A*: caminho mais barato

Este exercício vai oferecer uma matriz quadrada numérica e solicitar qual a menor soma de vizinhos contíguos saindo do elemento mais alto à esquerda até o mais baixo à direita. Ao fazer a soma, pode-se andar (somar) para qualquer um dos quatro vizinhos da célula atual. Este exercício está baseado no problema 83 do projeto Euler (projecteuler.net). Assim, por exemplo, na matriz

```
131 673 234 103  18
201  96 342 965 150
630 803 746 422 111
537 699 497 121 956
805 732 524  37 331
```

O caminho mais barato é: 131 + 201 + 96 + 342 + 234 + 103 + 18 + 150 + 111 + 422 + 121 + 37 + 331 = 2297. O referido caminho fica mais visível em

O fato do caminho poder voltar à esquerda inviabiliza o algoritmo da programação dinâmica.

Embora existam diversos algoritmos para calcular este caminho vai se usar aqui o algoritmo A* ligeiramente modificado.

Relembrando, o algoritmo A* está baseado na idéia de custo de um nodo, que sempre é calculado fazendo-se

$$C = F + H$$

onde C é o custo de um nodo, F é o valor real desde o início até o nodo em análise e H é uma heurística estimadora do caminho que falta para chegar até o final.

Neste caso, F é o valor real da soma dos valores da matriz desde a casa inicial (1,1) até a casa que se está examinando. Quanto a H, usaremos como função heurística o menor valor de custos da matriz original multiplicado pela distância manhattan da casa atual até o final.

Para entender este conceito, examine a matriz H do exemplo acima:

```
162 144 126 108  90
144 126 108  90  72
126 108  90  72  54
108  90  72  54  36
 90  72  54  36  18
```

Note que o menor valor da matriz original é 18, e este valor foi sendo multiplicado pela quantidade de casas necessárias a percorrer até o final da matriz. Eis a variavel LA do exemplo acima

	lin	col	som	custo	pai
1-	1	1	131	10000000293	-1
2-	1	2	804	10000000948	1
3-	2	1	332	10000000476	1
4-	2	2	428	10000000554	3
5-	3	1	962	10000001088	3
6-	2	3	770	10000000878	4
7-	3	2	1231	10000001339	4
8-	1	3	1004	10000001130	6
9-	2	4	1735	10000001825	6
10-	3	3	1516	10000001606	6
11-	4	1	1499	10000001607	5
12-	1	4	1107	10000001215	8
13-	1	5	1125	10000001215	12
14-	2	5	1275	10000001347	13
15-	4	2	1930	2020	7
16-	3	5	1386	10000001440	14
17-	4	5	2342	2378	16
18-	3	4	1808	10000001880	16
19-	4	3	2013	2085	10
20-	5	1	2304	2394	11
21-	4	4	1929	10000001983	18
22-	5	4	1966	10000002002	21
23-	5	5	2297	2315	22

Mais um exemplo

Seja uma matriz de ordem 14 a saber:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1|	70	149	101	892	457	539	404	230	254	321	833	902	962	592
2|	929	159	415	466	507	24	231	222	493	630	543	951	215	290
3|	729	517	916	72	261	688	316	980	354	842	91	333	613	471
4|	165	77	713	484	525	172	27	812	766	167	410	998	933	95
5|	509	881	977	282	402	990	93	955	198	979	45	885	889	79
6|	548	105	569	320	558	423	452	46	367	875	153	285	584	618
7|	446	417	658	606	249	53	935	473	264	23	771	148	827	770
8|	783	697	9	343	381	942	210	88	182	536	13	667	874	538
9|	531	47	819	345	331	125	992	600	69	194	954	604	246	563
10|	969	835	708	499	436	756	409	693	628	61	475	869	219	635
11|	828	769	786	442	138	679	73	186	145	96	128	137	247	431
12|	975	274	903	777	8	26	734	614	508	549	272	946	792	896
13|	831	386	97	595	853	256	774	414	968	801	991	545	883	656
14|	166	258	989	799	152	5	155	554	681	912	7	356	820	529

Nesta matriz o menor custo é 7036, com 27 casas percorridas, a saber: 14 14, 13 14, 12 14, 11 14, 11 13, 11 12, 11 11, 11 10, 10 10, 9 10, 9 9, 8 9, 8 8, 7 8, 6 8, 6 7, 5 7, 4 7, 4 6, 4 5, 3 5, 3 4, 2 4, 2 3, 1 3, 1 2, 1 1.

O algoritmo

```
1: leia mat
2: t ← ordem de mat
3: infi ← 10000000
4: men ← menor elemento de mat
5: LA[3000][5]
6: h[t][t]← men
7: para i=t; i≥ 1; i--
8:    para j=t; j≥ 1; j--
9:       h[i][j]← h[i][j] ×(−1) + (t − (i + 1)) + (t − (j + 1))
10:    fim{para}
11: fim{para}
12: LA[1][]← 1,1,mat[1][1],(mat[1][1]+h[1][1]),-1
13: enquanto (1=1)
14:    mm ← a primeira linha de LA com o menor valor
15:    LA[mm][4]←LA[mm][4]+infi
16:    f1 ← (LA[mm][1]-1),LA[mm][2] // acima
17:    f2 ← LA[mm][1],(LA[mm][2]+1) // direita
18:    f3 ← (LA[mm][1]+1),LA[mm][2] // baixo
19:    f4 ← LA[mm][1],(LA[mm][2]-1) // esquerda
20:    se f1 nos limites da matriz ∧ ainda não existente em LA
21:       LA[proximo][]← f1,
22:       (LA[mm][3]+mat[f1[1]][f1[2]]),
23:       (LA[mm][3]+mat[f1[1]][f1[2]]+h[f1[1]][f1[2]]),mm
24:       proximo++
25:       se f1 = t,t
26:          break
27:       fim{se}
28:    fim{se}// idem para f2, f3 e f4
29: fim{enquanto}
30: ultimo ← proximo - 1
31: imprima LA[ultimo][3]
32: enquanto ultimo ≠ -1
33:    imprima LA[ultimo][1] e LA[ultimo][2]
34:    ultimo ← LA[ultimo][5]
35: fim{enquanto}
```

Mais exemplos

Peça ao professor os arquivos EXEA00000, EXEA00001 e EXEA00002 e teste seu algoritmo neles. Deve achar as respostas 280366 101, 267899 103 e 274815 107.

☞ Para você fazer

Você receberá um arquivo de nome

```
XXXAAST01
```

contendo uma matriz quadrada de ordem 50 e contendo números menores do que 10000. Deve aplicar o algoritmo acima e achar a menor soma possível saindo da casa 1,1 e chegando na casa 50,50 podendo deslocar-se a qualquer um dos 4 vizinhos enquanto constrói o caminho.

Deve descobrir 2 informações:

1. Qual o valor da soma mínima
2. Quantas casas da matriz são percorridas para chegar a esse valor.

soma mínima	quantas casas