Uma imagem com texto

Descrição gerada automaticamente

**Instituto Politécnico do Cávado e do Ave**

**Escola Superior de Tecnologia**

**Uma imagem com arte, Gráficos, clipart, design

Descrição gerada automaticamente**

**Licenciatura**

**em**

**Engenharia Informática Médica**

**Inteligência Artificial**

Bruno Rafael Mendes Oliveira – a15566

Diogo Mário Sá Fernandes – a24017

**Dezembro de 2023**

Esta página foi deixada em branco propositadamente.

Uma imagem com texto

Descrição gerada automaticamente

**Instituto Politécnico do Cávado e do Ave**

**Escola Superior de Tecnologia**

**Licenciatura**

**em**

**Engenharia Informática Médica**

**Relatório do Projeto Engenharia de Software**

**Gestão de Armazéns por Inteligência Artificial**

**Unidade Curricular**

Inteligência Artificial

**Nome dos Alunos**

Bruno Oliveira

Diogo Fernandes

**Docente da Unidade Curricular:**

Profª. Joaquim Gonçalves

**Setembro de 2023**

Esta página foi deixada em branco propositadamente.

**Resumo**

Este relatório descreve o trabalho prático realizado na UC de Inteligência Artificial, envolvendo a análise e desenvolvimento de pesquisa num determinado armazém imaginário, onde este armazém tem um ponto de início onde existe um robot para coletar um produto e depois de coletar este produto o robot tem de ir para a saída. O objetivo foi aplicar os conceitos teóricos e desenvolver habilidades práticas lecionada na UC.

Além disso, foi desenvolvida uma interface gráfica em *Python* para processar e apresentar o processamento dos algoritmos gráfico.

O trabalho prático permitiu a aplicação dos conhecimentos teóricos e destacou a importância da Inteligência Artificial bem como a implementação de vários algoritmos como a perceção de uso. Assim como consolidar os conhecimentos adquiridos ao longo da UC.

**Palavras-Chaves:** Inteligência Artificial, Agentes, Armazéns Automáticos, *Python*, Algoritmos de pesquisa

**Abstract**

*This report describes the practical work carried out in the AI course, involving analysis and research development in a certain imaginary warehouse, where this warehouse has a starting point where there is a robot to collect a product and after collecting this product the robot has to go to the exit. The objective was to apply theoretical concepts and develop practical skills taught in the course. Furthermore, a graphical interface in Python was developed to process and present the graphical algorithms processing. The practical work allowed the application of theoretical knowledge and highlighted the importance of Artificial Intelligence as well as the implementation of various algorithms as a perception of use. As well as consolidating the knowledge acquired throughout the course.*

***Keywords****: Artificial Intelligence, Agents, Automated Warehouses, Python, Search Algorithms*

**Índice**

[Índice de Figuras 9](#_Toc152024778)

[Índice de Tabelas 10](#_Toc152024779)

[Lista de siglas e acrónimos 11](#_Toc152024780)

[1. Introdução 12](#_Toc152024781)

[1.1. 1. Introdução 12](#_Toc152024782)

[1.2. Enquadramento 12](#_Toc152024783)

[1.3. Objetivos 12](#_Toc152024784)

[1.4. Estrutura do Documento 13](#_Toc152024785)

[2. Enunciado para o projeto 14](#_Toc152024786)

[2.1. Objetivo do Projeto 14](#_Toc152024787)

[3. Algoritmos 15](#_Toc152024788)

[3.1. Algoritmo de Busca em Largura (BFS) 15](#_Toc152024789)

[3.1.1. Definição 15](#_Toc152024790)

[3.1.2. Processo: 15](#_Toc152024791)

[3.1.3. Propriedades: 16](#_Toc152024792)

[3.2. A Estrela 17](#_Toc152024793)

[3.2.1. Processo do Algoritmo A\*: 17](#_Toc152024794)

[3.2.2. Propriedades do Algoritmo A\*: 18](#_Toc152024795)

[*3.3.* *Greedy Search* 19](#_Toc152024796)

[*3.3.1.* Propriedades do Algoritmo *Greedy Search* 20](#_Toc152024797)

[4. Armazéns 21](#_Toc152024798)

[5. Desenvolvimento 22](#_Toc152024799)

[5.1. Matriz para simular Armazém 22](#_Toc152024800)

