Centro Federal de Educação Tecnológica de Minas Gerais

Engenharia da Computação
Disciplina: Inteligência Artificial

Trabalho 01:

Busca no Labirinto

Alunos: Eduardo Henrique Queiroz Almeida e Joaquim Cézar Santana da Cruz
Professor: Tiago Alves de Oliveira

Divinópolis/MG
outubro de 2025

Sumário

1 Fundamentação teórica

Esta seção estabelece os conceitos fundamentais sobre os algoritmos de busca em espaço de estados empregados neste trabalho. O problema de encontrar um caminho em um labirinto é um exemplo clássico para a aplicação e comparação dessas estratégias. Os algoritmos são divididos em duas categorias principais: buscas não informadas (cegas), que exploram o labirinto sistematicamente, e buscas informadas (heurísticas), que utilizam conhecimentoadicional sobree o problema para guiar a exploração de forma mais eficiente.

Buscas não informadas

As buscas não informadas, ou cegas, operam sem qualquer conhecimento sobre a localização do estado objetivo. Elas não sabem se o movimento as aproxima ou afasta do objetivo; sua única estratégia é explorar o espaço de estados de forma sistemática até que o objetivo seja encontrado ou o espaço destadosos se esgote.

A ideia por trás das buscas cegas tem origem no *British Museum Algorithm(Algoritmomo do Museu Britânico)*, este não sendo um algoritmo prático, mas sim uma metáfora para uma busca por força bruta, em que cada possibilidade é testada uma a uma, começando com aquelas que envolvem menos passos. A ideia central do algoritmo é que dado um tempo suficiente, um macaco digitando aleatoriamente em uma máquina de escrever, conseguiria eventualmente escrever todos os livros do museu britânico [1].

A ineficiência de tal método pode ser justificada pelo não uso de memória para evitar ciclos ou caminhos redundantes. Ele serve para ilustrar como algoritmos sistemáticos de busca cega, como o BFS e o DFS, são necessários. Embora cegos, esses métodos são fundamentalmente mais eficientes porque usam memória (listas de visitados) e uma estratégia de exploração ordenada (fila ou pilha) para garantir que o espaço seja explorado de forma completa, sem se perder em loops infinitos.

Busca em Largura

A Busca em largura (*BFS* - Breadth-First Search) é uma estratégia que explora o espaço de estados em "camadas". A partir do nó inicial, ela visita primeiro todos os seus vizinhos diretos (nível 1). Em seguida, visita todos os vizinhos dos nós do nível 1 (nível 2), e assim sucessivamente. A figura 1 representa o comportamento do algoritmo em uma árvore de busca.

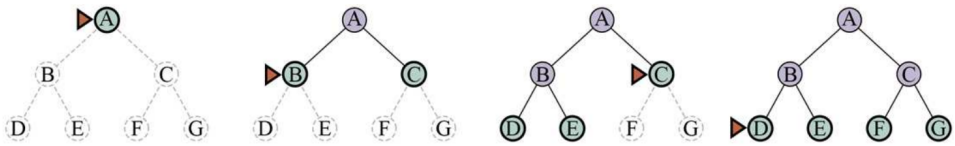

Figura 1: Exemplo de ordem de busca para o algoritmo BFS. Sua ordem de pesquisa é dada por níveis, pesquisando primeiro no primeiro nível e seguindo para níveis mais profundos [2].

A Busca em Largura possui as seguintes características:

- **Estrutura de dados utilizada:** Para implementar essa estratégia de exploração em níveis, o BFS utiliza uma *fila* (Queue) como estrutura de dados para sua fron-

teira. A natureza *FIFO* (First-In, First-Out) da fila garante que os nós sejam explorados na ordem exata em que foram descobertos, camada por camada.

- **Análise de Desempenho:**
  - **Tempo:** A complexidade de tempo do algoritmo é $O(b^d)$, onde b é o fator de ramificação - número máximo de vizinhos de um nó - e d é a profundidade da solução.
  - **Espaço:** A complexidade de espaço também é $O(b^d)$, pois a fronteira precisa armazenas todos os nós de um nível antes de passar para o próximo.
  - **Completude:** O *BFS* é completo para todo problema em que o fator de ramificação é finito.
  - **Optimalidade:** O *BFS* é ótimo se o custo de todas as ações for uniforme.

