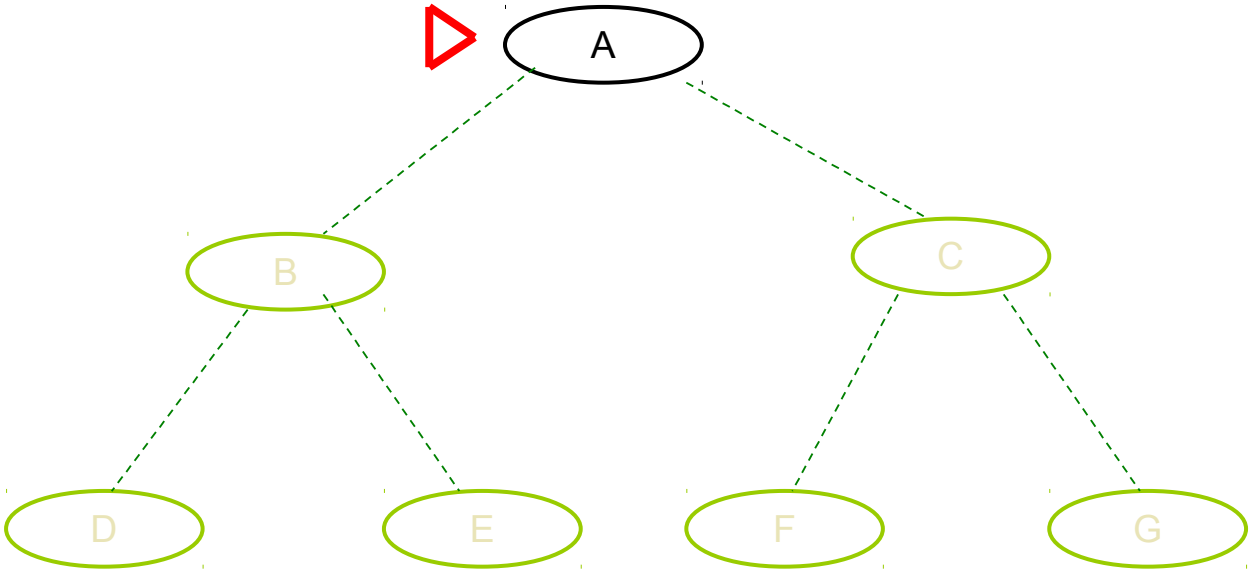
# Estratégias de pesquisa não informada

- As estratégias de pesquisa não informada usam apenas a informação que faz parte da definição do problema
  - Pesquisa em extensão ou largura (*breadth-first*)
  - Pesquisa de custo uniforme (*uniform-cost*)
  - Pesquisa em profundidade (*depth-first*)
  - Pesquisa em profundidade limitada (*depth-limited*)
  - Pesquisa por aprofundamento iterativo (*iterative deepening*)

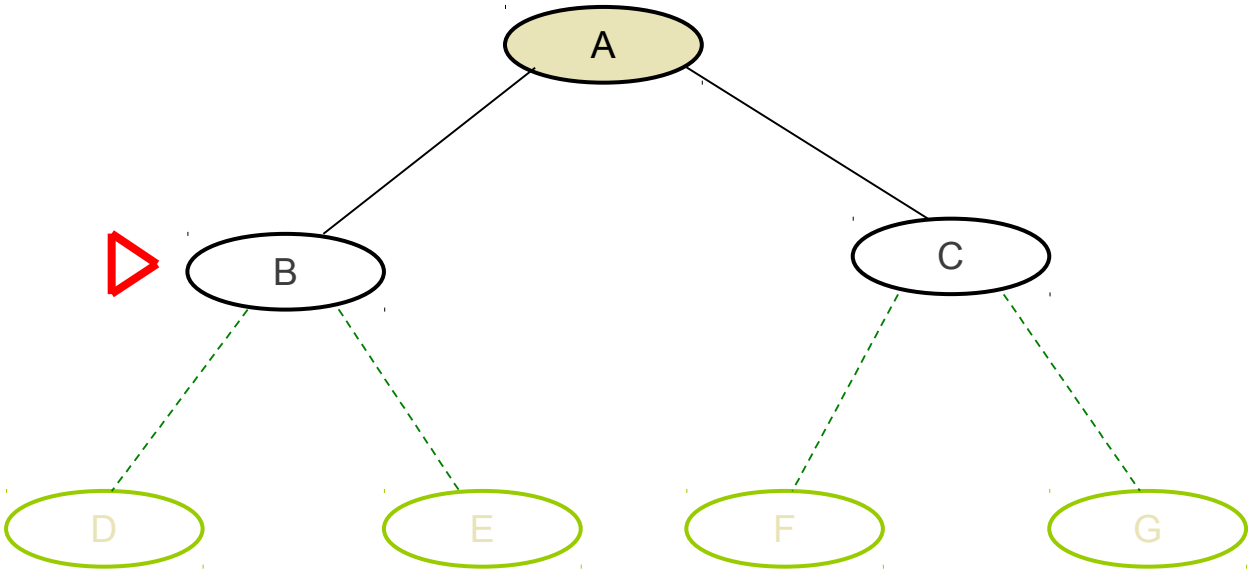
# Pesquisa em extensão ou largura

- Expande sempre o nó de menor profundidade
- Implementação:
  - fronteira é uma lista FIFO: os novos nós vão para o fim da fila



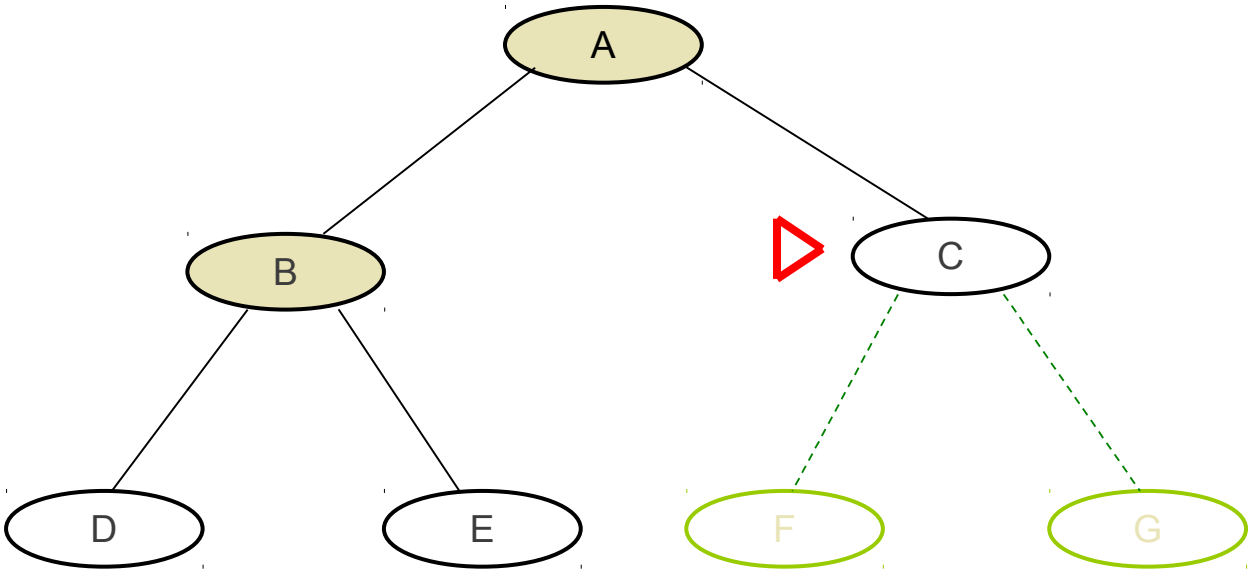
# Pesquisa em extensão ou largura (cont.)

- Expande sempre o nó de menor profundidade
- Implementação:
  - fronteira é uma lista FIFO: os novos nós vão para o fim da fila



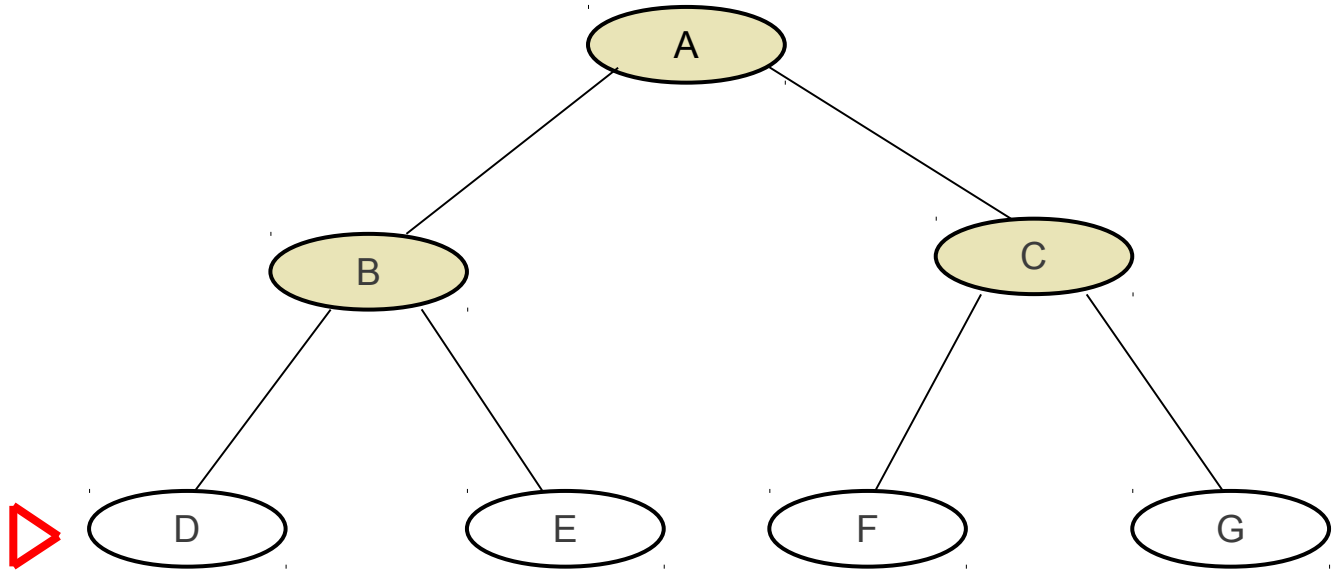
# Pesquisa em extensão ou largura (cont.)

