Sistemas Inteligentes

Prof. Elder Rizzon Santos

Universidade Federal de Santa Catarina

Sistemas de Informação

Alunos: Bruno Alexandre Schmitt Costa (17105532), Lucas dos Santos Ferreira (20205390)

**Atividade Prática 1**

**Trabalho sobre Métodos de busca (2024/1)**

### 

### Principais Funções e Como Elas Funcionam no Algoritmo A\*

#### Função distancia\_manhattan(estado)

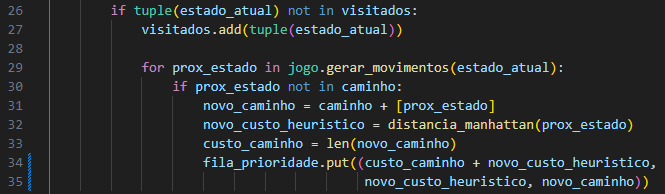
Esta função ajuda a gente a descobrir o quão longe estamos do final. Ela olha para cada peça no nosso quebra-cabeça e calcula a distância que ela precisa se mover para chegar ao lugar certo. No final, somamos todas essas distâncias e temos nossa estimativa de quão longe estamos do objetivo.

#### Função busca\_a\_estrela(estado\_inicial)

Usamos uma lista especial que guarda nossos estados para explorar. Essa lista é como uma fila de prioridade em um parque de diversões, onde os estados que parecem mais promissores (baseados na nossa estimativa da distância ao objetivo) são os primeiros a serem explorados. Vamos expandir cada estado, olhar para os estados que podemos alcançar a partir dele e adicionar esses novos estados na lista de espera, ordenados pelo quão promissores eles parecem ser.

### Como Gerenciamos Nossa Lista de Espera (Fronteira)

Na função busca\_a\_estrela, antes de colocar um novo estado na nossa lista de espera, verificamos algumas coisas:



1. Verificamos se já visitamos esse estado antes.
2. Se ainda não visitamos, marcamos ele como visitado.
3. Olhamos para todos os movimentos que podemos fazer a partir desse estado.
4. Verificamos se já passamos por esse estado para não ficarmos indo em círculos.
5. Calculamos o quão bom esse novo estado parece ser (com base na distância que calculamos antes) e colocamos ele na nossa lista de espera na posição certa.

### Explorando as Heurísticas

#### Heurística de Contagem de Peças Fora do Lugar (Mais Simples)

Esta heurística é bem direta. Basicamente, a gente olha para o nosso quebra-cabeça e conta quantas peças estão no lugar errado em relação ao que queremos alcançar. Quanto mais peças estiverem fora do lugar, mais difícil será chegar lá. No código, essa contagem é feita pela função calcular\_custo\_heuristico(estado). É uma maneira simples, mas ainda assim eficiente, de dar uma ideia de quão longe estamos do final.

#### Heurística de Distância de Manhattan (Mais Avançada)

Essa heurística nos permite calcular a distância que cada peça precisa percorrer para chegar à sua posição correta no tabuleiro objetivo, o que nos dá uma estimativa clara de quanto ainda falta para completar o quebra-cabeça.

No código que implementamos, essa distância é calculada pela função distancia\_manhattan(estado). Esta função é essencial para nos ajudar a encontrar o caminho certo de forma mais rápida, especialmente quando estamos lidando com quebra-cabeças mais complexos.

A distância é calculada para cada peça individualmente e, para cada peça, o valor da distância pode variar de 0 a 4, dependendo de sua posição atual em relação ao estado objetivo.

Para desenvolver nossa implementação da heurística de Distância de Manhattan, nos baseamos em um estudo realizado na Universidade Federal do Mato Grosso do Sul. Este estudo, orientado pelo Professor Edson Takashi Matsubara e com contribuições de José Augusto Lajo Vieira Vital, forneceu um alicerce teórico sólido para a nossa abordagem. O trabalho, intitulado "Implementação de diferentes métodos para a resolução do jogo 8Puzzle utilizando busca heurística", oferece uma explanação detalhada das diferentes abordagens heurísticas e métodos de busca utilizados para resolver o 8Puzzle.

Se você estiver interessado em explorar mais sobre este estudo e ver o código completo, ele está disponível no seguinte repositório: 8Puzzle-A-Star.

#### Comparação de Faixa de Valores e Precisão

Para a heurística simples, a faixa de valores varia de 0 a 9, pois o número máximo de peças fora do lugar é 9 (um para cada peça, excluindo o espaço vazio).

Para a heurística de Manhattan, a faixa de valores depende da configuração específica do estado, mas geralmente será maior do que a heurística simples, especialmente para estados mais complexos. A precisão desta heurística tende a ser maior, pois leva em consideração a distância real que cada peça precisa percorrer para alcançar sua posição correta.

### Breve Análise do Desempenho da Implementação

| Fácil  1 2 3  4 8 7  0 6 5 | Médio 4 8 6  3 1 2  0 7 5 | Difícil  3 7 8  0 5 1  4 2 6 |
| --- | --- | --- |

| **Cenário** | **Algoritmo** | **Tamanho do Caminho** | **Total de Nodos Visitados** | **Tempo de Execução (segundos)** |
| --- | --- | --- | --- | --- |
| Fácil | Custo Uniforme | 13 | 1459 | 0,018000 |
| Fácil | Heurística Simples | 13 | 382 | 0,005001 |
| Fácil | Heurística de Manhattan | 13 | 40 | 0,000000 |
| Médio | Custo Uniforme | 23 | 88461 | 1,440105 |
| Médio | Heurística Simples | 25 | 15731 | 0,238018 |
| Médio | Heurística de Manhattan | 23 | 511 | 0,019999 |
| Difícil | Custo Uniforme | 28 | 176970 | 3,068224 |
| Difícil | Heurística Simples | 30 | 91840 | 1,643120 |
| Difícil | Heurística de Manhattan | 28 | 2755 | 0,046003 |

A heurística de Manhattan se mostrou superior em eficiência e velocidade para resolver o problema do quebra-cabeça deslizante em comparação com o custo uniforme e a heurística simples. Embora o tamanho do caminho tenha ficado ligeiramente maior em dois casos específicos com a heurística simples, essa ocorrência não é comum e pode ser um resultado das particularidades dos estados iniciais escolhidos para esses cenários. Independentemente disso, a eficiência geral e a rapidez da heurística de Manhattan a tornam a escolha preferida para resolver esse tipo de problema.