Busca em Profundidade

A busca em profundidade (*DFS* - Depth-First Search) explora um ramo da árvore de busca o mais fundo possível, até atingir um nó terminal (folha) ou o objetivo. Caso não encontre o objetivo, ela retorna para o último nó onde havia uma escolha e explora o próximo ramo disponível. A figura 2 representa o comportamento do algoritmo em uma árvore de busca.

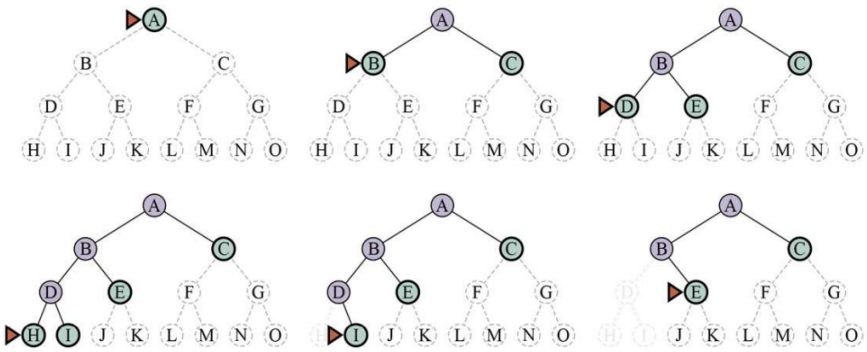

Figura 2: Exemplo de ordem d busca para o algoritmo *DFS*. Sua ordem de pesquisa é dada por profundidade, pesquisando até o nó folha e depois, caso não ache o objetivo, faz um backtraking [2].

A busca em profundidade possui as seguintes características:

- **Estrutura de dados utilizada:** Para implementar essa estraégia de busca ao nó mais profundo, o *DFS*d utiliza uma pilha (Stack) como estrutura de dados para a fronteira. A natureza *LIFO* (Last-In, First-Out) da pilha garante que o último nó descoberto seja o primeiro a ser explorado.
- **Análise de Desempenho:**
  - **Tempo:** A complexidade de tempo é $O(b^m)$, onde b é o fator de ramificação e m é a profundidade máxima do espaço de estados. Em um labirinto com caminhos longos, isso pode ser significativamente maior que o d do BFS.

  - **Espaço:** Esta é a grande vantagem do DFS. Sua complexidade de espaço é de apenas $O(b \cdot m)$, pois ele só precisa armazenar o caminho atual que está explorando.
  - **Completude:** O DFS é **completo** em espaços de estados finitos e sem ciclos (ou com detecção de visitados, como em nossa implementação).
  - **Optimalidade:** O DFS **não é ótimo**. Ele encontrará o primeiro caminho que descobrir, que raramente é o caminho mais curto.

Buscas Informadas

As buscas informadas, ao contrário das cegas, utilizam "conhecimento"adicional sobre o problema para tomar decisões mais inteligentes. Esse conhecimento é encapsulado em uma **função heurística**, $h(n)$, que estima o "custo"do caminho do nó n até o objetivo. A ideia é explorar primeiro os nós que *parecem* estar mais próximos da solução.

Heurística de Manhattan

Para que uma busca informada funcione, ela precisa de uma heurística. Neste trabalho, a **Distância de Manhattan** foi uma das heurísticas escolhidas.

- **Definição e Fórmula:** A Distância de Manhattan entre dois pontos (x_1, y_1) e (x_2, y_2) em um grid é a soma das diferenças absolutas de suas coordenadas [3]:

$$h(n) = |x_n - x_{\text{objetivo}}| + |y_n - y_{\text{objetivo}}| \tag{1}$$

- **Justificativa:** Esta heurística é ideal para o problema, pois o labirinto permite movimentos apenas em quatro direções (Norte, Sul, Leste, Oeste). A Distância de Manhattan representa exatamente o número mínimo de passos para ir de um ponto a outro se não houvesse paredes.

- **Propriedades Fundamentais:** A qualidade de uma heurística é definida por suas propriedades, que são cruciais para a avaliação:

  - **Admissibilidade:** Uma heurística é *admissível* se ela nunca superestima o custo real para alcançar o objetivo. A Distância de Manhattan é admissível, pois o caminho real (com paredes) será sempre igual ou maior que a distância em linha reta (sem paredes) que ela calcula.
  - **Consistência:** Uma heurística é *consistente* se o custo estimado de um nó n até o objetivo nunca é maior que o custo de se mover para um vizinho n' mais o custo estimado de n' até o objetivo. A Distância de Manhattan também é consistente.

  A admissibilidade é a propriedade mais crucial para este trabalho, pois é o requisito que garante a optimalidade do algoritmo A* [3].

