# Comparação de Algoritmos de Busca para o Quebra-cabeça dos 8 Números

Davi Emannoel Lopes de souza[1]
[1]Universidade Interdimensional Tuiuti do Paraná
Curitiba – PR
{davi.souza}@utp.edu..br

**Resumo**

Este trabalho apresenta uma análise comparativa de diferentes algoritmos de busca aplicados ao problema do quebra-cabeça dos 8 números (8-puzzle). Foram implementados e avaliados algoritmos de busca cega (Busca em Largura e Busca em Profundidade) e algoritmos de busca heurística (Busca Gulosa e A*). Os resultados obtidos demonstram que o algoritmo A* se destaca por combinar eficiência computacional e optimalidade das soluções, expandindo significativamente menos nós que os demais algoritmos enquanto mantém a garantia de soluções ótimas. A Busca Gulosa apresentou excelente desempenho em termos de eficiência, com qualidade próxima da ótima nas soluções, enquanto o DFS mostrou-se o menos eficaz para este problema, com menor taxa de sucesso e caminhos significativamente mais longos, foi utilizado inteligencia artificial para auxilio do código e organização do documento de relatório. [1].

## 1 Introdução

O quebra-cabeça dos 8 números (8-puzzle) é um problema clássico em Inteligência Artificial que consiste em organizar 8 peças numeradas em uma grade 3x3 com um espaço vazio. Este problema serve como um excelente caso de estudo para comparar diferentes estratégias de busca, pois possui um espaço de estados finito e bem definido, mas suficientemente complexo para desafiar algoritmos de busca.

Este relatório apresenta uma análise comparativa de algoritmos de busca aplicados ao problema do quebra-cabeça dos 8 números, incluindo algoritmos de busca cega e heurística. Os algoritmos analisados são:

- Busca Cega:
  - Busca em Largura (BFS)
  - Busca em Profundidade (DFS)
- Busca Heurística:
  - Busca Gulosa (Best-First Search)
  - Algoritmo A*

O objetivo principal deste estudo é analisar comparativamente o desempenho destes algoritmos em termos de eficiência computacional (tempo de execução e uso de memória) e qualidade das soluções encontradas.

# 2 Fundamentação Teórica

## 2.1 O Problema do Quebra-cabeça dos 8 Números

O 8-puzzle consiste em uma grade 3x3 contendo 8 peças numeradas (de 1 a 8) e um espaço vazio. O objetivo é reorganizar as peças a partir de uma configuração inicial para alcançar uma configuração objetivo, geralmente com os números em ordem crescente e o espaço vazio no canto inferior direito:

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 0 |

Onde 0 representa o espaço vazio.

Os movimentos permitidos são o deslizamento de peças adjacentes para o espaço vazio. A cada movimento, pode-se mover apenas uma peça que esteja adjacente (horizontal ou verticalmente) ao espaço vazio.

## 2.2 Algoritmos de Busca

### 2.2.1 Busca em Largura (BFS)

A busca em largura explora todos os nós de um determinado nível antes de passar para o próximo nível. Este algoritmo garante encontrar o caminho mais curto para a solução, mas pode consumir uma grande quantidade de memória ao armazenar todos os nós de cada nível.

### 2.2.2 Busca em Profundidade (DFS)

A busca em profundidade explora um caminho até o fim antes de retroceder e explorar caminhos alternativos. Este algoritmo é mais eficiente em termos de uso de memória, mas não garante encontrar a solução ótima e pode ficar preso em caminhos muito longos ou ciclos. Para evitar esse problema, geralmente é implementado com um limite de profundidade.

### 2.2.3 Busca Gulosa

A busca gulosa utiliza uma função heurística para estimar a proximidade de cada estado ao estado objetivo. A cada passo, o algoritmo expande o nó que parece estar mais próximo do objetivo, seguindo uma estratégia "gananciosa". Este algoritmo não considera o custo do caminho até o nó atual, focando apenas na estimativa do custo restante.