[5.2. Algoritmo BFS 23](#_Toc152024801)

[5.3. Algoritmo A Estrela 24](#_Toc152024802)

[5.4. Algoritmo *Gridy Search* 25](#_Toc152024803)

[6. Resultados 26](#_Toc152024804)

[6.1. Resultados Matrizes 75x75 27](#_Toc152024805)

[7. Pros e Contras 28](#_Toc152024806)

[8. Conclusão 28](#_Toc152024807)

[9. Bibliografia 29](#_Toc152024808)

[10. Anexos 30](#_Toc152024809)

# Índice de Figuras

# Índice de Tabelas

[Tabela 1 - Armazéns Simulados 25](#_Toc152024774)

# Lista de siglas e acrónimos

* AI: *Artificial Inteligence*;
* UC: Unidade Curricular
* DFS: *Depth -First Search*

# Introdução

## Enquadramento

No âmbito da UC de Inteligência Artificial, aprender sobre algoritmos de busca é crucial para entender como construir sistemas inteligentes. Estes algoritmos são fundamentais para a compreensão de como as máquinas podem imitar processos de raciocínio humano, especialmente em tarefas que envolvem resolução de problemas e tomada de decisões.

Os algoritmos de busca, como o Busca em Largura (BFS), A Estrela (A\*) e *Greedy Search*, exemplificam a aplicação de conceitos de Inteligência Artificial na solução de problemas complexos de navegação e otimização. No contexto de um armazém automatizado, estes algoritmos desempenham um papel crucial na orientação de robôs para a execução eficiente de tarefas de coleta e entrega, uma aplicação prática e altamente relevante dos princípios de IA.

Este estudo, portanto, não só contribui para o entendimento acadêmico e teórico de algoritmos de busca dentro do campo da Inteligência Artificial, mas também demonstra como esses conceitos são aplicados em cenários do mundo real. A análise do desempenho desses algoritmos em simulações de ambientes de armazém oferece insights valiosos sobre sua eficácia, limitações e potencial para futuras inovações na indústria de automação e logística.

Ao integrar teoria e prática, este trabalho alinha-se com os objetivos centrais da cadeira de Inteligência Artificial, proporcionando uma compreensão profunda tanto dos fundamentos teóricos quanto das aplicações práticas dos algoritmos de busca.

## 1.2. Objetivos

Compreender a aplicação dos algoritmos de busca em largura (*BFS*), A Estrela (A\*), e *Greedy Search* em ambientes de armazém automatizado.

Analisar a eficácia destes algoritmos em diferentes cenários de simulação de armazém, com variações no tamanho da matriz e nos níveis de obstáculos.

Comparar o desempenho desses algoritmos em termos de eficiência, tempo de execução, e precisão na navegação de um robô automatizado.

Explorar as limitações e potenciais melhorias desses métodos em ambientes de armazéns reais.

# Algoritmos

## Algoritmo de Busca em Largura (BFS)

### Definição

A busca em largura (BFS, do inglês "Breadth-First Search") é um algoritmo para percorrer ou pesquisar estruturas de dados em árvore ou grafo. O algoritmo começa no nó raiz (ou qualquer nó específico em um grafo) e explora todos os vizinhos desse nó antes de passar para os nós no próximo nível de profundidade. Ele visita os nós em camadas ou níveis, garantindo que todos os eles de um determinado nível sejam explorados antes de passar para o próximo. A BFS é útil para encontrar o caminho mais curto em grafos não ponderados e para travessias que precisam processar nós em ordem de proximidade ao nó inicial.

### Processo:

1. **Ponto de Partida:**

O BFS começa em um nó específico do grafo, que é geralmente referido como o nó raiz ou inicial. Este nó serve como o ponto de partida para a busca.

1. **Inicialização de Estruturas de Dados:**

Para rastrear a progressão da busca, o BFS utiliza um vetor. Todos os nós visitados são armazenados nesse vetor.

1. **Exploração do Primeiro Nível:**

O nó inicial é marcado como visitado e adicionado ao vetor. Em seguida, o algoritmo explora todos os vizinhos diretos desse nó.

1. **Passagem para Níveis Subsequentes:**

Uma vez que todos os vizinhos do nó inicial são explorados e adicionados ao vetor, o algoritmo remove o nó inicial da fila e passa para o próximo nó no vetor.

