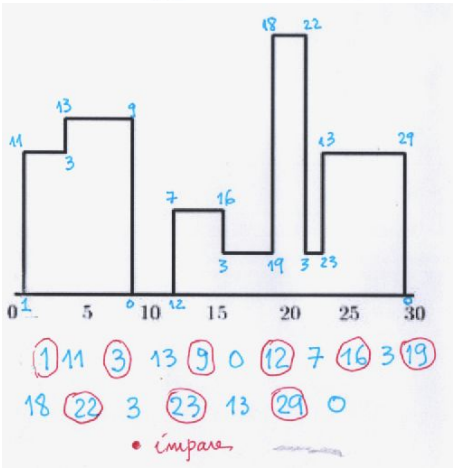
507-75822 -

CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 21/10/2024 - 12:07:31.8
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
VIVXhb1a V: 1.02
75839 IAN DOLABELLA OHATA
24PRG507 - 6 _____ / _____ / _____

# UVA105-Perfil da skyline

Tal como esta lá no site (`http:\\onlinejudge.org`): Com o advento das estações de trabalho gráficas de alta velocidade, CAD (design auxiliado por computador) e outras áreas (design CAM, VLSI) têm feito uso cada vez mais eficaz dos computadores. Um dos problemas com desenhar imagens é a eliminação de linhas ocultas (linhas obscurecidas por outras partes de um desenho). Você deverá projetar um programa para ajudar um arquiteto a desenhar o horizonte de uma cidade, considerando os locais dos edifícios da cidade. Para tornar o problema tratável, todos os edifícios são de forma retangular e eles compartilham um fundo comum (a cidade onde estão construídos é muito plana). A cidade também é vista como bidimensional. Um edifício é especificado por uma tripla ordenada $(L_i, O_i, R_i)$ onde $L_i$ e $R_i$ são as coordenadas à esquerda e à direita, respectivamente, do edifício $_i$, $(0 < L_i < R_i)$ e $H_i$ é a altura do edifício. No diagrama abaixo, os edifícios são mostrados à esquerda com triplas $(1, 11, 5), (2, 6, 7), (3, 13, 9), (12, 7, 16), (14, 3, 25),$ $(19, 18, 22), (23, 13, 29), (24, 4, 28)$ o horizonte, mostrado à direita, é representado pela sequência: $(1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)$ Estes dados correspondem à seguinte imagem:




**Entrada** A entrada é uma sequência de construção de triplas. Todas as coordenadas dos edifícios são números inteiros menores que 10.000 e haverá pelo menos um e no máximo 5.000 edifícios no arquivo de entrada. Cada tripla (edifício) é uma linha sozinha no arquivo de entrada. Todos os inteiros em uma tripla são separados por um ou mais espaços. As triplas serão classificados por $L_i$, a coordenada $x$ esquerda do edifício, então o edifício com o menor valor à esquerda (a coordenada $x$) é o primeiro no arquivo de entrada.

**Saída** A saída deve consistir no vetor que descreve o horizonte conforme mostrado no exemplo acima. No vetor skyline $(v_1, v_2, v_3, ..., v_{n-2}, v_{n-1}, v_n)$, o $v_i$ é tal que se $i$ é um número par representa um linha horizontal (altura). Se $i$ é um número ímpar representa uma linha vertical (coordenada $x$). O vetor skyline deve representar o 'caminho' percorrido, por exemplo, por uma formiga começando no menor coordenada $x$ e viajando horizontal e verticalmente sobre todas as linhas que definem o horizonte. Por isso a última entrada em todos os vetores do horizonte será '0'.

**Exemplo** O desenho acima está assim representado:

```
Entrada
1 11 5
2 6 7
3 13 9
12 7 16
14 3 25
19 18 22
23 13 29
24 4 28
Saída
1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0
```

**Algoritmo 1** Como quase todo problema não trivial, este admite diversas estratégias de solução. Uma possível é desdobrar a cada elemento da lista de entrada em dois eventos: O primeiro determina o início de um prédio e o segundo o seu final. Os eventos terão a estrutura: (esquerda, altura, 0=início) e (direita, altura, 1=final). Para o conjunto de dados do exemplo, ficará: [(1, 11, 1), (5, 11, 0), (2, 6, 1), (7, 6, 0), (3, 13, 1), (9, 13, 0), (12, 7, 1), (16, 7, 0), (14, 3, 1), (25, 3, 0), (19, 18, 1), (22, 18, 0), (23, 13, 1), (29, 13, 0), (24, 4, 1), (28, 4, 0)] A seguir ordena-se esta lista pela primeira posição (o $x$) cria-se uma lista chamada vista vazia ([]) e outra chamada segmentos vazia ([]) e corrente valendo 0 e depois processa-se a lista:

```
se infim==0 (início)
  insere a altura na lista segmentos
  ordena segmentos em ordem decrescente
senão
  remove esta altura de segmentos
nova-altura = primeiro elemento de segmentos
          (ou 0 se vazia)
se nova-altura <> corrente
  insere altura na vista
  corrente = nova-altura
retornar vista
```

Este algoritmo implementado reproduziu os dados do exemplo corretamente, mas foi recusado pelo online judge. Certamente algum conjunto de dados (oculto) ressalvou algum bug ou imprecisão.

**Algoritmo 2** Uma saída foi buscar outro algoritmo mais simples (embora menos eficiente) e baseado numa informação perdida do enunciado (os valores são menores do que 10000). Agora define-se um vetor de 10001 elementos que representa as ordenadas $x$ de todos os envolvidos no problema. Agora a estratégia ficou mais fácil: basta examinar o que acontece em cada ponto do vetor (já que todas as medidas são inteiras).

```
final=0
altura = [0] * 10001
processando (pcom, altu, pfin)
  se pfin>final
    final=pfin (localizando o final do desenho)
  variando i de pcom até pfin
    altura[i] = max(altura[i],altu)
  cara = altura[1]
  imprima (1, cara)
  variando i de 2 até final
    se cara <> altura[i]
      imprima (i, altura[i])
      cara = altura[i]
  imprima (final, 0)
```

## Exemplos resolvidos Caso 1

```
 1 18 11
 4 25  7
 7  6 15
17 26 25
19 25 25
21 12 26
23 28 26
29 19 30
37 16 42
40 22 47
```

Deu como resultado

```
1 18 4 25 7 18 11 6 15 0 17 26 23 28
26 0 29 19 30 0 37 16 40 22 47 0
```

**Caso 2**

```
 1  3  9
12 17 21
24  4 28
24  1 29
31  9 38
39 10 45
41  7 45
42  3 46
44 23 53
45 18 51
45  7 52
47 18 55
```

Deu como resultado

```
1 3 9 0 12 17 21 0 24 4 28 1 29 0 31 9
38 0 39 10 44 23 53 18 55 0
```

## ☞ Para você fazer

Obviamente, antes de executar este caso, você está convidado (intimado !) a submeter sua solução ao online.judge da UVA. Se você obtiver o ACCEPTED no site, pode exigir um 10 nesta folha independente do que o professor (e seu corretor) acharem. Combinado ?

Para deixar as coisas mais ágeis, considere que todos os números deste caso são menores do que 100.

Eis os seus dados aqui:

```
 1  4  2
12 23 22
19 20 28
20  2 29
22 24 30
30 18 31
36 11 45
40 22 47
47 25 52
48 11 53
```

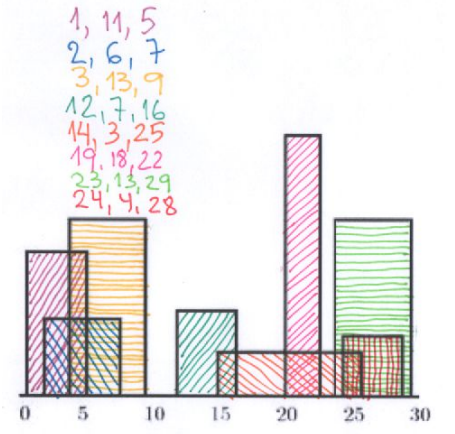
Responda aqui:

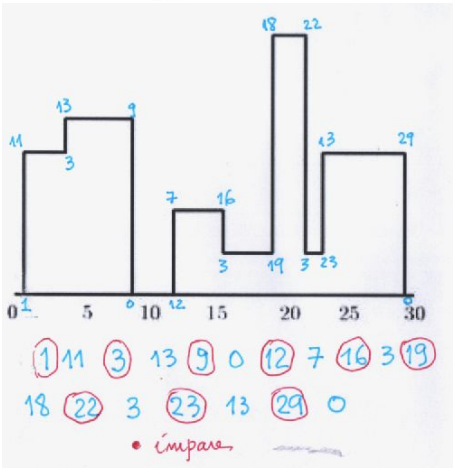
507-75839 -

CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 21/10/2024 - 12:07:31.8
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
VIVXhb1a V: 1.02
75846 JOAO BONFIN LINO
24PRG507 - 7 _____ / _____ / _____

## UVA105-Perfil da skyline

Tal como esta lá no site (`http:\\onlinejudge.org`): Com o advento das estações de trabalho gráficas de alta velocidade, CAD (design auxiliado por computador) e outras áreas (design CAM, VLSI) têm feito uso cada vez mais eficaz dos computadores. Um dos problemas com desenhar imagens é a eliminação de linhas ocultas (linhas obscurecidas por outras partes de um desenho). Você deverá projetar um programa para ajudar um arquiteto a desenhar o horizonte de uma cidade, considerando os locais dos edifícios da cidade. Para tornar o problema tratável, todos os edifícios são de forma retangular e eles compartilham um fundo comum (a cidade onde estão construídos é muito plana). A cidade também é vista como bidimensional. Um edifício é especificado por uma tripla ordenada $(L_i, O_i, R_i)$ onde $L_i$ e $R_i$ são as coordenadas à esquerda e à direita, respectivamente, do edifício $_i$, $(0 < L_i < R_i)$ e $H_i$ é a altura do edifício. No diagrama abaixo, os edifícios são mostrados à esquerda com triplas $(1, 11, 5), (2, 6, 7), (3, 13, 9), (12, 7, 16), (14, 3, 25),$ $(19, 18, 22), (23, 13, 29), (24, 4, 28)$ o horizonte, mostrado à direita, é representado pela sequência: $(1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)$ Estes dados correspondem à seguinte imagem:




**Entrada** A entrada é uma sequência de construção de triplas. Todas as coordenadas dos edifícios são números inteiros menores que 10.000 e haverá pelo menos um e no máximo 5.000 edifícios no arquivo de entrada. Cada tripla (edifício) é uma linha sozinha no arquivo de entrada. Todos os inteiros em uma tripla são separados por um ou mais espaços. As triplas serão classificados por $L_i$, a coordenada $x$ esquerda do edifício, então o edifício com o menor valor à esquerda (a coordenada $x$) é o primeiro no arquivo de entrada.

**Saída** A saída deve consistir no vetor que descreve o horizonte conforme mostrado no exemplo acima. No vetor skyline $(v_1, v_2, v_3, ..., v_{n-2}, v_{n-1}, v_n)$, o $v_i$ é tal que se $i$ é um número par representa um linha horizontal (altura). Se $i$ é um número ímpar representa uma linha vertical (coordenada $x$). O vetor skyline deve representar o 'caminho' percorrido, por exemplo, por uma formiga começando no menor coordenada $x$ e viajando horizontal e verticalmente sobre todas as linhas que definem o horizonte. Por isso a última entrada em todos os vetores do horizonte será '0'.

**Exemplo** O desenho acima está assim representado:

```
Entrada
1 11 5
2 6 7
3 13 9
12 7 16
14 3 25
19 18 22
23 13 29
24 4 28
Saída
1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0
```

**Algoritmo 1** Como quase todo problema não trivial, este admite diversas estratégias de solução. Uma possível é desdobrar a cada elemento da lista de entrada em dois eventos: O primeiro determina o início de um prédio e o segundo o seu final. Os eventos terão a estrutura: (esquerda, altura, 0=início) e (direita, altura, 1=final). Para o conjunto de dados do exemplo, ficará: [(1, 11, 1), (5, 11, 0), (2, 6, 1), (7, 6, 0), (3, 13, 1), (9, 13, 0), (12, 7, 1), (16, 7, 0), (14, 3, 1), (25, 3, 0), (19, 18, 1), (22, 18, 0), (23, 13, 1), (29, 13, 0), (24, 4, 1), (28, 4, 0)] A seguir ordena-se esta lista pela primeira posição (o $x$) cria-se uma lista chamada vista vazia ([]) e outra chamada segmentos vazia ([]) e corrente valendo 0 e depois processa-se a lista:

```
se infim==0 (início)
  insere a altura na lista segmentos
  ordena segmentos em ordem decrescente
senão
  remove esta altura de segmentos
nova-altura = primeiro elemento de segmentos
          (ou 0 se vazia)
se nova-altura <> corrente
  insere altura na vista
  corrente = nova-altura
retornar vista
```

Este algoritmo implementado reproduziu os dados do exemplo corretamente, mas foi recusado pelo online judge. Certamente algum conjunto de dados (oculto) ressalvou algum bug ou imprecisão.

**Algoritmo 2** Uma saída foi buscar outro algoritmo mais simples (embora menos eficiente) e baseado numa informação perdida do enunciado (os valores são menores do que 10000). Agora define-se um vetor de 10001 elementos que representa as ordenadas $x$ de todos os envolvidos no problema. Agora a estratégia ficou mais fácil: basta examinar o que acontece em cada ponto do vetor (já que todas as medidas são inteiras).

```
final=0
altura = [0] * 10001
processando (pcom, altu, pfin)
  se pfin>final
    final=pfin (localizando o final do desenho)
  variando i de pcom até pfin
    altura[i] = max(altura[i],altu)
  cara = altura[1]
  imprima (1, cara)
  variando i de 2 até final
    se cara <> altura[i]
      imprima (i, altura[i])
      cara = altura[i]
  imprima (final, 0)
```

## Exemplos resolvidos Caso 1

```
 1 18 11
 4 25  7
 7  6 15
17 26 25
19 25 25
21 12 26
23 28 26
29 19 30
37 16 42
40 22 47
```

Deu como resultado

```
1 18 4 25 7 18 11 6 15 0 17 26 23 28
26 0 29 19 30 0 37 16 40 22 47 0
```

**Caso 2**

```
 1  3  9
12 17 21
24  4 28
24  1 29
31  9 38
39 10 45
41  7 45
42  3 46
44 23 53
45 18 51
45  7 52
47 18 55
```

Deu como resultado

```
1 3 9 0 12 17 21 0 24 4 28 1 29 0 31 9
38 0 39 10 44 23 53 18 55 0
```

## ☞ Para você fazer

Obviamente, antes de executar este caso, você está convidado (intimado !) a submeter sua solução ao online.judge da UVA. Se você obtiver o ACCEPTED no site, pode exigir um 10 nesta folha independente do que o professor (e seu corretor) acharem. Combinado ?

Para deixar as coisas mais ágeis, considere que todos os números deste caso são menores do que 100.