#### 2.2.4 Algoritmo A*

O algoritmo A* combina o custo do caminho já percorrido ($g(n)$) com uma estimativa heurística do custo restante até o objetivo ($h(n)$). A função de avaliação $f(n) = g(n) + h(n)$ é usada para selecionar o próximo nó a ser expandido. Se a heurística utilizada for admissível (nunca superestima o custo real), o A* garante encontrar a solução ótima.

### 2.3 Heurísticas para o 8-Puzzle

#### 2.3.1 Distância de Hamming

A distância de Hamming conta o número de peças que estão fora da posição correta, excluindo o espaço vazio. Esta heurística é admissível, mas não muito informativa.

#### 2.3.2 Distância de Manhattan

A distância de Manhattan soma as distâncias de cada peça até sua posição correta, medida pelo número de movimentos horizontais e verticais necessários. Esta heurística é admissível e mais informativa que a distância de Hamming.

## 3 Implementação

### 3.1 Representação do Problema

O quebra-cabeça foi representado utilizando uma matriz 3x3, onde o valor 0 indica o espaço vazio. Cada estado do quebra-cabeça é encapsulado em uma classe `Puzzle` que contém:

- O estado atual (matriz 3x3)
- O nó pai na árvore de busca
- A ação que levou ao estado atual
- O custo do caminho até o estado atual
- A posição do espaço vazio

### 3.2 Implementação dos Algoritmos

A implementação dos algoritmos seguiu as seguintes estruturas:

- BFS: Utiliza uma fila FIFO (First-In-First-Out) para armazenar os nós de fronteira.
- DFS: Utiliza uma pilha LIFO (Last-In-First-Out) com limitação de profundidade.
- Busca Gulosa: Utiliza uma fila de prioridade baseada apenas na função heurística.
- A*: Utiliza uma fila de prioridade baseada na soma do custo do caminho e da função heurística.

### 3.3 Métricas de Avaliação

Para cada algoritmo, as seguintes métricas foram coletadas:

1. Tempo de execução: Medido em segundos.
2. Uso de memória: Diferença entre a memória utilizada no início e no fim da execução (em MB).
3. Nós expandidos: Número de nós visitados durante a busca.
4. Comprimento do caminho: Número de movimentos necessários para alcançar a solução.
5. Taxa de sucesso: Porcentagem de instâncias para as quais o algoritmo encontrou uma solução.

## 4 Experimentos e Resultados

### 4.1 Configuração dos Experimentos

Os experimentos foram realizados utilizando múltiplas instâncias do quebra-cabeça, carregadas a partir do arquivo "ed02-puzzle8.csv". Cada instância representa uma configuração inicial diferente, variando em dificuldade (medida pela distância até a solução).

O objetivo foi sempre o mesmo: a configuração ordenada com o espaço vazio no canto inferior direito.

### 4.2 Resultados

Os resultados obtidos na execução dos algoritmos são apresentados na tabela a seguir. Para cada algoritmo, são mostradas as médias das métricas coletadas nas instâncias onde a solução foi encontrada.

Tabela 1: Comparação do desempenho dos algoritmos

| **Algoritmo** | **%Soluções** | **Pass Méd.** | **Nós Exp.Méd.** | **Temp Méd.(s)** | **Memo Méd.(MB)** |
|---|---|---|---|---|---|
| BFS | 100.00% | 6.90 | 356.10 | 0.18914 | 37.09 |
| DFS | 40.00% | 22.67 | 26606.25 | 7.95649 | 36.89 |
| Gulosa | 100.00% | 7.70 | 37.40 | 0.01621 | 37.67 |
| A* | 100.00% | 6.90 | 16.10 | 0.00700 | 37.67 |

### 4.3 Análise Comparativa

#### 4.3.1 Qualidade da Solução

Como esperado, o BFS e o A* encontraram as soluções ótimas (menor número de passos), com média de 6.90 movimentos. A Busca Gulosa gerou soluções próximas da ótima, com 7.70 movimentos em média, representando um pequeno desvio de apenas 11.6% em relação à solução ótima. O DFS, como previsto, produziu os caminhos mais longos, com 22.67 movimentos em média, mais que o triplo da solução ótima.