Este processo é repetido para explorar todos os vizinhos não visitados do nó atual e adicioná-los ao vetor.

A busca continua seguindo a ordem dos nós na fila, garantindo que os eles sejam explorados nível por nível.

1. **Continuação até Esgotar os Nós ou chegar ao Objetivo:**

O processo continua até que a vetor esteja vazio ou chegue ao objetivo.

Durante a busca, cada nó é visitado exatamente uma vez, assegurando uma procura completa e sistemática.

1. **Reconstrução do Caminho:**

Uma vez que o objetivo é alcançado, o algoritmo reconstrói o caminho do nó objetivo de volta ao nó inicial.

### Propriedades:

* **Ótimo para Grafos Não Ponderados:** Em grafos não ponderados, o BFS garante encontrar o caminho mais curto entre o nó inicial e qualquer outro nó no grafo, pois ele explora todos os nós igualmente em cada nível antes de passar para o próximo.
* **Completo:** O BFS é completo, o que significa que se houver uma solução, o algoritmo a encontrará. Ele explora sistematicamente todos os nós e caminhos possíveis em um grafo finito.
* **Complexidade de Espaço:** A complexidade de espaço do BFS é proporcional à largura máxima do grafo. Ele precisa armazenar todos os nós de um nível para acessar os do próximo, o que é feito através de um vetor. Portanto, em grafos largos, o BFS pode consumir uma quantidade significativa de memória.

## A Estrela

O Algoritmo A\* é uma técnica de busca amplamente usada em computação para encontrar o caminho mais curto entre um ponto inicial e um ponto final em um grafo.

### Processo do Algoritmo A\*:

1. **Inicialização:**

O algoritmo começa com um conjunto aberto que contém apenas o nó inicial e um conjunto fechado vazio.

Cada nó mantém três valores importantes: o custo real do caminho do nó inicial até ele (g), a estimativa heurística do custo para alcançar o objetivo, onde é calculado o caminho de *Manhattan* (h) e a soma destes dois valores (f = g + h).

1. **Exploração de Nós:**

Em cada etapa, o algoritmo escolhe o nó com o menor valor de f no conjunto aberto.

Este nó é removido do conjunto aberto e adicionado ao conjunto fechado.

1. **Expansão do Nó Atual:**

O algoritmo expande o nó selecionado, examinando todos os seus vizinhos.

Para cada vizinho, o algoritmo calcula o valor de g (custo do caminho do nó inicial até o vizinho através do nó atual), h (caminho de *Manhattan)* e atualiza o valor de f.

1. **Atualização do Conjunto Aberto:**

Se um vizinho não está nem no conjunto aberto nem no fechado, ele é adicionado ao conjunto aberto.

Se o vizinho já está em um dos conjuntos, mas o novo caminho é melhor (menor g), o caminho até esse vizinho é atualizado.

1. **Repetição do Processo:**

O algoritmo repete esse processo de seleção, expansão e atualização até que o conjunto aberto esteja vazio ou até que o objetivo seja alcançado.

1. **Verificação de Objetivo:**

Se o nó selecionado for o nó objetivo, o algoritmo reconstrói o caminho até o nó inicial e termina.

1. **Reconstrução do Caminho:**

Uma vez que o objetivo é alcançado, o algoritmo reconstrói o caminho do nó objetivo de volta ao nó inicial, seguindo os caminhos de menor custo registados.

### Propriedades do Algoritmo A\*:

* **Ótimo e Completo (com a Heurística Adequada):** O algoritmo A\* é ótimo e completo se a função heurística usada for admissível, ou seja, nunca superestima o custo real para alcançar o objetivo. Isso garante que o A\* encontre o caminho mais curto em um espaço de busca finito.
* **Heurística:** O coração do A\* é sua função heurística, que estima o custo do caminho mais barato do nó atual para o objetivo.
* **Complexidade de Espaço:** A\* mantém todos os caminhos não explorados na memória. Em espaços de busca grandes, isso pode levar a um alto consumo de memória. A complexidade de espaço é, na pior das hipóteses, exponencial em relação ao comprimento do caminho.
* **Complexidade de Tempo:** A complexidade de tempo depende fortemente da heurística. Na pior das hipóteses, pode ser exponencial, especialmente se a heurística não for eficaz. Com uma boa heurística, no entanto, A\* é significativamente mais rápido do que algoritmos de busca sem informação.
* **Balanceamento entre Exploração e Eficiência:** A\* equilibra a busca que se move rapidamente em direção ao objetivo, e o método de menor custo, que explora caminhos de baixo custo. Este balanceamento é alcançado através da heurística.