Eis os seus dados aqui:

```
 1 21  9
 7 30 10
 8  1 11
12  9 14
18 12 20
18  8 28
23  1 24
27 29 28
33 19 37
38  8 39
```

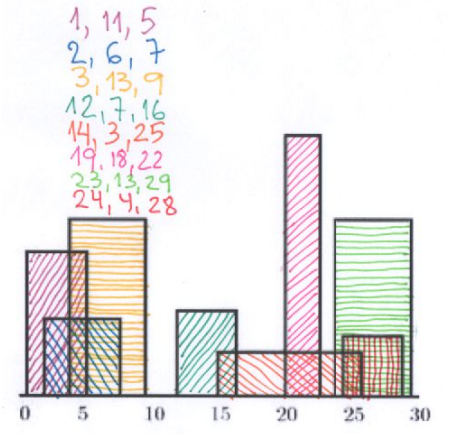
Responda aqui:

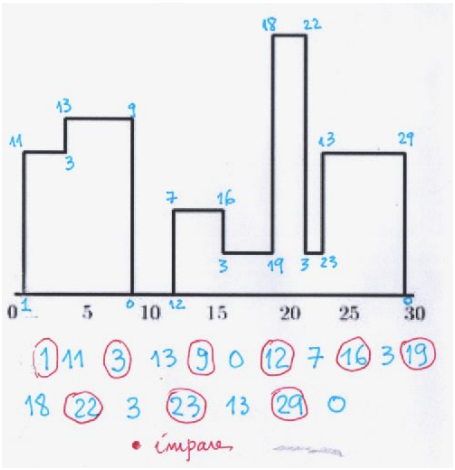
507-75846 -

CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 21/10/2024 - 12:07:31.8
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
VIVXhb1a V: 1.02
75853 JOAO LUCA FERNANDES DE OLIVE
24PRG507 - 8 _____ / _____ / _____

## UVA105-Perfil da skyline

Tal como esta lá no site (`http:\\onlinejudge.org`): Com o advento das estações de trabalho gráficas de alta velocidade, CAD (design auxiliado por computador) e outras áreas (design CAM, VLSI) têm feito uso cada vez mais eficaz dos computadores. Um dos problemas com desenhar imagens é a eliminação de linhas ocultas (linhas obscurecidas por outras partes de um desenho). Você deverá projetar um programa para ajudar um arquiteto a desenhar o horizonte de uma cidade, considerando os locais dos edifícios da cidade. Para tornar o problema tratável, todos os edifícios são de forma retangular e eles compartilham um fundo comum (a cidade onde estão construídos é muito plana). A cidade também é vista como bidimensional. Um edifício é especificado por uma tripla ordenada $(L_i, O_i, R_i)$ onde $L_i$ e $R_i$ são as coordenadas à esquerda e à direita, respectivamente, do edifício $_i$, $(0 < L_i < R_i)$ e $H_i$ é a altura do edifício. No diagrama abaixo, os edifícios são mostrados à esquerda com triplas $(1, 11, 5), (2, 6, 7), (3, 13, 9), (12, 7, 16), (14, 3, 25),$ $(19, 18, 22), (23, 13, 29), (24, 4, 28)$ o horizonte, mostrado à direita, é representado pela sequência: $(1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)$ Estes dados correspondem à seguinte imagem:




**Entrada** A entrada é uma sequência de construção de triplas. Todas as coordenadas dos edifícios são números inteiros menores que 10.000 e haverá pelo menos um e no máximo 5.000 edifícios no arquivo de entrada. Cada tripla (edifício) é uma linha sozinha no arquivo de entrada. Todos os inteiros em uma tripla são separados por um ou mais espaços. As triplas serão classificados por $L_i$, a coordenada $x$ esquerda do edifício, então o edifício com o menor valor à esquerda (a coordenada $x$) é o primeiro no arquivo de entrada.

**Saída** A saída deve consistir no vetor que descreve o horizonte conforme mostrado no exemplo acima. No vetor skyline $(v_1, v_2, v_3, ..., v_{n-2}, v_{n-1}, v_n)$, o $v_i$ é tal que se $i$ é um número par representa um linha horizontal (altura). Se $i$ é um número ímpar representa uma linha vertical (coordenada $x$). O vetor skyline deve representar o 'caminho' percorrido, por exemplo, por uma formiga começando no menor coordenada $x$ e viajando horizontal e verticalmente sobre todas as linhas que definem o horizonte. Por isso a última entrada em todos os vetores do horizonte será '0'.

**Exemplo** O desenho acima está assim representado:

```
Entrada
1 11 5
2 6 7
3 13 9
12 7 16
14 3 25
19 18 22
23 13 29
24 4 28
Saída
1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0
```

**Algoritmo 1** Como quase todo problema não trivial, este admite diversas estratégias de solução. Uma possível é desdobrar a cada elemento da lista de entrada em dois eventos: O primeiro determina o início de um prédio e o segundo o seu final. Os eventos terão a estrutura: (esquerda, altura, 0=início) e (direita, altura, 1=final). Para o conjunto de dados do exemplo, ficará: [(1, 11, 1), (5, 11, 0), (2, 6, 1), (7, 6, 0), (3, 13, 1), (9, 13, 0), (12, 7, 1), (16, 7, 0), (14, 3, 1), (25, 3, 0), (19, 18, 1), (22, 18, 0), (23, 13, 1), (29, 13, 0), (24, 4, 1), (28, 4, 0)] A seguir ordena-se esta lista pela primeira posição (o $x$) cria-se uma lista chamada vista vazia ([]) e outra chamada segmentos vazia ([]) e corrente valendo 0 e depois processa-se a lista:

```
se infim==0 (início)
  insere a altura na lista segmentos
  ordena segmentos em ordem decrescente
senão
  remove esta altura de segmentos
nova-altura = primeiro elemento de segmentos
          (ou 0 se vazia)
se nova-altura <> corrente
  insere altura na vista
  corrente = nova-altura
retornar vista
```

Este algoritmo implementado reproduziu os dados do exemplo corretamente, mas foi recusado pelo online judge. Certamente algum conjunto de dados (oculto) ressalvou algum bug ou imprecisão.

**Algoritmo 2** Uma saída foi buscar outro algoritmo mais simples (embora menos eficiente) e baseado numa informação perdida do enunciado (os valores são menores do que 10000). Agora define-se um vetor de 10001 elementos que representa as ordenadas $x$ de todos os envolvidos no problema. Agora a estratégia ficou mais fácil: basta examinar o que acontece em cada ponto do vetor (já que todas as medidas são inteiras).

```
final=0
altura = [0] * 10001
processando (pcom, altu, pfin)
  se pfin>final
    final=pfin (localizando o final do desenho)
  variando i de pcom até pfin
    altura[i] = max(altura[i],altu)
  cara = altura[1]
  imprima (1, cara)
  variando i de 2 até final
    se cara <> altura[i]
      imprima (i, altura[i])
      cara = altura[i]
  imprima (final, 0)
```

## Exemplos resolvidos Caso 1

```
 1 18 11
 4 25  7
 7  6 15
17 26 25
19 25 25
21 12 26
23 28 26
29 19 30
37 16 42
40 22 47
```

Deu como resultado

```
1 18 4 25 7 18 11 6 15 0 17 26 23 28
26 0 29 19 30 0 37 16 40 22 47 0
```

**Caso 2**

```
 1  3  9
12 17 21
24  4 28
24  1 29
31  9 38
39 10 45
41  7 45
42  3 46
44 23 53
45 18 51
45  7 52
47 18 55
```

Deu como resultado

```
1 3 9 0 12 17 21 0 24 4 28 1 29 0 31 9
38 0 39 10 44 23 53 18 55 0
```

## ☞ Para você fazer

Obviamente, antes de executar este caso, você está convidado (intimado !) a submeter sua solução ao online.judge da UVA. Se você obtiver o ACCEPTED no site, pode exigir um 10 nesta folha independente do que o professor (e seu corretor) acharem. Combinado ?

Para deixar as coisas mais ágeis, considere que todos os números deste caso são menores do que 100.

Eis os seus dados aqui:

```
 1  8  7
 5 23 10
12 27 21
13 11 22
14 24 20
20 13 23
24 30 29
33 20 37
36  6 42
42 26 51
```

Responda aqui:

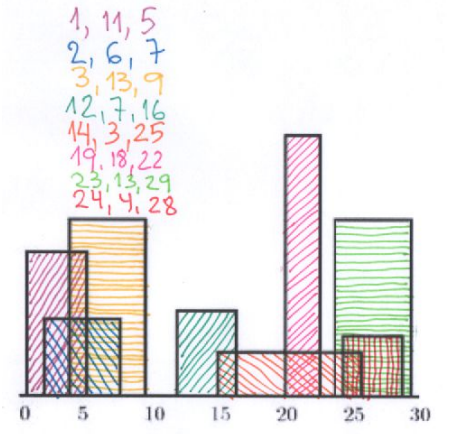
507-75853 -

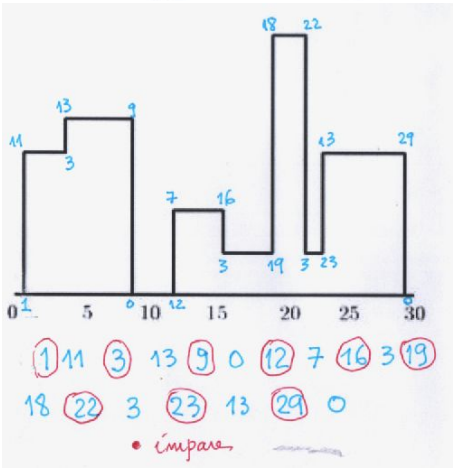
CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 21/10/2024 - 12:07:31.8
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)

VIVXhb1a V: 1.02
75860 JOAO PEDRO DE BRITO DUARTE
24PRG507 - 9 _____ / _____ / _____

## UVA105-Perfil da skyline

Tal como esta lá no site (`http:\\onlinejudge.org`): Com o advento das estações de trabalho gráficas de alta velocidade, CAD (design auxiliado por computador) e outras áreas (design CAM, VLSI) têm feito uso cada vez mais eficaz dos computadores. Um dos problemas com desenhar imagens é a eliminação de linhas ocultas (linhas obscurecidas por outras partes de um desenho). Você deverá projetar um programa para ajudar um arquiteto a desenhar o horizonte de uma cidade, considerando os locais dos edifícios da cidade. Para tornar o problema tratável, todos os edifícios são de forma retangular e eles compartilham um fundo comum (a cidade onde estão construídos é muito plana). A cidade também é vista como bidimensional. Um edifício é especificado por uma tripla ordenada $(L_i, O_i, R_i)$ onde $L_i$ e $R_i$ são as coordenadas à esquerda e à direita, respectivamente, do edifício $_i$, $(0 < L_i < R_i)$ e $H_i$ é a altura do edifício. No diagrama abaixo, os edifícios são mostrados à esquerda com triplas $(1, 11, 5), (2, 6, 7), (3, 13, 9), (12, 7, 16), (14, 3, 25),$ $(19, 18, 22), (23, 13, 29), (24, 4, 28)$ o horizonte, mostrado à direita, é representado pela sequência: $(1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)$ Estes dados correspondem à seguinte imagem:




**Entrada** A entrada é uma sequência de construção de triplas. Todas as coordenadas dos edifícios são números inteiros menores que 10.000 e haverá pelo menos um e no máximo 5.000 edifícios no arquivo de entrada. Cada tripla (edifício) é uma linha sozinha no arquivo de entrada. Todos os inteiros em uma tripla são separados por um ou mais espaços. As triplas serão classificados por $L_i$, a coordenada $x$ esquerda do edifício, então o edifício com o menor valor à esquerda (a coordenada $x$) é o primeiro no arquivo de entrada.

**Saída** A saída deve consistir no vetor que descreve o horizonte conforme mostrado no exemplo acima. No vetor skyline $(v_1, v_2, v_3, ..., v_{n-2}, v_{n-1}, v_n)$, o $v_i$ é tal que se $i$ é um número par representa um linha horizontal (altura). Se $i$ é um número ímpar representa uma linha vertical (coordenada $x$). O vetor skyline deve representar o 'caminho' percorrido, por exemplo, por uma formiga começando no menor coordenada $x$ e viajando horizontal e verticalmente sobre todas as linhas que definem o horizonte. Por isso a última entrada em todos os vetores do horizonte será '0'.

**Exemplo** O desenho acima está assim representado:

```
Entrada
1 11 5
2 6 7
3 13 9
12 7 16
14 3 25
19 18 22
23 13 29
24 4 28
Saída
1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0
```

**Algoritmo 1** Como quase todo problema não trivial, este admite diversas estratégias de solução. Uma possível é desdobrar a cada elemento da lista de entrada em dois eventos: O primeiro determina o início de um prédio e o segundo o seu final. Os eventos terão a estrutura: (esquerda, altura, 0=início) e (direita, altura, 1=final). Para o conjunto de dados do exemplo, ficará: [(1, 11, 1), (5, 11, 0), (2, 6, 1), (7, 6, 0), (3, 13, 1), (9, 13, 0), (12, 7, 1), (16, 7, 0), (14, 3, 1), (25, 3, 0), (19, 18, 1), (22, 18, 0), (23, 13, 1), (29, 13, 0), (24, 4, 1), (28, 4, 0)] A seguir ordena-se esta lista pela primeira posição (o $x$) cria-se uma lista chamada vista vazia ([]) e outra chamada segmentos vazia ([]) e corrente valendo 0 e depois processa-se a lista:

```
se infim==0 (início)
  insere a altura na lista segmentos
  ordena segmentos em ordem decrescente
senão
  remove esta altura de segmentos
nova-altura = primeiro elemento de segmentos
          (ou 0 se vazia)
se nova-altura <> corrente
  insere altura na vista
  corrente = nova-altura
retornar vista
```

Este algoritmo implementado reproduziu os dados do exemplo corretamente, mas foi recusado pelo online judge. Certamente algum conjunto de dados (oculto) ressalvou algum bug ou imprecisão.

**Algoritmo 2** Uma saída foi buscar outro algoritmo mais simples (embora menos eficiente) e baseado numa informação perdida do enunciado (os valores são menores do que 10000). Agora define-se um vetor de 10001 elementos que representa as ordenadas $x$ de todos os envolvidos no problema. Agora a estratégia ficou mais fácil: basta examinar o que acontece em cada ponto do vetor (já que todas as medidas são inteiras).

```
final=0
altura = [0] * 10001
processando (pcom, altu, pfin)
  se pfin>final
    final=pfin (localizando o final do desenho)
  variando i de pcom até pfin
    altura[i] = max(altura[i],altu)
  cara = altura[1]
  imprima (1, cara)
  variando i de 2 até final
    se cara <> altura[i]
      imprima (i, altura[i])
      cara = altura[i]
  imprima (final, 0)
```

## Exemplos resolvidos Caso 1

```
 1 18 11
 4 25  7
 7  6 15
17 26 25
19 25 25
21 12 26
23 28 26
29 19 30
37 16 42
40 22 47
```

Deu como resultado

```
1 18 4 25 7 18 11 6 15 0 17 26 23 28
26 0 29 19 30 0 37 16 40 22 47 0
```

**Caso 2**

```
 1  3  9
12 17 21
24  4 28
24  1 29
31  9 38
39 10 45
41  7 45
42  3 46
44 23 53
45 18 51
45  7 52
47 18 55
```

Deu como resultado

```
1 3 9 0 12 17 21 0 24 4 28 1 29 0 31 9
38 0 39 10 44 23 53 18 55 0
```

## ☞ Para você fazer

Obviamente, antes de executar este caso, você está convidado (intimado !) a submeter sua solução ao online.judge da UVA. Se você obtiver o ACCEPTED no site, pode exigir um 10 nesta folha independente do que o professor (e seu corretor) acharem. Combinado ?

Para deixar as coisas mais ágeis, considere que todos os números deste caso são menores do que 100.