#### 4.3.2 Eficiência Computacional

Em termos de nós expandidos, o A* foi excepcionalmente eficiente, expandindo uma média de apenas 16.10 nós, seguido pela Busca Gulosa (37.40 nós) - ambos muito econômicos. O BFS expandiu significativamente mais nós (356.10), mas ainda manteve um desempenho razoável. O DFS apresentou um comportamento muito menos eficiente, expandindo uma média de 26606.25 nós, cerca de 1651 vezes mais nós que o A*, o que explica seu alto tempo de execução.

#### 4.3.3 Tempo de Execução

O A* foi o algoritmo mais rápido por ampla margem, com tempo médio de apenas 0.00700 segundos, seguido pela Busca Gulosa (0.01621s) - ambos extremamente rápidos. O BFS apresentou um tempo médio de 0.18914s, ainda aceitável. O DFS foi significativamente mais lento (7.95649s), cerca de 1136 vezes mais lento que o A*, consistente com o extraordinário número de nós expandidos.

#### 4.3.4 Uso de Memória

Todos os algoritmos apresentaram consumo de memória semelhante, variando entre 36.89 MB (DFS) e 37.67 MB (A* e Busca Gulosa), com o BFS ligeiramente no meio (37.09 MB). Esta diferença mínima sugere que, para o tamanho do problema abordado, o consumo de memória base da implementação domina em relação às diferenças teóricas de consumo de memória entre os algoritmos.

#### 4.3.5 Taxa de Sucesso

O BFS, A* e Busca Gulosa obtiveram 100% de taxa de sucesso, encontrando soluções para todas as instâncias testadas. Em contraste, o DFS conseguiu resolver apenas 40% das instâncias, reforçando sua inadequação para este tipo de problema quando implementado com limitação de profundidade.

# 5 Discussão

## 5.1 Comparação dos Algoritmos

A análise dos resultados revela padrões que confirmam e, em alguns casos, excedem as expectativas teóricas:

1. BFS vs. DFS:
    - O BFS garantiu soluções ótimas em todas as instâncias, com um custo computacional razoável.
    - O DFS apresentou um desempenho significativamente inferior, falhando em 60% das instâncias e produzindo caminhos muito mais longos quando bem-sucedido. Sua vantagem teórica em economia de memória não se mostrou significativa no contexto deste problema.
2. Busca Heurística vs. Busca Cega:

- Os algoritmos heurísticos (A* e Busca Gulosa) foram dramaticamente mais eficientes em termos de nós expandidos e tempo de execução.
- A Busca Gulosa foi extremamente rápida e, surpreendentemente, produziu soluções muito próximas da ótima.
- O A* combinou o melhor dos dois mundos: eficiência excepcional e garantia de optimalidade.

3. A* vs. Demais:
   - O A* destacou-se como o algoritmo mais equilibrado e eficiente, combinando a optimalidade do BFS com uma eficiência computacional superior a todos os outros algoritmos.
   - Sua capacidade de expandir apenas 16.10 nós em média demonstra o poder da combinação inteligente do custo do caminho com a estimativa heurística.

## 5.2 Impacto da Heurística

A distância de Manhattan provou ser uma heurística excepcionalmente eficaz para o problema do quebra-cabeça dos 8 números, proporcionando uma estimativa precisa da distância até o objetivo. Isso permitiu que o A* e a Busca Gulosa direcionassem a busca de forma extremamente eficiente, com o A* expandindo 22 vezes menos nós que o BFS, apesar de ambos garantirem soluções ótimas.

A admissibilidade da heurística de Manhattan garantiu que o A* encontrasse soluções ótimas, enquanto sua informatividade permitiu uma redução dramática no número de nós expandidos em comparação com as buscas cegas.