- Expande sempre o nó de menor profundidade
- Implementação:
  - fronteira é uma lista FIFO: os novos nós vão para o fim da fila



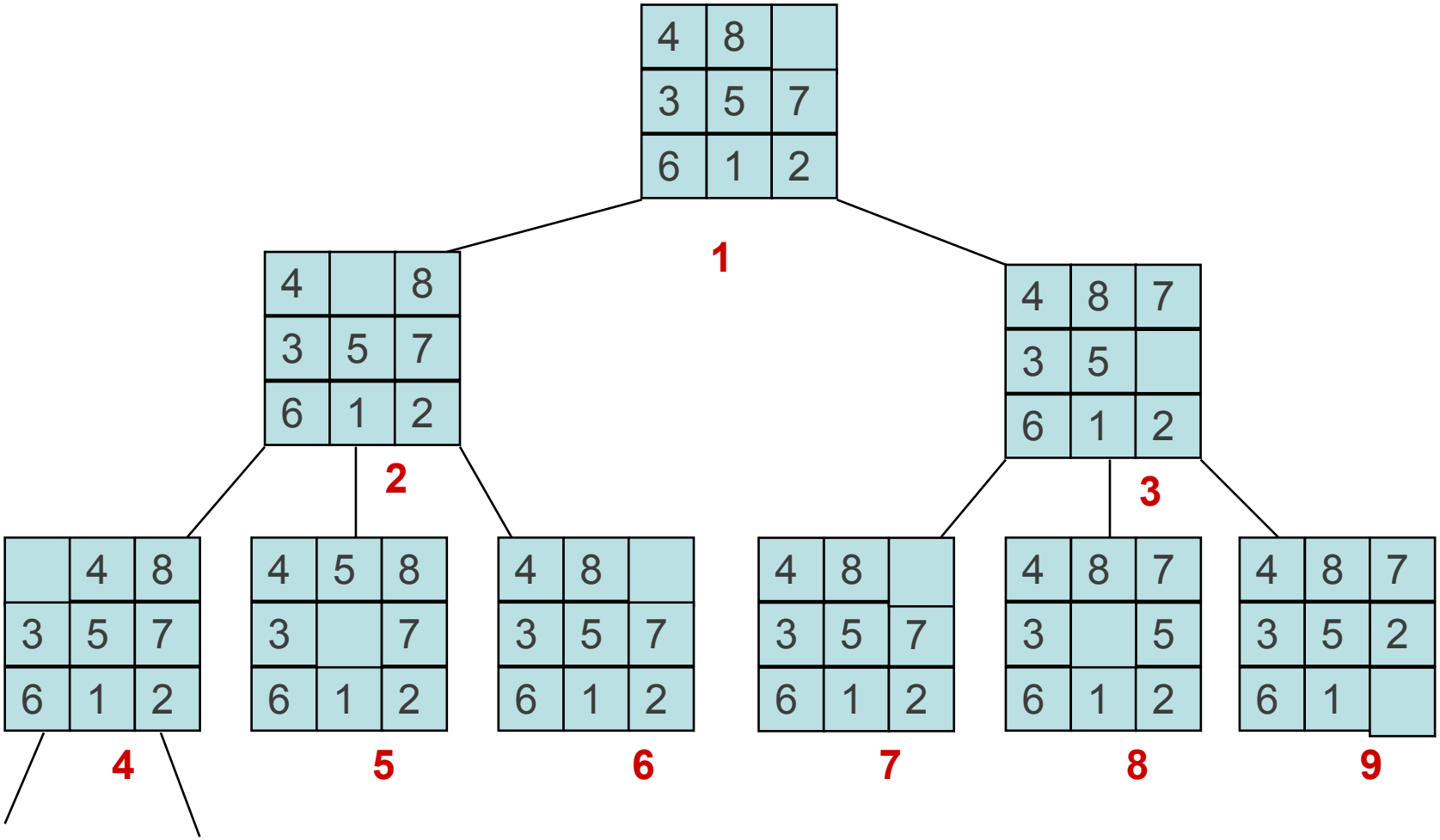
# Pesquisa em extensão ou largura (cont.)

- Expande sempre o nó de menor profundidade
- Implementação:
  - fronteira é uma lista FIFO: os novos nós vão para o fim da fila




**etc...**

# Pesquisa em extensão ou largura (cont.)




etc...

# Pesquisa em extensão ou largura -- propriedades

- Completa ??
  - Sim, se b for finito
- Tempo ??
  - $1 + b + b^2 + b^3 + ... + b^d + b^{d+1} = O( b^{d+1})$
- Espaço ??
  - $O( b^{d+1})$ – os nós são todos guardados em memória
- Ótimo ??
  - Sim, se o custo de cada passo for 1
- Exemplo: fator de ramificação = 10:

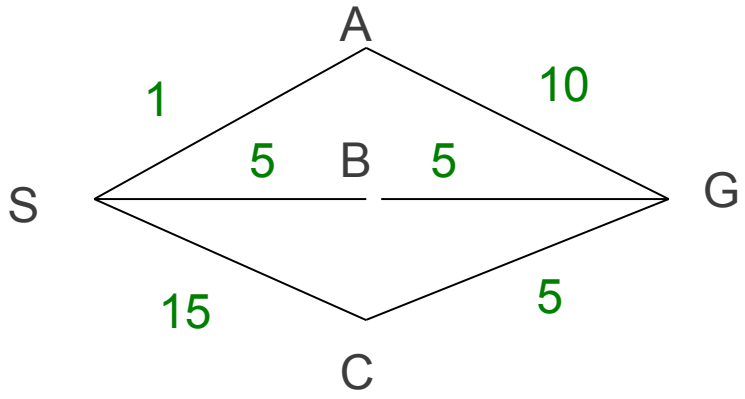
| Profundidade | Nº nós | Tempo | Memória |
|---|---|---|---|
| 0 | 1 | 1 ms | 100 bytes |
| 2 | 111 | 100 ms | 11 kbyte |
| 4 | 11 111 | 11 s | 1 Mbyte |
| 8 | $10^8$ | 31 h | 11 Gbyte |
| 14 | $10^{14}$ | 3 500 anos | 11 111 Tbyte |

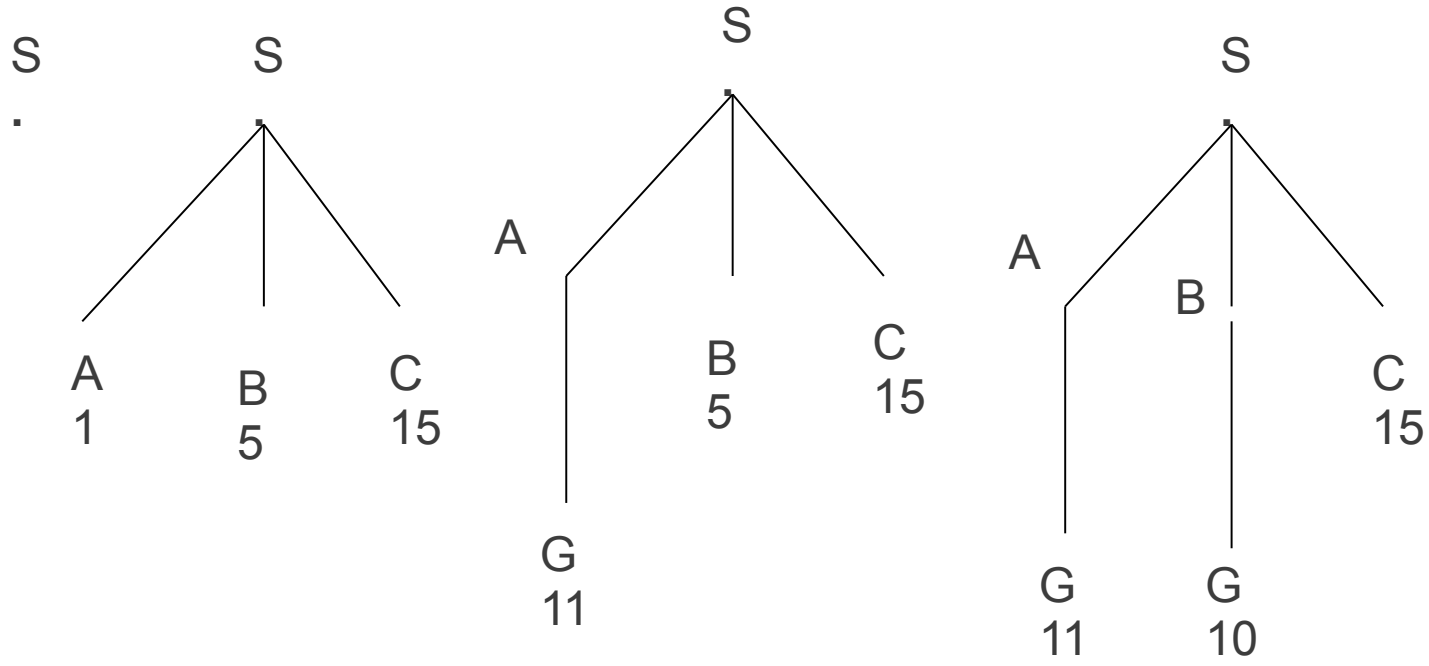
# Pesquisa de custo uniforme

- Expande o nó com menor custo do caminho
- Implementação:
  - fronteira: lista ordenada pelo custo do caminho
- Se os passos tiverem todos o mesmo custo, é equivalente à pesquisa em extensão ou largura

