Lista 1

Entrega: 22/04/2019

O exercício será composto de 2 partes:

1. Implementar *BFS-Graph*, *Iterative Deepening*, A^*, *IDA**, *Greedy Best-first Search* para o problema 8-puzzle.
2. Implementar A^* para o problema 15-puzzle.

Antes de iniciar as tarefas, preste atenção nas regras do padrão de entrada abaixo.
O trabalho deverá ser realizado em grupos de 2 pessoas.

1 Entrada

O seu código receberá como entrada um parâmetro definindo o algoritmo `-bfs, -idfs, -astar, -idastar, -gbfs`, e um estado inicial do puzzle ou lista de estados iniciais (todos de mesmo tamanho) separados por virgula.

O estado inicial é representado por uma lista de nove ou dezesseis números, onde o número na posição n indica a posição do tile $n-1$ no grid $n \times n$. A lista `7 2 4 5 0 6 8 3 1` corresponde ao estado inicial do 8-puzzle abaixo.

$$\begin{bmatrix} 7 & 2 & 4 \\ 5 & 0 & 6 \\ 8 & 3 & 1 \end{bmatrix}$$

Abaixo seguem alguns exemplos de entradas.

```
-bfs 8 3 5 7 2 6 0 4 1
-bfs 3 0 8 7 5 2 1 6 4, 5 4 6 3 2 7 0 1 8, 0 1 8 4 2 7 3 6 5
```

2 Heurística

A função heurística a ser implementada é a distancia manhattan.

3 Estado Objetivo

Os estados objetivos correspondem aos grids ordenados para 8 e 16-puzzle, respectivamente.

$$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{bmatrix}$$

4 Implementação

As únicas linguagens aceitas são C e C++. Além disso, a compilação deve ser realizada utilizando um `Makefile` apenas digitando `make` por linha de comando. Deve ser gerado um executável chamado – *obrigatoriamente* – `main`. Qualquer biblioteca não nativa das linguagens é proibida.

Ordem de geração de sucessores: cima, esquerda, direita e baixo.

A*: ordem de prioridade valor-f, valor-h, e LIFO. GBFS: valor-h, e LIFO.

Como os algoritmos *Iterative Deepening* e *IDA** não possuem *closed-list* evite gerar o estado-pai na geração dos sucessores de um estado.

5 Saída

Para cada estado inicial na entrada a sua implementação devera apresentar em uma nova linha separados por virgula: número de nodos expandidos, comprimento da solução ótima, tempo para solução, valor médio da função heurística, valor da função heurística no estado inicial.

Exemplo 8-puzzle com A*:

```
./main -astar 0 6 1 7 4 2 3 8 5, 5 0 2 6 4 8 1 7 3, 2 4 7 0 3 6 8 1 5
```

A saída esperada seria:

```
91,16,0.0000479221,6.72561,10
612,21,0.000200987,9.81136,11
511,23,0.000167847,9.97655,15
```

Exemplo 8-puzzle com *BFS-Graph*:

```
./main -bfs 0 6 1 7 4 2 3 8 5, 5 0 2 6 4 8 1 7 3, 2 4 7 0 3 6 8 1 5
```

A saída esperada seria:

```
5209,16,0.00154996,0,10
42704,21,0.011677,0,11
86288,23,0.022454,0,15
```

Exemplo 8-puzzle com IDA*:

```
./main -idastar 0 6 1 7 4 2 3 8 5, 5 0 2 6 4 8 1 7 3, 2 4 7 0 3 6 8 1 5
```

A saída esperada seria:

```
165,16,0.0000319481,7.13781,10
439,21,0.0000369549,10.75,11
724,23,0.0000538826,11.7241,15
```

Exemplo 8-puzzle com *Iterative Deepening*:

```
./main -idfs 0 6 1 7 4 2 3 8 5, 5 0 2 6 4 8 1 7 3, 2 4 7 0 3 6 8 1 5
```

A saída esperada seria:

```
20305,16,0.000360012,0,10
336804,21,0.00481009,0,11
1183127,23,0.017087,0,15
```

Exemplo 8-puzzle com *Greedy Best-first Search*:

```
./main -gbfs 0 6 1 7 4 2 3 8 5, 5 0 2 6 4 8 1 7 3, 2 4 7 0 3 6 8 1 5
```

A saída esperada seria:

```
16,16,0.0000259876,5.5,10
96,27,0.0000660419,8.05422,11
90,57,0.0000729561,6.80982,15
```

Exemplo 15-puzzle com A*:

```
./main -astar 7 11 8 3 14 0 6 15 1 4 13 9 5 12 2 10,
      12 9 0 6 8 3 5 14 2 4 11 7 10 1 15 13,
       13 0 9 12 11 6 3 5 15 8 1 10 4 14 2 7
```

A saída esperada seria:

```
154092,46,0.0424099,22.9871,36
2731989,50,0.799799,24.6075,32
744209,53,0.206705,28.0517,39
```

6 Eficiência

Para o 8-puzzle sua implementação para cada um dos algoritmos deverá revolver todo o conjunto de teste em menos de 10 minutos usando no máximo 4 GB. Para o 15-puzzle sua implementação deverá resolver cada estado inicial no conjunto de testes usando menos de 30 segundos usando no máximo 8 GB.

7 Relatório

Entregar um relatório descrevendo na primeira página a sua implementação, estratégias e desafios. O relatório também deve apresentar na primeira página uma tabela com duas linhas para cada um dos algoritmos, uma para cada conjunto de estados iniciais, com a média de número de: nodos expandidos, comprimento da solução ótima, tempo para solução, valor médio da função heurística, valor da função heurística no estado inicial. Informe também o número de estados iniciais resolvidos do 15-puzzle. Para o cálculo do valor médio da função heurística utilize os valores no momento em que são calculados. Para facilitar, mantenha na implementação da função heurística uma variável acumuladora das estimativas e uma variável contadora para o número de vezes que a função é chamada. Ao final para reportar o valor médio basta dividir a variável acumuladora pela contadora. Nas páginas seguintes o relatório deve apresentar para cada um dos estados iniciais e para cada um dos algoritmos os valores requeridos acima. Enviar com o relatório um arquivo .csv para cada algoritmo com os dados obtidos na sua execução. A ordem das linhas deve ser igual a ordem das entradas. O número de linhas deve ser igual ao número total de instâncias, sendo que em instâncias não resolvidas os atributos devem ser preenchidos com um traço (−).

Ainda deve apresentar a configuração do computador em que os testes foram executados.

8 Formato de Entrega

O exercício deve ser enviado em um arquivo compactado `numero`, substituindo `numero` pelo número de matrícula do componente do grupo que submeter o trabalho no Moodle. Dentro deste arquivo deve haver um diretório com o mesmo nome do arquivo e dentro deste diretório os arquivos do trabalho e o relatório.

9 Código

O código entregue deve satisfazer todos os requisitos especificados neste enunciado. Programas que *i*) não compilam; *ii*) não respeitam as normas de entrada e saída; ou que *iii*) apresentam erros em tempo de execução, irão receber nota 0.

Descreva no cabeçalho de cada arquivo a idéia geral do código e detalhes específicos de partes que mereçam uma explicação.