Disciplina: Inteligência Artificial

Professora: Cristiane Neri Nobre

Data de entrega: 04/05

Valor: 1,5 pontos

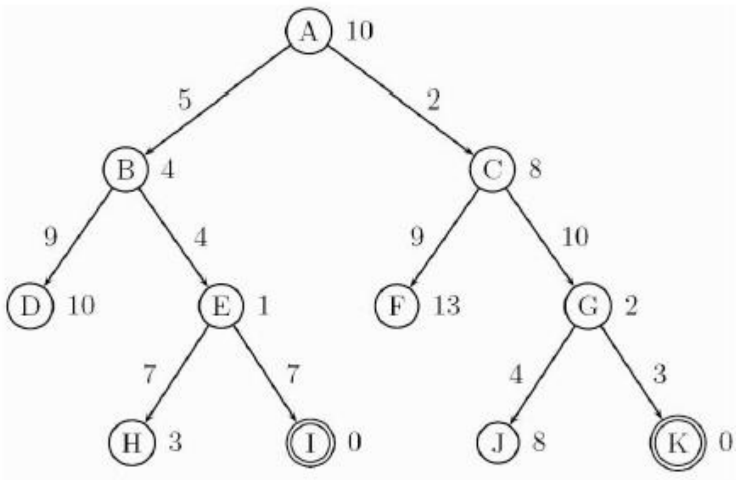
Aluno: Lucas Henrique Rocha Hauck

Questão 01

Considere o espaço de busca a seguir. Cada nó é rotulado por uma letra. Cada nó objetivo é representado por um círculo duplo. Existe uma heurística estimada para cada dado nó (indicada por um valor ao lado do nó). Arcos representam os operadores e seus custos associados. Para cada um dos algoritmos a seguir, pede-se:

1) Os **nós visitados** na ordem em que eles são examinados, começando pelo nó A
2) Forneça também a **solução obtida** por cada método
3) Pergunta-se: a **heurística** é admissível? Justifique.

No caso de escolhas equivalentes entre diferentes nodos, prefira o nodo mais próximo da raiz, seguido pelo nodo mais à esquerda na árvore. O algoritmo para a busca quando encontra o I ou o K. Ou seja, não é necessário encontrar os dois objetivos.



1) Algoritmo de Busca em Largura
   1 A, B, C, D, E, F, G, H, I, J, K
   2 A, B, E, I = 16 e A, C, G, K = 15
   3 A BFS não utiliza heurística no algoritmo
2) Algoritmo de Busca em Profundidade
   1 A, B, D, E, H, I, C, F, G, J, K
   2 A, B, E, I = 16 e A, C, G, K = 15
   3 A DFS não utiliza heurística no algoritmo
3) Custo Uniforme
   1 A, C, B, G, E, F, D, K, I

2 A, B, E, I = 16 e A, C, G, K = 15
3 O algoritmo de custo uniforme não utiliza heurística no algoritmo

4) Algoritmo de Busca Gulosa
   1 A, C, G, K
   2 A, C, G, K = 15
   3 A busca gulosa utiliza apenas a heurística (h(n)) como critério de escolha. A heurística não é necessariamente admissível, pois ela pode superestimar o custo real até o objetivo. Por isso, **a solução pode não ser ótima**.

5) Algoritmo A*
   1 A, C, B, G, E, K
   2 A, C, G, K = 15
   3 O algoritmo A* utiliza tanto o custo acumulado (g(n)) quanto a heurística (h(n)), isto é, f(n) = g(n) + h(n). Para que A* garanta a **solução ótima**, a heurística precisa ser **admissível**, ou seja, **nunca superestimar o custo real até o objetivo**. Neste caso, como todos os valores heurísticos são menores ou iguais ao custo real até o objetivo, **a heurística é admissível**.

Questão 02

Para o problema do Puzzle de 8, pede-se:

1. A heurística de Manhattan é admissível? Justifique.
   Sim. Ela soma a distância em linhas e colunas que cada peça está da posição correta. Como nunca passa do número real de movimentos, é admissível.
2. Proponha uma outra heurística para este problema. Ela é admissível? Justifique.
   Pode ser o número de peças fora do lugar. Também é admissível, pois nunca exagera no número de movimentos — só conta quantas peças ainda precisam ser movidas.