- Completo??
  - Sim, se o custo de cada passo >= ε
    - Pode ficar preso num ciclo de passos com custo 0
- Tempo ??
  - $O(b^{1+C^*/\varepsilon})$ em que C* é o custo da solução ótima $O(b^d)$
- Espaço ??
  - $O(b^{1+C^*/\varepsilon})$ em que C* é o custo da solução ótima $O(b^d)$
- Ótimo ??
  - Sim; os nós são expandidos por valores crescentes de g(n)

# Pesquisa de custo uniforme (cont.)



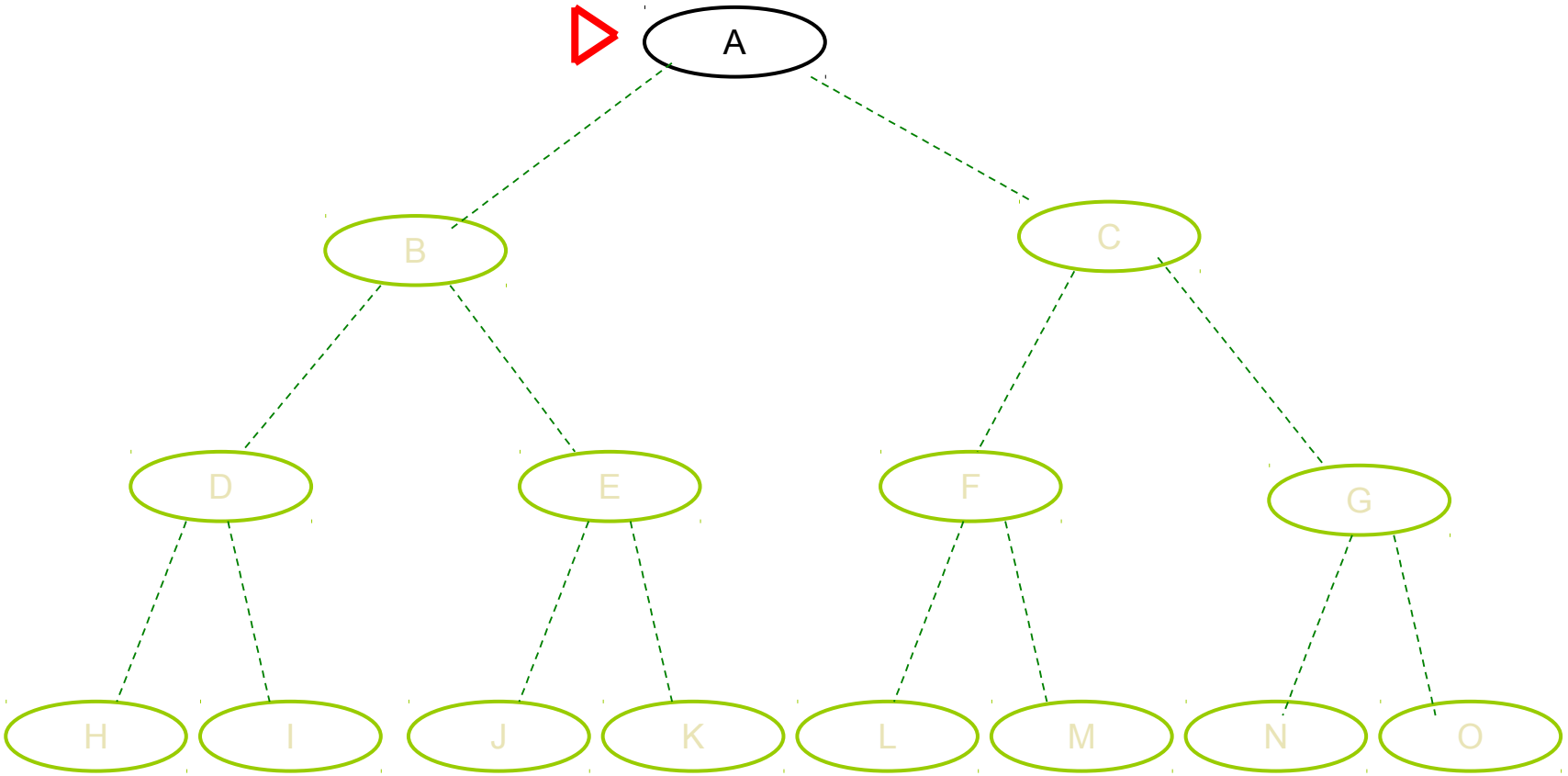

- **Um nó só é a solução quando for o MENOR da lista de nós fronteira**

# Pesquisa de custo uniforme - otimização

- A eficiência da pesquisa pode ser melhorada se garantirmos que não há nós de estados duplicados na lista de nós fronteira, nem repetimos a pesquisa de nós com estados que já foram explorados
  - Antes de juntar um nó à lista de nós fronteira, verificar se o estado já faz parte desta lista, e escolher o nó com menor custo
  - Registar numa coleção os nós já explorados, e verificar se o novo nó tem um estado que faz parte desta lista, e portanto já foi explorado
  - Para aumentar a eficiência, e dado que a coleção não precisa de estar ordenada, pode ser usado um HashMap

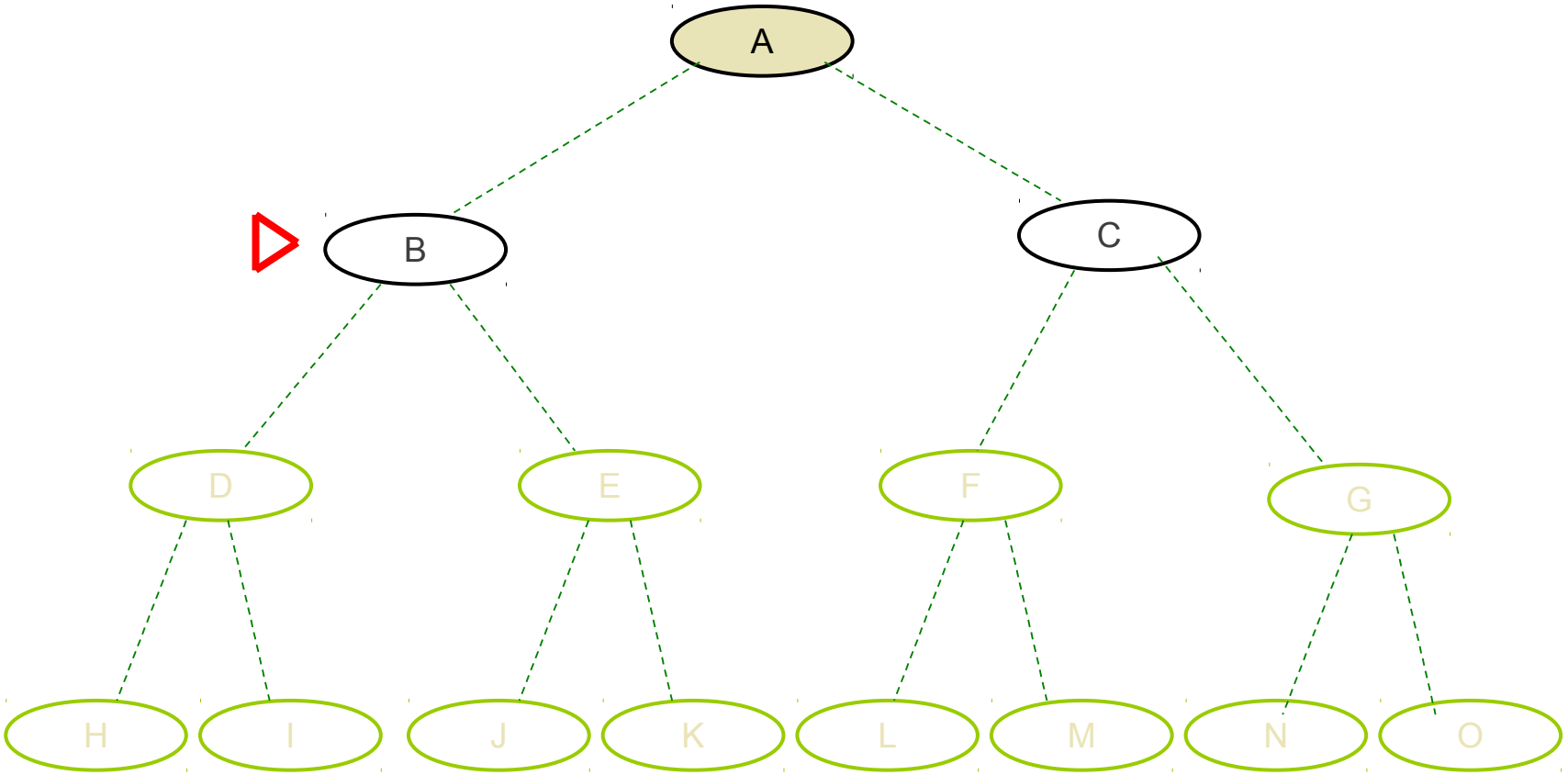
# Pesquisa em profundidade

- Expande o nó mais profundo da lista de nós fronteira
- Implementação:
  - fronteira é uma lista LIFO, isto é, os nós novos ficam à cabeça