Heurística de Euclidiana

A **Distância Euclidiana** foi a outra heurística sugerida para este trabalho.

- **Definição e Fórmula:** A Distância Euclidiana é a distância em linha reta ("o menor caminho entre dois pontos") no plano, calculada pelo Teorema de Pitágoras:

$$h(n) = \sqrt{(x_n - x_{\text{objetivo}})^2 + (y_n - y_{\text{objetivo}})^2} \tag{2}$$

- **Justificativa:** Esta heurística representa a distância física real entre os dois pontos. Embora não reflita o custo de movimento em um grid de 4 direções (como a Manhattan), ela é uma estimativa válida e comumente usada em problemas que permitem movimento diagonal (8 direções) ou em espaços contínuos.

- **Propriedades Fundamentais:**
  - **Admissibilidade:** A Distância Euclidiana é *admissível*. O caminho mais curto possível entre dois pontos é, por definição, a linha reta. Qualquer caminho real em um labirinto (restrito a N/S/L/O) será sempre igual ou maior que este valor.
  - **Consistência:** A heurística Euclidiana também é *consistente*, pois a distância em linha reta sempre obedece à desigualdade triangular.

  Por ser admissível, o uso da Distância Euclidiana também garante a optimalidade do A*. No entanto, para este problema de 4 direções, ela geralmente subestima mais o custo real do que a Distância de Manhattan, podendo resultar em uma busca menos eficiente (mais nós expandidos).

Busca Gulosa

A Busca Gulosa pelo melhor caminho (Greedy Best-First Search) é a implementação mais simples de uma busca informada. Sua estratégia é "míope": ela sempre escolhe expandir o nó que *parece* estar mais perto do objetivo, ignorando o custo que já levou para chegar até ali. A sua função de avaliação é $f(n) = h(n)$.

- **Estrutura de Dados:** Utiliza uma fila de prioridade (*Priority Queue*, ordenada crescentemente pelo valor da heurística $h(n)$.

- **Análise de Desempenho:**
  - **Tempo e Espaço:** A complexidade é $O(b^m)$ no pior caso, mas na prática, com uma boa heurística, ela é muito mais rápida.
  - **Completude:** Não é completa. Pode entrar em loops se não houver detecção de visitados e pode se perder em "armadilhas"da heurística.
  - **Optimalidade:** Não é ótima. Por ser gulosa, ela pode escolher um caminho que parece promissor no início, mas que se revela um caminho longo e subótimo no final.

Busca A*

O A* é amplamente considerado o algoritmo de busca informada mais eficiente e robusto. Ele combina a eficiência da Busca Gulosa com a garantia de otimalidadee do Custo Uniforme.

O A* avalia os nós na fronteira usando uma função que soma o custo real do caminho percorrido desde o início, $g(n)$, com a estimativa heurística do custo restante até o objetivo, $h(n)$:

$$f(n) = g(n) + h(n)$$

- **Estrutura de Dados:** Utiliza uma Fila de Prioridade (Priority Queue), ordenada crescentemente pelo valor da função $f(n)$.
- **Análise de Desempenho:**
  - **Tempo e Espaço:** A complexidade depende da qualidade da heurística. No pior caso, é $O(b^d)$, mas com uma boa heurística (como a Distância de Manhattan), ela é dramaticamente mais eficiente, pois "poda"grandes partes irrelevantes da árvore de busca.
  - **Completude:** O A* é completo.
  - **Optimalidade:** O A* é ótimo, desde que a heurística utilizada seja admissível.

2 Metodologia Aplicada

Esta seção descreve a metodologia empregada para a implementação e avaliação dos algoritmos de busca.

Estrutura do Projeto

O projeto foi estruturado dividindo as tarefas em em diferentes arquivos. A linguagem de programação utilizada foi Python.

- **src/maze.py:** Contém a classe `Maze`, que define o problema.
- **src/heuristics.py:** Implementa a função de heurística (Distância de Manhattan), que é utilizada pelos algoritmos de busca informada.
- **src/search.py:** Contém as implementações de todos os algoritmos de busca (BFS, DFS, A*, Gulosa). Cada algoritmo é uma função independente que recebe uma instância da classe `Maze` e retorna a tupla `(caminho, métricas)`. As estruturas de dados da fronteira (Fila, Pilha, Fila de Prioridade) são gerenciadas internamente em cada função.
- **run_search.py:** Script principal que orquestra os experimentos, lendo os mapas do arquivo de entrada, executando todos os algoritmos de busca em sequência e salvando os resultados consolidados em um arquivo de relatório.

