Introdução à Análise de Algoritmos
2º semestre de 2020 - Turmas 04 e 94
Exercício Programa 2: Labirinto

1 Objetivo

O objetivo deste exercício programa é, através do desenvolvimento de um algoritmo do tipo *tentativa e erro*, ajudar um explorador a encontrar um caminho dentro de um labirinto, dada uma posição de partida e outra de destino. Havendo mais de um caminho possível, seu algoritmo deve selecionar aquele que é considerado o melhor a partir de um certo critério (escolhido para a execução do programa). O labirinto é representado por uma matriz e a movimentação pelo labirinto pode se dar em 4 direções: para cima, para baixo, para a esquerda e para a direita. Um caminho que leva o explorador da origem ao destino só pode passar por cada posição uma única vez. Além disso, podem haver itens, cada um com um determinado valor e peso, espalhados pelo labirinto. Sempre que o explorador encontra um item, ele é obrigado a coletar o item e levá-lo consigo. Embora coletar itens seja vantajoso pelo valor em si de cada item, os itens também possuem peso, o que aumenta o tempo que o explorador leva para percorrer um determinado caminho. Quatro são os critérios que devem ser considerados para determinar qual o melhor caminho que leva o explorador da origem ao destino:

1. **Caminho mais curto. Isto é, que passe pelo menor número possível de casas.**
2. **Caminho mais longo. Isto é, que passe pelo maior número possível de casas** (ok, este é um critério estranho, mas pense que o explorador possui motivos para visitar o maior número de casas possível sem passar mais de uma vez por uma mesma casa).
3. **Caminho mais valioso. Isto é, que maximiza o valor dos itens coletados.**
4. **Caminho mais rápido. Ou seja, aquele minimiza o tempo que se leva para chegar no destino** (note que não necessariamente equivale ao mais curto, pois depende dos itens que são coletados no caminho, o que por sua vez afeta o tempo necessário para percorrê-lo).

O tempo t que o explorador leva para realizar um movimento, indo de uma posição qualquer para uma das posições adjacentes é dado em função do peso p dos itens que ele carrega no momento (note que, tipicamente, o valor inicial de p deve ser zero – a não ser que a posição de partida já contenha algum item. Note também que o valor de p irá variar, conforme um caminho for sendo traçado, em função dos itens encontrados no caminho). Este tempo, referente a execução de um passo, é dado pela fórmula apresentada na Equação 1.

$$t = \left(1 + \frac{p}{10}\right)^2 \tag{1}$$

Seu programa deve ler um arquivo de entrada, contendo a definição do labirinto (representado por uma matriz de caracteres), lista de itens espalhados pelo labirinto (coordenadas, valor e peso) e as posições de partida e de destino. Deve ser gerada como saída uma listagem de coordenadas, referentes às posições que

fazem parte do caminho encontrado, assim como uma listagem dos itens coletados. Maiores detalhes sobre o arquivo de entrada e a formatação da saída são apresentados a seguir.

2 Arquivo de Entrada

Seu programa deve receber dois parâmetros na linha de comando, da seguinte forma:

```
java EP <arquivo com definição do labirinto> <criterio>
```

onde o primeiro parâmetro especifica o nome de um arquivo que contem a especificação do labirinto, e o segundo especifica um dos quatro critérios a ser considerado na procura pelo caminho (um valor inteiro entre 1 e 4). Abaixo um exemplo de como deve ser chamado o EP na linha de comando:

```
java EP mapa1.txt 2
```

O arquivo de entrada deve conter a definição do labirinto, a lista de itens espalhados pelo mesmo, e as posições de partida/destino, respeitando a seguinte formatação:

```
<numero de linhas do labirinto> <numero de colunas do labirinto>
<linha 0>
<linha 1>
<linha 2>
(...)
<número de itens>
<linha item 0> <coluna item 0> <valor item 0> <peso do item 0>
<linha item 1> <coluna item 1> <valor item 1> <peso do item 1>
<linha item 2> <coluna item 2> <valor item 2> <peso do item 2>
(...)
<linha posição de partida> <coluna posição de partida>
<linha posição de destino> <coluna posição de destino>
```

