Aluno: Breno Pires Santos

Matricula 808238

Questão 1

Algoritmos:

Busca em Largura (BFS):

- **Ordem dos nós visitados:** A, B, C, D, E, F, G, H, I
- **Solução obtida:** A → B → E → I

Busca em Profundidade (DFS):

- **Ordem dos nós visitados:** A, B, D, H, I
- **Solução obtida:** A → B → D → H → I

Custo Uniforme:

- **Ordem dos nós visitados:** A, C, B, F, G, D, E, I
- **Solução obtida:** A → B → E → I

Busca Gulosa:

- **Ordem dos nós visitados:** A, C, G, K
- **Solução obtida:** A → C → G → K

Algoritmo A* (A estrela):

- **Ordem dos nós visitados:** A, C, F, G, J, K
- **Solução obtida:** A → C → G → K

A heurística é admissível?
A heurística não pode ser considerada admissível, pois em alguns casos ela subestima o custo real até o objetivo. Um exemplo claro disso é o nó C: o caminho mais barato de C até o objetivo K custa 19, mas a heurística associada a C é apenas 2. Essa diferença mostra que a heurística está sendo otimista demais, o que fere a definição de admissibilidade.

Questão 2

A heurística de Manhattan é admissível? Justifique

Sim, a heurística de Manhattan é admissível. Ela calcula para cada peça a distância total em movimentos verticais e horizontais entre sua posição atual e a posição correta no estado objetivo. Como essa heurística nunca superestima o número real de movimentos necessários ela é considerada admissível.Isso acontece por que, no pior caso, cada movimento de uma peça no quebra-cabeça equivale a uma unidade na distância de Manhattan ou seja, ela considera o mínimo necessário para cada peça alcançar sua posição final, sem assumir movimentos impossíveis.

Proponha uma outra heurística para este problema. Ela é admissível? Justifique.

Uma outra heurística possível é o número de peças fora do lugar também chamada de heurística da contagem de erros. Essa heurística simplesmente conta quantas peças estão em posições diferentes do objetivo. Ela também é admissível, pois nunca superestima a quantidade real de movimentos: se uma peça está fora do lugar, pelo menos um movimento será necessário para posicioná-la corretamente. No entanto, essa heurística costuma ser menos informativa do que a de Manhattan, pois trata todos os erros com o mesmo peso, sem considerar quão longe cada peça está da posição correta.

Questão 3

Resposta: B

Questão 4

Resposta: A

Questão 5

Resposta: E

Questão 6

Resposta: A

Questão 7

Resposta: B

Questão 8

Resposta: B

Questão 9

Quando w = 0: A função objetivo se torna$f(n)=2\cdot g(n)$f(n)=2·g(n). Isso resulta em uma busca de custo uniforme, que expande os nós com o menor custo acumulado.

Quando w = 1: A função objetivo se torna$f(n)=g(n)+h(n)$f(n)=g(n)+h(n). Isso corresponde a uma busca A*, que combina o custo real$g(n)$g(n) com a heurística$h(n)$h(n).

Quando w = 2: A função objetivo se torna$f(n)=g(n)+2\cdot h(n)$f(n)=g(n)+2·h(n). Isso resulta em uma busca gulosa, que prioriza a heurística$h(n)$h(n) e ignora o custo real$g(n)$g(n).

Questão 10

Busca A*

Nós expandidos:

h_0: S, A, C, G

h_1: S, A, C, G

h_2: S, A, C, G

Caminho encontrado:

h_0: S → A → C → G

h_1: S → A → C → G

h_2: S → A → C → G

Heurísticas admissíveis:

h_0: Sim

h_1: Sim

h_2: Não (superestima o custo real em A)