Métricas de Desempenho

Para avaliar e comparar os algoritmos, foram coletadas as métricas de desempenho especificadas no escopo do trabalho. Cada função em `search.py` foi instrumentada para rastrear e retornar as seguintes métricas:

- **Tempo de Execução:** Medido em segundos, utilizando a biblioteca `time` do Python, para cronometrar o tempo total desde a chamada da função de busca até o retorno da solução.
- **Memória:** Medida conforme a definição específica do trabalho. Durante a execução, é retreado o tamanho da fronteira (fila/pilha/heap) e o número de nós no conjunto de "visitados"ou "explorados". A métrica registrada é o valor máximo que a soma `(tamanho_fronteira + tamanho_explorados)` atingiu durante a busca.
- **Nós Expandidos:** Um contador que é incrementado a cada vez que um nó é removido da fronteira para ter seus vizinhos avaliados.
- **Completude:** Avaliada de forma binária, verificando se o algoritmo foi capaz de encontrar uma solução nos mapas onde ela existe.
- **Optimalidade:** Avaliada se a solução encontrada é o menor caminho possível.

3 Análise Comparativa

Para a análise de desempenho, foi selecionado o Mapa 10 (criado a partir do labirinto 9), um labirinto de dimensões 30x30 com uma estrutura complexa. Esta escolha se justifica por ser um cenário desafiador, ideal para evidenciar as diferenças de desempenho e optimalidade entre os algoritmos, bem como o impacto da escolha da heurística nas buscas informadas. Os resultados completos da execução para este mapa estão consolidados na Tabela 1.

Tabela 1: Resultados dos algoritmos de busca para o Mapa 10.

Algoritmo	**Custo (Passos)**	**Nós Expandidos**	**Memória (Nós)**	**Tempo (s)**
DFS	120	271	355	0.000893
BFS	60	403	414	0.001281
A* (Manhattan)	60	237	268	0.001060
A* (Euclidiana)	60	274	299	0.001203
Greedy (Manhattan)	68	110	187	0.000504
Greedy (Euclidiana)	62	70	149	0.000430

Análise das Buscas Não Informadas (DFS vs. BFS)

- **Optimalidade e Custo:** A superioridade do BFS em garantir o caminho ótimo ficou evidente. Ele encontrou a solução com custo 60, enquanto o DFS, ao seguir um caminho profundo sem critério de custo, encontrou uma rota com o dobro do tamanho (custo 120). Isso reafirma que o DFS não é ótimo e pode entregar soluções ineficientes em mapas complexos.

- **Eficiência (Tempo e Nós Expandidos):** Embora tenha encontrado um caminho muito pior, o DFS foi mais rápido e expandiu menos nós (271) que o BFS (403). O BFS precisou explorar uma área muito maior do labirinto para garantir a optimalidade, resultando em maior consumo de tempo e processamento.

- **Uso de Memória:** O BFS apresentou o maior pico de uso de memória entre todos os algoritmos (414 nós), consequência de sua necessidade de armazenar uma fronteira de nós muito extensa. O DFS, por sua vez, consumiu menos memória (355 nós), pois armazena apenas o ramo atual da busca.

Análise das Buscas Informadas e o Impacto da Heurística

- **Optimalidade (A* vs. Greedy):** A diferença fundamental entre as estratégias informadas foi exposta. Ambas as versões do **A*** encontraram o caminho ótimo de custo 60, confirmando sua robustez. Em contrapartida, nenhuma das buscas **Greedy** alcançou a optimalidade. A versão com heurística Euclidiana chegou mais perto (custo 62), e a com Manhattan ficou ainda mais distante (custo 68). Este resultado demonstra a principal fraqueza da busca gulosa: sua estratégia "míope", focada apenas em $h(n)$, a levou a caminhos subótimos, enquanto o A*, ao balancear o custo real percorrido ($g(n)$) com a heurística, manteve-se no caminho certo.

- **Eficiência e a Escolha da Heurística:** As buscas gulosas foram as mais rápidas e as que menos expandiram nós, com a versão **Euclidiana** se destacando como a mais eficiente de todas (70 nós expandidos e 0.000430s). Isso mostra o poder da heurística em direcionar a busca agressivamente para o objetivo. Entre as versões do A*, a que utilizou a distância de **Manhattan** foi superior, expandindo menos nós (237 vs. 274) e sendo mais rápida. Isso sugere que, para um labirinto com movimentos restritos a 4 direções, a heurística de Manhattan é mais informativa e consistente, guiando o A* de forma mais eficaz e evitando explorações desnecessárias.