A seguir, um exemplo de um arquivo de entrada:

```
7 5
.....
.X.X.
.X.X.
.....
.X.X.
.X.X.
.....
3
4 0 4 3
3 2 1 8
2 0 8 6
6 2
0 2
```

A primeira linha do arquivo contem 2 valores inteiros que definem o número de linhas e o número de colunas do labirinto (no caso do exemplo, 7 linhas e 5 colunas). Cada uma das 7 (no caso do exemplo) linhas seguintes contem uma sequência de caracteres de tamanho igual ao número de colunas (5, para o arquivo de exemplo), e cada caractere representa uma posição do labirinto. O caractere "." indica uma posição livre, enquanto que o caractere "X" representa uma posição bloqueada (ou seja, não pode fazer parte de um caminho). Após a última linha referente ao labirinto, há uma nova linha com um valor inteiro que define a quantidade de itens espelhados pelo labirinto (3 para o exemplo). E em seguida uma linha referente a cada item. Cada linha referente a um item possui 4 valores inteiros: linha e coluna do item no labirinto, seguidos do valor e do peso do item. Finalmente, a penúltima linha contem as coordenadas (linha e coluna, respectivamente) da posição de partida do explorador (linha 6 e coluna 2, para o exemplo), e a última linha as coordenadas da posição de destino (linha 0 e coluna 2, para o exemplo).

3 Saída

A saída gerada pelo seu programa deve ser impressa na saída padrão e deve respeitar a seguinte formatação:

```
<tamanho do caminho encontrado> <tempo para percorrer o caminho>
<linha da posição 0> <coluna da posição 0>
<linha da posição 1> <coluna da posição 1>
<linha da posição 2> <coluna da posição 2>
(...)
<quantidade de itens coletados> <valor total dos itens> <peso total dos itens>
<linha item coletado 0> <coluna item coletado 0>
<linha item coletado 1> <coluna item coletado 1>
<linha item coletado 2> <coluna item coletado 2>
(...)
```

Para o arquivo de entrada apresentado como exemplo, quando a opção de critério 1 é utilizada, a seguinte saída deve ser gerada:

```
7 12.72
6 2
5 2
4 2
3 2
2 2
1 2
0 2
1 1 8
3 2
```

Quando a opção 2 é escolhida, devemos ter a seguinte saída:

```
15 39.94
6 2
6 1
6 0
5 0
```

```
4 0
3 0
3 1
3 2
3 3
3 4
2 4
1 4
0 4
0 3
0 2
2 5 11
4 0
3 2
```

Já para a opção 3, a saída deve ser:

```
11 21.82
6 2
6 1
6 0
5 0
4 0
3 0
2 0
1 0
0 0
0 1
0 2
2 12 9
4 0
2 0
```

E finalmente, para a opção 4, devemos ter a seguinte saída:

```
11 10.0
6 2
6 3
6 4
5 4
4 4
3 4
2 4
1 4
0 4
0 3
0 2
0 0 0
```

4 Prazos e Entrega

Este EP deve ser desenvolvido **individualmente** ou em **duplas**, e deve ser implementado na linguagem **Java**. Sua entrega deve ser feita até o dia 23/01/2021 às 23:59. Para entregá-lo, crie uma pasta nomeada com seu número USP e coloque dentro dela os arquivos fontes criados. Adicione também um arquivo **README** com informações básicas sobre como compilar e rodar o EP, bugs conhecidos e outras informações relevantes. Não esqueça também de identificar de forma clara (no README e nos cabeçalhos dos arquivos fonte) o(s) autor(es) do EP. Em seguida compacte a pasta no formato **ZIP** (criando um arquivo também nomeado com seu número USP mais a extensão ".zip") e envie este arquivo através da tarefa criada no eDisciplinas para este propósito. Para EPs desenvolvidos em dupla, apenas um dos componentes deve fazer o envio.

Atenção:

- Seu EP deve seguir à risca as especificações de entrada e saída deste enunciado. A não conformidade com as especificações irá acarretar em descontos na nota.

Boa diversão!