Inteligência Artificial

Lista 4 – Métodos de Busca

Vitor Dias de Britto Militão

Questão 1:

1) Algoritmo de Busca em Largura (BFS)

- Nós visitados: A, B, C, D, E, F, G, H, I
- Solução obtida: A → B → E → I
- Justificativa: A Busca em Largura expande todos os nós de um nível antes de passar para o próximo, garantindo encontrar o nó mais próximo do inicial.

2) Algoritmo de Busca em Profundidade (DFS)

- Nós visitados: A, B, D, H, I
- Solução obtida: A → B → D → H
- Justificativa: A Busca em Profundidade explora o caminho mais profundo antes de retroceder, encontrando o primeiro objetivo ao percorrer o caminho mais longo possível antes de retornar.

3) Custo Uniforme

- Nós visitados: A, B, C, D, E, F, I
- Solução obtida: A → B → E → I
- Justificativa: O Custo Uniforme expande o nó com o menor custo acumulado, garantindo a solução com o menor custo.

4) Algoritmo de Busca Gulosa

- Nós visitados: A, C, F, I
- Solução obtida: A → C → F → I
- Justificativa: A Busca Gulosa segue a heurística que parece prometer a solução mais rápida, sem considerar o custo total acumulado, mas focando no nó mais promissor.

5) Algoritmo A*

- Nós visitados: A, C, F, I
- Solução obtida: A → C → F → I
- Justificativa: O A* combina o custo acumulado e a heurística estimada para escolher o próximo nó a ser expandido. A heurística aqui é admissível porque não superestima os custos reais.

Questão 2:

1) Sim, a heurística de Manhattan é admissível. A heurística de Manhattan para o Puzzle de 8 é a soma das distâncias de cada peça da posição atual para a posição final desejada, considerando a movimentação em blocos ortogonais (vertical e horizontal). Ela é admissível porque nunca superestima o custo real para alcançar a solução. O custo real de mover uma peça é sempre 1 por movimento, e a distância de Manhattan sempre conta o número mínimo de movimentos necessários para levar uma peça à sua posição correta. Isso significa que a heurística nunca dará um valor maior do que o número real de movimentos necessários, o que cumpre a condição de admissibilidade.
2) Uma outra heurística possível para o Puzzle de 8 é contar o número de peças fora do lugar (também chamada de heurística de peças deslocadas). Este método simplesmente conta quantas peças não estão na posição correta em comparação com o estado final. Essa heurística também é admissível. Ela, assim como a heurística de Manhattan, nunca superestima o custo real para chegar ao estado final, pois o número de peças fora de lugar é sempre menor ou igual ao número real de movimentos necessários para corrigir o estado. Assim, embora simples, essa heurística oferece uma estimativa mínima do número de movimentos restantes, o que garante sua admissibilidade.

Questão 3:

Alternativa B.

Questão 4:

Alternativa A.

Questão 5:

Alternativa E.

Questão 6:

Alternativa E.

Questão 7:

Alternativa B.

Questão 8:

Alternativa B.

Questão 9:

Quando w=0: O algoritmo realiza uma **busca de custo uniforme**, considerando apenas o custo real g(n).

Quando w=1: O algoritmo realiza uma **busca A***, equilibrando o custo real g(n) e a heurística h(n), garantindo uma solução ótima se a heurística for admissível.

Quando w=2: O algoritmo realiza uma **busca gulosa**, baseando-se exclusivamente na heurística h(n) para decidir quais nós expandir, sem garantir uma solução ótima.

Questão 10:

1) Busca A*

a) Nós expandidos:

- Com h1: S, B, D, G.
- Com h2: S, A, G.
- Com h0: S, B, D, G.

b) Caminho encontrado:

- Com h1: S→B→D→G
- Com h2: S→A→G
- Com h0: S→B→D→G

c) Heurísticas admissíveis:

- Admissíveis: h1, h2, h0 (não superestimam o custo).

2) Busca Gulosa

a) Nós expandidos:

- Com h1: S, B, D, G.
- Com h2: S, A, G.

b) Caminho encontrado:

- Com h1: S→B→D→GS
- Com h2: S→A→G

3) Busca em Profundidade

c) Nós expandidos: **S, B, D, G.**
d) Caminho encontrado: **S→B→D→G**

4) Busca em Largura

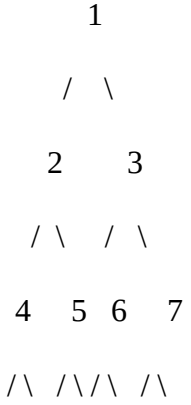
e) Nós expandidos: **S, A, B, C, D, G.**
f) Caminho encontrado: **S→A→G**

Questão 11:

Alternativa A.

Questão 12:

a)



b) Busca com estado objetivo 11:

1. Busca em largura (extensão):
Ordem dos nós visitados:
1,2,3,4,5,6,7,8,9,10,111, 2, 3, 4, 5, 6, 7, 8, 9, 10, 111,2,3,4,5,6,7,8,9,10,11.