- **Uso de Memória:** A busca gulosa com a heurística Euclidiana foi a mais econômica em memória (149 nós). As versões do A* consumiram mais memória que as buscas gulosas, mas significativamente menos que o BFS, demonstrando como a heurística ajuda a "podar"a árvore de busca e a manter a fronteira mais controlada.

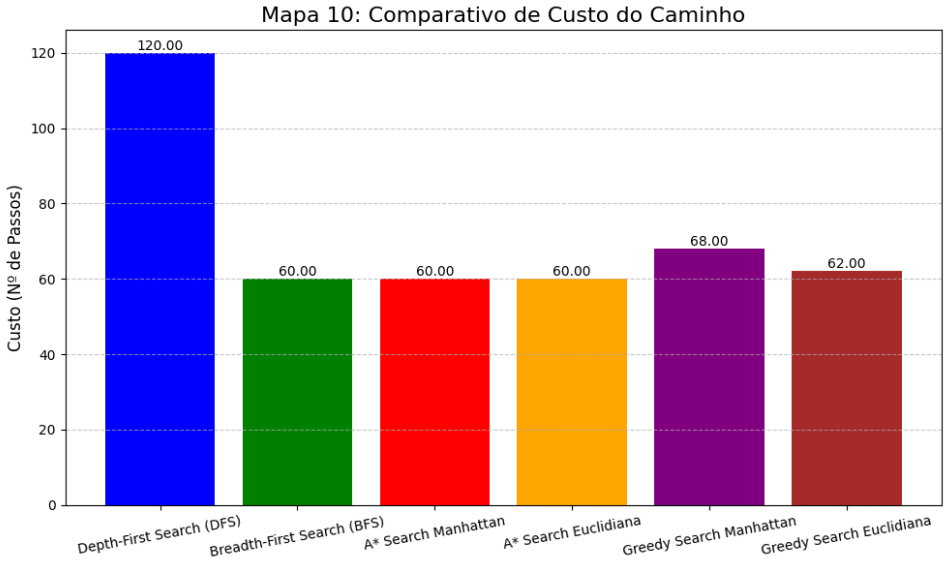

Figura 3: Comparativo de Custo do Caminho para o Mapa 10.

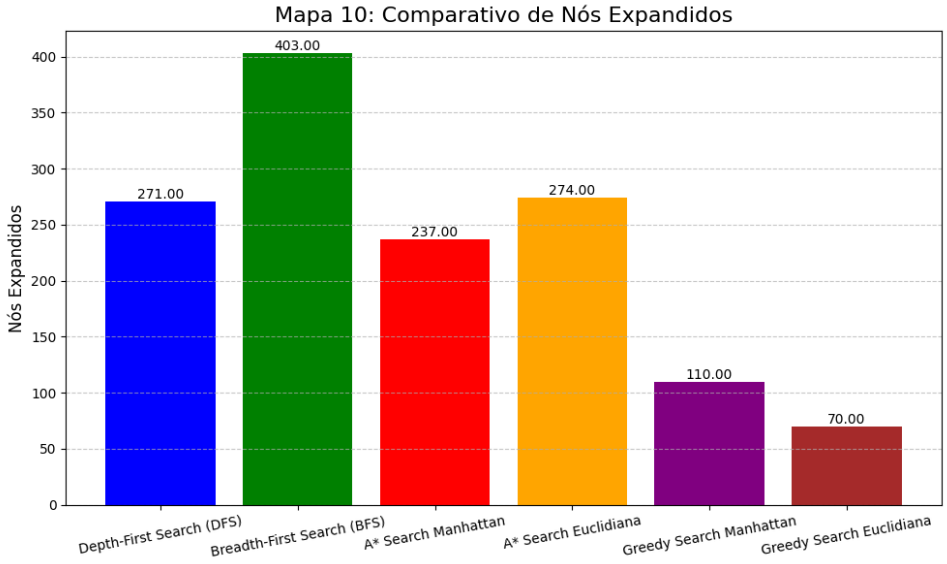

Figura 4: Comparativo de Nós Expandidos para o Mapa 10.