## *Greedy Search*

Um algoritmo *Greedy Search* é uma abordagem para resolver um problema selecionando a melhor opção disponível no momento. Não se preocupa se o melhor resultado atual trará o resultado global ótimo.

* 1. **Ponto de Partida:**

O algoritmo inicia a partir de um nó específico, considerado o ponto de partida da procura.

* 1. **Decisão Baseada em Heurística:**

Em cada passo, o algoritmo seleciona o próximo nó para explorar com base em uma função heurística. Essa função avalia qual opção parece levar mais rapidamente ao objetivo, sem preocupar-se com o caminho percorrido até então.

* 1. **Avanço Direcionado:**

O algoritmo avança para o nó escolhido, baseando-se unicamente que ele oferece em direção ao objetivo. Ele não reconsidera as escolhas anteriores, mantendo-se focado no caminho à frente.

* 1. **Procura pelo Objetivo:**

A busca continua seguindo a orientação da heurística, até que o objetivo seja encontrado ou até que fique claro que não há caminho viável para o objetivo a partir do trajeto escolhido.

* 1. **Conclusão da Busca:**

Uma vez que o objetivo é alcançado, o algoritmo reconstrói o caminho do nó objetivo de volta ao nó inicial, seguindo os caminhos de menor custo registados.

### Propriedades do Algoritmo *Greedy Search*

**Não Ótimo:** O Greedy Search não garante encontrar a solução mais ótima para um problema. Ele faz escolhas que parecem ser as melhores no momento, sem considerar o custo acumulado ou o impacto futuro dessas escolhas.

**Não Completo:** Este algoritmo não é completo, o que significa que não há garantia de que ele sempre encontrará uma solução, se ela existir. Em cenários com múltiplos caminhos ou obstáculos, pode falhar em encontrar um caminho até o objetivo.

**Dependência da Heurística:** A eficácia do *Greedy Search* é altamente dependente da qualidade da função heurística utilizada. Uma heurística apropriada pode conduzir rapidamente ao objetivo, enquanto uma heurística inadequada pode levar a resultados não atinge a mais alta qualidade ou falhas.

**Eficiência em Termos de Espaço e Tempo:** O *Greedy Search* é geralmente eficiente em termos de uso de memória e tempo de execução, principalmente porque não mantém um registo de todos os caminhos explorados.

**Aplicabilidade Prática Limitada:** Embora o *Greedy Search* seja útil em alguns cenários específicos, sua aplicabilidade é limitada em problemas complexos, onde a escolha mais promissora a curto prazo não necessariamente leva ao melhor resultado global.

# Armazéns

Neste relatório, exploramos a aplicação de algoritmos de busca em um ambiente de armazém automatizado, onde um robô é utilizado para realizar tarefas de coleta e entrega de produtos. A eficiência e eficácia do robô em navegar pelo armazém são fundamentais para otimizar as operações logísticas. Para isso, modelamos o ambiente do armazém como uma matriz, onde cada célula representa uma área específica, variando de espaços transitáveis a obstáculos.

**Componentes do Armazém:**

* **Robô (R):** Localizado inicialmente em uma posição específica da matriz, o robô é a unidade móvel encarregada de coletar e entregar produtos.
* **Espaço Livre (.):** Representa as áreas livres do armazém por onde o robô pode mover-se, permitindo-lhe traçar rotas para alcançar diferentes pontos.
* **Obstáculos (#):** Simulam paredes ou pilares no armazém, criando barreiras físicas. Estes obstáculos são intransponíveis para o robô, requerendo que ele planeje rotas alternativas para alcançar seu destino.
* **Produto (P):** Localização de um produto que o robô deve coletar. Este ponto é o destino inicial do robô, onde ele deve chegar para realizar a coleta.
* **Ponto de Entrega (O):** Após a coleta do produto, o robô deve levar o item até este ponto para completar a entrega.

**Aplicação dos Algoritmos de Busca:**

Os algoritmos de busca como DFS, BFS e A\* são implementados para guiar o robô pelo armazém.