Questão 03

Julgue os itens a seguir, relativos a métodos de busca com informação (busca heurística) e sem informação (busca cega), aplicados a problemas em que todas as ações têm o mesmo custo, o grafo de busca tem fator de ramificação finito e as ações não retornam a estados já visitados.

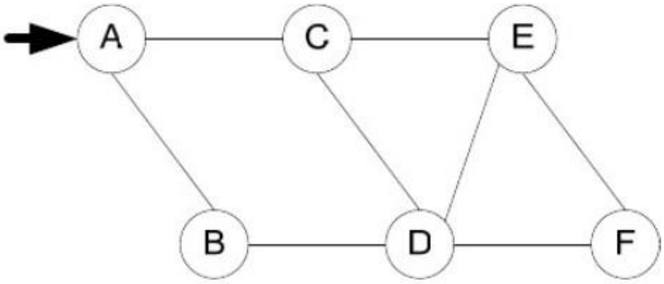
I. A primeira solução encontrada pela estratégia de busca em largura é a solução ótima.
II. A primeira solução encontrada pela estratégia de busca em profundidade é a solução ótima.
III. As estratégias de busca com informação usam funções heurísticas que, quando bem definidas, permitem melhorar a eficiência da busca.
IV. A estratégia de busca gulosa é eficiente porque expande apenas os nós que estão no caminho da solução.

Estão certos apenas os itens

a) I e II.
b) I e III. Resposta
c) I e IV.
d) II e IV.
e) III e IV.

Questão 04

Considere o algoritmo de busca em largura em grafos. Dado o grafo a seguir e o vértice A como ponto de partida, a ordem em que os vértices são descobertos é dada por:



A) A B C D E F
B) A B D C E F
C) A C D B F E
D) A B C E D F
E) A B D F E C

Resposta: Letra A

Questão 05

Analise as seguintes as seguintes afirmativas:

I. A estratégia de busca em largura encontra a solução ótima quando todos os operadores de mudança de estado têm o mesmo custo.

II. A estratégia de busca em profundidade sempre expande um menor número de nós que a estratégia de busca em largura, quando aplicadas ao mesmo problema.

III. A estratégia de busca heurística encontra sempre a solução de menor custo.

IV. A estratégia de busca heurística expande um número de nós em geral menor que o algoritmo de busca em largura, mas não garante encontrar a solução ótima.

V. O algoritmo de busca heurística que utiliza uma função heurística admissível encontra a solução ótima.

A esse respeito, pode-se concluir que

(a) apenas a afirmativa V é correta.
(b) todas as afirmativas são corretas.
(c) todas as afirmativas são falsas.
(d) apenas as afirmativas II e V são corretas.
(e) apenas as afirmativas I, IV e V são corretas.

Resposta: Letra E

Questão 06 - POSCOMP 2007

[**TE**] Considerando que $h(n)$ é o custo estimado do nó n até o objetivo, em relação à busca informada, pode-se afirmar que

(a) a busca *gulosa* minimiza $h(n)$.

(b) a busca A^* minimiza $h(n)$.

(c) a busca de custo uniforme minimiza $h(n)$.

(d) a busca *gulosa* minimiza $h(n)$ somente se a heurística for admissível.

(e) a busca A^* minimiza $h(n)$ somente se a heurística for admissível.

Resposta: letra A

Questão 07 - POSCOMP 2005

Considere $h(x)$ como uma função heurística que define a distância de x até a meta; considere ainda $h^r(x)$ como a distância real de x até a meta. $h(x)$ é dita admissível se e somente se:

(a) $\exists n\ h(n) \leq h^r(n)$.

(b) $\forall n\ h(n) \leq h^r(n)$.

(c) $\forall n\ h(n) > h^r(n)$.

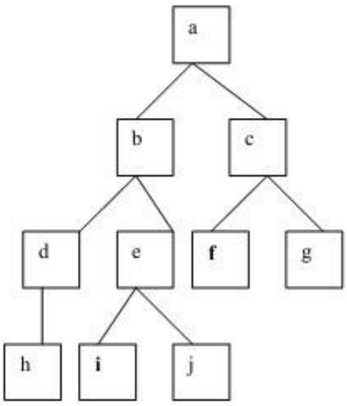
(d) $\exists n\ h(n) > h^r(n)$.

(e) $\exists n\ h(n) < h^r(n)$.

Resposta: letra B

Questão 8

59. Seja a árvore binária abaixo a representação de um espaço de estados para um problema p, em que o estado inicial é a, e i e f são estados finais.