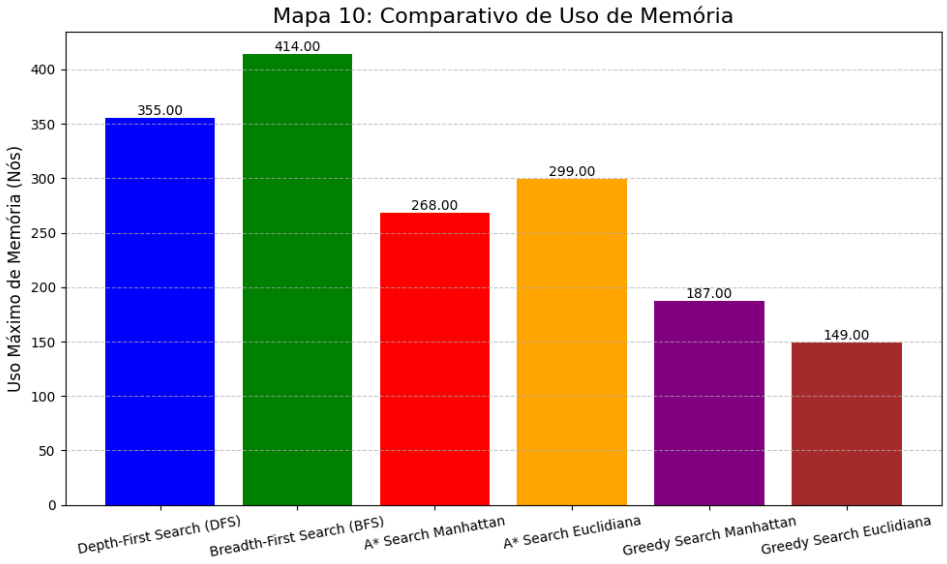

Figura 5: Comparativo de Uso de Memória para o Mapa 10.

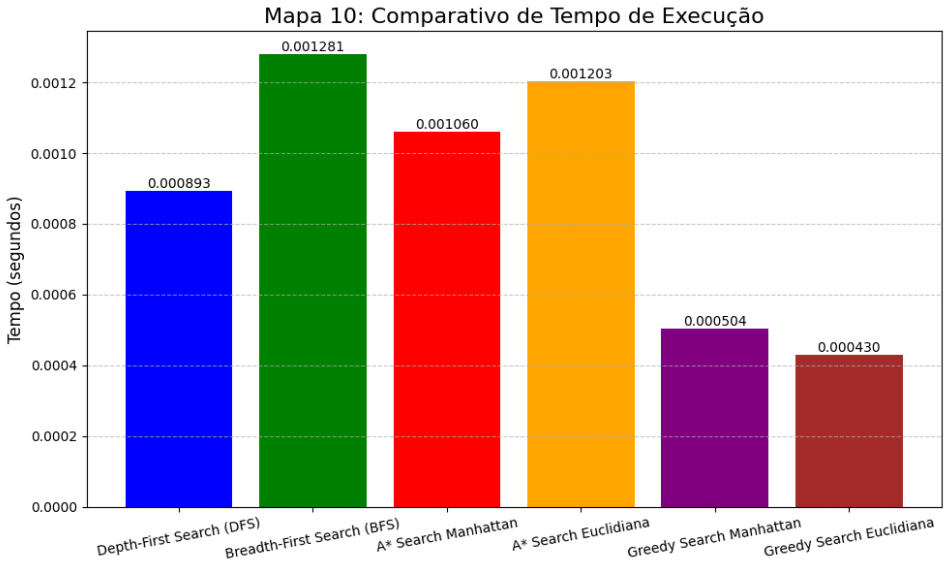

Figura 6: Comparativo de Tempo de Execução para o Mapa 10.

Impacto da Função de Avaliação (*A* vs. Greedy*)

A análise do complexo Mapa 9 evidencia a diferença fundamental entre as estratégias informadas e o risco inerente à abordagem "míope"da Busca Gulosa. Conforme a Tabela 1 da análise atualizada, a Busca Gulosa, guiada unicamente pela heurística ($f(n) = h(n)$), foi induzida ao erro e selecionou caminhos subótimos, com custo 68 (Manhattan) e 62 (Euclidiana). Em contrapartida, o algoritmo A*, em ambas as suas variantes, manteve a optimalidade, identificando corretamente o caminho de custo 60.

Este comportamento é explicado pela função de avaliação de cada um. A Busca Gulosa foi "enganada"por caminhos que pareciam promissores (baixo $h(n)$), mas que eram, na verdade, armadilhas longas. O A*, por sua vez, ao incorporar o custo real percorrido ($g(n)$) em sua função de avaliação ($f(n) = g(n) + h(n)$), percebeu que o custo total

da armadilha estava crescendo muito. O algoritmo abandonou estes ramos ineficientes e priorizou o caminho ótimo.

