UTFPR

Universidade Tecnológica Federal do Paraná
Mateus Tomoo Yonemoto Peixoto, Renan Kodama Rodrigues

Ciências da Computação – Universidade Tecnológica Federal do Paraná (UTFPR)

Caixa Postal 271 – 87301-899 – Campo Mourão – PR – Brasil
Departamento de Ciências da Computação – Universidade de Campo Mourão PR.

{mateus yonemoto} mateus_tomoo@hotmail.com,
{renan kodama} renan_kdm_rdg@hotmail.com.

Abstract. In this article, the goal is the behavior of the search algorithm. To solve the problem of an 8-piece puzzle in different board approaches, using two different heuristics, a distance from Manhattan and a distance from Canberra. After selecting an initial configuration of the board and a heuristic to be used, the results were analyzed in both results. It will also be approached as the main functions of the mentioned algorithm and its relation in relation to the performance with different heuristics.

Resumo. Neste artigo, será apresentado o comportamento do algoritmo de busca A*, para resolver o problema de um Puzzle de 8 peças em diferentes configurações iniciais do tabuleiro, usando duas heuristicas diferentes, a distância de Manhattan e a distância de Camberra. Após selecionada a configuração inicial do tabuleiro e a heuristica a ser usada iremos analisar ambos os resultados. Também será abordado as principais funções do algoritmo mencionado e a sua comparação em relação ao desempenho com diferentes heuristicas.

1. Informações Gerais

O algoritmo de busca A* foi implementado usando a linguagem Java, foi escolhida por trazer maior abstração e estruturas já implementadas, contribuindo assim para uma maior facilidade para a implementação. Para a execução do algoritmo é necessário informar a entrada em um arquivo *".txt"*, informando também qual heurística utilizar, dentre as implementadas, estão: distância de Manhattan e a distância de Canberra, levando em consideração a entrada como sendo a configuração inicial do Puzzle, e o objetivo como sendo a configuração de aceitação para o algoritmo, por padrão adotamos a confinguração final sendo o arquivo *"objetivo.txt"* contendo a matriz do puzzle (*{1,2,3}; {4,5,6};{7,8,0}).* Vale lembrar que o espaço em branco do puzzle é representado pelo valor "0" da matriz.

2. A*(A Star)

O objetivo do algoritmo A*(A Star) é buscar o caminho com o menor custo entre um vértice inicial e um final, através dos resultados das heuristicas, que acabam guiando a busca, convergindo para um resultado aceitável. É utilizado uma estrutura de pilha para armazenar os caminhos adjacentes ao estado atual, ou em outras palavras os caminhos adjacentes ao espaço em branco. Durante a execução se a pilha estiver vazia e não atingiu o estado objetivo então a busca não encontrou uma solução e é interrompida, caso a pilha tenha mais de um estado armazenado é realizado a verificação do custo para

todos estes estados, selecionando aquele que tiver o custo igual ou menor ao estado atual, tomando este como estado atual e adicionado à uma lista de visitados, para este novo nó, é realizado a sua expansão adicionando seus caminhos derivados novamente à pilha e o processo repetido, caso a configuração final do tabuleiro é atingida em algum momento da busca então o algoritmo será interrompido pois ele atingiu seu objetivo.

4. Heurísticas

A heurística é utilizada para “guiar” uma busca, informando se está próximo ou distânte do objetivo final, os algoritmos de busca baseiam-se nos resultados das heurísticas para determinar qual caminho seguir. Para todos os possíveis caminhos a partir de um estado atual é computado o seu custo, selecionando para o próximo estado aquele nó que apresentou o menor custo dentre todos os outros caminhos em relação ao estado atual. Pois no conceito de busca informada, a cada nó de menor custo escolhido, este estará mais perto da solução final. Foram utilizadas duas heuristicas para calcular a distância entre os estados, entre os métodos implementados estão, distância de Manhattan e distância de Canberra.

4.1 Distância Manhattan

É a distância entre dois pontos em um espaço euclidiano de coordenadas fixas, dada pela fórmula matemática d = |a1-b1| + |a2-b2|, e expressa na forma de algoritmo por:

```
for (int i = 0; i < tamanho; i++){
        posicaoAtual = mesaAtual.getposicaoMesa(i)
        destino = objetivo.getposicaoMesa(i)
        aux1 = Math.abs(posicaoAtual[0]-destino[0])
        aux2 = Math.abs(posicaoAtual[1]-destino[1])
        distancia = aux1+aux2
        somaDistancia += distancia;
}
```

4.3 Distância Canberra

É obtido através dos pares entre os vetores destino e origem em um espaço n-dimensional. Expressa na fórmula matemática d = (|ai - bi| / |ai| + |bi|), onde i vai de 0 até o número de dimensões, e expressa na forma de algoritmo por:

```
for ( int i = 0; i < tamanho; i++){
        posicaoAtual = mesaAtual.getposicaoMesa(i)
        destino = objetivo.getposicaoMesa(i)
        aux1 = Math.abs(posicaoAtual[0]-destino[0])
        aux2 = (Math.abs(posicaoAtual[1])
        aux3 = Math.abs(destino[1])+1)
        distancia = (aux1/aux2)+aux3
        somaDistancia += distancia;
}
```

5. Resultados

Foram utilizados três diferentes entradas sendo elas “teste1.txt”, “teste2.txt”, “teste3.txt” e “teste4.txt”, sendo este último arquivo impossível de se encontrar uma solução. Todas elas tem a mesma configuração final do tabuleiro expressa na forma matricial como {(1,2,3) ; (4,5,6) ; (7,8,0)}. As confugurações iniciais do tabuleiro expressas de forma matricial 3x3 podem ser vista a seguir:

teste1→ {(1,2,3); (0,4,6); (7,5,8)}
teste2→ {(2,3,1); (0,5,6); (4,7,8)}
teste3→ {(1,2,3); (8,7,6); (5,4,0)}
teste4→ {(1,0,3); (2,4,6); (7,5,8)}

5.1 Resultados Obtidos

Arquivo	Heuristica	Tempo de Execução	Nós Expandidos	Menor Caminho
“teste2.txt”	Manhattan	171ms	361	20
“teste2.txt”	Camberra	341ms	361	20
“teste3.txt”	Manhattan	135ms	104	19
“teste3.txt”	Camberra	44 ms	104	19

6. Conclusão

Após realizar o experimento podemos concluir que, dependendo da função heuristica o algoritmo A* apresenta diferentes custos, como no experimento realizado, ambas as heuristicas obtiveram bons e mals desempenhos usando o tempo como medida, porém o algoritmo sempre retornará o melhor caminho para se chegar no resultado, no experimento ambas heuristicas retonaram a mesma quantidade de nós que deverá ser expantido para atingir o objetivo.