Um algoritmo de busca em largura-primeiro forneceria a seguinte seqüência de estados como primeira alternativa a um caminho-solução para o problema p:

a) a b d h e i

b) a b c d e f

c) a b e i

d) a c f

e) a b d e f

Resposta: letra c

Questão 9

Suponha um algoritmo de busca pelo melhor primeiro (best-first ou busca gulosa) em que a função objetivo é f(n) = (2 – w).g(n) + w.h(n). Que tipo de busca ele realiza quando w = 0? Quando w = 1? E quando w = 2?

Quando w = 0:

- $f(n) = (2 - 0)\cdot g(n) + 0\cdot h(n) = 2\cdot g(n)$
- Neste caso, o algoritmo se comporta como uma busca de custo uniforme, priorizando o caminho com menor custo g(n) acumulado até o momento. Como o fator 2 é aplicado uniformemente a todos os nós, a ordenação relativa não muda.

Quando w = 1:

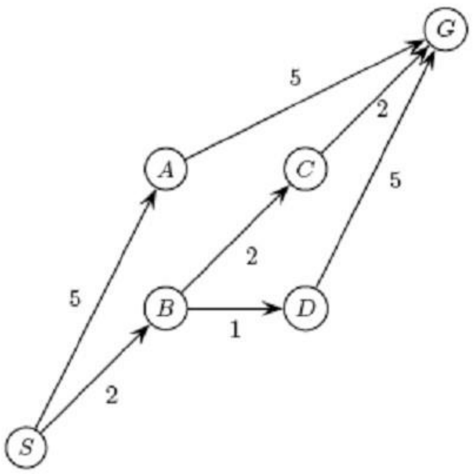
- $f(n) = (2 - 1)\cdot g(n) + 1\cdot h(n) = g(n) + h(n)$
- Neste caso, o algoritmo se comporta como A*, considerando tanto o custo já percorrido quanto a estimativa heurística.

Quando w = 2:

- $f(n) = (2 - 2)\cdot g(n) + 2\cdot h(n) = 0\cdot g(n) + 2\cdot h(n) = 2\cdot h(n)$
- Neste caso, o algoritmo se comporta como uma busca gulosa pura (greedy best-first search), considerando apenas a heurística h(n). O fator 2 não altera a ordem de seleção dos nós.

Questão 10

Considere o espaço de busca abaixo, onde S é o estado inicial e G é o único estado que satisfaz o teste de objetivo. Os rótulos nas arestas indicam o custo de percorrê-las e a tabela ao lado mostra o valor de três heurísticas h1, h2 e h3 para cada estado.



Node	h_0	h_1	h_2
S	0	5	6
A	0	3	5
B	0	4	2
C	0	2	5
D	0	5	3
G	0	0	0

1) Em relação à busca A*

a) Nós expandidos pela busca A* usando cada heurística

Para A*, a função de avaliação é f(n) = g(n) + h(n), onde g(n) é o custo real do caminho do início até o nó n.

Usando h1:

- Começamos com S: f(S) = 0 + 5 = 5
- Expandimos S, gerando A e B:
  - f(A) = 5 + 3 = 8
  - f(B) = 2 + 4 = 6
- Expandimos B (menor f), gerando C e D:
  - f(C) = 2+2 = 4 + 2 = 6
  - f(D) = 2+1 = 3 + 5 = 8
- Expandimos C (empatado com B, mas já expandimos B), gerando G:
  - f(G) = 2+2+2 = 6 + 0 = 6
- Expandimos G (empatado com C, mas já foi expandido)
- Nós expandidos: S, B, C, G

Usando h2:

- Começamos com S: f(S) = 0 + 6 = 6
- Expandimos S, gerando A e B:
  - f(A) = 5 + 5 = 10
  - f(B) = 2 + 2 = 4
- Expandimos B (menor f), gerando C e D:
  - f(C) = 4 + 5 = 9
  - f(D) = 3 + 3 = 6
- Expandimos D (menor f), gerando G:
  - f(G) = 8 + 0 = 8
- Expandimos G
- Nós expandidos: S, B, D, G

b) Caminho encontrado por cada heurística

Usando h1:

- Caminho: S → B → C → G com custo total 2+2+2 = 6

Usando h2:

- Caminho: S → B → D → G com custo total 2+1+5 = 8

c) Heurísticas admissíveis

Uma heurística é admissível se nunca superestima o custo real para alcançar o objetivo a partir de qualquer nó.