Eis os seus dados aqui:

```
 1 10  3
 9  5 16
14 26 18
14  2 17
18 30 26
32 23 38
33 18 34
38 20 46
44  9 48
50 11 58
```

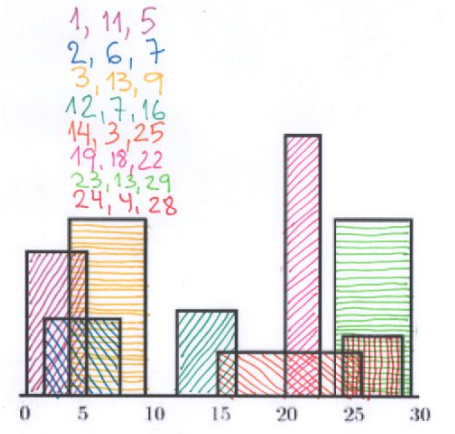
Responda aqui:

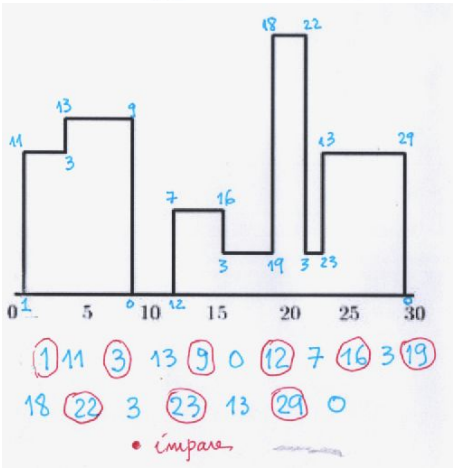
507-75860 -

CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 21/10/2024 - 12:07:31.8
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
VIVXhb1a V: 1.02
75877 JOAO PEDRO MARIANO SUEKE
24PRG507 - 10 _____ / _____ / _____

## UVA105-Perfil da skyline

Tal como esta lá no site (`http:\\onlinejudge.org`): Com o advento das estações de trabalho gráficas de alta velocidade, CAD (design auxiliado por computador) e outras áreas (design CAM, VLSI) têm feito uso cada vez mais eficaz dos computadores. Um dos problemas com desenhar imagens é a eliminação de linhas ocultas (linhas obscurecidas por outras partes de um desenho). Você deverá projetar um programa para ajudar um arquiteto a desenhar o horizonte de uma cidade, considerando os locais dos edifícios da cidade. Para tornar o problema tratável, todos os edifícios são de forma retangular e eles compartilham um fundo comum (a cidade onde estão construídos é muito plana). A cidade também é vista como bidimensional. Um edifício é especificado por uma tripla ordenada $(L_i, O_i, R_i)$ onde $L_i$ e $R_i$ são as coordenadas à esquerda e à direita, respectivamente, do edifício $_i$, $(0 < L_i < R_i)$ e $H_i$ é a altura do edifício. No diagrama abaixo, os edifícios são mostrados à esquerda com triplas $(1, 11, 5), (2, 6, 7), (3, 13, 9), (12, 7, 16), (14, 3, 25),$ $(19, 18, 22), (23, 13, 29), (24, 4, 28)$ o horizonte, mostrado à direita, é representado pela sequência: $(1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)$ Estes dados correspondem à seguinte imagem:




**Entrada** A entrada é uma sequência de construção de triplas. Todas as coordenadas dos edifícios são números inteiros menores que 10.000 e haverá pelo menos um e no máximo 5.000 edifícios no arquivo de entrada. Cada tripla (edifício) é uma linha sozinha no arquivo de entrada. Todos os inteiros em uma tripla são separados por um ou mais espaços. As triplas serão classificados por $L_i$, a coordenada $x$ esquerda do edifício, então o edifício com o menor valor à esquerda (a coordenada $x$) é o primeiro no arquivo de entrada.

**Saída** A saída deve consistir no vetor que descreve o horizonte conforme mostrado no exemplo acima. No vetor skyline $(v_1, v_2, v_3, ..., v_{n-2}, v_{n-1}, v_n)$, o $v_i$ é tal que se $i$ é um número par representa um linha horizontal (altura). Se $i$ é um número ímpar representa uma linha vertical (coordenada $x$). O vetor skyline deve representar o 'caminho' percorrido, por exemplo, por uma formiga começando no menor coordenada $x$ e viajando horizontal e verticalmente sobre todas as linhas que definem o horizonte. Por isso a última entrada em todos os vetores do horizonte será '0'.

**Exemplo** O desenho acima está assim representado:

```
Entrada
1 11 5
2 6 7
3 13 9
12 7 16
14 3 25
19 18 22
23 13 29
24 4 28
Saída
1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0
```

**Algoritmo 1** Como quase todo problema não trivial, este admite diversas estratégias de solução. Uma possível é desdobrar a cada elemento da lista de entrada em dois eventos: O primeiro determina o início de um prédio e o segundo o seu final. Os eventos terão a estrutura: (esquerda, altura, 0=início) e (direita, altura, 1=final). Para o conjunto de dados do exemplo, ficará: [(1, 11, 1), (5, 11, 0), (2, 6, 1), (7, 6, 0), (3, 13, 1), (9, 13, 0), (12, 7, 1), (16, 7, 0), (14, 3, 1), (25, 3, 0), (19, 18, 1), (22, 18, 0), (23, 13, 1), (29, 13, 0), (24, 4, 1), (28, 4, 0)] A seguir ordena-se esta lista pela primeira posição (o $x$) cria-se uma lista chamada vista vazia ([]) e outra chamada segmentos vazia ([]) e corrente valendo 0 e depois processa-se a lista:

```
se infim==0 (início)
  insere a altura na lista segmentos
  ordena segmentos em ordem decrescente
senão
  remove esta altura de segmentos
nova-altura = primeiro elemento de segmentos
          (ou 0 se vazia)
se nova-altura <> corrente
  insere altura na vista
  corrente = nova-altura
retornar vista
```

Este algoritmo implementado reproduziu os dados do exemplo corretamente, mas foi recusado pelo online judge. Certamente algum conjunto de dados (oculto) ressalvou algum bug ou imprecisão.

**Algoritmo 2** Uma saída foi buscar outro algoritmo mais simples (embora menos eficiente) e baseado numa informação perdida do enunciado (os valores são menores do que 10000). Agora define-se um vetor de 10001 elementos que representa as ordenadas $x$ de todos os envolvidos no problema. Agora a estratégia ficou mais fácil: basta examinar o que acontece em cada ponto do vetor (já que todas as medidas são inteiras).

```
final=0
altura = [0] * 10001
processando (pcom, altu, pfin)
  se pfin>final
    final=pfin (localizando o final do desenho)
  variando i de pcom até pfin
    altura[i] = max(altura[i],altu)
  cara = altura[1]
  imprima (1, cara)
  variando i de 2 até final
    se cara <> altura[i]
      imprima (i, altura[i])
      cara = altura[i]
  imprima (final, 0)
```

## Exemplos resolvidos Caso 1

```
 1 18 11
 4 25  7
 7  6 15
17 26 25
19 25 25
21 12 26
23 28 26
29 19 30
37 16 42
40 22 47
```

Deu como resultado

```
1 18 4 25 7 18 11 6 15 0 17 26 23 28
26 0 29 19 30 0 37 16 40 22 47 0
```

**Caso 2**

```
 1  3  9
12 17 21
24  4 28
24  1 29
31  9 38
39 10 45
41  7 45
42  3 46
44 23 53
45 18 51
45  7 52
47 18 55
```

Deu como resultado

```
1 3 9 0 12 17 21 0 24 4 28 1 29 0 31 9
38 0 39 10 44 23 53 18 55 0
```

## ☞ Para você fazer

Obviamente, antes de executar este caso, você está convidado (intimado !) a submeter sua solução ao online.judge da UVA. Se você obtiver o ACCEPTED no site, pode exigir um 10 nesta folha independente do que o professor (e seu corretor) acharem. Combinado ?

Para deixar as coisas mais ágeis, considere que todos os números deste caso são menores do que 100.

Eis os seus dados aqui:

```
 1  9 10
12  8 18
20 22 24
23 26 33
26  9 32
34 17 35
36  1 39
39  1 42
40 28 41
47 28 57
```

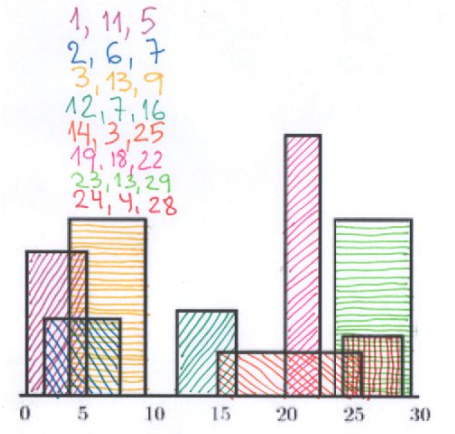
Responda aqui:

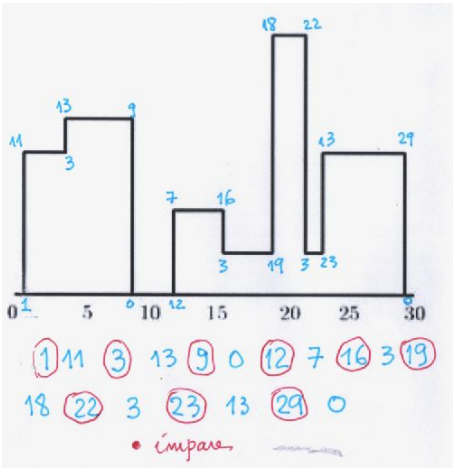
507-75877 -

CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 21/10/2024 - 12:07:31.8
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
VIVXhb1a V: 1.02
75884 JOAO VICTOR GOMES VILELA GER
24PRG507 - 11 _____ / _____ / _____

## UVA105-Perfil da skyline

Tal como esta lá no site (`http:\\onlinejudge.org`): Com o advento das estações de trabalho gráficas de alta velocidade, CAD (design auxiliado por computador) e outras áreas (design CAM, VLSI) têm feito uso cada vez mais eficaz dos computadores. Um dos problemas com desenhar imagens é a eliminação de linhas ocultas (linhas obscurecidas por outras partes de um desenho). Você deverá projetar um programa para ajudar um arquiteto a desenhar o horizonte de uma cidade, considerando os locais dos edifícios da cidade. Para tornar o problema tratável, todos os edifícios são de forma retangular e eles compartilham um fundo comum (a cidade onde estão construídos é muito plana). A cidade também é vista como bidimensional. Um edifício é especificado por uma tripla ordenada $(L_i, O_i, R_i)$ onde $L_i$ e $R_i$ são as coordenadas à esquerda e à direita, respectivamente, do edifício $_i$, $(0 < L_i < R_i)$ e $H_i$ é a altura do edifício. No diagrama abaixo, os edifícios são mostrados à esquerda com triplas $(1, 11, 5), (2, 6, 7), (3, 13, 9), (12, 7, 16), (14, 3, 25),$ $(19, 18, 22), (23, 13, 29), (24, 4, 28)$ o horizonte, mostrado à direita, é representado pela sequência: $(1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)$ Estes dados correspondem à seguinte imagem:




**Entrada** A entrada é uma sequência de construção de triplas. Todas as coordenadas dos edifícios são números inteiros menores que 10.000 e haverá pelo menos um e no máximo 5.000 edifícios no arquivo de entrada. Cada tripla (edifício) é uma linha sozinha no arquivo de entrada. Todos os inteiros em uma tripla são separados por um ou mais espaços. As triplas serão classificados por $L_i$, a coordenada $x$ esquerda do edifício, então o edifício com o menor valor à esquerda (a coordenada $x$) é o primeiro no arquivo de entrada.

**Saída** A saída deve consistir no vetor que descreve o horizonte conforme mostrado no exemplo acima. No vetor skyline $(v_1, v_2, v_3, ..., v_{n-2}, v_{n-1}, v_n)$, o $v_i$ é tal que se $i$ é um número par representa um linha horizontal (altura). Se $i$ é um número ímpar representa uma linha vertical (coordenada $x$). O vetor skyline deve representar o 'caminho' percorrido, por exemplo, por uma formiga começando no menor coordenada $x$ e viajando horizontal e verticalmente sobre todas as linhas que definem o horizonte. Por isso a última entrada em todos os vetores do horizonte será '0'.

**Exemplo** O desenho acima está assim representado:

```
Entrada
1 11 5
2 6 7
3 13 9
12 7 16
14 3 25
19 18 22
23 13 29
24 4 28
Saída
1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0
```

**Algoritmo 1** Como quase todo problema não trivial, este admite diversas estratégias de solução. Uma possível é desdobrar a cada elemento da lista de entrada em dois eventos: O primeiro determina o início de um prédio e o segundo o seu final. Os eventos terão a estrutura: (esquerda, altura, 0=início) e (direita, altura, 1=final). Para o conjunto de dados do exemplo, ficará: [(1, 11, 1), (5, 11, 0), (2, 6, 1), (7, 6, 0), (3, 13, 1), (9, 13, 0), (12, 7, 1), (16, 7, 0), (14, 3, 1), (25, 3, 0), (19, 18, 1), (22, 18, 0), (23, 13, 1), (29, 13, 0), (24, 4, 1), (28, 4, 0)] A seguir ordena-se esta lista pela primeira posição (o $x$) cria-se uma lista chamada vista vazia ([]) e outra chamada segmentos vazia ([]) e corrente valendo 0 e depois processa-se a lista:

```
se infim==0 (início)
  insere a altura na lista segmentos
  ordena segmentos em ordem decrescente
senão
  remove esta altura de segmentos
nova-altura = primeiro elemento de segmentos
          (ou 0 se vazia)
se nova-altura <> corrente
  insere altura na vista
  corrente = nova-altura
retornar vista
```

Este algoritmo implementado reproduziu os dados do exemplo corretamente, mas foi recusado pelo online judge. Certamente algum conjunto de dados (oculto) ressalvou algum bug ou imprecisão.

**Algoritmo 2** Uma saída foi buscar outro algoritmo mais simples (embora menos eficiente) e baseado numa informação perdida do enunciado (os valores são menores do que 10000). Agora define-se um vetor de 10001 elementos que representa as ordenadas $x$ de todos os envolvidos no problema. Agora a estratégia ficou mais fácil: basta examinar o que acontece em cada ponto do vetor (já que todas as medidas são inteiras).

```
final=0
altura = [0] * 10001
processando (pcom, altu, pfin)
  se pfin>final
    final=pfin (localizando o final do desenho)
  variando i de pcom até pfin
    altura[i] = max(altura[i],altu)
  cara = altura[1]
  imprima (1, cara)
  variando i de 2 até final
    se cara <> altura[i]
      imprima (i, altura[i])
      cara = altura[i]
  imprima (final, 0)
```

## Exemplos resolvidos Caso 1

```
 1 18 11
 4 25  7
 7  6 15
17 26 25
19 25 25
21 12 26
23 28 26
29 19 30
37 16 42
40 22 47
```

Deu como resultado

```
1 18 4 25 7 18 11 6 15 0 17 26 23 28
26 0 29 19 30 0 37 16 40 22 47 0
```

**Caso 2**

```
 1  3  9
12 17 21
24  4 28
24  1 29
31  9 38
39 10 45
41  7 45
42  3 46
44 23 53
45 18 51
45  7 52
47 18 55
```

Deu como resultado

```
1 3 9 0 12 17 21 0 24 4 28 1 29 0 31 9
38 0 39 10 44 23 53 18 55 0
```

## ☞ Para você fazer

Obviamente, antes de executar este caso, você está convidado (intimado !) a submeter sua solução ao online.judge da UVA. Se você obtiver o ACCEPTED no site, pode exigir um 10 nesta folha independente do que o professor (e seu corretor) acharem. Combinado ?

Para deixar as coisas mais ágeis, considere que todos os números deste caso são menores do que 100.