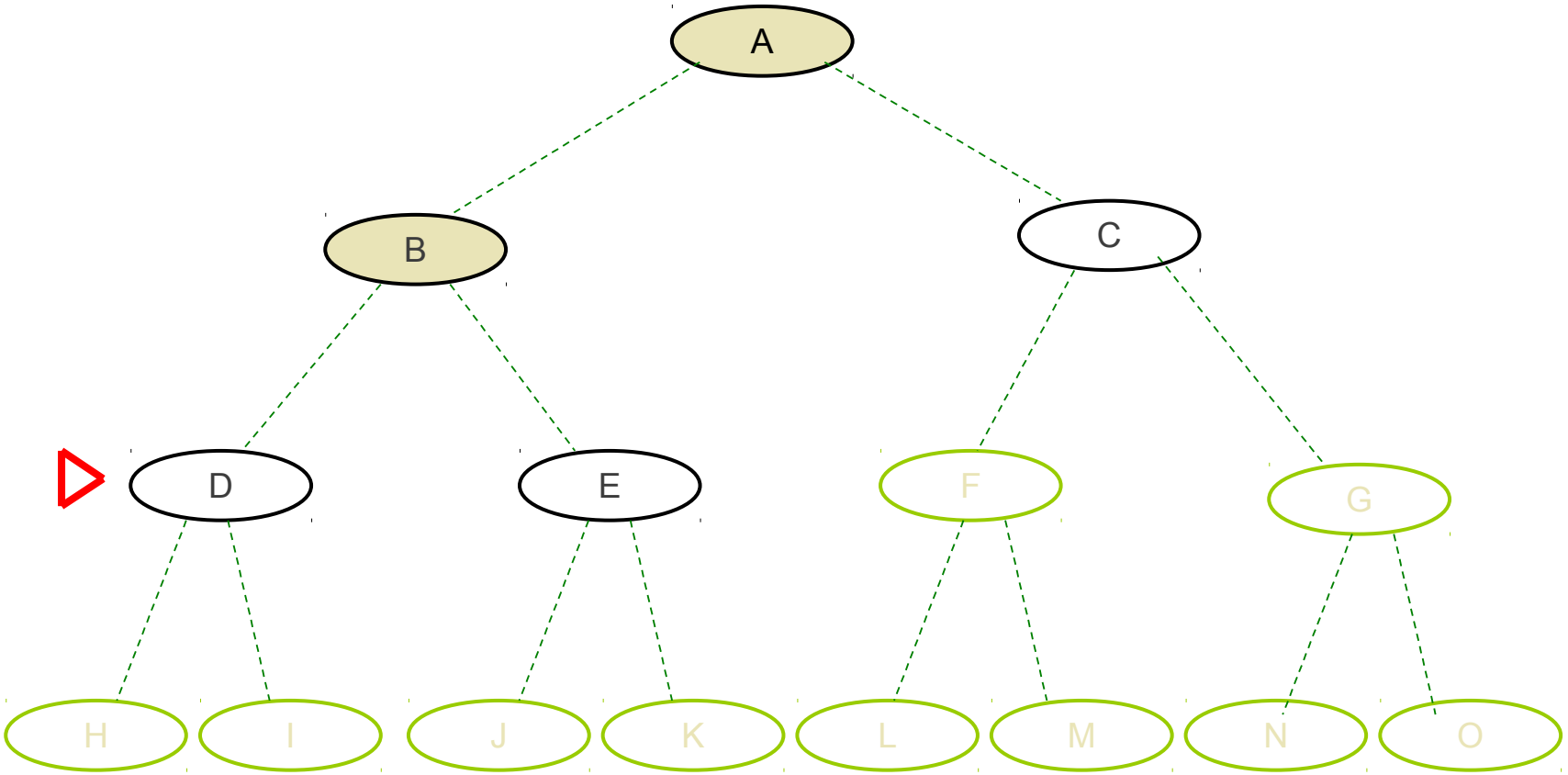
# Pesquisa em profundidade (cont.)

- Expande o nó mais profundo da lista de nós fronteira
- Implementação:
  - fronteira é uma lista LIFO, isto é, os nós novos ficam à cabeça



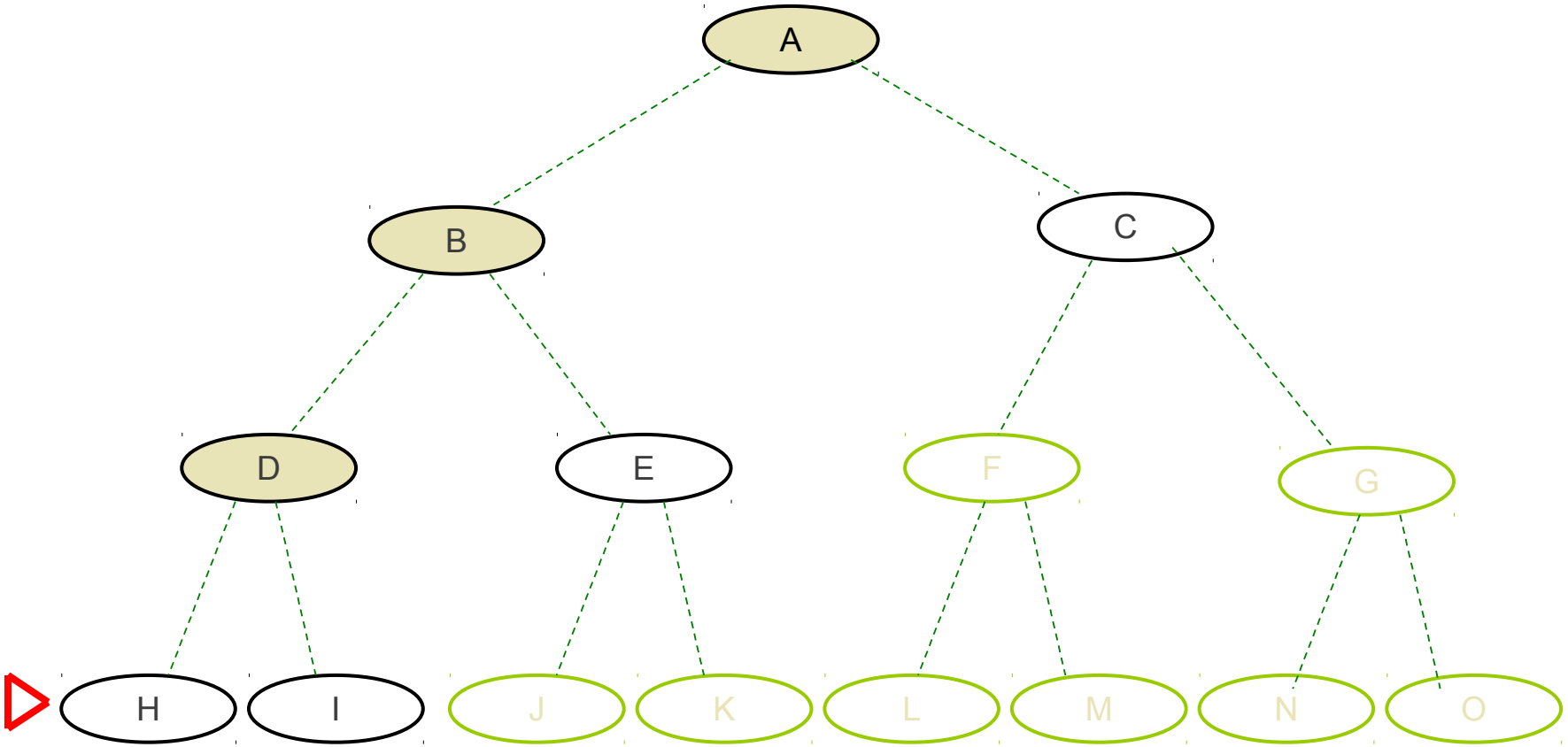
# Pesquisa em profundidade (cont.)

