Atividade 2 Moodle

João Augusto Lima (up201605314)
Mestrado Integrado em Engenharia
Informática e de Computação
Faculdade de Engenharia e da
Universidade do Porto
Porto, Portugal
up201605314@fe.up.pt

***Resumo*—O artigo em questão contém uma pequena descrição do problema N-Puzzle e uma formulação da resolução deste puzzle com métodos de inteligência artificial.**

Keywords—Inteligência Artificial, N-Puzzle, Algoritmo A*, Pesquisa em Largura, Pesquisa Gananciosa.

I. Introdução

Na segunda atividade do moodle da unidade curricular de Inteligência Artificial, do MIEIC, foi proposto obter a resolução para um problema. O problema a ser explorado é o N-Puzzle e consiste no movimento de peças até que estas estejam num determinado estado final.

II. Formulação do problema

Representação de estados:

Os estados são representados numa matriz de números. O número 0 representa um espaço vazio no qual qualquer peça pode ir. No entanto será usada como se fosse uma peça que pode ir para o lugar de outras e estas para o lugar dela. A posição do estado vazio é guardada para futuros cálculos. O tamanho que o mapa tem é representado pela variável n.

Estado inicial:

Depende do tabuleiro escolhido e da disposição das peças.

Operadores:

<direção>, representa o movimento que a peça 0 pode fazer, sendo 1(Top), 2(Left), 3(Bottom), 4(Right).

Pré-Condições:

A pré-condição para efetuar qualquer movimento é a peça escolhida estar dentro dos limites do mapa. Sendo que não pode ser menos que 0 ou maior ou igual a n.

Efeitos:

A peça com o número 0 ocupa o lugar da outra peça para a direção selecionada e essa peça passa a estar na posição antiga do número 0.

Custo:

Cada movimento feito tem um custo de 1.

Teste de objetivo:

Quando todas as peças estão por ordem crescente da esquerda para a direita, cima para baixo, sendo que a última peça, a mais a direito e mais em baixo, tem de ser o número 0.

Custo da solução:

O custo da solução é o número de movimentos total feito pela peça número 0 para o tabuleiro estar na posição final esperada.

III. Implementação do Problema

O problema foi implementado em C/C++, sendo que este recebe input de um ficheiro input.txt e faz output para a linha de comandos. Ao correr o programa este corre todos os algoritmos desejados e mostra os tempos de resolução, bem como o nós expandidos para chegar a solução. A frente é mostrada um sequencia de números correspondente ao que a peça número 0 precisa de fazer para chegar ao estado objetivo. Estes números correspondem aos operadores do problema.

O código fonte está localizado todo num ficheiro, com uma classe para representar informação sobre o estado do jogo e variáveis globais para guardar valores globais.

Existem várias funções, sendo a bfs, astar e greedy que correm os algoritmos.

A função move serve para fazer o movimento da peça número 0 no tabuleiro. A função checkFim verifica se o tabuleiro esta na posição final pretendida.

As funções que calculam as heurísticas do programa são: quantForaSitio que calcula a quantidade de peças que não estão na posição final pretendida; e a manhttanForaSitio calcula o somatório de todas as distancia Manhattan da pessoas que não estão na posição final pretendida.

IV. Algoritmos de pesquisa

Os algoritmos de pesquisa utilizados foram o Breadth-First Search A-Star e Greedy.

O algoritmo Breadth-First Search guarda todos os estados do programa numa fila bem como os estados já visitados. A fila é percorrida até não ter nenhum elemento ou até a solução ser encontrada.

O algoritmo A-Star guarda todos os estados do programa numa fila de prioridade ordenada segundo o custo (distancia percorrida + distancia que falta até ao objetivo). Também é guardada os nodes já percorridos para não de ter de os processar novamente.

O algoritmo Greedy é exatamente igual ao A-Star no entanto o custo é a distancia que falta até ao objetivo.

V. RESULTADOS

Os seguintes resultados foram obtidos:

Probl1: tamanho 3x3

Algoritmo	Nós Percorridos	Movimentos	Tempo(s)
Breadth-First Search	28	4	0.0005056
A-Star peças fora do sítio	4	4	0.0000978
A-Star Manhattan distance	4	4	0.0001085
Greedy peças fora do sítio	4	4	0.0000981
Greedy Manhattan distance	4	4	0.0000972

Probl2: tamanho 3x3

Algoritmo	Nós Percorridos	Movimentos	Tempo(s)
Breadth-First Search	114	7	0.0020449
A-Star peças fora do sítio	8	7	0.0001787
A-Star Manhattan distance	12	7	0.0002526
Greedy peças fora do sítio	8	7	0.0001696
Greedy Manhattan distance	8	7	0.0001691

Probl3: tamanho 3x3

Algoritmo	Nós Percorridos	Movimentos	Tempo(s)
Breadth-First Search	588	10	0.0113459
A-Star peças fora do sítio	37	10	0.0008828
A-Star Manhattan distance	20	10	0.0004385
Greedy peças fora do sítio	16	10	0.0003478
Greedy Manhattan distance	16	10	0.0003494

Probl4: tamanho 4x4

Algoritmo	Nós Percorridos	Movimentos	Tempo(s)
Breadth-First Search	3628	10	0.128804
A-Star peças fora do sítio	23	10	0.0018199
A-Star Manhattan distance	38	10	0.0010673
Greedy peças fora do sítio	442	10	0.0172477
Greedy Manhattan distance	442	10	0.0145482

Nos resultados obtidos o melhor algoritmo foi o A-Star tendo sido mais rápido quando o tamanho do tabuleiro aumentou.

REFERENCES

[1] Luis Paulo Reis, Apontamentos de Inteligencia Artificial 2018/2019

[2] Antti Laaksonen, Competitive Programmer's Handbook, Edição 3 de Julho, 2018.