Eis os seus dados aqui:

```
 1 28  4
12 13 15
14  6 15
14 15 20
18  8 25
26 11 27
31 12 32
33 22 40
34 11 36
49 15 57
```

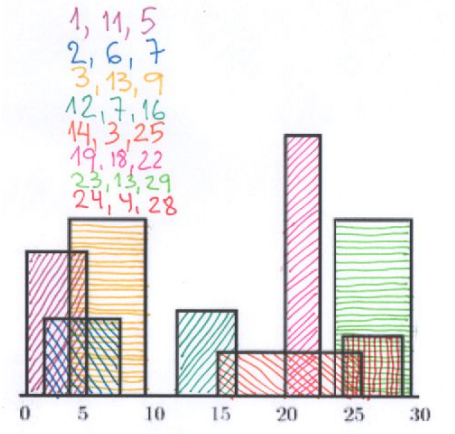
Responda aqui:

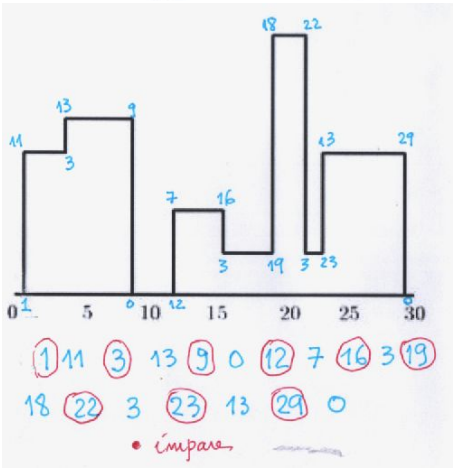
507-75884 -

CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 21/10/2024 - 12:07:31.8
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
VIVXhb1a V: 1.02
75989 LUCAS ANTOSZCZYSZEN
24PRG507 - 12 _____ / _____ / _____

# UVA105-Perfil da skyline

Tal como esta lá no site (`http:\\onlinejudge.org`): Com o advento das estações de trabalho gráficas de alta velocidade, CAD (design auxiliado por computador) e outras áreas (design CAM, VLSI) têm feito uso cada vez mais eficaz dos computadores. Um dos problemas com desenhar imagens é a eliminação de linhas ocultas (linhas obscurecidas por outras partes de um desenho). Você deverá projetar um programa para ajudar um arquiteto a desenhar o horizonte de uma cidade, considerando os locais dos edifícios da cidade. Para tornar o problema tratável, todos os edifícios são de forma retangular e eles compartilham um fundo comum (a cidade onde estão construídos é muito plana). A cidade também é vista como bidimensional. Um edifício é especificado por uma tripla ordenada $(L_i, O_i, R_i)$ onde $L_i$ e $R_i$ são as coordenadas à esquerda e à direita, respectivamente, do edifício $_i$, $(0 < L_i < R_i)$ e $H_i$ é a altura do edifício. No diagrama abaixo, os edifícios são mostrados à esquerda com triplas $(1, 11, 5), (2, 6, 7), (3, 13, 9), (12, 7, 16), (14, 3, 25),$ $(19, 18, 22), (23, 13, 29), (24, 4, 28)$ o horizonte, mostrado à direita, é representado pela sequência: $(1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)$ Estes dados correspondem à seguinte imagem:




**Entrada** A entrada é uma sequência de construção de triplas. Todas as coordenadas dos edifícios são números inteiros menores que 10.000 e haverá pelo menos um e no máximo 5.000 edifícios no arquivo de entrada. Cada tripla (edifício) é uma linha sozinha no arquivo de entrada. Todos os inteiros em uma tripla são separados por um ou mais espaços. As triplas serão classificados por $L_i$, a coordenada $x$ esquerda do edifício, então o edifício com o menor valor à esquerda (a coordenada $x$) é o primeiro no arquivo de entrada.

**Saída** A saída deve consistir no vetor que descreve o horizonte conforme mostrado no exemplo acima. No vetor skyline $(v_1, v_2, v_3, ..., v_{n-2}, v_{n-1}, v_n)$, o $v_i$ é tal que se $i$ é um número par representa um linha horizontal (altura). Se $i$ é um número ímpar representa uma linha vertical (coordenada $x$). O vetor skyline deve representar o 'caminho' percorrido, por exemplo, por uma formiga começando no menor coordenada $x$ e viajando horizontal e verticalmente sobre todas as linhas que definem o horizonte. Por isso a última entrada em todos os vetores do horizonte será '0'.

**Exemplo** O desenho acima está assim representado:

```
Entrada
1 11 5
2 6 7
3 13 9
12 7 16
14 3 25
19 18 22
23 13 29
24 4 28
Saída
1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0
```

**Algoritmo 1** Como quase todo problema não trivial, este admite diversas estratégias de solução. Uma possível é desdobrar a cada elemento da lista de entrada em dois eventos: O primeiro determina o início de um prédio e o segundo o seu final. Os eventos terão a estrutura: (esquerda, altura, 0=início) e (direita, altura, 1=final). Para o conjunto de dados do exemplo, ficará: [(1, 11, 1), (5, 11, 0), (2, 6, 1), (7, 6, 0), (3, 13, 1), (9, 13, 0), (12, 7, 1), (16, 7, 0), (14, 3, 1), (25, 3, 0), (19, 18, 1), (22, 18, 0), (23, 13, 1), (29, 13, 0), (24, 4, 1), (28, 4, 0)] A seguir ordena-se esta lista pela primeira posição (o $x$) cria-se uma lista chamada vista vazia ([]) e outra chamada segmentos vazia ([]) e corrente valendo 0 e depois processa-se a lista:

```
se infim==0 (início)
  insere a altura na lista segmentos
  ordena segmentos em ordem decrescente
senão
  remove esta altura de segmentos
nova-altura = primeiro elemento de segmentos
          (ou 0 se vazia)
se nova-altura <> corrente
  insere altura na vista
  corrente = nova-altura
retornar vista
```

Este algoritmo implementado reproduziu os dados do exemplo corretamente, mas foi recusado pelo online judge. Certamente algum conjunto de dados (oculto) ressalvou algum bug ou imprecisão.

**Algoritmo 2** Uma saída foi buscar outro algoritmo mais simples (embora menos eficiente) e baseado numa informação perdida do enunciado (os valores são menores do que 10000). Agora define-se um vetor de 10001 elementos que representa as ordenadas $x$ de todos os envolvidos no problema. Agora a estratégia ficou mais fácil: basta examinar o que acontece em cada ponto do vetor (já que todas as medidas são inteiras).

```
final=0
altura = [0] * 10001
processando (pcom, altu, pfin)
  se pfin>final
    final=pfin (localizando o final do desenho)
  variando i de pcom até pfin
    altura[i] = max(altura[i],altu)
  cara = altura[1]
  imprima (1, cara)
  variando i de 2 até final
    se cara <> altura[i]
      imprima (i, altura[i])
      cara = altura[i]
  imprima (final, 0)
```

## Exemplos resolvidos Caso 1

```
 1 18 11
 4 25  7
 7  6 15
17 26 25
19 25 25
21 12 26
23 28 26
29 19 30
37 16 42
40 22 47
```

Deu como resultado

```
1 18 4 25 7 18 11 6 15 0 17 26 23 28
26 0 29 19 30 0 37 16 40 22 47 0
```

**Caso 2**

```
 1  3  9
12 17 21
24  4 28
24  1 29
31  9 38
39 10 45
41  7 45
42  3 46
44 23 53
45 18 51
45  7 52
47 18 55
```

Deu como resultado

```
1 3 9 0 12 17 21 0 24 4 28 1 29 0 31 9
38 0 39 10 44 23 53 18 55 0
```

## ☞ Para você fazer

Obviamente, antes de executar este caso, você está convidado (intimado !) a submeter sua solução ao online.judge da UVA. Se você obtiver o ACCEPTED no site, pode exigir um 10 nesta folha independente do que o professor (e seu corretor) acharem. Combinado ?

Para deixar as coisas mais ágeis, considere que todos os números deste caso são menores do que 100.

Eis os seus dados aqui:

```
 1 18  2
 3  3  4
 9 13 18
10 13 13
18 19 26
30 26 33
32 25 40
33 10 37
41 21 42
47 16 53
```

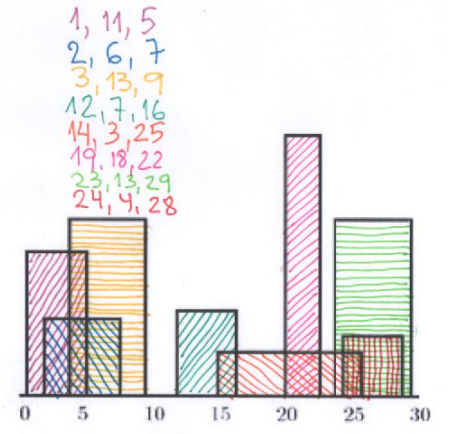
Responda aqui:

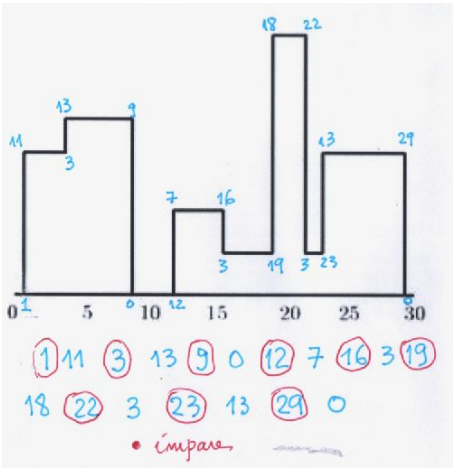
507-75989 -

CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 21/10/2024 - 12:07:31.8
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
VIVXhb1a V: 1.02
75891 LUIZ FELIPE GUEDES BUCCHERI
24PRG507 - 13 _____ / _____ / _____

## UVA105-Perfil da skyline

Tal como esta lá no site (`http:\\onlinejudge.org`): Com o advento das estações de trabalho gráficas de alta velocidade, CAD (design auxiliado por computador) e outras áreas (design CAM, VLSI) têm feito uso cada vez mais eficaz dos computadores. Um dos problemas com desenhar imagens é a eliminação de linhas ocultas (linhas obscurecidas por outras partes de um desenho). Você deverá projetar um programa para ajudar um arquiteto a desenhar o horizonte de uma cidade, considerando os locais dos edifícios da cidade. Para tornar o problema tratável, todos os edifícios são de forma retangular e eles compartilham um fundo comum (a cidade onde estão construídos é muito plana). A cidade também é vista como bidimensional. Um edifício é especificado por uma tripla ordenada $(L_i, O_i, R_i)$ onde $L_i$ e $R_i$ são as coordenadas à esquerda e à direita, respectivamente, do edifício $_i$, $(0 < L_i < R_i)$ e $H_i$ é a altura do edifício. No diagrama abaixo, os edifícios são mostrados à esquerda com triplas $(1, 11, 5), (2, 6, 7), (3, 13, 9), (12, 7, 16), (14, 3, 25),$ $(19, 18, 22), (23, 13, 29), (24, 4, 28)$ o horizonte, mostrado à direita, é representado pela sequência: $(1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)$ Estes dados correspondem à seguinte imagem:




**Entrada** A entrada é uma sequência de construção de triplas. Todas as coordenadas dos edifícios são números inteiros menores que 10.000 e haverá pelo menos um e no máximo 5.000 edifícios no arquivo de entrada. Cada tripla (edifício) é uma linha sozinha no arquivo de entrada. Todos os inteiros em uma tripla são separados por um ou mais espaços. As triplas serão classificados por $L_i$, a coordenada $x$ esquerda do edifício, então o edifício com o menor valor à esquerda (a coordenada $x$) é o primeiro no arquivo de entrada.

**Saída** A saída deve consistir no vetor que descreve o horizonte conforme mostrado no exemplo acima. No vetor skyline $(v_1, v_2, v_3, ..., v_{n-2}, v_{n-1}, v_n)$, o $v_i$ é tal que se $i$ é um número par representa um linha horizontal (altura). Se $i$ é um número ímpar representa uma linha vertical (coordenada $x$). O vetor skyline deve representar o 'caminho' percorrido, por exemplo, por uma formiga começando no menor coordenada $x$ e viajando horizontal e verticalmente sobre todas as linhas que definem o horizonte. Por isso a última entrada em todos os vetores do horizonte será '0'.

**Exemplo** O desenho acima está assim representado:

```
Entrada
1 11 5
2 6 7
3 13 9
12 7 16
14 3 25
19 18 22
23 13 29
24 4 28
Saída
1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0
```

**Algoritmo 1** Como quase todo problema não trivial, este admite diversas estratégias de solução. Uma possível é desdobrar a cada elemento da lista de entrada em dois eventos: O primeiro determina o início de um prédio e o segundo o seu final. Os eventos terão a estrutura: (esquerda, altura, 0=início) e (direita, altura, 1=final). Para o conjunto de dados do exemplo, ficará: [(1, 11, 1), (5, 11, 0), (2, 6, 1), (7, 6, 0), (3, 13, 1), (9, 13, 0), (12, 7, 1), (16, 7, 0), (14, 3, 1), (25, 3, 0), (19, 18, 1), (22, 18, 0), (23, 13, 1), (29, 13, 0), (24, 4, 1), (28, 4, 0)] A seguir ordena-se esta lista pela primeira posição (o $x$) cria-se uma lista chamada vista vazia ([]) e outra chamada segmentos vazia ([]) e corrente valendo 0 e depois processa-se a lista:

```
se infim==0 (início)
  insere a altura na lista segmentos
  ordena segmentos em ordem decrescente
senão
  remove esta altura de segmentos
nova-altura = primeiro elemento de segmentos
          (ou 0 se vazia)
se nova-altura <> corrente
  insere altura na vista
  corrente = nova-altura
retornar vista
```

Este algoritmo implementado reproduziu os dados do exemplo corretamente, mas foi recusado pelo online judge. Certamente algum conjunto de dados (oculto) ressalvou algum bug ou imprecisão.

**Algoritmo 2** Uma saída foi buscar outro algoritmo mais simples (embora menos eficiente) e baseado numa informação perdida do enunciado (os valores são menores do que 10000). Agora define-se um vetor de 10001 elementos que representa as ordenadas $x$ de todos os envolvidos no problema. Agora a estratégia ficou mais fácil: basta examinar o que acontece em cada ponto do vetor (já que todas as medidas são inteiras).

```
final=0
altura = [0] * 10001
processando (pcom, altu, pfin)
  se pfin>final
    final=pfin (localizando o final do desenho)
  variando i de pcom até pfin
    altura[i] = max(altura[i],altu)
  cara = altura[1]
  imprima (1, cara)
  variando i de 2 até final
    se cara <> altura[i]
      imprima (i, altura[i])
      cara = altura[i]
  imprima (final, 0)
```

## Exemplos resolvidos Caso 1

```
 1 18 11
 4 25  7
 7  6 15
17 26 25
19 25 25
21 12 26
23 28 26
29 19 30
37 16 42
40 22 47
```

Deu como resultado

```
1 18 4 25 7 18 11 6 15 0 17 26 23 28
26 0 29 19 30 0 37 16 40 22 47 0
```

**Caso 2**

```
 1  3  9
12 17 21
24  4 28
24  1 29
31  9 38
39 10 45
41  7 45
42  3 46
44 23 53
45 18 51
45  7 52
47 18 55
```

Deu como resultado

```
1 3 9 0 12 17 21 0 24 4 28 1 29 0 31 9
38 0 39 10 44 23 53 18 55 0
```

## ☞ Para você fazer

Obviamente, antes de executar este caso, você está convidado (intimado !) a submeter sua solução ao online.judge da UVA. Se você obtiver o ACCEPTED no site, pode exigir um 10 nesta folha independente do que o professor (e seu corretor) acharem. Combinado ?

Para deixar as coisas mais ágeis, considere que todos os números deste caso são menores do que 100.