- Expande o nó mais profundo da lista de nós fronteira
- Implementação:
  - fronteira é uma lista LIFO, isto é, os nós novos ficam à cabeça



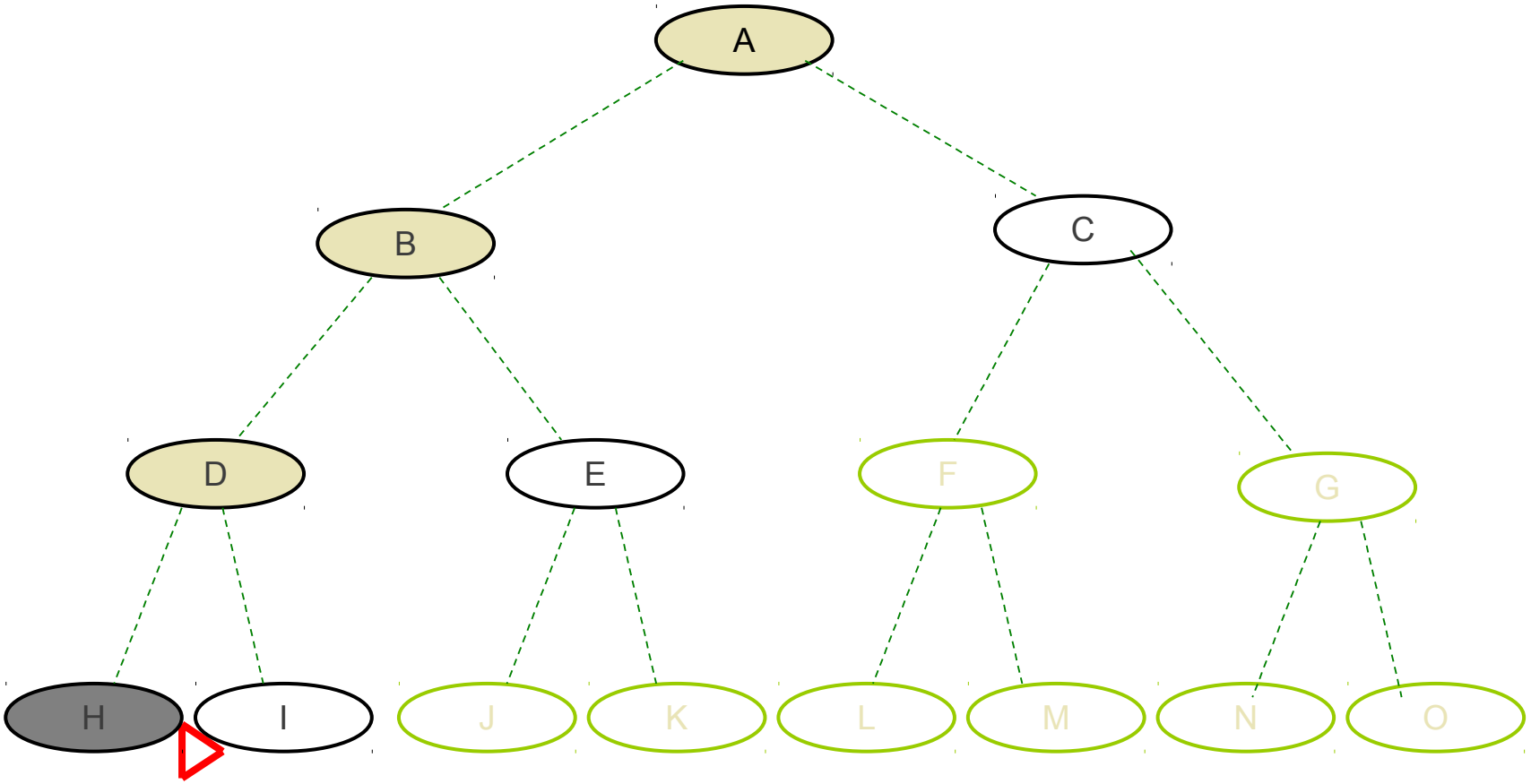
# Pesquisa em profundidade (cont.)

- Expande o nó mais profundo da lista de nós fronteira
- Implementação:
  - fronteira é uma lista LIFO, isto é, os nós novos ficam à cabeça



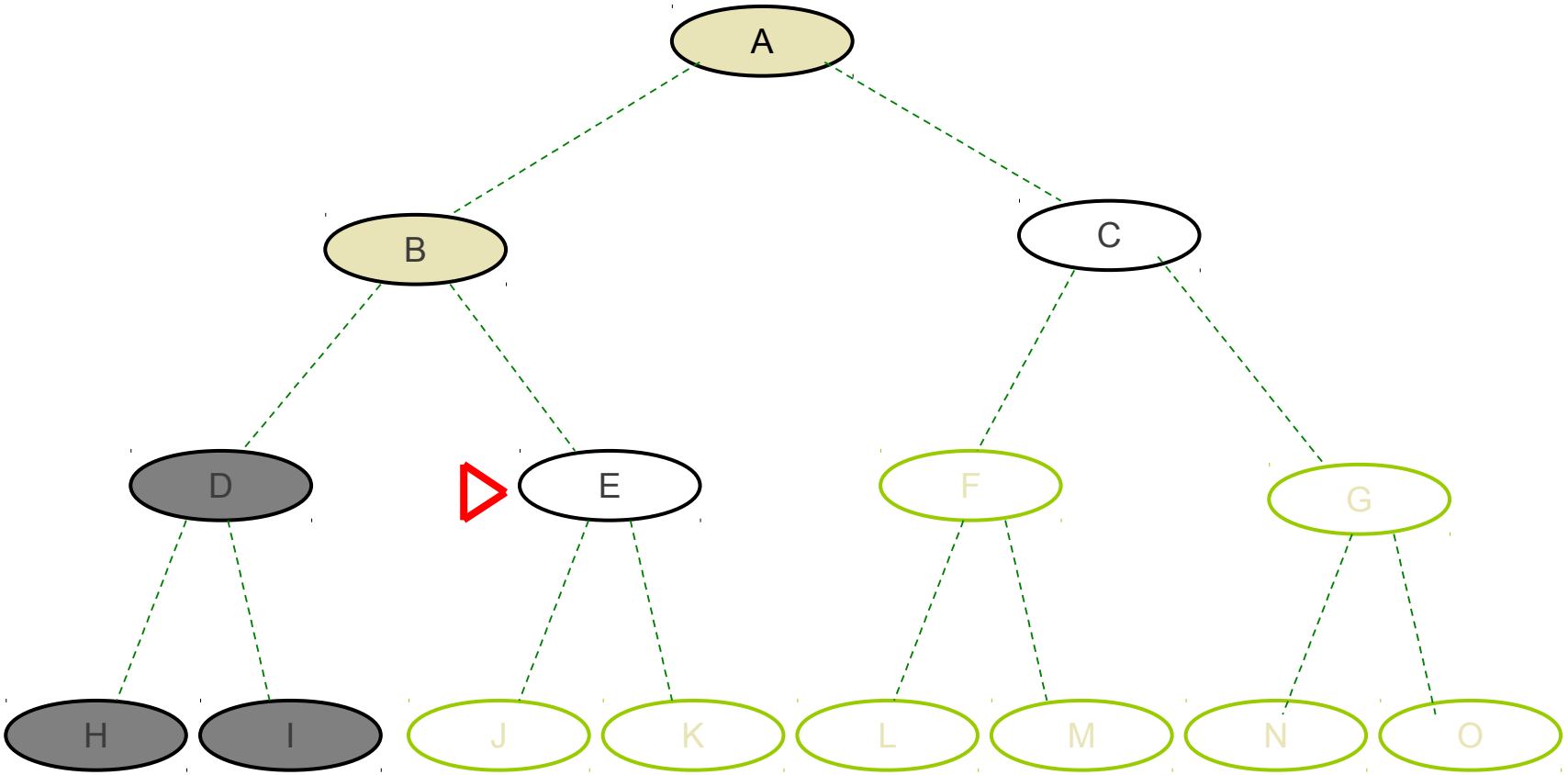
# Pesquisa em profundidade (cont.)

- Expande o nó mais profundo da lista de nós fronteira
- Implementação:
  - fronteira é uma lista LIFO, isto é, os nós novos ficam à cabeça