Para h1:

- Verificando cada nó:
    - S: h1(S) = 5, custo real = min(5+5, 2+2+2, 2+1+5) = 6 ✓
    - A: h1(A) = 3, custo real = 5 ✓
    - B: h1(B) = 4, custo real = min(2+2, 1+5) = 4 ✓
    - C: h1(C) = 2, custo real = 2 ✓
    - D: h1(D) = 5, custo real = 5 ✓
    - G: h1(G) = 0, custo real = 0 ✓

h1 é admissível pois nunca superestima o custo real para nenhum nó.

Para h2:

- Verificando cada nó:
    - S: h2(S) = 6, custo real = 6 ✓
    - A: h2(A) = 5, custo real = 5 ✓
    - B: h2(B) = 2, custo real = 4 ✗ (subestima)
    - C: h2(C) = 5, custo real = 2 ✗ (superestima)
    - D: h2(D) = 3, custo real = 5 ✓
    - G: h2(G) = 0, custo real = 0 ✓

h2 não é admissível pois superestima o custo para o nó C.

2) Em relação à busca gulosa

a) Nós expandidos

Na busca gulosa, expandimos sempre o nó com menor valor de h1:

- Começamos com S: h1(S) = 5
- Expandimos S, gerando A e B:
    - h1(A) = 3
    - h1(B) = 4
- Expandimos A (menor h1), gerando G:
    - h1(G) = 0
- Expandimos G (objetivo alcançado)
- Nós expandidos: S, A, G

b) Caminho encontrado

- Caminho: S → A → G com custo total 5+5 = 10

3) Em relação à busca em profundidade

c) Nós expandidos

Na busca em profundidade, expandimos primeiramente os filhos e depois os irmãos:

- Começamos com S
- Expandimos S, gerando A e B (assumindo que exploramos A primeiro)
- Expandimos A, gerando G
- Expandimos G (objetivo alcançado)
- Nós expandidos: S, A, G

d) Caminho encontrado

- Caminho: S → A → G com custo total 5+5 = 10

4) Em relação à busca em largura

e) Nós expandidos

Na busca em largura, expandimos nível por nível:

- Começamos com S
- Expandimos S, gerando A e B
- Expandimos A, gerando G
- Expandimos B, gerando C e D
- Nós expandidos: S, A, B, G (objetivo já encontrado, mas completamos o nível)

f) Caminho encontrado

- Caminho: S → A → G com custo total 5+5 = 10

Questão 11

Considere um jogo do tipo 8-puzzle, cujo objetivo é conduzir o tabuleiro esquematizado na figura abaixo para o seguinte estado final.

1	2	3
8		4
7	6	5

Considere, ainda, que, em determinado instante do jogo, se tenha o estado E0 a seguir.

3	4	6
5	8	
2	1	7

Pelas regras desse jogo, sabe-se que os próximos estados possíveis são os estados E1, E2 e E3 mostrados abaixo.

3	4	6
5		8
2	1	7

E1

3	4	6
5	8	7
2	1	

E2

3	4	
5	8	6
2	1	7

E3

Considere uma função heurística **h** embasada na soma das distâncias das peças em relação ao estado final desejado, em que a distância **d** a que uma peça **p** está da posição final é dada pela soma do número de linhas com o número de colunas que a separam da posição final desejada.

Por exemplo, em E1, **d(1) = 2 + 1 = 3**. A partir dessas informações analise as asserções a seguir.

Utilizando-se um algoritmo de busca gulosa pela melhor escolha que utiliza a função h, o próximo estado no desenvolvimento do jogo a partir do estado E0 tem de ser E3

porque,

dos três estados E1, E2 e E3 possíveis, o est ado com menor soma das distâncias entre a posição atual das peças e a posição final é o estado E3.

Assinale a opção correta a respeito dessas asserções.