Eis os seus dados aqui:

```
 1  6  5
 3 20  9
11 29 17
13 13 19
16  8 23
25 23 30
35  4 42
39 10 47
40 11 43
44 22 50
```

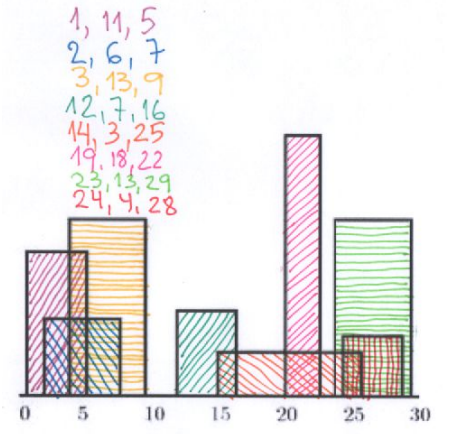
Responda aqui:

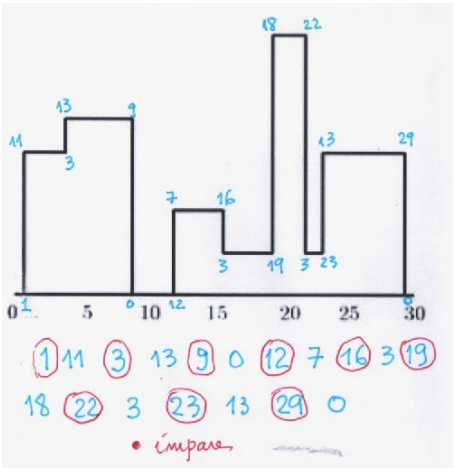
507-75891 -

CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 21/10/2024 - 12:07:31.8
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
VIVXhb1a V: 1.02
75903 MANUELLA LEAL DE MEIRELLES F
24PRG507 - 14 _____ / _____ / _____

## UVA105-Perfil da skyline

Tal como esta lá no site (`http:\\onlinejudge.org`): Com o advento das estações de trabalho gráficas de alta velocidade, CAD (design auxiliado por computador) e outras áreas (design CAM, VLSI) têm feito uso cada vez mais eficaz dos computadores. Um dos problemas com desenhar imagens é a eliminação de linhas ocultas (linhas obscurecidas por outras partes de um desenho). Você deverá projetar um programa para ajudar um arquiteto a desenhar o horizonte de uma cidade, considerando os locais dos edifícios da cidade. Para tornar o problema tratável, todos os edifícios são de forma retangular e eles compartilham um fundo comum (a cidade onde estão construídos é muito plana). A cidade também é vista como bidimensional. Um edifício é especificado por uma tripla ordenada $(L_i, O_i, R_i)$ onde $L_i$ e $R_i$ são as coordenadas à esquerda e à direita, respectivamente, do edifício $_i$, $(0 < L_i < R_i)$ e $H_i$ é a altura do edifício. No diagrama abaixo, os edifícios são mostrados à esquerda com triplas $(1, 11, 5), (2, 6, 7), (3, 13, 9), (12, 7, 16), (14, 3, 25),$ $(19, 18, 22), (23, 13, 29), (24, 4, 28)$ o horizonte, mostrado à direita, é representado pela sequência: $(1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)$ Estes dados correspondem à seguinte imagem:




**Entrada** A entrada é uma sequência de construção de triplas. Todas as coordenadas dos edifícios são números inteiros menores que 10.000 e haverá pelo menos um e no máximo 5.000 edifícios no arquivo de entrada. Cada tripla (edifício) é uma linha sozinha no arquivo de entrada. Todos os inteiros em uma tripla são separados por um ou mais espaços. As triplas serão classificados por $L_i$, a coordenada $x$ esquerda do edifício, então o edifício com o menor valor à esquerda (a coordenada $x$) é o primeiro no arquivo de entrada.

**Saída** A saída deve consistir no vetor que descreve o horizonte conforme mostrado no exemplo acima. No vetor skyline $(v_1, v_2, v_3, ..., v_{n-2}, v_{n-1}, v_n)$, o $v_i$ é tal que se $i$ é um número par representa um linha horizontal (altura). Se $i$ é um número ímpar representa uma linha vertical (coordenada $x$). O vetor skyline deve representar o 'caminho' percorrido, por exemplo, por uma formiga começando no menor coordenada $x$ e viajando horizontal e verticalmente sobre todas as linhas que definem o horizonte. Por isso a última entrada em todos os vetores do horizonte será '0'.

**Exemplo** O desenho acima está assim representado:

```
Entrada
1 11 5
2 6 7
3 13 9
12 7 16
14 3 25
19 18 22
23 13 29
24 4 28
Saída
1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0
```

**Algoritmo 1** Como quase todo problema não trivial, este admite diversas estratégias de solução. Uma possível é desdobrar a cada elemento da lista de entrada em dois eventos: O primeiro determina o início de um prédio e o segundo o seu final. Os eventos terão a estrutura: (esquerda, altura, 0=início) e (direita, altura, 1=final). Para o conjunto de dados do exemplo, ficará: [(1, 11, 1), (5, 11, 0), (2, 6, 1), (7, 6, 0), (3, 13, 1), (9, 13, 0), (12, 7, 1), (16, 7, 0), (14, 3, 1), (25, 3, 0), (19, 18, 1), (22, 18, 0), (23, 13, 1), (29, 13, 0), (24, 4, 1), (28, 4, 0)] A seguir ordena-se esta lista pela primeira posição (o $x$) cria-se uma lista chamada vista vazia ([]) e outra chamada segmentos vazia ([]) e corrente valendo 0 e depois processa-se a lista:

```
se infim==0 (início)
  insere a altura na lista segmentos
  ordena segmentos em ordem decrescente
senão
  remove esta altura de segmentos
nova-altura = primeiro elemento de segmentos
          (ou 0 se vazia)
se nova-altura <> corrente
  insere altura na vista
  corrente = nova-altura
retornar vista
```

Este algoritmo implementado reproduziu os dados do exemplo corretamente, mas foi recusado pelo online judge. Certamente algum conjunto de dados (oculto) ressalvou algum bug ou imprecisão.

**Algoritmo 2** Uma saída foi buscar outro algoritmo mais simples (embora menos eficiente) e baseado numa informação perdida do enunciado (os valores são menores do que 10000). Agora define-se um vetor de 10001 elementos que representa as ordenadas $x$ de todos os envolvidos no problema. Agora a estratégia ficou mais fácil: basta examinar o que acontece em cada ponto do vetor (já que todas as medidas são inteiras).

```
final=0
altura = [0] * 10001
processando (pcom, altu, pfin)
  se pfin>final
    final=pfin (localizando o final do desenho)
  variando i de pcom até pfin
    altura[i] = max(altura[i],altu)
  cara = altura[1]
  imprima (1, cara)
  variando i de 2 até final
    se cara <> altura[i]
      imprima (i, altura[i])
      cara = altura[i]
  imprima (final, 0)
```

## Exemplos resolvidos Caso 1

```
 1 18 11
 4 25  7
 7  6 15
17 26 25
19 25 25
21 12 26
23 28 26
29 19 30
37 16 42
40 22 47
```

Deu como resultado

```
1 18 4 25 7 18 11 6 15 0 17 26 23 28
26 0 29 19 30 0 37 16 40 22 47 0
```

**Caso 2**

```
 1  3  9
12 17 21
24  4 28
24  1 29
31  9 38
39 10 45
41  7 45
42  3 46
44 23 53
45 18 51
45  7 52
47 18 55
```

Deu como resultado

```
1 3 9 0 12 17 21 0 24 4 28 1 29 0 31 9
38 0 39 10 44 23 53 18 55 0
```

## ☞ Para você fazer

Obviamente, antes de executar este caso, você está convidado (intimado !) a submeter sua solução ao online.judge da UVA. Se você obtiver o ACCEPTED no site, pode exigir um 10 nesta folha independente do que o professor (e seu corretor) acharem. Combinado ?

Para deixar as coisas mais ágeis, considere que todos os números deste caso são menores do que 100.

Eis os seus dados aqui:

```
 1 22  9
18 30 20
18 15 21
22  1 24
28 11 38
31 15 32
39  5 48
41  3 44
46 13 55
50 14 55
```

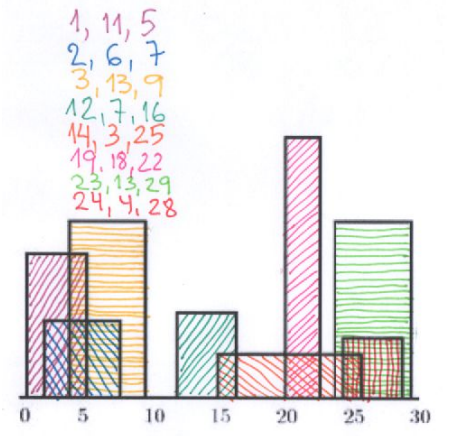
Responda aqui:

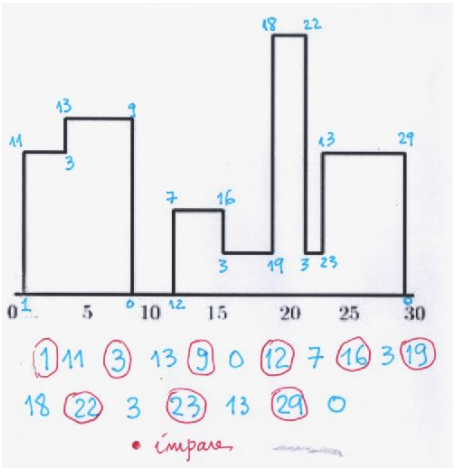
507-75903 -

CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 21/10/2024 - 12:07:31.8
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
VIVXhb1a V: 1.02
75910 MARIA EDUARDA DA SILVA
24PRG507 - 15 _____ / _____ / _____

## UVA105-Perfil da skyline

Tal como esta lá no site (`http:\\onlinejudge.org`): Com o advento das estações de trabalho gráficas de alta velocidade, CAD (design auxiliado por computador) e outras áreas (design CAM, VLSI) têm feito uso cada vez mais eficaz dos computadores. Um dos problemas com desenhar imagens é a eliminação de linhas ocultas (linhas obscurecidas por outras partes de um desenho). Você deverá projetar um programa para ajudar um arquiteto a desenhar o horizonte de uma cidade, considerando os locais dos edifícios da cidade. Para tornar o problema tratável, todos os edifícios são de forma retangular e eles compartilham um fundo comum (a cidade onde estão construídos é muito plana). A cidade também é vista como bidimensional. Um edifício é especificado por uma tripla ordenada $(L_i, O_i, R_i)$ onde $L_i$ e $R_i$ são as coordenadas à esquerda e à direita, respectivamente, do edifício $_i$, $(0 < L_i < R_i)$ e $H_i$ é a altura do edifício. No diagrama abaixo, os edifícios são mostrados à esquerda com triplas $(1, 11, 5), (2, 6, 7), (3, 13, 9), (12, 7, 16), (14, 3, 25),$ $(19, 18, 22), (23, 13, 29), (24, 4, 28)$ o horizonte, mostrado à direita, é representado pela sequência: $(1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)$ Estes dados correspondem à seguinte imagem:




**Entrada** A entrada é uma sequência de construção de triplas. Todas as coordenadas dos edifícios são números inteiros menores que 10.000 e haverá pelo menos um e no máximo 5.000 edifícios no arquivo de entrada. Cada tripla (edifício) é uma linha sozinha no arquivo de entrada. Todos os inteiros em uma tripla são separados por um ou mais espaços. As triplas serão classificados por $L_i$, a coordenada $x$ esquerda do edifício, então o edifício com o menor valor à esquerda (a coordenada $x$) é o primeiro no arquivo de entrada.

**Saída** A saída deve consistir no vetor que descreve o horizonte conforme mostrado no exemplo acima. No vetor skyline $(v_1, v_2, v_3, ..., v_{n-2}, v_{n-1}, v_n)$, o $v_i$ é tal que se $i$ é um número par representa um linha horizontal (altura). Se $i$ é um número ímpar representa uma linha vertical (coordenada $x$). O vetor skyline deve representar o 'caminho' percorrido, por exemplo, por uma formiga começando no menor coordenada $x$ e viajando horizontal e verticalmente sobre todas as linhas que definem o horizonte. Por isso a última entrada em todos os vetores do horizonte será '0'.

**Exemplo** O desenho acima está assim representado:

```
Entrada
1 11 5
2 6 7
3 13 9
12 7 16
14 3 25
19 18 22
23 13 29
24 4 28
Saída
1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0
```

**Algoritmo 1** Como quase todo problema não trivial, este admite diversas estratégias de solução. Uma possível é desdobrar a cada elemento da lista de entrada em dois eventos: O primeiro determina o início de um prédio e o segundo o seu final. Os eventos terão a estrutura: (esquerda, altura, 0=início) e (direita, altura, 1=final). Para o conjunto de dados do exemplo, ficará: [(1, 11, 1), (5, 11, 0), (2, 6, 1), (7, 6, 0), (3, 13, 1), (9, 13, 0), (12, 7, 1), (16, 7, 0), (14, 3, 1), (25, 3, 0), (19, 18, 1), (22, 18, 0), (23, 13, 1), (29, 13, 0), (24, 4, 1), (28, 4, 0)] A seguir ordena-se esta lista pela primeira posição (o $x$) cria-se uma lista chamada vista vazia ([]) e outra chamada segmentos vazia ([]) e corrente valendo 0 e depois processa-se a lista:

```
se infim==0 (início)
  insere a altura na lista segmentos
  ordena segmentos em ordem decrescente
senão
  remove esta altura de segmentos
nova-altura = primeiro elemento de segmentos
          (ou 0 se vazia)
se nova-altura <> corrente
  insere altura na vista
  corrente = nova-altura
retornar vista
```

Este algoritmo implementado reproduziu os dados do exemplo corretamente, mas foi recusado pelo online judge. Certamente algum conjunto de dados (oculto) ressalvou algum bug ou imprecisão.

**Algoritmo 2** Uma saída foi buscar outro algoritmo mais simples (embora menos eficiente) e baseado numa informação perdida do enunciado (os valores são menores do que 10000). Agora define-se um vetor de 10001 elementos que representa as ordenadas $x$ de todos os envolvidos no problema. Agora a estratégia ficou mais fácil: basta examinar o que acontece em cada ponto do vetor (já que todas as medidas são inteiras).

```
final=0
altura = [0] * 10001
processando (pcom, altu, pfin)
  se pfin>final
    final=pfin (localizando o final do desenho)
  variando i de pcom até pfin
    altura[i] = max(altura[i],altu)
  cara = altura[1]
  imprima (1, cara)
  variando i de 2 até final
    se cara <> altura[i]
      imprima (i, altura[i])
      cara = altura[i]
  imprima (final, 0)
```

## Exemplos resolvidos Caso 1

```
 1 18 11
 4 25  7
 7  6 15
17 26 25
19 25 25
21 12 26
23 28 26
29 19 30
37 16 42
40 22 47
```

Deu como resultado

```
1 18 4 25 7 18 11 6 15 0 17 26 23 28
26 0 29 19 30 0 37 16 40 22 47 0
```

**Caso 2**

```
 1  3  9
12 17 21
24  4 28
24  1 29
31  9 38
39 10 45
41  7 45
42  3 46
44 23 53
45 18 51
45  7 52
47 18 55
```

Deu como resultado

```
1 3 9 0 12 17 21 0 24 4 28 1 29 0 31 9
38 0 39 10 44 23 53 18 55 0
```

## ☞ Para você fazer

Obviamente, antes de executar este caso, você está convidado (intimado !) a submeter sua solução ao online.judge da UVA. Se você obtiver o ACCEPTED no site, pode exigir um 10 nesta folha independente do que o professor (e seu corretor) acharem. Combinado ?

Para deixar as coisas mais ágeis, considere que todos os números deste caso são menores do que 100.