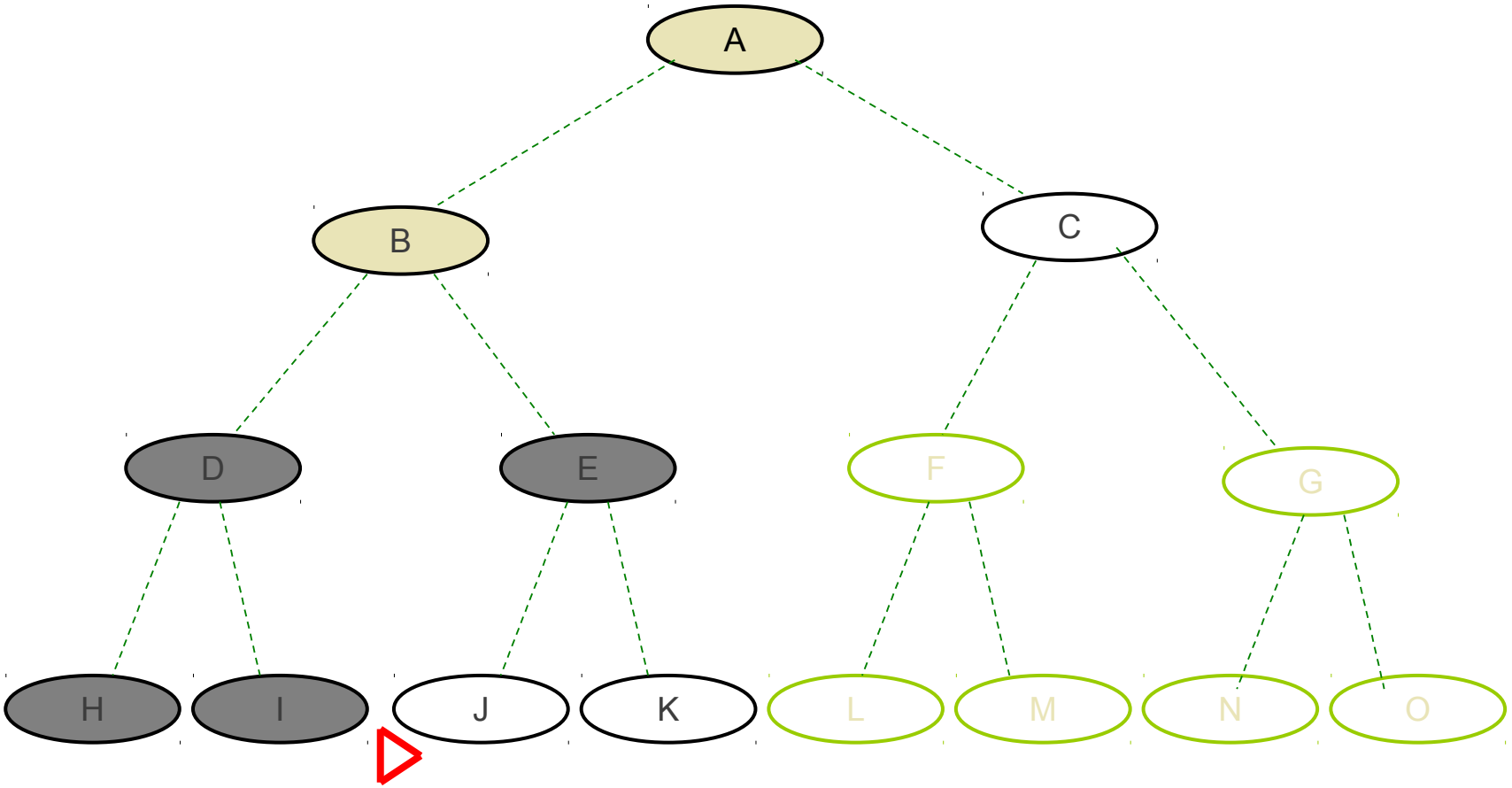
# Pesquisa em profundidade (cont.)

- Expande o nó mais profundo da lista de nós fronteira
- Implementação:
  - fronteira é uma lista LIFO, isto é, os nós novos ficam à cabeça



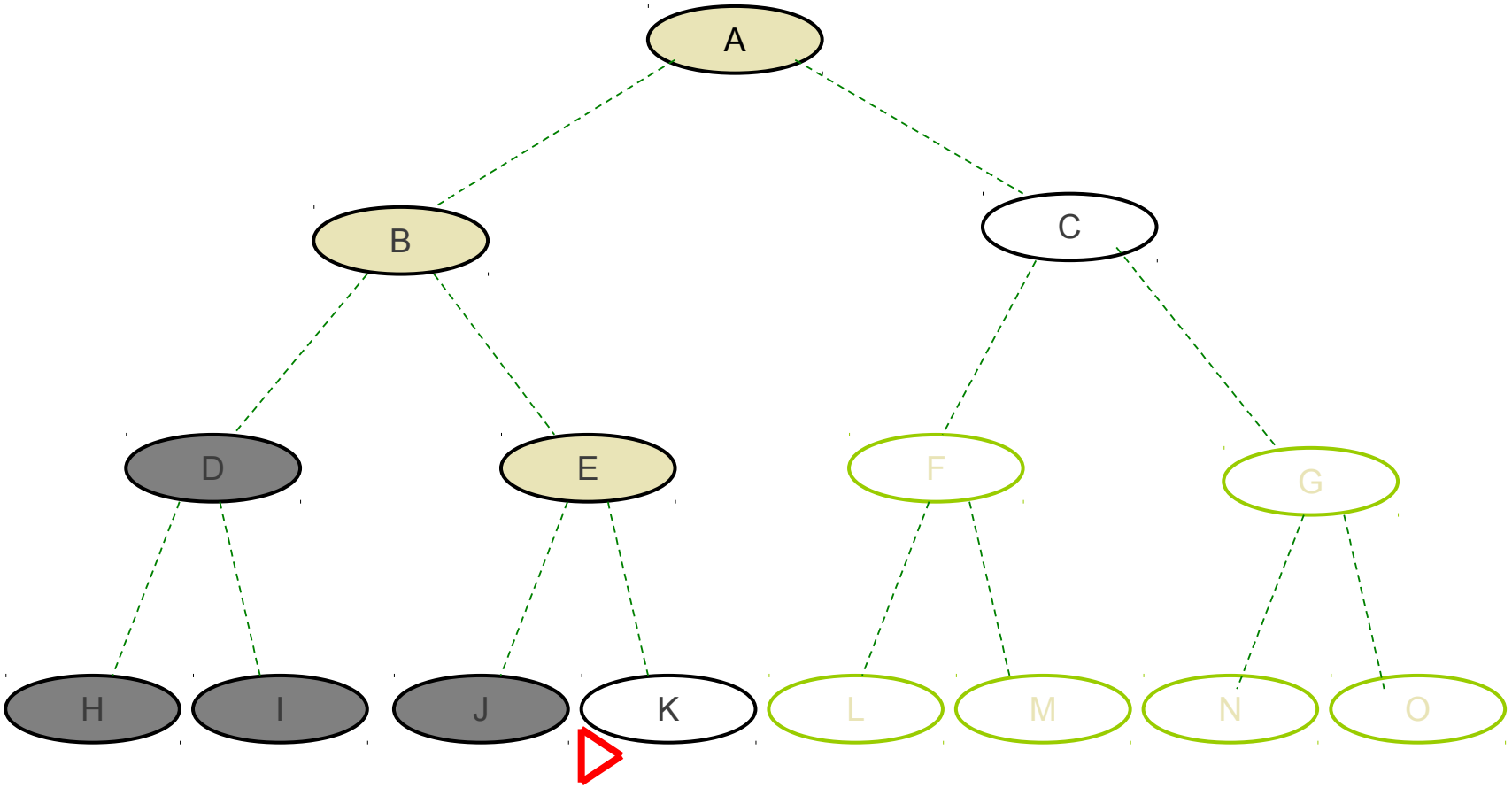
# Pesquisa em profundidade (cont.)

- Expande o nó mais profundo da lista de nós fronteira
- Implementação:
  - fronteira é uma lista LIFO, isto é, os nós novos ficam à cabeça



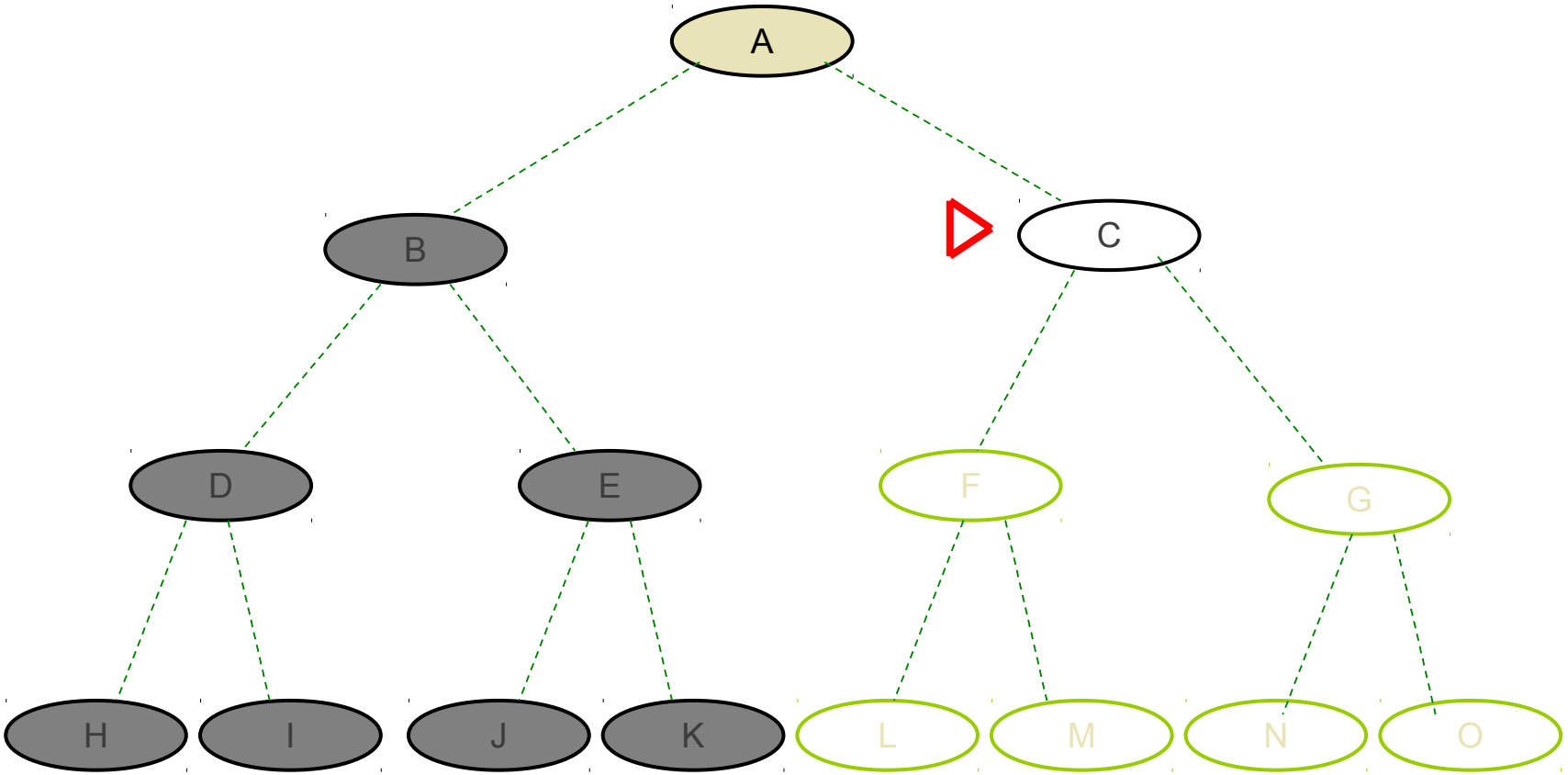
# Pesquisa em profundidade (cont.)