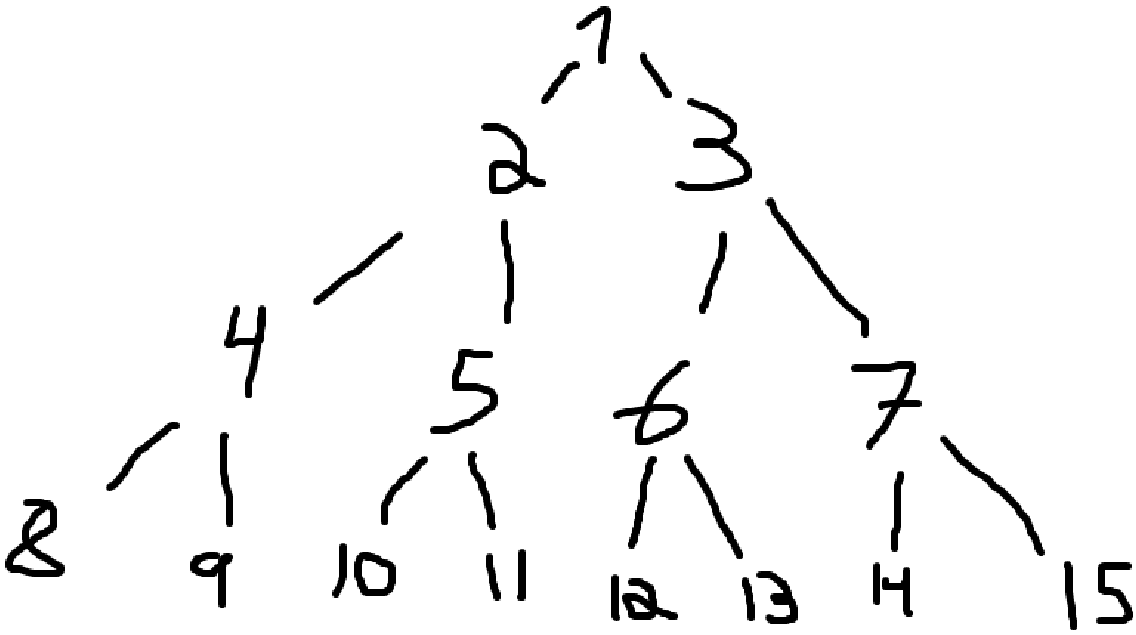
a) As duas asserções são proposições verdadeiras, e a segunda é uma justificativa correta da primeira.
b) As duas asserções são proposições verdadeiras, e a segunda não é uma justificativa correta da primeira.
c) A primeira asserção é uma proposição verdadeira, e a segunda é uma proposição falsa.
d) A primeira asserção é uma proposição falsa, e a segunda é uma proposição verdadeira.
e) As duas asserções são proposições falsas. Resposta

Questão 12

Considere um espaço de estados onde o estado inicial é o número 1 e a função sucessor para o estado *n* retorna dois estados, com os números *2n* e *2n+1*.

a. Desenhe a porção do espaço de estados correspondente aos estados 1 a 15.

- Nível 0: 1
- Nível 1: 2, 3
- Nível 2: 4, 5, 6, 7
- Nível 3: 8, 9, 10, 11, 12, 13, 14, 15



b. Suponha que o estado objetivo seja 11. Liste a ordem em que os nós serão visitados no caso da busca em extensão, da busca em profundidade limitada com limite 3 e da busca por aprofundamento iterativo.

Busca em Extensão (Largura):

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

Busca em Profundidade Limitada (limite 3):

1, 2, 4, 8, 9, 5, 10, 11

Busca por Aprofundamento Iterativo:

- Limite 0: 1
- Limite 1: 1, 2, 3
- Limite 2: 1, 2, 4, 5, 3, 6, 7
- Limite 3: 1, 2, 4, 8, 9, 5, 10, 11

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

Questão 13

Investigue vantagens e desvantagens do algoritmo A*.

Vantagens:

1. **Otimalidade**: Quando usado com uma heurística admissível, A* garante encontrar a solução ótima (de menor custo).
2. **Eficiência**: Geralmente expande menos nós que outros algoritmos de busca completos, como busca em largura.
3. **Flexibilidade**: Pode ser adaptado para diferentes problemas através da escolha apropriada da função heurística.
4. **Completude**: Se houver uma solução e o fator de ramificação for finito, A* garantidamente a encontrará.
5. **Balanceamento**: Equilibra a consideração do custo já percorrido e a estimativa do custo futuro.

Desvantagens:

1. **Requisitos de memória**: Mantém todos os nós gerados na memória, o que pode ser proibitivo para problemas de grande escala.
2. **Dependência da heurística**: O desempenho depende fortemente da qualidade da heurística escolhida.
3. **Complexidade computacional**: Em casos onde a heurística não é informativa o suficiente, pode degenerar para uma busca exaustiva.
4. **Dificuldade de paralelização**: A natureza sequencial do algoritmo dificulta implementações paralelas eficientes.
5. **Não adaptável**: Não aprende durante a execução para melhorar seu desempenho em buscas futuras.