Eis os seus dados aqui:

```
 1 25  7
 9 23 10
13  3 18
17 18 26
24 15 26
27 19 33
31 11 39
33 17 38
41  8 51
45 15 53
```

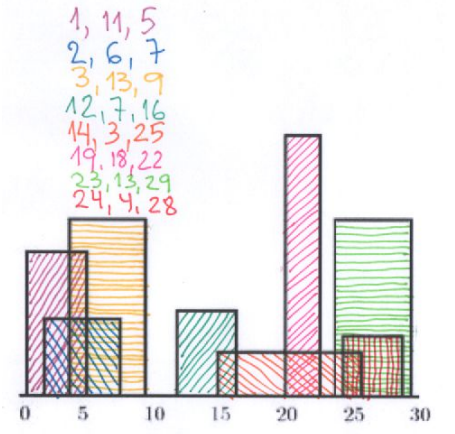
Responda aqui:

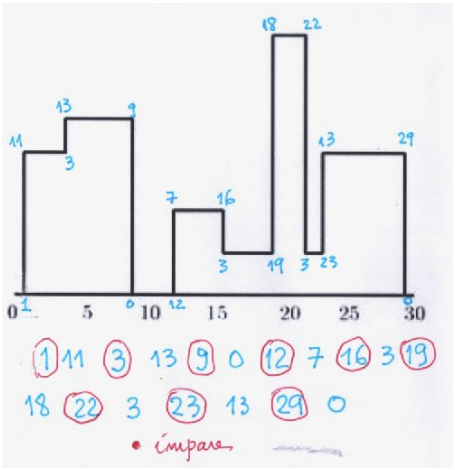
507-75910 -

CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 21/10/2024 - 12:07:31.8
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
VIVXhb1a V: 1.02
75927 MATHEUS AITA FABRICIO DE CAR
24PRG507 - 16 _____ / _____ / _____

## UVA105-Perfil da skyline

Tal como esta lá no site (`http:\\onlinejudge.org`): Com o advento das estações de trabalho gráficas de alta velocidade, CAD (design auxiliado por computador) e outras áreas (design CAM, VLSI) têm feito uso cada vez mais eficaz dos computadores. Um dos problemas com desenhar imagens é a eliminação de linhas ocultas (linhas obscurecidas por outras partes de um desenho). Você deverá projetar um programa para ajudar um arquiteto a desenhar o horizonte de uma cidade, considerando os locais dos edifícios da cidade. Para tornar o problema tratável, todos os edifícios são de forma retangular e eles compartilham um fundo comum (a cidade onde estão construídos é muito plana). A cidade também é vista como bidimensional. Um edifício é especificado por uma tripla ordenada $(L_i, O_i, R_i)$ onde $L_i$ e $R_i$ são as coordenadas à esquerda e à direita, respectivamente, do edifício $_i$, $(0 < L_i < R_i)$ e $H_i$ é a altura do edifício. No diagrama abaixo, os edifícios são mostrados à esquerda com triplas $(1, 11, 5), (2, 6, 7), (3, 13, 9), (12, 7, 16), (14, 3, 25),$ $(19, 18, 22), (23, 13, 29), (24, 4, 28)$ o horizonte, mostrado à direita, é representado pela sequência: $(1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)$ Estes dados correspondem à seguinte imagem:




**Entrada** A entrada é uma sequência de construção de triplas. Todas as coordenadas dos edifícios são números inteiros menores que 10.000 e haverá pelo menos um e no máximo 5.000 edifícios no arquivo de entrada. Cada tripla (edifício) é uma linha sozinha no arquivo de entrada. Todos os inteiros em uma tripla são separados por um ou mais espaços. As triplas serão classificados por $L_i$, a coordenada $x$ esquerda do edifício, então o edifício com o menor valor à esquerda (a coordenada $x$) é o primeiro no arquivo de entrada.

**Saída** A saída deve consistir no vetor que descreve o horizonte conforme mostrado no exemplo acima. No vetor skyline $(v_1, v_2, v_3, ..., v_{n-2}, v_{n-1}, v_n)$, o $v_i$ é tal que se $i$ é um número par representa um linha horizontal (altura). Se $i$ é um número ímpar representa uma linha vertical (coordenada $x$). O vetor skyline deve representar o 'caminho' percorrido, por exemplo, por uma formiga começando no menor coordenada $x$ e viajando horizontal e verticalmente sobre todas as linhas que definem o horizonte. Por isso a última entrada em todos os vetores do horizonte será '0'.

**Exemplo** O desenho acima está assim representado:

```
Entrada
1 11 5
2 6 7
3 13 9
12 7 16
14 3 25
19 18 22
23 13 29
24 4 28
Saída
1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0
```

**Algoritmo 1** Como quase todo problema não trivial, este admite diversas estratégias de solução. Uma possível é desdobrar a cada elemento da lista de entrada em dois eventos: O primeiro determina o início de um prédio e o segundo o seu final. Os eventos terão a estrutura: (esquerda, altura, 0=início) e (direita, altura, 1=final). Para o conjunto de dados do exemplo, ficará: [(1, 11, 1), (5, 11, 0), (2, 6, 1), (7, 6, 0), (3, 13, 1), (9, 13, 0), (12, 7, 1), (16, 7, 0), (14, 3, 1), (25, 3, 0), (19, 18, 1), (22, 18, 0), (23, 13, 1), (29, 13, 0), (24, 4, 1), (28, 4, 0)] A seguir ordena-se esta lista pela primeira posição (o $x$) cria-se uma lista chamada vista vazia ([]) e outra chamada segmentos vazia ([]) e corrente valendo 0 e depois processa-se a lista:

```
se infim==0 (início)
  insere a altura na lista segmentos
  ordena segmentos em ordem decrescente
senão
  remove esta altura de segmentos
nova-altura = primeiro elemento de segmentos
          (ou 0 se vazia)
se nova-altura <> corrente
  insere altura na vista
  corrente = nova-altura
retornar vista
```

Este algoritmo implementado reproduziu os dados do exemplo corretamente, mas foi recusado pelo online judge. Certamente algum conjunto de dados (oculto) ressalvou algum bug ou imprecisão.

**Algoritmo 2** Uma saída foi buscar outro algoritmo mais simples (embora menos eficiente) e baseado numa informação perdida do enunciado (os valores são menores do que 10000). Agora define-se um vetor de 10001 elementos que representa as ordenadas $x$ de todos os envolvidos no problema. Agora a estratégia ficou mais fácil: basta examinar o que acontece em cada ponto do vetor (já que todas as medidas são inteiras).

```
final=0
altura = [0] * 10001
processando (pcom, altu, pfin)
  se pfin>final
    final=pfin (localizando o final do desenho)
  variando i de pcom até pfin
    altura[i] = max(altura[i],altu)
  cara = altura[1]
  imprima (1, cara)
  variando i de 2 até final
    se cara <> altura[i]
      imprima (i, altura[i])
      cara = altura[i]
  imprima (final, 0)
```

## Exemplos resolvidos Caso 1

```
 1 18 11
 4 25  7
 7  6 15
17 26 25
19 25 25
21 12 26
23 28 26
29 19 30
37 16 42
40 22 47
```

Deu como resultado

```
1 18 4 25 7 18 11 6 15 0 17 26 23 28
26 0 29 19 30 0 37 16 40 22 47 0
```

**Caso 2**

```
 1  3  9
12 17 21
24  4 28
24  1 29
31  9 38
39 10 45
41  7 45
42  3 46
44 23 53
45 18 51
45  7 52
47 18 55
```

Deu como resultado

```
1 3 9 0 12 17 21 0 24 4 28 1 29 0 31 9
38 0 39 10 44 23 53 18 55 0
```

## ☞ Para você fazer

Obviamente, antes de executar este caso, você está convidado (intimado !) a submeter sua solução ao online.judge da UVA. Se você obtiver o ACCEPTED no site, pode exigir um 10 nesta folha independente do que o professor (e seu corretor) acharem. Combinado ?

Para deixar as coisas mais ágeis, considere que todos os números deste caso são menores do que 100.

Eis os seus dados aqui:

```
 1 20  8
 8  3 16
 9 15 18
13 26 23
16 20 26
16 21 23
24 28 28
35 22 39
38 13 42
47 10 57
```

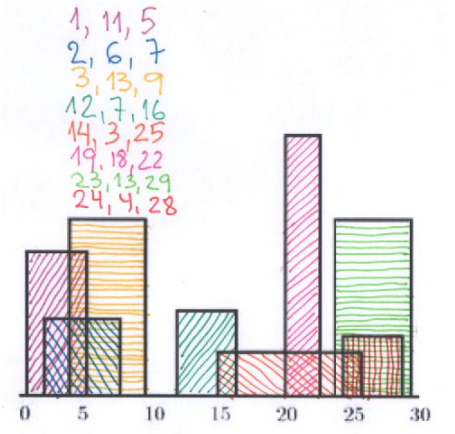
Responda aqui:

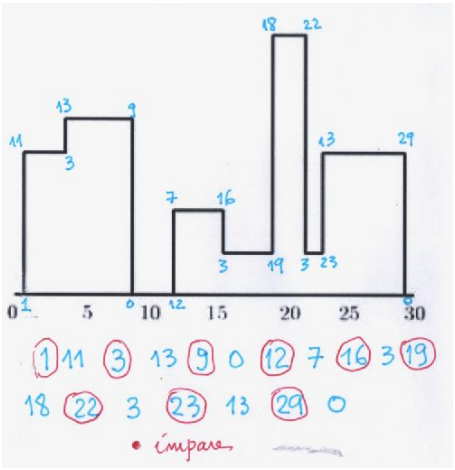
507-75927 -

CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 21/10/2024 - 12:07:31.8
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
VIVXhb1a V: 1.02
75934 MATHEUS LEANDRO DE BITENCOUR
24PRG507 - 17 _____ / _____ / _____

## UVA105-Perfil da skyline

Tal como esta lá no site (`http:\\onlinejudge.org`): Com o advento das estações de trabalho gráficas de alta velocidade, CAD (design auxiliado por computador) e outras áreas (design CAM, VLSI) têm feito uso cada vez mais eficaz dos computadores. Um dos problemas com desenhar imagens é a eliminação de linhas ocultas (linhas obscurecidas por outras partes de um desenho). Você deverá projetar um programa para ajudar um arquiteto a desenhar o horizonte de uma cidade, considerando os locais dos edifícios da cidade. Para tornar o problema tratável, todos os edifícios são de forma retangular e eles compartilham um fundo comum (a cidade onde estão construídos é muito plana). A cidade também é vista como bidimensional. Um edifício é especificado por uma tripla ordenada $(L_i, O_i, R_i)$ onde $L_i$ e $R_i$ são as coordenadas à esquerda e à direita, respectivamente, do edifício $_i$, $(0 < L_i < R_i)$ e $H_i$ é a altura do edifício. No diagrama abaixo, os edifícios são mostrados à esquerda com triplas $(1, 11, 5), (2, 6, 7), (3, 13, 9), (12, 7, 16), (14, 3, 25),$ $(19, 18, 22), (23, 13, 29), (24, 4, 28)$ o horizonte, mostrado à direita, é representado pela sequência: $(1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)$ Estes dados correspondem à seguinte imagem:




**Entrada** A entrada é uma sequência de construção de triplas. Todas as coordenadas dos edifícios são números inteiros menores que 10.000 e haverá pelo menos um e no máximo 5.000 edifícios no arquivo de entrada. Cada tripla (edifício) é uma linha sozinha no arquivo de entrada. Todos os inteiros em uma tripla são separados por um ou mais espaços. As triplas serão classificados por $L_i$, a coordenada $x$ esquerda do edifício, então o edifício com o menor valor à esquerda (a coordenada $x$) é o primeiro no arquivo de entrada.

**Saída** A saída deve consistir no vetor que descreve o horizonte conforme mostrado no exemplo acima. No vetor skyline $(v_1, v_2, v_3, ..., v_{n-2}, v_{n-1}, v_n)$, o $v_i$ é tal que se $i$ é um número par representa um linha horizontal (altura). Se $i$ é um número ímpar representa uma linha vertical (coordenada $x$). O vetor skyline deve representar o 'caminho' percorrido, por exemplo, por uma formiga começando no menor coordenada $x$ e viajando horizontal e verticalmente sobre todas as linhas que definem o horizonte. Por isso a última entrada em todos os vetores do horizonte será '0'.

**Exemplo** O desenho acima está assim representado:

```
Entrada
1 11 5
2 6 7
3 13 9
12 7 16
14 3 25
19 18 22
23 13 29
24 4 28
Saída
1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0
```

**Algoritmo 1** Como quase todo problema não trivial, este admite diversas estratégias de solução. Uma possível é desdobrar a cada elemento da lista de entrada em dois eventos: O primeiro determina o início de um prédio e o segundo o seu final. Os eventos terão a estrutura: (esquerda, altura, 0=início) e (direita, altura, 1=final). Para o conjunto de dados do exemplo, ficará: [(1, 11, 1), (5, 11, 0), (2, 6, 1), (7, 6, 0), (3, 13, 1), (9, 13, 0), (12, 7, 1), (16, 7, 0), (14, 3, 1), (25, 3, 0), (19, 18, 1), (22, 18, 0), (23, 13, 1), (29, 13, 0), (24, 4, 1), (28, 4, 0)] A seguir ordena-se esta lista pela primeira posição (o $x$) cria-se uma lista chamada vista vazia ([]) e outra chamada segmentos vazia ([]) e corrente valendo 0 e depois processa-se a lista:

```
se infim==0 (início)
  insere a altura na lista segmentos
  ordena segmentos em ordem decrescente
senão
  remove esta altura de segmentos
nova-altura = primeiro elemento de segmentos
          (ou 0 se vazia)
se nova-altura <> corrente
  insere altura na vista
  corrente = nova-altura
retornar vista
```

Este algoritmo implementado reproduziu os dados do exemplo corretamente, mas foi recusado pelo online judge. Certamente algum conjunto de dados (oculto) ressalvou algum bug ou imprecisão.

**Algoritmo 2** Uma saída foi buscar outro algoritmo mais simples (embora menos eficiente) e baseado numa informação perdida do enunciado (os valores são menores do que 10000). Agora define-se um vetor de 10001 elementos que representa as ordenadas $x$ de todos os envolvidos no problema. Agora a estratégia ficou mais fácil: basta examinar o que acontece em cada ponto do vetor (já que todas as medidas são inteiras).

```
final=0
altura = [0] * 10001
processando (pcom, altu, pfin)
  se pfin>final
    final=pfin (localizando o final do desenho)
  variando i de pcom até pfin
    altura[i] = max(altura[i],altu)
  cara = altura[1]
  imprima (1, cara)
  variando i de 2 até final
    se cara <> altura[i]
      imprima (i, altura[i])
      cara = altura[i]
  imprima (final, 0)
```

## Exemplos resolvidos Caso 1

```
 1 18 11
 4 25  7
 7  6 15
17 26 25
19 25 25
21 12 26
23 28 26
29 19 30
37 16 42
40 22 47
```

Deu como resultado

```
1 18 4 25 7 18 11 6 15 0 17 26 23 28
26 0 29 19 30 0 37 16 40 22 47 0
```

**Caso 2**

```
 1  3  9
12 17 21
24  4 28
24  1 29
31  9 38
39 10 45
41  7 45
42  3 46
44 23 53
45 18 51
45  7 52
47 18 55
```

Deu como resultado

```
1 3 9 0 12 17 21 0 24 4 28 1 29 0 31 9
38 0 39 10 44 23 53 18 55 0
```

## ☞ Para você fazer

Obviamente, antes de executar este caso, você está convidado (intimado !) a submeter sua solução ao online.judge da UVA. Se você obtiver o ACCEPTED no site, pode exigir um 10 nesta folha independente do que o professor (e seu corretor) acharem. Combinado ?

Para deixar as coisas mais ágeis, considere que todos os números deste caso são menores do que 100.