Notavelmente, embora tenha falhado na optimalidade, a Busca Gulosa provou ser mais eficiente em velocidade e processamento. A versão com heurística Euclidiana expandiu apenas 70 nós, contra 237 do A* (Manhattan), e consumiu menos memória. Isso ilustra o clássico trade-off: a Busca Gulosa oferece uma solução rápida, mas não confiável, enquanto o A* garante a melhor solução com um custo computacional maior.

Tabela 2: Comparativo de Desempenho A* vs. Greedy no Mapa 9.

Algoritmo	**Custo (Passos)**	**Nós Expandidos**	**Memória (Nós)**	**Tempo (s)**
A* (Manhattan)	**60**	237	268	0.001060
A* (Euclidiana)	**60**	274	299	0.001203
Greedy (Manhattan)	68	110	187	0.000504
Greedy (Euclidiana)	62	70	149	0.000430

O Impacto da Heurística (BFS vs. A*)

Uma comparação direta entre os algoritmos que garantem a optimalidade — a Busca em Largura (BFS) e o A* — revela o impacto prático de uma boa heurística nos resultados do Mapa 10. Conforme observado na Tabela 1, todos os algoritmos ótimos (BFS, A* Manhattan, A* Euclidiana) cumpriram sua garantia, encontrando consistentemente o mesmo custo de caminho de 60 passos.

A diferença crucial entre eles, no entanto, reside na eficiência computacional. O BFS, por ser uma busca cega, precisou expandir **403 nós** para encontrar a solução. O A*, utilizando a heurística de Manhattan (a mais adequada para este tipo de grid), alcançou o mesmo resultado expandindo apenas **237 nós**.

Este resultado empírico demonstra de forma inequívoca o poder da heurística. O BFS explora o labirinto sistematicamente em "camadas", expandindo inevitavelmente um grande número de nós irrelevantes. O A*, por outro lado, utiliza a heurística para guiar sua exploração de forma inteligente. Ao priorizar nós que não estão apenas próximos da origem (baixo $g(n)$), mas que também estão estimados como próximos do objetivo (baixo $h(n)$), o A* consegue "podar"vastas áreas do espaço de busca, alcançando a mesma solução ótima com um esforço computacional significativamente menor.

4 Manhattan vs. Euclidiana

Para aprofundar a análise sobre o impacto das heurísticas, foi implementada e testada uma segunda função, a **distância Euclidiana**, que calcula a distância em linha reta entre o nó atual e o objetivo. Esta seção compara diretamente o desempenho das duas heurísticas quando aplicadas aos algoritmos de busca informada no mapa 9.

Comparativo de Desempenho no A*

Quando aplicadas ao algoritmo A*, ambas as heurísticas guiaram a busca para o mesmo caminho ótimo de **custo 60**. A garantia de optimalidade do A* foi mantida, mas a eficiência da exploração variou:

- **Heurística de Manhattan:** Expandiu **237 nós** para encontrar a solução.
- **Heurística Euclidiana:** Precisou expandir **274 nós**, ou seja, 15.6% a mais que a de Manhattan.

A análise mostra que, para o A*, a heurística de **Manhattan** foi mais eficiente. Por ser mais fiel ao custo real de movimento no grid (uma estimativa mais informada), ela conseguiu guiar o A* de forma mais direta, "podando"mais ramos irrelevantes da árvore de busca e, consequentemente, expandindo menos nós.

Comparativo de Desempenho na Busca Gulosa (Greedy)

Na Busca Gulosa, cuja estratégia é inteiramente guiada pela heurística, a escolha entre Manhattan e Euclidiana teve um impacto drástico em todas as métricas de desempenho:

- **Heurística de Manhattan:** Encontrou um caminho subótimo de custo 68, expandindo 110 nós.
- **Heurística Euclidiana:** Também encontrou um caminho subótimo, mas de qualidade superior (custo 62), e foi muito mais eficiente, expandindo apenas 70 nós.

Neste cenário, a **heurística Euclidiana provou ser superior** para a estratégia gulosa: encontrou um **caminho 8.8% melhor** (mais curto) e **explorou 36.4% menos nós**, resultando em uma busca consideravelmente mais rápida e direta.