- Expande o nó mais profundo da lista de nós fronteira
- Implementação:
  - fronteira é uma lista LIFO, isto é, os nós novos ficam à cabeça



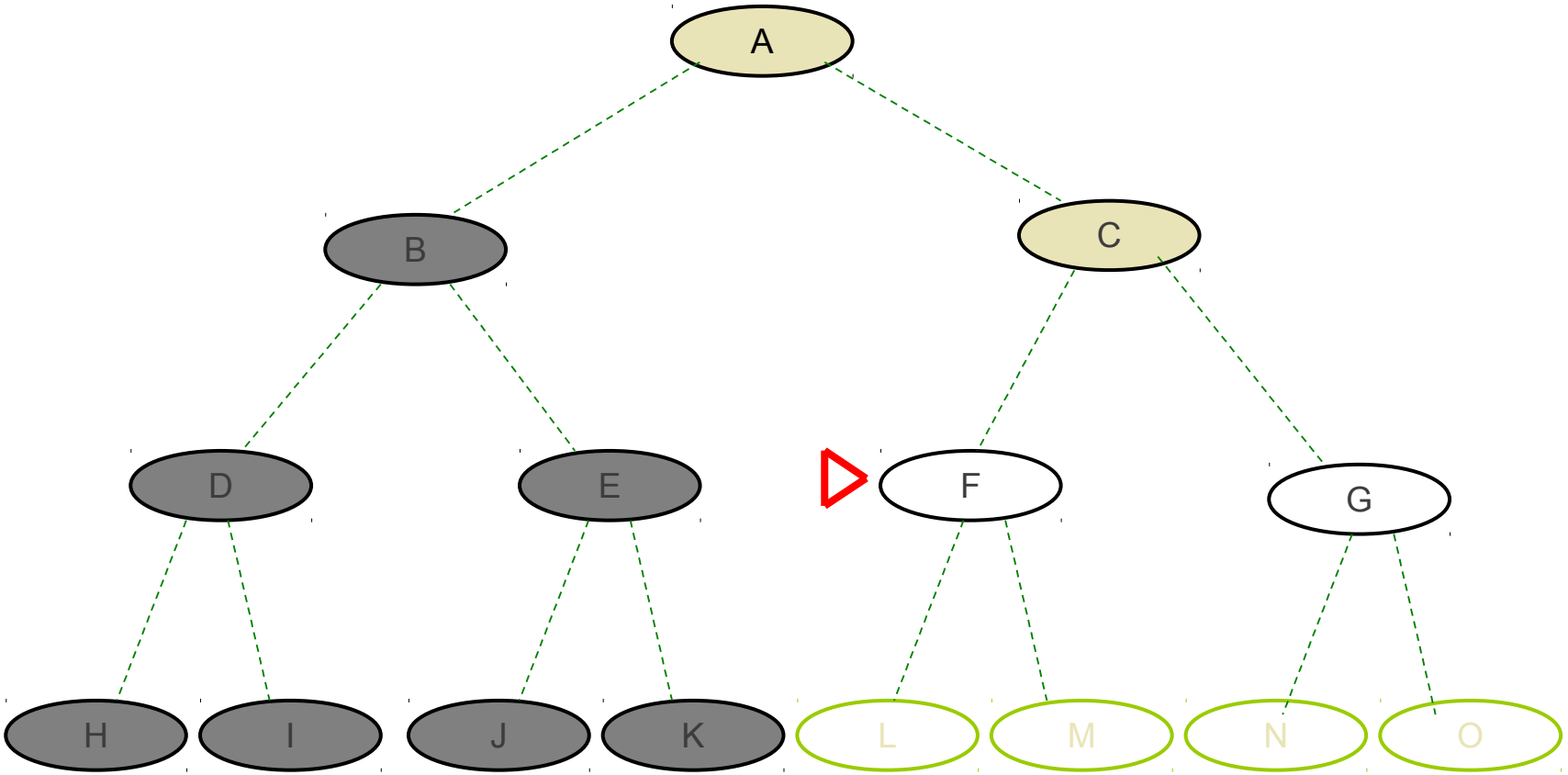
# Pesquisa em profundidade (cont.)

- Expande o nó mais profundo da lista de nós fronteira
- Implementação:
  - fronteira é uma lista LIFO, isto é, os nós novos ficam à cabeça



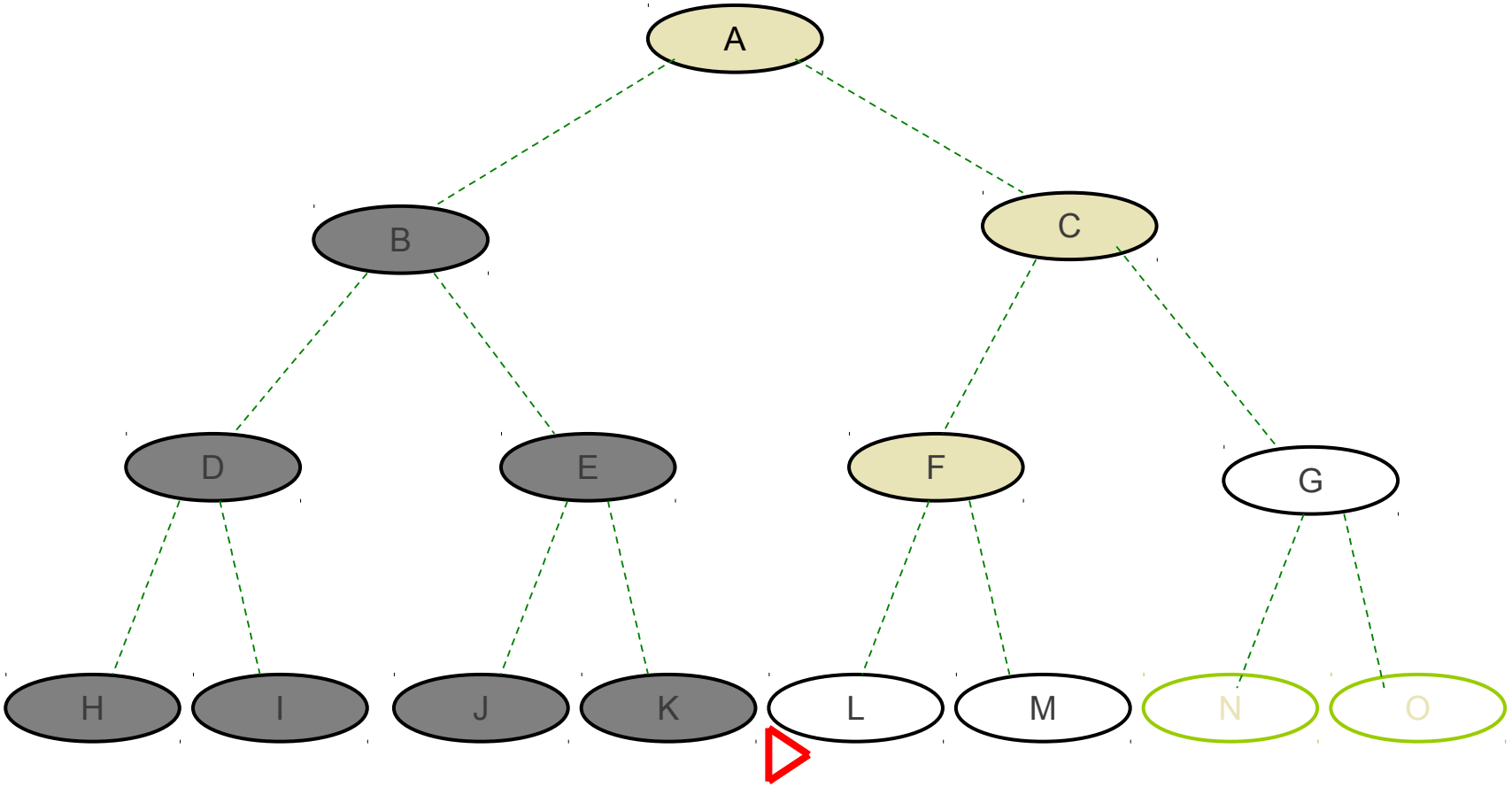
# Pesquisa em profundidade (cont.)