Eis os seus dados aqui:

```
 1  3  3
 6 27  7
 8 13 14
 8  4 12
14 28 15
15 27 21
30 30 31
35 27 39
44 27 50
44  2 54
```

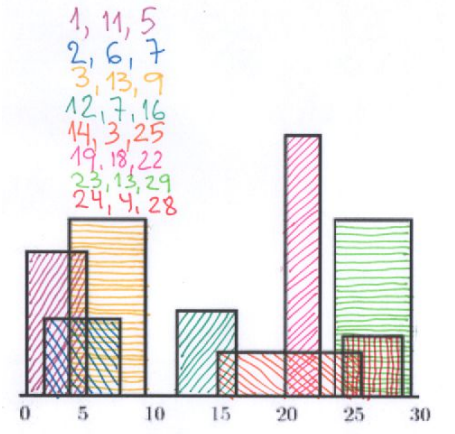
Responda aqui:

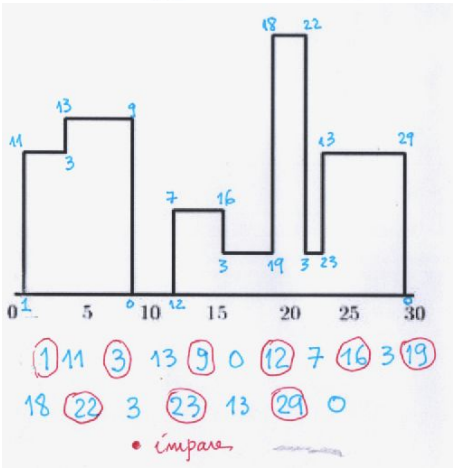
507-75934 -

CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 21/10/2024 - 12:07:31.8
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
VIVXhb1a V: 1.02
75941 MAYCON TAVARES WOLPE
24PRG507 - 18 _____ / _____ / _____

## UVA105-Perfil da skyline

Tal como esta lá no site (`http:\\onlinejudge.org`): Com o advento das estações de trabalho gráficas de alta velocidade, CAD (design auxiliado por computador) e outras áreas (design CAM, VLSI) têm feito uso cada vez mais eficaz dos computadores. Um dos problemas com desenhar imagens é a eliminação de linhas ocultas (linhas obscurecidas por outras partes de um desenho). Você deverá projetar um programa para ajudar um arquiteto a desenhar o horizonte de uma cidade, considerando os locais dos edifícios da cidade. Para tornar o problema tratável, todos os edifícios são de forma retangular e eles compartilham um fundo comum (a cidade onde estão construídos é muito plana). A cidade também é vista como bidimensional. Um edifício é especificado por uma tripla ordenada $(L_i, O_i, R_i)$ onde $L_i$ e $R_i$ são as coordenadas à esquerda e à direita, respectivamente, do edifício $_i$, $(0 < L_i < R_i)$ e $H_i$ é a altura do edifício. No diagrama abaixo, os edifícios são mostrados à esquerda com triplas $(1, 11, 5), (2, 6, 7), (3, 13, 9), (12, 7, 16), (14, 3, 25),$ $(19, 18, 22), (23, 13, 29), (24, 4, 28)$ o horizonte, mostrado à direita, é representado pela sequência: $(1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)$ Estes dados correspondem à seguinte imagem:




**Entrada** A entrada é uma sequência de construção de triplas. Todas as coordenadas dos edifícios são números inteiros menores que 10.000 e haverá pelo menos um e no máximo 5.000 edifícios no arquivo de entrada. Cada tripla (edifício) é uma linha sozinha no arquivo de entrada. Todos os inteiros em uma tripla são separados por um ou mais espaços. As triplas serão classificados por $L_i$, a coordenada $x$ esquerda do edifício, então o edifício com o menor valor à esquerda (a coordenada $x$) é o primeiro no arquivo de entrada.

**Saída** A saída deve consistir no vetor que descreve o horizonte conforme mostrado no exemplo acima. No vetor skyline $(v_1, v_2, v_3, ..., v_{n-2}, v_{n-1}, v_n)$, o $v_i$ é tal que se $i$ é um número par representa um linha horizontal (altura). Se $i$ é um número ímpar representa uma linha vertical (coordenada $x$). O vetor skyline deve representar o 'caminho' percorrido, por exemplo, por uma formiga começando no menor coordenada $x$ e viajando horizontal e verticalmente sobre todas as linhas que definem o horizonte. Por isso a última entrada em todos os vetores do horizonte será '0'.

**Exemplo** O desenho acima está assim representado:

```
Entrada
1 11 5
2 6 7
3 13 9
12 7 16
14 3 25
19 18 22
23 13 29
24 4 28
Saída
1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0
```

**Algoritmo 1** Como quase todo problema não trivial, este admite diversas estratégias de solução. Uma possível é desdobrar a cada elemento da lista de entrada em dois eventos: O primeiro determina o início de um prédio e o segundo o seu final. Os eventos terão a estrutura: (esquerda, altura, 0=início) e (direita, altura, 1=final). Para o conjunto de dados do exemplo, ficará: [(1, 11, 1), (5, 11, 0), (2, 6, 1), (7, 6, 0), (3, 13, 1), (9, 13, 0), (12, 7, 1), (16, 7, 0), (14, 3, 1), (25, 3, 0), (19, 18, 1), (22, 18, 0), (23, 13, 1), (29, 13, 0), (24, 4, 1), (28, 4, 0)] A seguir ordena-se esta lista pela primeira posição (o $x$) cria-se uma lista chamada vista vazia ([]) e outra chamada segmentos vazia ([]) e corrente valendo 0 e depois processa-se a lista:

```
se infim==0 (início)
  insere a altura na lista segmentos
  ordena segmentos em ordem decrescente
senão
  remove esta altura de segmentos
nova-altura = primeiro elemento de segmentos
          (ou 0 se vazia)
se nova-altura <> corrente
  insere altura na vista
  corrente = nova-altura
retornar vista
```

Este algoritmo implementado reproduziu os dados do exemplo corretamente, mas foi recusado pelo online judge. Certamente algum conjunto de dados (oculto) ressalvou algum bug ou imprecisão.

**Algoritmo 2** Uma saída foi buscar outro algoritmo mais simples (embora menos eficiente) e baseado numa informação perdida do enunciado (os valores são menores do que 10000). Agora define-se um vetor de 10001 elementos que representa as ordenadas $x$ de todos os envolvidos no problema. Agora a estratégia ficou mais fácil: basta examinar o que acontece em cada ponto do vetor (já que todas as medidas são inteiras).

```
final=0
altura = [0] * 10001
processando (pcom, altu, pfin)
  se pfin>final
    final=pfin (localizando o final do desenho)
  variando i de pcom até pfin
    altura[i] = max(altura[i],altu)
  cara = altura[1]
  imprima (1, cara)
  variando i de 2 até final
    se cara <> altura[i]
      imprima (i, altura[i])
      cara = altura[i]
  imprima (final, 0)
```

## Exemplos resolvidos Caso 1

```
 1 18 11
 4 25  7
 7  6 15
17 26 25
19 25 25
21 12 26
23 28 26
29 19 30
37 16 42
40 22 47
```

Deu como resultado

```
1 18 4 25 7 18 11 6 15 0 17 26 23 28
26 0 29 19 30 0 37 16 40 22 47 0
```

**Caso 2**

```
 1  3  9
12 17 21
24  4 28
24  1 29
31  9 38
39 10 45
41  7 45
42  3 46
44 23 53
45 18 51
45  7 52
47 18 55
```

Deu como resultado

```
1 3 9 0 12 17 21 0 24 4 28 1 29 0 31 9
38 0 39 10 44 23 53 18 55 0
```

## ☞ Para você fazer

Obviamente, antes de executar este caso, você está convidado (intimado !) a submeter sua solução ao online.judge da UVA. Se você obtiver o ACCEPTED no site, pode exigir um 10 nesta folha independente do que o professor (e seu corretor) acharem. Combinado ?

Para deixar as coisas mais ágeis, considere que todos os números deste caso são menores do que 100.

Eis os seus dados aqui:

```
 1 14  5
 4  8  9
 7 15 15
22 12 29
43 21 44
45 21 48
45 11 53
46  4 47
46 29 47
50 18 52
```

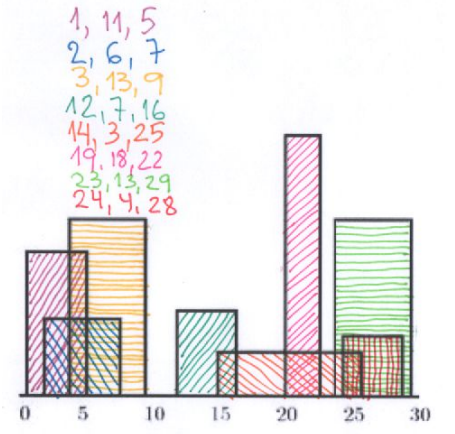
Responda aqui:

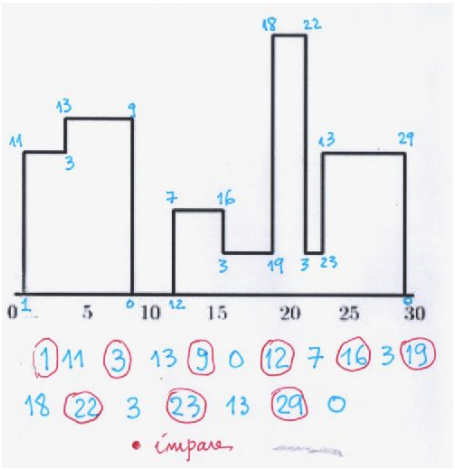
507-75941 -

CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 21/10/2024 - 12:07:31.8
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
VIVXhb1a V: 1.02
75958 PEDRO TOLEDO LEAL
24PRG507 - 19 _____ / _____ / _____

## UVA105-Perfil da skyline

Tal como esta lá no site (`http:\\onlinejudge.org`): Com o advento das estações de trabalho gráficas de alta velocidade, CAD (design auxiliado por computador) e outras áreas (design CAM, VLSI) têm feito uso cada vez mais eficaz dos computadores. Um dos problemas com desenhar imagens é a eliminação de linhas ocultas (linhas obscurecidas por outras partes de um desenho). Você deverá projetar um programa para ajudar um arquiteto a desenhar o horizonte de uma cidade, considerando os locais dos edifícios da cidade. Para tornar o problema tratável, todos os edifícios são de forma retangular e eles compartilham um fundo comum (a cidade onde estão construídos é muito plana). A cidade também é vista como bidimensional. Um edifício é especificado por uma tripla ordenada $(L_i, O_i, R_i)$ onde $L_i$ e $R_i$ são as coordenadas à esquerda e à direita, respectivamente, do edifício $_i$, $(0 < L_i < R_i)$ e $H_i$ é a altura do edifício. No diagrama abaixo, os edifícios são mostrados à esquerda com triplas $(1, 11, 5), (2, 6, 7), (3, 13, 9), (12, 7, 16), (14, 3, 25),$ $(19, 18, 22), (23, 13, 29), (24, 4, 28)$ o horizonte, mostrado à direita, é representado pela sequência: $(1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)$ Estes dados correspondem à seguinte imagem:




**Entrada** A entrada é uma sequência de construção de triplas. Todas as coordenadas dos edifícios são números inteiros menores que 10.000 e haverá pelo menos um e no máximo 5.000 edifícios no arquivo de entrada. Cada tripla (edifício) é uma linha sozinha no arquivo de entrada. Todos os inteiros em uma tripla são separados por um ou mais espaços. As triplas serão classificados por $L_i$, a coordenada $x$ esquerda do edifício, então o edifício com o menor valor à esquerda (a coordenada $x$) é o primeiro no arquivo de entrada.

**Saída** A saída deve consistir no vetor que descreve o horizonte conforme mostrado no exemplo acima. No vetor skyline $(v_1, v_2, v_3, ..., v_{n-2}, v_{n-1}, v_n)$, o $v_i$ é tal que se $i$ é um número par representa um linha horizontal (altura). Se $i$ é um número ímpar representa uma linha vertical (coordenada $x$). O vetor skyline deve representar o 'caminho' percorrido, por exemplo, por uma formiga começando no menor coordenada $x$ e viajando horizontal e verticalmente sobre todas as linhas que definem o horizonte. Por isso a última entrada em todos os vetores do horizonte será '0'.

**Exemplo** O desenho acima está assim representado:

```
Entrada
1 11 5
2 6 7
3 13 9
12 7 16
14 3 25
19 18 22
23 13 29
24 4 28
Saída
1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0
```

**Algoritmo 1** Como quase todo problema não trivial, este admite diversas estratégias de solução. Uma possível é desdobrar a cada elemento da lista de entrada em dois eventos: O primeiro determina o início de um prédio e o segundo o seu final. Os eventos terão a estrutura: (esquerda, altura, 0=início) e (direita, altura, 1=final). Para o conjunto de dados do exemplo, ficará: [(1, 11, 1), (5, 11, 0), (2, 6, 1), (7, 6, 0), (3, 13, 1), (9, 13, 0), (12, 7, 1), (16, 7, 0), (14, 3, 1), (25, 3, 0), (19, 18, 1), (22, 18, 0), (23, 13, 1), (29, 13, 0), (24, 4, 1), (28, 4, 0)] A seguir ordena-se esta lista pela primeira posição (o $x$) cria-se uma lista chamada vista vazia ([]) e outra chamada segmentos vazia ([]) e corrente valendo 0 e depois processa-se a lista:

```
se infim==0 (início)
  insere a altura na lista segmentos
  ordena segmentos em ordem decrescente
senão
  remove esta altura de segmentos
nova-altura = primeiro elemento de segmentos
          (ou 0 se vazia)
se nova-altura <> corrente
  insere altura na vista
  corrente = nova-altura
retornar vista
```

Este algoritmo implementado reproduziu os dados do exemplo corretamente, mas foi recusado pelo online judge. Certamente algum conjunto de dados (oculto) ressalvou algum bug ou imprecisão.

**Algoritmo 2** Uma saída foi buscar outro algoritmo mais simples (embora menos eficiente) e baseado numa informação perdida do enunciado (os valores são menores do que 10000). Agora define-se um vetor de 10001 elementos que representa as ordenadas $x$ de todos os envolvidos no problema. Agora a estratégia ficou mais fácil: basta examinar o que acontece em cada ponto do vetor (já que todas as medidas são inteiras).

```
final=0
altura = [0] * 10001
processando (pcom, altu, pfin)
  se pfin>final
    final=pfin (localizando o final do desenho)
  variando i de pcom até pfin
    altura[i] = max(altura[i],altu)
  cara = altura[1]
  imprima (1, cara)
  variando i de 2 até final
    se cara <> altura[i]
      imprima (i, altura[i])
      cara = altura[i]
  imprima (final, 0)
```

## Exemplos resolvidos Caso 1

```
 1 18 11
 4 25  7
 7  6 15
17 26 25
19 25 25
21 12 26
23 28 26
29 19 30
37 16 42
40 22 47
```

Deu como resultado

```
1 18 4 25 7 18 11 6 15 0 17 26 23 28
26 0 29 19 30 0 37 16 40 22 47 0
```

**Caso 2**

```
 1  3  9
12 17 21
24  4 28
24  1 29
31  9 38
39 10 45
41  7 45
42  3 46
44 23 53
45 18 51
45  7 52
47 18 55
```

Deu como resultado

```
1 3 9 0 12 17 21 0 24 4 28 1 29 0 31 9
38 0 39 10 44 23 53 18 55 0
```

## ☞ Para você fazer

Obviamente, antes de executar este caso, você está convidado (intimado !) a submeter sua solução ao online.judge da UVA. Se você obtiver o ACCEPTED no site, pode exigir um 10 nesta folha independente do que o professor (e seu corretor) acharem. Combinado ?

Para deixar as coisas mais ágeis, considere que todos os números deste caso são menores do que 100.