Conclusões sobre as Heurísticas

A comparação entre as duas heurísticas no Mapa 10 permite extrair as seguintes conclusões:

- **Natureza da Heurística:** A distância de Manhattan é mais **conservadora** e informada, pois se alinha melhor às restrições de movimento do problema. A distância Euclidiana é mais **otimista**, pois representa um "atalho"que não é fisicamente possível no labirinto.
- **Interação com o Algoritmo:**
  - Para o **A***, a natureza mais informada da heurística de **Manhattan** foi mais vantajosa, resultando em uma busca mais eficiente e com menos nós expandidos.
  - Para a **Busca Gulosa**, a natureza otimista da **Euclidiana** foi benéfica, guiando a busca de forma mais agressiva e direta ao objetivo, resultando em superioridade em todas as métricas: velocidade, eficiência de exploração e qualidade da solução encontrada.

5 Divisão de Tarefas e uso de *IA*

Autores e Papéis

- **Eduardo Henrique Queiroz ALmeida:** Implementação dos algorítmos *BFS* e Busca gulosa pelo melhor caminho, redação das seções 3,5 e 6.

- **Joaquim Cézar Santana da Cruz:** Implementação dos algoritmos *DFS* e A*, redação das seções 1, 2 e 4.

Uso de *IA*

- **Ferramenta(s)**: Google Gemini
- **Finalidade**: A ferramenta foi utilizada como assistente para a estruturação e revisão textual deste relatório. Ademais, ela foi usada para fazer a função de geração dos gráficos pela biblioteca *matplotlib* no código.
- **Declaração de conformidade**: Em conformidade com a política da disciplina, declaramos que nenhuma parte do código de implementação dos algoritmos de busca avaliados (*DFS*, *BFS*, *A**, e busca gulosa pelo melhor caminho) foi gerada pela ferramenta de *IA*. A lógica de implementação destes algoritmos foi auxiliada pelos pseudocódigos, pela pesquisa em sites e vídeos no youtube. Sendo as fontes: [3], [4], [5]

6 Conclusão

Este trabalho implementou e comparou múltiplos algoritmos de busca — DFS, BFS, Greedy e A*, com as heurísticas de Manhattan e Euclidiana — aplicados à resolução de labirintos. A análise dos resultados, com foco no complexo Mapa 10, permitiu extrair conclusões robustas e alinhadas com a teoria da Inteligência Artificial.

O A* e a Busca em Largura (BFS) se destacaram pela **optimalidade**, ambos encontrando o caminho mais curto. No entanto, o teste no mapa complexo evidenciou a superioridade do A*. Enquanto o BFS garantiu a optimalidade ao custo de expandir o maior número de nós e consumir mais memória, o A*, por sua vez, confirmou seu status como o algoritmo mais robusto: utilizando uma heurística bem-informada como a de Manhattan, ele garantiu o mesmo resultado ótimo de forma muito mais eficiente, "podando"ramos de busca irrelevantes.

Por outro lado, a Busca em Profundidade (DFS) e a Busca Gulosa se mostraram mais rápidas, mas sem qualquer garantia de optimalidade. O DFS encontrou o pior caminho de todos, com o dobro do custo da solução ótima, sendo útil apenas quando a memória é uma restrição severa e qualquer solução é aceitável. A Busca Gulosa, com sua estratégia "míope", provou ser a mais rápida, especialmente com a heurística Euclidiana. No entanto, essa eficiência veio com um risco claro: em um labirinto complexo como o analisado, ela foi facilmente atraída para caminhos subótimos, falhando em encontrar a melhor solução.

Em suma, a escolha do algoritmo de busca ideal depende intrinsecamente do trade-off entre a qualidade da solução e a eficiência. Se a **optimalidade é indispensável**, o **A* com uma heurística bem ajustada ao problema (neste caso, Manhattan)** é a escolha superior, superando com folga a abordagem de força bruta do BFS. Se a **velocidade é a prioridade máxima** e uma solução "boa o suficiente"é aceitável, a **Busca Gulosa** é uma excelente alternativa. As buscas cegas, BFS e DFS, servem como uma base fundamental de comparação, ressaltando o imenso valor da informação heurística para resolver problemas complexos de forma inteligente.

Referências

[1] American Psychological Association. Apa dictionary of psychology - british museum algorithm, 2018.

[2] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Appoach - Fourth Edition*. Pearon series.

[3] datacamp. O que é a distância de manhattan, 2024.

[4] datacamp. O algoritmo a*: Um gruia completo, 2024.

[5] freeCodCamp. Harvard cs50's artificial intelligence with python – full university cours, 2023.