O objetivo é encontrar o caminho mais eficiente do ponto inicial (R) até o produto (P), e em seguida, do produto até o ponto de entrega (O), considerando os obstáculos presentes.

# Desenvolvimento

## Matriz para simular Armazém

A matriz serve como uma representação simplificada de um armazém, onde cada célula corresponde a uma área específica dentro do espaço físico do armazém. Esta representação é fundamental para simular e analisar o movimento do robô dentro deste ambiente.

* **Criação da Matriz:**

A matriz é inicialmente criada como uma matriz de células vazias, representadas por pontos ('.'). Cada ponto simboliza um espaço livre por onde o robô pode se mover.

* **Definição de Percentagem de Obstáculos**:

A quantidade de obstáculos é determinada por uma percentagem definida, assegurando que um certo número de células na matriz seja convertido em obstáculos.

* **Adição de Obstáculos**

Obstáculos são distribuídos aleatoriamente pela matriz para simular obstáculos no armazém. Estes são representados por '#' e indicam áreas pelas quais o robô não pode passar.

* **Posicionamento de Elementos-Chave:**

**Robô (R):** Uma célula é designada como o ponto de partida do robô.

**Produto (P):** Um local na matriz é escolhido aleatoriamente para representar onde o produto está localizado.

**Ponto de Entrega (O):** Um ponto é estabelecido como o destino final do robô, após a coleta do produto.

* **Aleatoriedade e Realismo:**

A posição dos obstáculos e do produto é aleatória, imitando a natureza imprevisível e variável de um armazém real.

* **Processo de Geração:**

A matriz é gerada automaticamente com estes critérios, utilizando algoritmos para assegurar a distribuição aleatória dos obstáculos e a colocação dos elementos-chave.

## Algoritmo BFS

O Algoritmo de Busca em Largura (BFS) é utilizado para orientar o movimento de um robô em um armazém simulado. O objetivo é que o robô encontre e siga o caminho mais eficiente para coletar um produto (P) e, em seguida, entregá-lo em um ponto de saída (O), navegando por um espaço repleto de obstáculos (#).

* **Implementação e Funcionamento:**

Uma interface gráfica mostra o armazém, com cores distintas marcando o robô, o produto, o ponto de saída, os caminhos explorados e os obstáculos. Isso proporciona uma visualização clara do ambiente e da trajetória do robô.

* **Navegação do Robô:**

O robô, partindo de sua posição inicial, utiliza o BFS para explorar o ambiente.

O BFS busca o produto e o ponto de saída, movendo-se nível por nível, avaliando todas as células acessíveis antes de prosseguir para as próximas.

* **Visualização do Caminho:**

À medida que o robô se move, o caminho percorrido é destacado na interface, permitindo que os observadores acompanhem sua progressão e as decisões tomadas pelo algoritmo.

* **Desafios de Navegação:**

Obstáculos representam barreiras físicas que o robô não pode atravessar, exigindo que o BFS encontre rotas alternativas.

A eficiência do robô em alcançar o produto e o ponto de saída é crucial, demonstrando a eficácia do BFS em resolver tarefas de navegação complexas.

* **Análise de Desempenho:**

O tempo necessário para encontrar os caminhos é monitorado, fornecendo dados importantes sobre o desempenho do algoritmo em diferentes configurações do armazém.

## Algoritmo A Estrela

No cenário do armazém automatizado, o algoritmo A Estrela é utilizado para guiar um robô na coleta e entrega de produtos. O armazém é simulado por uma matriz, onde o robô deve encontrar o caminho mais eficiente até um produto (P), e depois até um ponto de saída (O), superando obstáculos representados por '#' na matriz.

* **Implementação e Funcionamento:**

Uma interface gráfica mostra o armazém, com cores distintas marcando o robô, o produto, o ponto de saída, os caminhos explorados e os obstáculos. Isso proporciona uma visualização clara do ambiente e da trajetória do robô.

* **Navegação Inteligente:**

O robô utiliza o A Estrela para calcular a rota mais eficaz até o produto e posteriormente até a saída.

O algoritmo combina o custo real do caminho percorrido com uma estimativa heurística da distância até o objetivo, equilibrando eficiência e previsibilidade.

* **Funcionamento do Algoritmo:**

O A Estrela prioriza os nós com o menor custo estimado total (somatório do custo real até o ponto atual e a estimativa até o objetivo).