2. Busca em profundidade limitada (limite 3):
Ordem dos nós visitados:
1,2,4,8,9,5,10,111, 2, 4, 8, 9, 5, 10, 111,2,4,8,9,5,10,11.

3. Busca por aprofundamento iterativo:
Ordem dos nós visitados:
1,2,3,4,5,6,7,8,9,10,111, 2, 3, 4, 5, 6, 7, 8, 9, 10, 111,2,3,4,5,6,7,8,9,10,11.

Questão 13:

Vantagens do A*:

1. **Ótimo e Completo**: Quando a heurística é admissível (não superestima o custo) e consistente, A* sempre encontra a solução ótima (menor custo) e é completo, ou seja, encontrará uma solução se houver.
2. **Balanceamento de Custo e Heurística**: A* combina o custo acumulado g(n) com a estimativa heurística h(n), permitindo uma busca eficiente que considera o caminho percorrido e o custo restante.
3. **Flexibilidade**: Funciona bem para diversos tipos de problemas e pode ser ajustado com diferentes heurísticas para se adaptar a domínios específicos.

Desvantagens do A*:

1. **Consumo de Memória**: A* mantém todos os nós gerados em memória, o que pode ser problemático em grandes espaços de busca, levando ao consumo excessivo de memória.
2. **Dependência da Heurística**: Se a heurística não for bem projetada (admissível e consistente), o algoritmo pode perder eficiência ou não garantir a solução ótima.
3. **Custo Computacional**: Embora seja eficiente, A* pode expandir muitos nós, o que pode ser lento em espaços de busca grandes ou quando a solução está muito distante.

Questão 14:

SMA (Simplified Memory-Bounded A)**

- **Descrição**: O SMA* é uma versão limitada em memória do A*. Ele expande os nós de maneira semelhante ao A*, mas limita o número de nós armazenados em função da capacidade de memória disponível. Quando a memória está cheia, ele descarta nós menos promissores, mas mantém informações para que eles possam ser reexplorados, se necessário.
- **Vantagem**: É uma versão de A* que funciona em ambientes com restrições de memória.
- **Desvantagem**: O desempenho pode diminuir ao recarregar estados descartados quando necessário.

A_1 (A One)

- **Descrição**: O A_1 ajusta dinamicamente o peso w que controla a importância entre g(n)e h(n). Começa como uma busca gulosa (com maior peso na heurística) e, gradualmente, se torna uma busca A* à medida que se aproxima da solução.
- **Vantagem**: Pode acelerar a busca inicial como uma busca gulosa, mantendo a garantia de encontrar a solução ótima no final.
- **Desvantagem**: O ajuste dinâmico pode ser complexo e não é sempre eficiente em todos os domínios.

Theta (Theta Star)*

- **Descrição**: Theta* é uma variação do A* projetada para ambientes contínuos (ex.: mapas 2D ou 3D), permitindo a movimentação em linha reta entre nós, ao invés de restringir-se a movimentação por arestas de grafo. Isso reduz o custo do caminho em muitos cenários.
- **Vantagem**: Gera caminhos mais curtos e realistas em espaços contínuos, como em robótica ou jogos.
- **Desvantagem**: Pode ser mais custoso computacionalmente em espaços com muitos obstáculos.

Estes são apenas alguns exemplos, mas existem mais alguns.

Questão 15:

Para verificar se o jogador MAX pode ganhar o jogo usando a busca MINIMAX, podemos analisar a situação inicial de 5 palitos. A estratégia envolve considerar as possíveis jogadas de ambos os jogadores (MAX e MIN).

1. **Nodos do Jogo**: Cada nodo representa uma configuração do jogo com um determinado número de palitos. A partir de 5 palitos, MAX pode fazer as seguintes jogadas:
    - Retirar 1 palito (4 palitos restantes).
    - Retirar 2 palitos (3 palitos restantes).
    - Retirar 3 palitos (2 palitos restantes).
2. **Cenários**:
    - Se MAX retirar 1 palito, MIN terá 4 palitos. MIN pode retirar 1, 2 ou 3 palitos, levando a resultados variados.
    - Se MAX retirar 2 palitos, MIN terá 3 palitos e a situação se repetirá.
    - Se MAX retirar 3 palitos, MIN ficará com 2 palitos, novamente com várias opções.
3. **Análise de Perdas**: MAX perde se:
    - Ao final de sua jogada, deixar MIN em uma posição onde ele (MIN) sempre possa forçar a vitória.
4. **Avaliação**:
    - A configuração de 1 palito é uma posição perdedora para quem jogar, pois o jogador terá que retirar o último palito.
    - Com 2 palitos, o jogador pode retirar 1, levando o oponente a 1 (perdendo), então é uma posição ganhadora.
    - Com 3 palitos, MAX pode ganhar.
    - Com 4 palitos, MIN pode levar MAX a uma posição perdedora, e assim por diante.
5. **Conclusão**:
    - A partir de 5 palitos, MAX pode garantir uma vitória, retirando 2 palitos e levando o jogo a 3, garantindo que, após as jogadas subsequentes, ele pode sempre manter a vantagem.

Portanto, MAX pode ganhar o jogo.

Questão 16:

Alternativa C.