## 5.3 Limitações e Considerações

Algumas limitações e considerações importantes sobre os resultados:

1. Limite de profundidade no DFS: O limite de profundidade imposto ao DFS influenciou sua baixa taxa de sucesso (40%) e a qualidade inferior das soluções encontradas.

2. Complexidade das instâncias: As instâncias testadas parecem ter uma complexidade moderada (média de 6.90 passos na solução ótima), o que permitiu que os algoritmos heurísticos demonstrassem pleno potencial.

3. Uso de memória consistente: A similaridade no uso de memória entre os algoritmos sugere que, para problemas desta escala, o consumo base da implementação domina sobre as diferenças teóricas de comportamento em memória.

4. Desempenho excepcional da Busca Gulosa: A pequena diferença entre as soluções da Busca Gulosa e as soluções ótimas sugere que, para este problema específico, a heurística de Manhattan é particularmente bem-alinhada com o caminho de custo mínimo.

# 6 Conclusão

Este estudo comparou o desempenho de quatro algoritmos de busca (BFS, DFS, Busca Gulosa e A*) aplicados ao problema do quebra-cabeça dos 8 números. Os resultados não apenas confirmaram mas excederam as expectativas teóricas, demonstrando que:

1. O algoritmo A* destacou-se como a estratégia claramente superior, combinando garantia de soluções ótimas com um desempenho extraordinário em termos de nós expandidos (16.10 em média) e tempo de execução (0.00700s).

2. A Busca Gulosa ofereceu um excelente equilíbrio entre eficiência e qualidade, expandindo poucos nós (37.40) e produzindo soluções muito próximas da ótima (apenas 11.6% mais longas), sendo uma alternativa viável quando a garantia de optimalidade não é crítica.

3. O BFS garantiu soluções ótimas com um desempenho razoável, expandindo uma média de 356.10 nós.

4. O DFS mostrou-se inadequado para este problema, com baixa taxa de sucesso (40%), soluções de qualidade muito inferior (3.3 vezes mais longas) e um custo computacional excepcionalmente alto (26606.25 nós expandidos e 7.95649s de tempo médio).

Estes resultados enfatizam a importância crucial da escolha adequada do algoritmo de busca conforme as necessidades específicas de cada aplicação, e destacam o valor extraordinário das heurísticas bem projetadas na resolução eficiente de problemas combinatórios.

## 6.1 Trabalhos Futuros

Para trabalhos futuros, sugerimos:

1. Estender a análise para o quebra-cabeça dos 15 números (15-puzzle), que apresenta um espaço de estados muito maior e pode desafiar até mesmo os algoritmos heurísticos.

2. Investigar outras heurísticas, como o número de conflitos lineares ou padrões de banco de dados, que poderiam melhorar ainda mais o desempenho do A*.

3. Implementar e avaliar variantes dos algoritmos, como IDA* (Iterative Deepening A*) ou SMA* (Simplified Memory-Bounded A*), que poderiam reduzir o consumo de memória.

4. Analisar o impacto de diferentes estruturas de dados e otimizações de implementação no desempenho dos algoritmos, especialmente para problemas de maior escala.

5. Investigar por que o DFS teve um desempenho tão inferior, identificando possíveis otimizações ou alternativas como busca bidirecional ou algoritmos de retrocesso inteligente.

# Referências

[1] Russell, S. J., & Norvig, P. (2020). Artificial Intelligence: A Modern Approach (4th ed.). Pearson.

[2] Korf, R. E. (1985). Depth-first Iterative-Deepening: An Optimal Admissible Tree Search. Artificial Intelligence, 27(1), 97-109.

[3] Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics, 4(2), 100-107.

[4] Korf, R. E., & Taylor, L. A. (1996). Finding Optimal Solutions to the Twenty-Four Puzzle. Proceedings of the Thirteenth National Conference on Artificial Intelligence, 1202-1207.