Questão 14

Investigue outros algoritmos que são melhoria do algoritmo A*

1. IDA (Iterative Deepening A)**

- Combina A* com busca por aprofundamento iterativo
- Usa muito menos memória que A*
- Mantém a garantia de otimalidade
- Pode reexplorar estados múltiplas vezes, o que reduz a eficiência em alguns casos

2. SMA (Simplified Memory-Bounded A)**

- Versão de A* com limite de memória
- Quando a memória se esgota, o algoritmo descarta os nós menos promissores
- Mantém a otimalidade se houver memória suficiente para armazenar o caminho mais longo
- Mais adaptável a problemas com restrições de memória

3. D (Dynamic A) e D* Lite**

- Variantes para ambientes dinâmicos onde os custos podem mudar
- Replanejamento eficiente quando o ambiente muda, sem recomeçar a busca
- Muito usado em robótica e planejamento de trajetórias
- Mais complexo de implementar que A* padrão

4. Weighted A (WA)**

- Usa uma heurística ponderada: $f(n) = g(n) + w * h(n)$, onde $w > 1$
- Encontra soluções mais rapidamente, porém não garante otimalidade
- O fator de ponderação controla o balanço entre velocidade e qualidade da solução

5. Anytime Weighted A*

- Começa com uma alta ponderação (w) para encontrar rapidamente uma solução
- Continua a busca com valores decrescentes de w para melhorar a solução iterativamente
- Útil quando há restrições de tempo e precisamos de qualquer solução viável

6. Jump Point Search (JPS)

- Otimização específica para grades uniformes

- Elimina nós simétricos e redundantes
- Pode ser várias vezes mais rápido que A* em mapas de grade
- Mantém a otimalidade mas é limitado a certos tipos de problemas

7. Beam Search

- Limita o número de nós expandidos em cada nível
- Drasticamente reduz o uso de memória
- Não garante otimalidade ou completude
- Útil para problemas onde uma solução aproximada é aceitável

Questão 15

Considere a seguinte situação: Dados 5 palitos, cada jogador pode retirar 1, 2 ou 3 por turno. Perde o jogador que retira o último palito. Utilize a busca MINIMAX para verificar se MAX pode ganhar o jogo.

Árvore de Jogo MINIMAX

Representarei os estados como o número de palitos restantes. Utilizarei os valores:

- +1 para vitória de MAX
- -1 para vitória de MIN
- 0 para empate (se houver)

Começo com MAX e 5 palitos disponíveis:

Estado: 5 palitos (MAX joga)

MAX pode escolher remover 1, 2 ou 3 palitos:

1. **MAX remove 1 palito** → Restam 4 palitos (MIN joga)
   - MIN remove 1 palito → Restam 3 palitos (MAX joga)
     - MAX remove 1 palito → Restam 2 palitos (MIN joga)
       - MIN remove 1 palito → Resta 1 palito (MAX joga)
         - MAX remove 1 palito → Restam 0 palitos (MAX perde) = **-1**
       - MIN remove 2 palitos → Restam 0 palitos (MIN perde) = **+1**
     - MAX remove 2 palitos → Resta 1 palito (MIN joga)
       - MIN remove 1 palito → Restam 0 palitos (MIN perde) = **+1**
     - MAX remove 3 palitos → Restam 0 palitos (MAX perde) = **-1**
   - MIN remove 2 palitos → Restam 2 palitos (MAX joga)
     - MAX remove 1 palito → Resta 1 palito (MIN joga)
       - MIN remove 1 palito → Restam 0 palitos (MIN perde) = **+1**
     - MAX remove 2 palitos → Restam 0 palitos (MAX perde) = **-1**
   - MIN remove 3 palitos → Resta 1 palito (MAX joga)
     - MAX remove 1 palito → Restam 0 palitos (MAX perde) = **-1**
2. **MAX remove 2 palitos** → Restam 3 palitos (MIN joga)
   - MIN remove 1 palito → Restam 2 palitos (MAX joga)
     - MAX remove 1 palito → Resta 1 palito (MIN joga)
       - MIN remove 1 palito → Restam 0 palitos (MIN perde) = **+1**
     - MAX remove 2 palitos → Restam 0 palitos (MAX perde) = **-1**
   - MIN remove 2 palitos → Resta 1 palito (MAX joga)
     - MAX remove 1 palito → Restam 0 palitos (MAX perde) = **-1**
   - MIN remove 3 palitos → Restam 0 palitos (MIN perde) = **+1**
3. **MAX remove 3 palitos** → Restam 2 palitos (MIN joga)
   - MIN remove 1 palito → Resta 1 palito (MAX joga)
     - MAX remove 1 palito → Restam 0 palitos (MAX perde) = **-1**
   - MIN remove 2 palitos → Restam 0 palitos (MIN perde) = **+1**