Eis os seus dados aqui:

```
 1 11  3
 4 19  6
 5  5  6
12  9 18
23 23 26
23 16 30
28  7 29
28 29 36
31  5 34
39 27 44
```

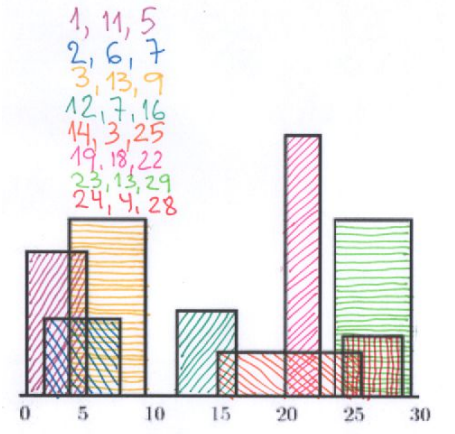
Responda aqui:

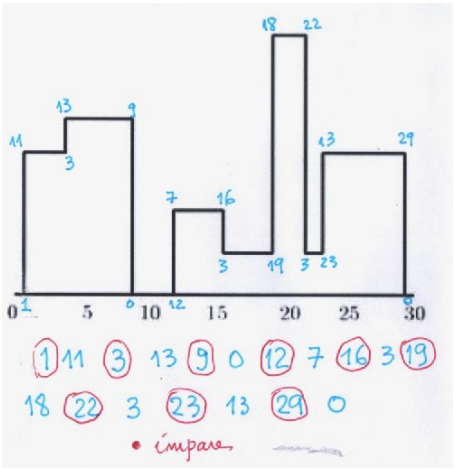
507-75958 -

CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 21/10/2024 - 12:07:31.8
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
VIVXhb1a V: 1.02
75965 PHELIPE GABRIEL LIMA DA SILV
24PRG507 - 20 _____ / _____ / _____

## UVA105-Perfil da skyline

Tal como esta lá no site (`http:\\onlinejudge.org`): Com o advento das estações de trabalho gráficas de alta velocidade, CAD (design auxiliado por computador) e outras áreas (design CAM, VLSI) têm feito uso cada vez mais eficaz dos computadores. Um dos problemas com desenhar imagens é a eliminação de linhas ocultas (linhas obscurecidas por outras partes de um desenho). Você deverá projetar um programa para ajudar um arquiteto a desenhar o horizonte de uma cidade, considerando os locais dos edifícios da cidade. Para tornar o problema tratável, todos os edifícios são de forma retangular e eles compartilham um fundo comum (a cidade onde estão construídos é muito plana). A cidade também é vista como bidimensional. Um edifício é especificado por uma tripla ordenada $(L_i, O_i, R_i)$ onde $L_i$ e $R_i$ são as coordenadas à esquerda e à direita, respectivamente, do edifício $_i$, $(0 < L_i < R_i)$ e $H_i$ é a altura do edifício. No diagrama abaixo, os edifícios são mostrados à esquerda com triplas $(1, 11, 5), (2, 6, 7), (3, 13, 9), (12, 7, 16), (14, 3, 25),$ $(19, 18, 22), (23, 13, 29), (24, 4, 28)$ o horizonte, mostrado à direita, é representado pela sequência: $(1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)$ Estes dados correspondem à seguinte imagem:




**Entrada** A entrada é uma sequência de construção de triplas. Todas as coordenadas dos edifícios são números inteiros menores que 10.000 e haverá pelo menos um e no máximo 5.000 edifícios no arquivo de entrada. Cada tripla (edifício) é uma linha sozinha no arquivo de entrada. Todos os inteiros em uma tripla são separados por um ou mais espaços. As triplas serão classificados por $L_i$, a coordenada $x$ esquerda do edifício, então o edifício com o menor valor à esquerda (a coordenada $x$) é o primeiro no arquivo de entrada.

**Saída** A saída deve consistir no vetor que descreve o horizonte conforme mostrado no exemplo acima. No vetor skyline $(v_1, v_2, v_3, ..., v_{n-2}, v_{n-1}, v_n)$, o $v_i$ é tal que se $i$ é um número par representa um linha horizontal (altura). Se $i$ é um número ímpar representa uma linha vertical (coordenada $x$). O vetor skyline deve representar o 'caminho' percorrido, por exemplo, por uma formiga começando no menor coordenada $x$ e viajando horizontal e verticalmente sobre todas as linhas que definem o horizonte. Por isso a última entrada em todos os vetores do horizonte será '0'.

**Exemplo** O desenho acima está assim representado:

```
Entrada
1 11 5
2 6 7
3 13 9
12 7 16
14 3 25
19 18 22
23 13 29
24 4 28
Saída
1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0
```

**Algoritmo 1** Como quase todo problema não trivial, este admite diversas estratégias de solução. Uma possível é desdobrar a cada elemento da lista de entrada em dois eventos: O primeiro determina o início de um prédio e o segundo o seu final. Os eventos terão a estrutura: (esquerda, altura, 0=início) e (direita, altura, 1=final). Para o conjunto de dados do exemplo, ficará: [(1, 11, 1), (5, 11, 0), (2, 6, 1), (7, 6, 0), (3, 13, 1), (9, 13, 0), (12, 7, 1), (16, 7, 0), (14, 3, 1), (25, 3, 0), (19, 18, 1), (22, 18, 0), (23, 13, 1), (29, 13, 0), (24, 4, 1), (28, 4, 0)] A seguir ordena-se esta lista pela primeira posição (o $x$) cria-se uma lista chamada vista vazia ([]) e outra chamada segmentos vazia ([]) e corrente valendo 0 e depois processa-se a lista:

```
se infim==0 (início)
  insere a altura na lista segmentos
  ordena segmentos em ordem decrescente
senão
  remove esta altura de segmentos
nova-altura = primeiro elemento de segmentos
          (ou 0 se vazia)
se nova-altura <> corrente
  insere altura na vista
  corrente = nova-altura
retornar vista
```

Este algoritmo implementado reproduziu os dados do exemplo corretamente, mas foi recusado pelo online judge. Certamente algum conjunto de dados (oculto) ressalvou algum bug ou imprecisão.

**Algoritmo 2** Uma saída foi buscar outro algoritmo mais simples (embora menos eficiente) e baseado numa informação perdida do enunciado (os valores são menores do que 10000). Agora define-se um vetor de 10001 elementos que representa as ordenadas $x$ de todos os envolvidos no problema. Agora a estratégia ficou mais fácil: basta examinar o que acontece em cada ponto do vetor (já que todas as medidas são inteiras).

```
final=0
altura = [0] * 10001
processando (pcom, altu, pfin)
  se pfin>final
    final=pfin (localizando o final do desenho)
  variando i de pcom até pfin
    altura[i] = max(altura[i],altu)
  cara = altura[1]
  imprima (1, cara)
  variando i de 2 até final
    se cara <> altura[i]
      imprima (i, altura[i])
      cara = altura[i]
  imprima (final, 0)
```

## Exemplos resolvidos Caso 1

```
 1 18 11
 4 25  7
 7  6 15
17 26 25
19 25 25
21 12 26
23 28 26
29 19 30
37 16 42
40 22 47
```

Deu como resultado

```
1 18 4 25 7 18 11 6 15 0 17 26 23 28
26 0 29 19 30 0 37 16 40 22 47 0
```

**Caso 2**

```
 1  3  9
12 17 21
24  4 28
24  1 29
31  9 38
39 10 45
41  7 45
42  3 46
44 23 53
45 18 51
45  7 52
47 18 55
```

Deu como resultado

```
1 3 9 0 12 17 21 0 24 4 28 1 29 0 31 9
38 0 39 10 44 23 53 18 55 0
```

## ☞ Para você fazer

Obviamente, antes de executar este caso, você está convidado (intimado !) a submeter sua solução ao online.judge da UVA. Se você obtiver o ACCEPTED no site, pode exigir um 10 nesta folha independente do que o professor (e seu corretor) acharem. Combinado ?

Para deixar as coisas mais ágeis, considere que todos os números deste caso são menores do que 100.

Eis os seus dados aqui:

```
 1  5  6
12 25 21
21  6 30
29 27 30
33  7 39
39  7 42
41 20 44
42 19 46
43 22 51
45  8 54
```

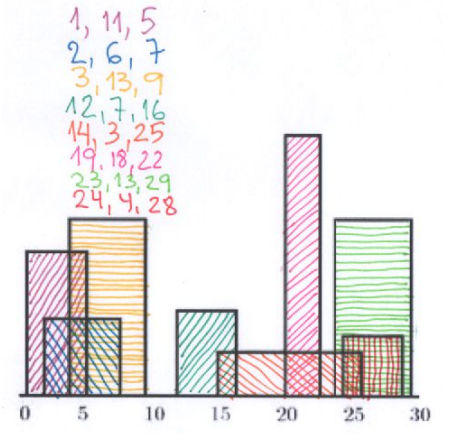
Responda aqui:

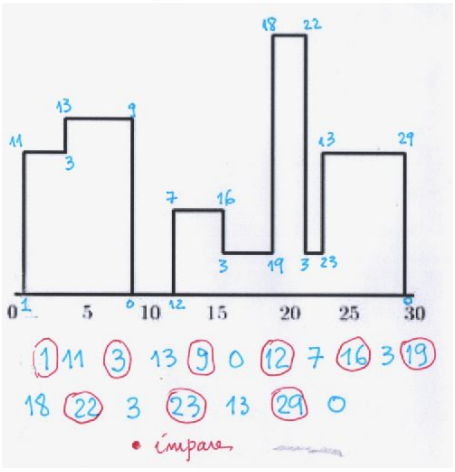
507-75965 -

CEP-UFPR-UTFPR-PUC/Pr-UP Sistemas de Informação 21/10/2024 - 12:07:31.8
Matemática aplicada Prof Dr P Kantek (pkantek@gmail.com)
VIVXhb1a V: 1.02
75972 REBECA CABRAL DOS SANTOS
24PRG507 - 21 _____ / _____ / _____

## UVA105-Perfil da skyline

Tal como esta lá no site (`http:\\onlinejudge.org`): Com o advento das estações de trabalho gráficas de alta velocidade, CAD (design auxiliado por computador) e outras áreas (design CAM, VLSI) têm feito uso cada vez mais eficaz dos computadores. Um dos problemas com desenhar imagens é a eliminação de linhas ocultas (linhas obscurecidas por outras partes de um desenho). Você deverá projetar um programa para ajudar um arquiteto a desenhar o horizonte de uma cidade, considerando os locais dos edifícios da cidade. Para tornar o problema tratável, todos os edifícios são de forma retangular e eles compartilham um fundo comum (a cidade onde estão construídos é muito plana). A cidade também é vista como bidimensional. Um edifício é especificado por uma tripla ordenada $(L_i, O_i, R_i)$ onde $L_i$ e $R_i$ são as coordenadas à esquerda e à direita, respectivamente, do edifício $_i$, $(0 < L_i < R_i)$ e $H_i$ é a altura do edifício. No diagrama abaixo, os edifícios são mostrados à esquerda com triplas $(1, 11, 5), (2, 6, 7), (3, 13, 9), (12, 7, 16), (14, 3, 25),$ $(19, 18, 22), (23, 13, 29), (24, 4, 28)$ o horizonte, mostrado à direita, é representado pela sequência: $(1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0)$ Estes dados correspondem à seguinte imagem:




**Entrada** A entrada é uma sequência de construção de triplas. Todas as coordenadas dos edifícios são números inteiros menores que 10.000 e haverá pelo menos um e no máximo 5.000 edifícios no arquivo de entrada. Cada tripla (edifício) é uma linha sozinha no arquivo de entrada. Todos os inteiros em uma tripla são separados por um ou mais espaços. As triplas serão classificados por $L_i$, a coordenada $x$ esquerda do edifício, então o edifício com o menor valor à esquerda (a coordenada $x$) é o primeiro no arquivo de entrada.

**Saída** A saída deve consistir no vetor que descreve o horizonte conforme mostrado no exemplo acima. No vetor skyline $(v_1, v_2, v_3, ..., v_{n-2}, v_{n-1}, v_n)$, o $v_i$ é tal que se $i$ é um número par representa um linha horizontal (altura). Se $i$ é um número ímpar representa uma linha vertical (coordenada $x$). O vetor skyline deve representar o 'caminho' percorrido, por exemplo, por uma formiga começando na menor coordenada $x$ e viajando horizontal e verticalmente sobre todas as linhas que definem o horizonte. Por isso a última entrada em todos os vetores do horizonte será '0'.

**Exemplo** O desenho acima está assim representado:

```
Entrada
1 11 5
2 6 7
3 13 9
12 7 16
14 3 25
19 18 22
23 13 29
24 4 28
Saída
1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0
```

**Algoritmo 1** Como quase todo problema não trivial, este admite diversas estratégias de solução. Uma possível é desdobrar a cada elemento da lista de entrada em dois eventos: O primeiro determina o início de um prédio e o segundo o seu final. Os eventos terão a estrutura: (esquerda, altura, 0=início) e (direita, altura, 1=final). Para o conjunto de dados do exemplo, ficará: [(1, 11, 1), (5, 11, 0), (2, 6, 1), (7, 6, 0), (3, 13, 1), (9, 13, 0), (12, 7, 1), (16, 7, 0), (14, 3, 1), (25, 3, 0), (19, 18, 1), (22, 18, 0), (23, 13, 1), (29, 13, 0), (24, 4, 1), (28, 4, 0)] A seguir ordena-se esta lista pela primeira posição (o $x$) cria-se uma lista chamada vista vazia ([]) e outra chamada segmentos vazia ([]) e corrente valendo 0 e depois processa-se a lista:

```
se infim==0 (início)
  insere a altura na lista segmentos
  ordena segmentos em ordem decrescente
senão
  remove esta altura de segmentos
nova-altura = primeiro elemento de segmentos
          (ou 0 se vazia)
se nova-altura <> corrente
  insere altura na vista
  corrente = nova-altura
retornar vista
```

Este algoritmo implementado reproduziu os dados do exemplo corretamente, mas foi recusado pelo online judge. Certamente algum conjunto de dados (oculto) ressalvou algum bug ou imprecisão.

**Algoritmo 2** Uma saída foi buscar outro algoritmo mais simples (embora menos eficiente) e baseado numa informação perdida do enunciado (os valores são menores do que 10000). Agora define-se um vetor de 10001 elementos que representa as ordenadas $x$ de todos os envolvidos no problema. Agora a estratégia ficou mais fácil: basta examinar o que acontece em cada ponto do vetor (já que todas as medidas são inteiras).

```
final=0
altura = [0] * 10001
processando (pcom, altu, pfin)
  se pfin>final
    final=pfin (localizando o final do desenho)
  variando i de pcom até pfin
    altura[i] = max(altura[i],altu)
  cara = altura[1]
  imprima (1, cara)
  variando i de 2 até final
    se cara <> altura[i]
      imprima (i, altura[i])
      cara = altura[i]
  imprima (final, 0)
```

## Exemplos resolvidos Caso 1

```
 1 18 11
 4 25  7
 7  6 15
17 26 25
19 25 25
21 12 26
23 28 26
29 19 30
37 16 42
40 22 47
```

Deu como resultado

```
1 18 4 25 7 18 11 6 15 0 17 26 23 28
26 0 29 19 30 0 37 16 40 22 47 0
```

**Caso 2**

```
 1  3  9
12 17 21
24  4 28
24  1 29
31  9 38
39 10 45
41  7 45
42  3 46
44 23 53
45 18 51
45  7 52
47 18 55
```

Deu como resultado

```
1 3 9 0 12 17 21 0 24 4 28 1 29 0 31 9
38 0 39 10 44 23 53 18 55 0
```

## ☞ Para você fazer

Obviamente, antes de executar este caso, você está convidado (intimado !) a submeter sua solução ao online.judge da UVA. Se você obtiver o ACCEPTED no site, pode exigir um 10 nesta folha independente do que o professor (e seu corretor) acharem. Combinado ?

Para deixar as coisas mais ágeis, considere que todos os números deste caso são menores do que 100.

Eis os seus dados aqui:

```
 1  9 11
 6  7 15
10  7 12
13 24 15
13 19 20
31 15 38
32 18 41
33 25 41
36  5 38
39 23 41
```

Responda aqui:

507-75972 -