O caminho é reconstruído uma vez que o destino é alcançado, mostrando a rota completa que o robô deve seguir.

* **Análise de Desempenho:**

O tempo necessário para o robô encontrar o caminho é monitorado, fornecendo uma métrica de eficiência do algoritmo em diferentes configurações do armazém.

* **Desafios e Soluções:**

Obstáculos no armazém exigem que o algoritmo encontre rotas alternativas, demonstrando a adaptabilidade do A Estrela em ambientes complexos.

A precisão da heurística utilizada impacta diretamente na eficiência do caminho encontrado.

## Algoritmo *Gridy Search*

No ambiente simulado de um armazém, o algoritmo *Greedy Search* é empregue para guiar um robô na tarefa de localizar e coletar um produto, e posteriormente transportá-lo para um ponto de saída. O armazém é mapeado por uma matriz, com obstáculos, caminhos e locais de interesse claramente definidos.

* **Implementação e Funcionamento:**

Uma interface gráfica mostra o armazém, com cores distintas marcando o robô, o produto, o ponto de saída, os caminhos explorados e os obstáculos. Isso proporciona uma visualização clara do ambiente e da trajetória do robô.

* **Funcionamento do Algoritmo:**

O *Greedy Search* move o robô passo a passo em direção ao objetivo, selecionando a cada momento a opção que parece mais direta, baseada em uma função heurística.

Esta abordagem não reavalia decisões passadas nem considera o custo total percorrido, focando unicamente na estimativa até o destino.

* **Desafios de Procura:**

A natureza direta do *Greedy Search* pode não sempre garantir o caminho mais eficiente ou seguro, mas busca o trajeto mais direto baseado na heurística utilizada.

* **Monitoramento e Análise:**

O tempo e a eficiência do robô em alcançar seus objetivos são monitorados para avaliar a eficácia do algoritmo *Greedy Search* em diferentes configurações do armazém.

* **Visualização do Processo:**

A interface gráfica atualiza em tempo real para mostrar o caminho sendo explorado pelo robô, os locais já visitados e a progressão em direção ao objetivo.

# Resultados

Foram criadas 72 matrizes para simular os armazéns e avaliar o desempenho de cada algoritmo. As matrizes criadas são do tipo:

Tabela - Armazéns Simulados

|  |  |  |
| --- | --- | --- |
| Número de Matrizes | Tamanho da Matriz | Tipo de Matriz |
| 5 | 10x10 | Nível Baixo de Obstáculos |
| 25x25 | Nível Baixo de Obstáculos |
| 48x52 | Nível Baixo de Obstáculos |
| 50x50 | Nível Baixo de Obstáculos |
| 75x75 | Nível Baixo de Obstáculos |
| 10x10 | Nível Médio de Obstáculos |
| 25x25 | Nível Médio de Obstáculos |
| 48x52 | Nível Médio de Obstáculos |
| 50x50 | Nível Médio de Obstáculos |
| 75x75 | Nível Médio de Obstáculos |
| 10x10 | Nível Alto de Obstáculos |
| 25x25 | Nível Alto de Obstáculos |
| 48x52 | Nível Alto de Obstáculos |
| 50x50 | Nível Alto de Obstáculos |
| 75x75 | Nível Alto de Obstáculos |

Foi criado um procedimento para validar o desempenho de cada algoritmo em cada um destes armazéns. Para validar estes procedimentos foram retiradas várias métricas como, media, mediana, desvio padrão, máximo e mínimo, para verificar qual a performance de todos os algoritmos.

Os dados foram separados da seguinte forma:

* Custo de Robô para Produto (onde é calculado em unidades de tempo);
* Custo de Produto para Output (onde é calculado em unidades de tempo);
* Custo de Robô para Produto em segundos (estimado que cada casa o robô demora 0.1segundos a percorrer);
* Custo de Produto para Output em segundos (estimado que cada casa o robô demora 0.1segundos a percorrer);
* Tempo de procura do caminho do Robô para Produto em segundos;
* Tempo de procura do caminho do Produto para Output em segundos;

## Resultados Matrizes 10x10



|  |  |  |  |
| --- | --- | --- | --- |
|  | ***BFS*** | *Greedy Search* | A\* |
| Tempo Total | 2,537442446 | 2,49467907 | 2,387797281 |

# Pros e Contras

# Conclusão

# Bibliografia

# Anexos