Busca Gulosa

Nós expandidos: S, B, D, G

Caminho encontrado: S → B → D → G

Busca em Profundidade

Nós expandidos: S, A, C, G

Caminho encontrado: S → A → C → G

Busca em Largura

Nós expandidos: S, A, B, C, G

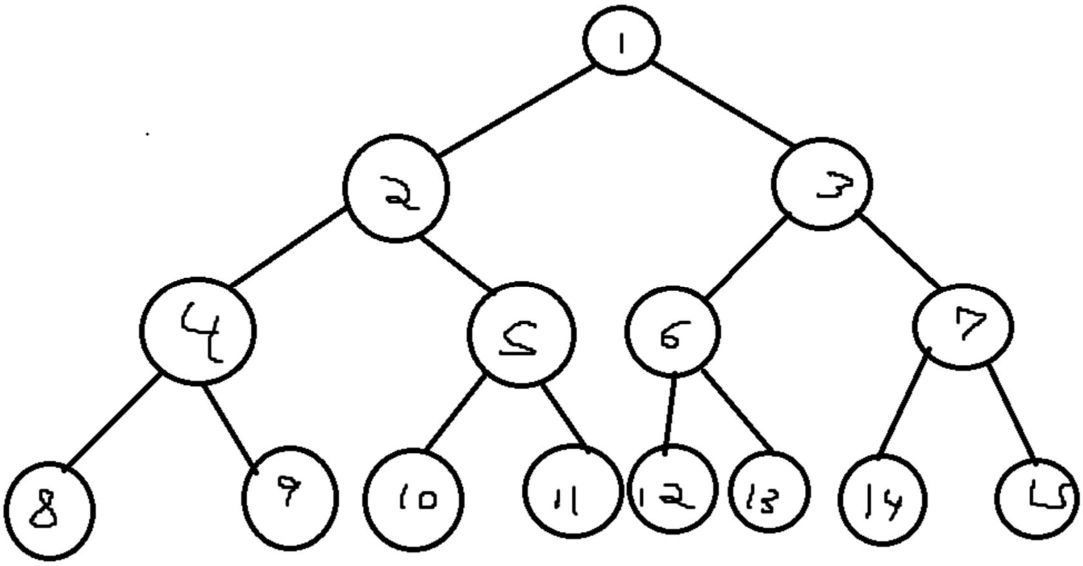
Caminho encontrado: S → A → C → G

Questão 11

Resposta: A

Questão 12

a)



b)

Busca em Largura

Ordem dos nós: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

Busca em Profundidade

Ordem dos nós: 1, 2, 4, 8, 9, 5, 10, 11

Busca por aprofundamento iterativo

Ordem dos nós: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

Questão 13

O algoritmo A* é uma técnica de busca informada que combina o custo acumulado do caminho (g) com uma estimativa heurística do custo restante até o objetivo (h), garantindo assim a obtenção de soluções ótimas quando a heurística é admissível. Sua principal vantagem é a eficiência na exploração do espaço de busca, focando em caminhos promissores e evitando a expansão desnecessária de nós, o que o torna adequado para aplicações como jogos, navegação e planejamento. No entanto, seu consumo elevado de memória pode ser um obstáculo em espaços de busca muito grandes, e seu desempenho depende fortemente da qualidade da heurística utilizada, podendo tornar-se ineficiente ou até incorreto se a heurística for mal projetada.

Questão 14

Diversos algoritmos foram desenvolvidos como melhorias ou variações do A*, buscando reduzir seu consumo de memória ou acelerar o tempo de execução. Um exemplo é o IDA* que combina a busca em profundidade com os princípios do A*, utilizando menos memória ao explorar os caminhos por profundidade crescente baseada em limites de custo. Outra variação é o ARA* que encontra rapidamente uma solução subótima e a melhora gradualmente, sendo útil quando há restrições de tempo. Já o D* e o D* Lite são versões adaptadas para ambientes dinâmicos, recalculando caminhos de forma eficiente sempre que o ambiente muda, o que é especialmente útil em robótica e navegação em tempo real. Essas melhorias mantêm a ideia central do A*, mas adaptam seu comportamento a diferentes demandas de desempenho, memória e aplicabilidade.

Questão 15

|- tira 1 → MIN (4)
| |- tira 1 → MAX (3)
| | |- tira 1 → MIN (2)
| | | |- tira 1 → MAX (1)
| | | | |- tira 1 → MIN (0) → MIN perde → MAX vence
| | | |- tira 2 → MAX (0) → MAX perde
| | | |- tira 3 → impossível
| | |- tira 2 → MIN (1)
| | | |- tira 1 → MAX (0) → MAX perde
| | |- tira 3 → MIN (0) → MIN perde → MAX vence
| |- tira 2 → MAX (2)
| | |- tira 1 → MIN (1)
| | | |- tira 1 → MAX (0) → MAX perde
| | |- tira 2 → MIN (0) → MIN perde → MAX vence
| |- tira 3 → MAX (1)
| | |- tira 1 → MIN (0) → MIN perde → MAX vence

|- tira 2 → MIN (3)
| |- tira 1 → MAX (2)
| | |- tira 1 → MIN (1)
| | | |- tira 1 → MAX (0) → MAX perde
| | |- tira 2 → MIN (0) → MIN perde → MAX vence
| |- tira 2 → MAX (1)
| | |- tira 1 → MIN (0) → MIN perde → MAX vence
| |- tira 3 → MAX (0) → MAX perde

|- tira 3 → MIN (2)
| |- tira 1 → MAX (1)
| | |- tira 1 → MIN (0) → MIN perde → MAX vence

| |- tira 2 → MAX (0) → MAX perde
| |- tira 3 → impossível

Melhor jogada de MAX: tirar 2 palitos no primeiro turno → deixa 3. Qualquer jogada de MIN a partir daí leva à vitória de MAX. Portanto, MAX pode ganhar.

Questão 16

Resposta: E