Manuseio de tabelas II

Para todos os algoritmos abaixo suponha-se duas variáveis globais:

```
1: inteiro TABELA[num]
2: inteiro ULTIMO
```

Onde $TABELA$ contém os dados a serem acessados. num é um valor suficiente para conter todos os dados necessários e $ULTIMO$ é um valor que aponta para o último valor válido da tabela.

Inclusão de itens em tabelas desordenadas Esta inclusão é fácil, bastando acrescentar o item ao final da área útil da tabela.

```
1: função INCLUSAONORD (inteiro CHAVE)
2: ULTIMO ← ULTIMO + 1
3: se ULTIMO > num então
4:     ...erro...
5: fim se
6: TABELA[ULTIMO] ← CHAVE
```

Qual a quantidade de operações para fazer uma inclusão em tabela não ordenada no caso de uma tabela de

elementos	operações
100	
10.000	
1.000.000	
n	

Exclusão de itens em tabelas desordenadas Há duas estratégias aqui: a primeira é eliminar o item puxando os que ficaram abaixo da exclusão para "tampar o buraco".

```
1: função EXCLUSAO1NORD (inteiro CHAVE)
2: inteiro i ← 1
3: enquanto TABELA[i] ≠ CHAVE faça
4:     i++
5:     se i > ULTIMO então
6:         ...erro...
7:     fim se
8: fim enquanto
9: enquanto i < ULTIMO faça
10:     TABELA[i] ← TABELA [i+1]
11:     i++
12: fim enquanto
13: ULTIMO ← ULTIMO - 1
```

Qual a quantidade de operações para fazer uma exclusão do tipo 1; qual o número médio de operações para achar um item que está na tabela e qual o número médio de operações para concluir que um dado número não está na tabela

elementos	ops para exclusão	ops para achar	ops para concluir que não está
100			
10.000			
1.000.000			
n			

A segunda estratégia é incluir algum "indicador" ($filler$) que informe que o item foi eliminado.

```
1: função EXCLUSAO2NORD (inteiro CHAVE)
2: inteiro i ← 1
3: enquanto TABELA[i] ≠ CHAVE faça
4:     i++
5:     se i > ULTIMO então
6:         ...erro...
7:     fim se
8: fim enquanto
9: TABELA[i] ← 9999999
```

Qual a quantidade de operações para fazer uma exclusão do tipo 2; qual o número médio de operações para achar um item que está na tabela e qual o número médio de operações para concluir que um dado número não está na tabela

elementos	ops para exclusão	ops para achar	ops para concluir que não está
100			
10.000			
1.000.000			
n			

A vantagem desta segunda alternativa é que a exclusão é mais rápida, mas em compensação as buscas tendem a ficar mais demoaradas.

Busca binária Conjunto ordenado, ótimo desempenho

```
1: função BUSCABIN(inteiro CHAVE)
2: inteiro INIC,METADE,FIM
3: INIC ← 1
4: FIM ← ULTIMO
5: repita
6:     METADE ← ⌊ ((INIC + FIM)/2)
7:     se CHAVE ≤ TABELA[METADE] então
8:         FIM ← METADE - 1
9:     senão
10:        INIC ← METADE + 1
11:    fim se
12: até TABELA[METADE] = CHAVE ∨ INIC > FIM
13: se TABELA[METADE] = CHAVE então
14:     devolva METADE
15: senão
16:     devolva -1
17: fim se
```

Qual a quantidade de operações para fazer pesquisa e qual o número médio de operações para concluir que um dado número não está na tabela

elementos	ops para pesquisa	ops para concluir que não está
100		
10.000		
1.000.000		
n		

Inclusão em tabela ordenada A contrapartida para poder fazer a busca ordenada é criar e manter a tabela em ordem. Para isso, a inclusão de novos elementos deve ocorrer em seus locais específicos e não no final, como vimos acima.

```
1: função INCLUSAOORD (inteiro CHAVE)
2: ULTIMO ← ULTIMO + 1
3: se ULTIMO > num então
4:     ...erro...
5: fim se
6: inteiro i ← 1
7: enquanto TABELA[i] < CHAVE faça
8:     i++
9: fim enquanto
10: k ← ULTIMO
11: enquanto k > i faça
12:     TABELA[k] ← TABELA[k-1]
13:     k−−
14: fim enquanto
15: TABELA[k] ← CHAVE
```

Qual a quantidade de operações para fazer uma inclusão em tabela ordenada no caso de uma tabela de

elementos	operações
100	
10.000	
1.000.000	
n	

Para você fazer

Escreva a seguir um algoritmo de uma função que Receba uma matriz global, de nome MAT-ENT, cujo conteúdo é:

MAT-ENT é uma matriz de 3 colunas, sendo todas inteiros, formada por

código da transação	valor da transação	código da cidade

O algoritmo deverá criar e imprimir uma segunda matriz, de apenas duas colunas

MAT-SAI é uma matriz de 2 colunas, sendo ambos inteiros, formada por

código da cidade	total dos valores de transações

Nesta matriz de saída, os valores de débito (cód 2) deverão ser totalizados, e quando houver **quebra** de cidade, uma nova linha deve ser introduzida na matriz de saida com o total. Note que a matriz de entrada está ordenada por cidade. (ENUNCIADO 2)

Orientações:

- Anexe a esta folha a listagem deste algoritmo produzida pelo software VISUALG. Anexe também as listagens das matrizes de entrada e de saída.
- Para baixar, www.apoioinformatica.inf.br