Métodos de Busca

IMPLEMENTAÇÃO DE MÉTODOS DE BUSCA PARA RESOLUÇÃO DE LABIRINTO COM OBSTÁCULO

TÓPICOS ASSOCIADOS

- O que são Métodos de Busca?
- Para que Servem os Métodos de Busca?
- Categorias
- Modelagem do Problema
- Conclusão

O que são Métodos de Busca?

Métodos de busca são algoritmos que buscam explorar e resolver problemas ao percorrer um espaço de estados. Eles tentam encontrar um caminho ou solução através de uma série de transformações do estado inicial até um estado objetivo, de acordo com um conjunto de regras ou restrições definidas para o problema.

Objetivo dos Métodos de Busca:

Identificar a sequência de ações ou passos que levam de uma condição inicial até a solução do problema. No contexto da Disciplina (IA), a busca pode ser utilizada para encontrar soluções para problemas de planejamento, navegação e otimização.

Para que Servem os Métodos de Busca?

Busca de Caminhos:

Encontrar o caminho mais curto entre dois pontos. Em jogos e navegação, isso é utilizado para determinar a trajetória mais eficiente, minimizando o custo (tempo, distância, etc.).

Otimização:

Métodos de busca são cruciais em problemas de otimização, como alocação de recursos ou planejamento de rotas, para encontrar soluções que maximizem ou minimizem algum critério (ex: minimizar o custo ou o tempo de execução).

Planejamento:

Planejamento de ações, como em robótica, onde é necessário calcular uma sequência de movimentos para atingir um objetivo, evitando colisões ou obstáculos.

Categorias

1. Subida de Encosta (Hill Climbing)

O algoritmo de subida de encosta é um algoritmo de busca local. Ele começa em um estado inicial e tenta fazer melhorias sucessivas para chegar à solução. A cada passo, o algoritmo escolhe o estado "vizinho" que tem a melhor avaliação segundo uma heurística. Se não encontrar um vizinho melhor, o algoritmo termina.

Estado Inicial: O algoritmo começa de uma das entradas do labirinto.

Vizinhos: São as células adjacentes que podem ser visitadas (não são obstáculos e não foram visitadas anteriormente).

Heurística: A heurística poderia ser a distância em linha reta até a saída.

Limitação: O algoritmo pode ficar preso em um mínimo local (onde uma solução boa, mas não ótima, é alcançada sem possibilidade de melhorar).

Pontos Positivos:
Simples e rápido para problemas pequenos.

Pontos Negativos:
Não garante a solução ótima.
Pode ficar preso em pontos de mínimo local.

Categorias

2. Busca Gulosa (Greedy Best-First Search)

A busca gulosa é um algoritmo de busca informada que seleciona o próximo nó com base em uma heurística, sem se preocupar com o custo acumulado. Ele sempre expande o nó que parece ser o mais promissor em direção à solução.

Estado Inicial: Começa de uma das entradas.

Heurística: A heurística poderia ser a distância direta até a saída, como a distância em linha reta (usando a fórmula de distância euclidiana ou Manhattan).

Expansão: O algoritmo expande o nó que parece ser o mais próximo da saída, sem considerar o custo até esse ponto.

Pontos Positivos:
Pode ser rápido, pois foca na direção da solução.

Pontos Negativos:
Não garante uma solução ótima, pois pode explorar caminhos longos ou inúteis.
Pode ser ineficiente dependendo da distribuição dos obstáculos.

Categorias

3. Busca A* (A-star)

A busca A* é um algoritmo informada que combina a busca de custo uniforme com uma heurística. Ele seleciona o próximo nó a ser expandido com base no custo total acumulado até aquele ponto e uma estimativa do custo restante até a solução. A* é otimizado, garantindo uma solução ótima se a heurística for admissível.

Estado Inicial: Começa de uma das entradas.

Heurística: A heurística seria a distância em linha reta até a saída.

Custo Real (g(n)): O custo acumulado para alcançar o nó até o momento.

Custo Estimado (h(n)): A estimativa do custo para alcançar a saída (heurística).

Função de Avaliação (f(n)): f(n)=g(n)+h(n). O próximo nó a ser expandido é o que tem o menor valor de f(n)f(n)f(n).

Pontos Positivos:
Ótimo e Completo: Se a heurística for admissível, A* sempre encontra a solução ótima..

Pontos Negativos:
Mais complexo e mais lento que outros algoritmos devido à necessidade de calcular tanto o custo real quanto a heurística.

Categorias

4. Busca em Profundidade (Depth-First Search - DFS)

A busca em profundidade é um algoritmo não informado que explora o mais profundo possível em um caminho antes de retroceder. Em cada nível, o algoritmo escolhe um nó e o expande até que não seja mais possível, retrocedendo para explorar caminhos alternativos.

Estado Inicial: Começa de uma das entradas.

Expansão: O algoritmo expande os nós até o final do caminho, retrocedendo ao atingir um beco sem saída.

Problema: DFS pode ficar preso em ciclos ou em caminhos longos sem encontrar a solução.

Pontos Positivos:
Simples de implementar e com uso de memória reduzido.
Funciona bem para problemas com solução próxima ao fundo da árvore de busca.

Pontos Negativos:
Não garante encontrar o caminho mais curto.
Pode ser ineficiente e ficar preso em caminhos longos ou ciclos.

Categorias

Busca em Largura (Breadth-First Search - BFS)

A busca em largura é um algoritmo não informado que expande todos os nós no nível atual antes de passar para o próximo nível. BFS garante que o caminho mais curto (em termos de número de passos) será encontrado, mas pode ser mais lento para labirintos grandes.

Estado Inicial: Começa de uma das entradas.

Expansão: Expande todos os nós de cada nível antes de mover para o próximo nível.

Garantia: Sempre encontrará o caminho mais curto para a saída, desde que a distância seja contada de forma uniforme (sem obstáculos insuperáveis).

Pontos Positivos:
Ótimo: Garante encontrar o caminho mais curto.
Completo: Sempre encontra uma solução se existir.

Pontos Negativos:
Uso de memória muito alto, especialmente em labirintos grandes.
Pode ser muito lento devido à grande quantidade de nós a serem expandidos.

Categorias

Algoritmo	Garantia de Ótima Solução	Complexidade	Vantagens	Desvantagens
Subida de Encosta	Não	Baixa	Simples, rápido	Pode ficar preso em mínimos locais.
Busca Gulosa	Não	Média	Foca em soluções promissoras	Não garante solução ótima.
Busca A*	Sim (se heurística for boa)	Alta	Ótimo e completo, eficiente se bem configurado	Complexo, exige mais memória e cálculo.
Busca em Profundidade	Não	Baixa	Simples, baixo uso de memória	Pode ser muito ineficiente e não encontrar a solução ótima.
Busca em Largura	Sim (para labirintos sem pesos)	Alta	Garante solução mais curta	Alto uso de memória, lento em grandes espaços.