Resolução bottom-up do MINIMAX

Agora, vamos calcular os valores MINIMAX de baixo para cima:

Nível dos estados com 1 palito (MAX joga)

- MAX só pode remover 1 palito → Valor = -1 (MAX perde)

Nível dos estados com 2 palitos (MIN joga)

- MIN pode remover 1 palito → Leva a um estado de valor -1

- MIN pode remover 2 palitos → Leva a um estado de valor +1
- MIN escolhe o menor valor: -1

Nível dos estados com 3 palitos (MAX joga)

- MAX pode remover 1 palito → Leva a um estado de valor -1
- MAX pode remover 2 palitos → Leva a um estado de valor -1
- MAX pode remover 3 palitos → MAX ganha diretamente (+1)
- MAX escolhe o maior valor: +1

Nível dos estados com 4 palitos (MIN joga)

- MIN pode remover 1 palito → Leva a um estado de valor +1
- MIN pode remover 2 palitos → Leva a um estado de valor -1
- MIN pode remover 3 palitos → Leva a um estado de valor -1
- MIN escolhe o menor valor: -1

Nível inicial: 5 palitos (MAX joga)

- MAX pode remover 1 palito → Leva a um estado de valor -1
- MAX pode remover 2 palito → Leva a um estado de valor +1
- MAX pode remover 3 palito → Leva a um estado de valor -1
- MAX escolhe o maior valor: +1

Conclusão

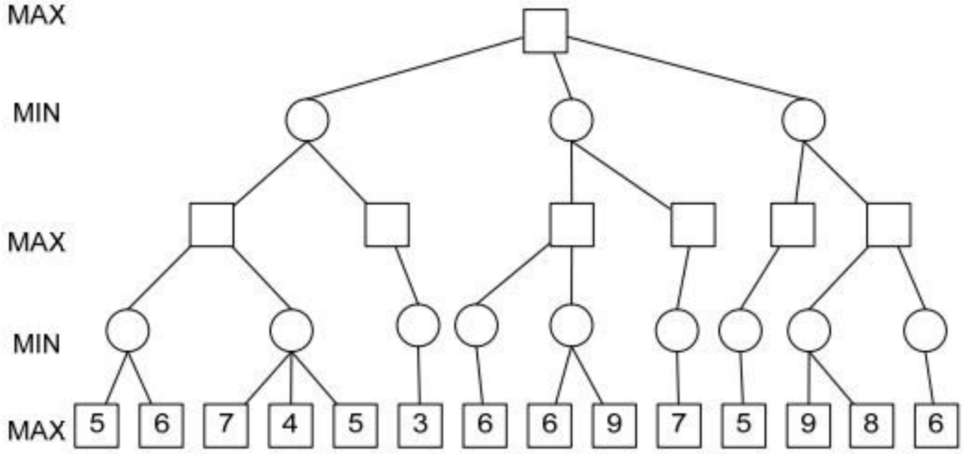
A análise MINIMAX mostra que o valor do estado inicial (5 palitos) é +1, o que significa que MAX pode garantir a vitória jogando otimamente. A jogada ótima para MAX é remover 2 palitos no primeiro turno, deixando 3 palitos para MIN.

Se MIN remover 1 ou 2 palitos, MAX pode remover respectivamente 2 ou 1 palito, deixando 0 palitos para MIN e vencendo o jogo. Se MIN remover 3 palitos, já terá perdido por definição das regras do jogo.

Portanto, MAX pode ganhar o jogo usando a estratégia determinada pelo algoritmo MINIMAX.

Questão 16

Considere a árvore minimax abaixo, representando um jogo onde queremos maximizar o valor da função de avaliação estática:



Assinale a alternativa que apresenta a quantidade de folhas que não deverão ser visitados em uma busca da melhor jogada se a estratégia de **poda alfa-beta** for utilizada.

a) 5 Resposta

b) 8

c) 9

d) 10

e) 11