- Expande o nó mais profundo da lista de nós fronteira
- Implementação:
  - fronteira é uma lista LIFO, isto é, os nós novos ficam à cabeça



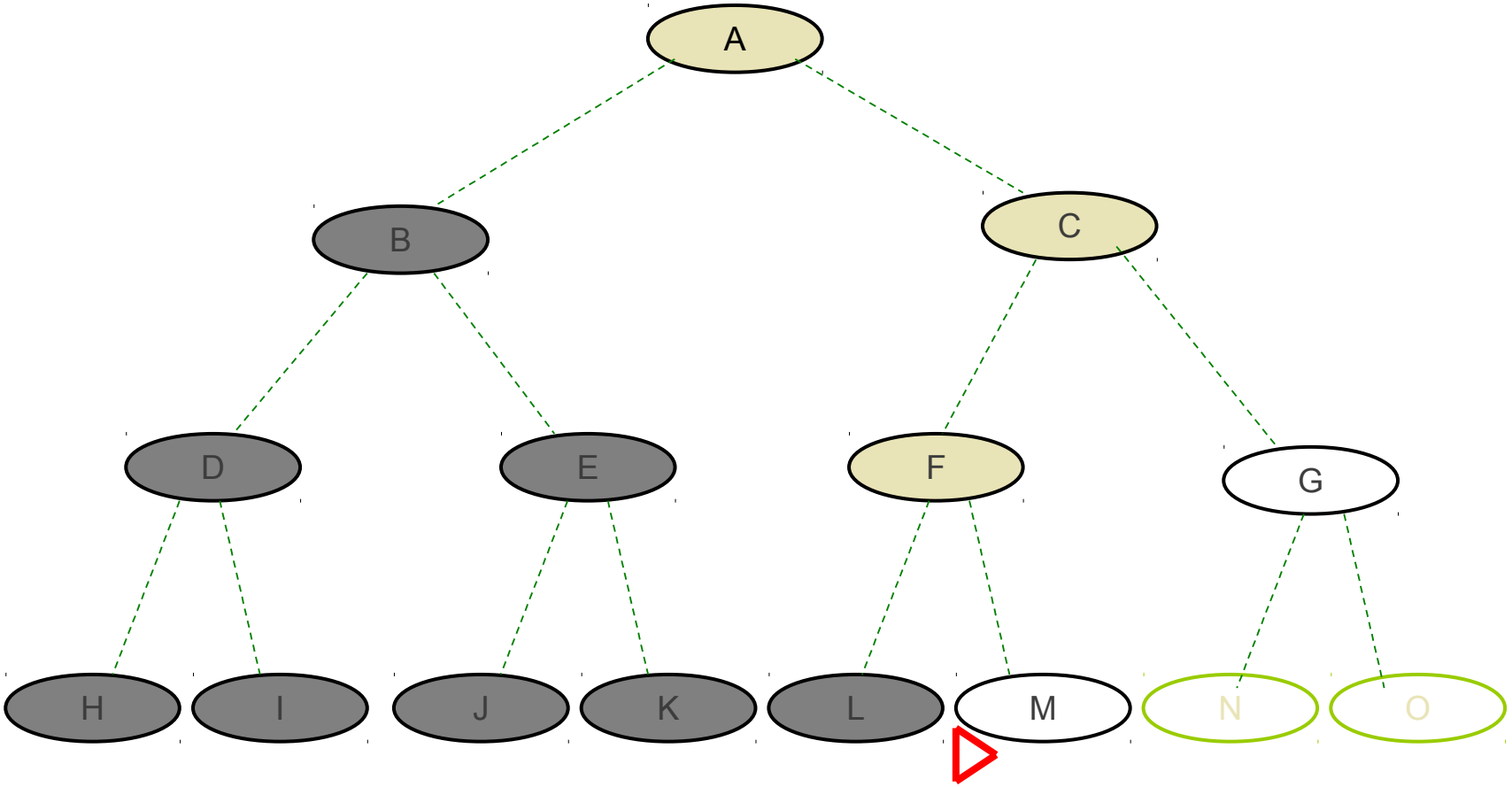
# Pesquisa em profundidade (cont.)

- Expande o nó mais profundo da lista de nós fronteira
- Implementação:
  - fronteira é uma lista LIFO, isto é, os nós novos ficam à cabeça

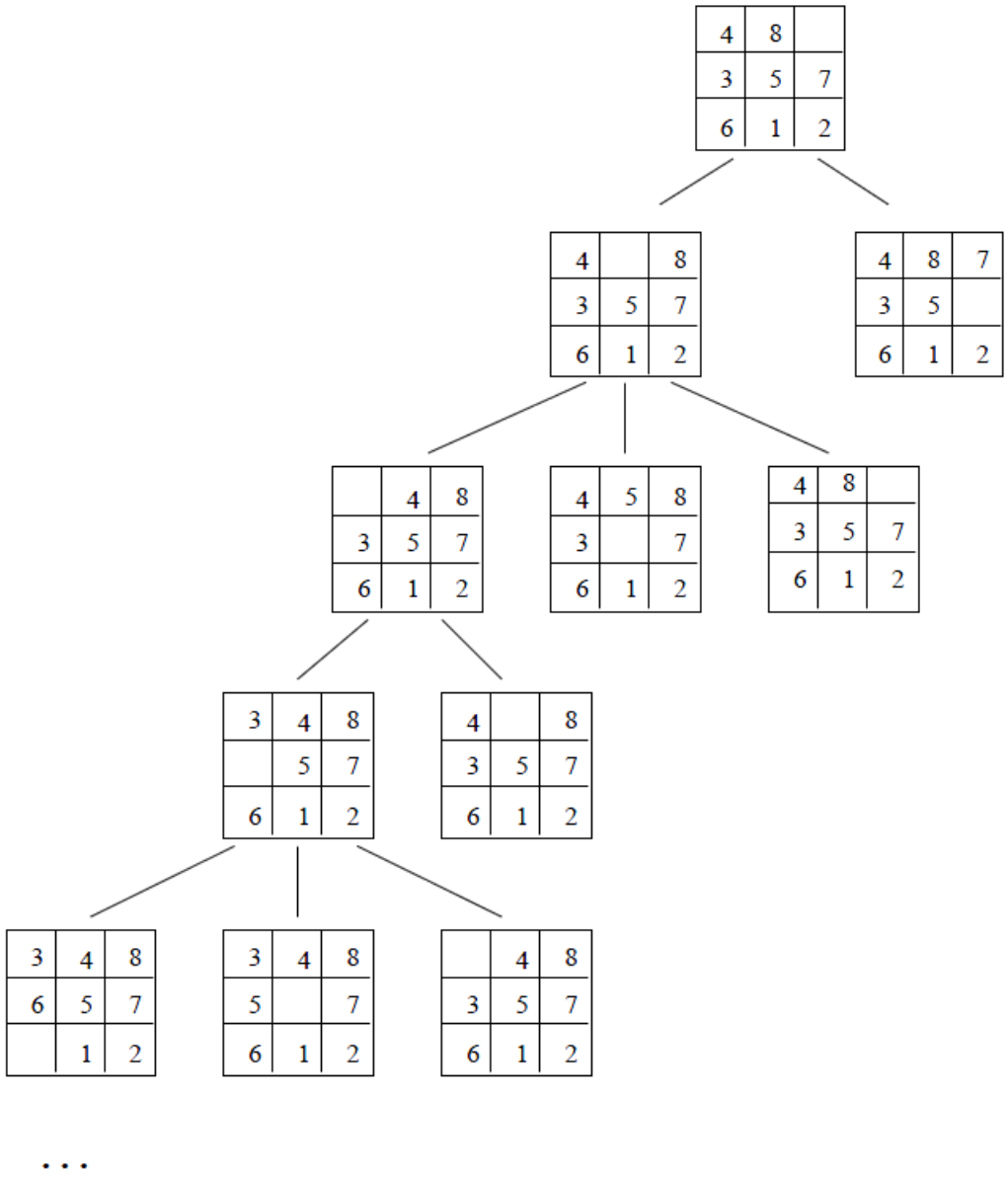


# Pesquisa em profundidade (cont.)

- Expande o nó mais profundo da lista de nós fronteira
- Implementação:
  - fronteira é uma lista LIFO, isto é, os nós novos ficam à cabeça



# Pesquisa em profundidade (cont.)



# Pesquisa em profundidade (cont.)

- Completo ??
  - Não. Falha em espaços com profundidade ilimitada, ou quando existem ciclos
  - Se for modificado para detetar e eliminar estados repetidos ao longo do caminho (que originam ciclos), então é completo em espaços finitos (limitados)
- Tempo ??
  - $O( b^m)$. Horrível se m >> d
  - Se houver muitas soluções, pode ser muito eficiente, porque aumenta a probabilidade de encontrar uma solução
- Espaço ??
  - O( b m) isto é, linear!
- Ótimo ??
  - Não

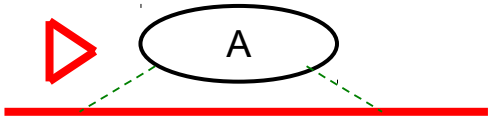
# Pesquisa em profundidade limitada

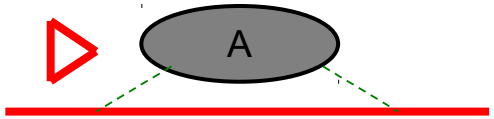
- igual a pesquisa em profundidade, mas com limite de profundidade l, isto é, nós com profundidade l não têm sucessores

- Completo??
  - Sim, mas apenas se a profundidade da solução, d, for menor ou igual a l
- Tempo ??
  - $O( b^l)$, se encontrar uma solução
- Espaço ??
  - O( b l)
- Ótimo ??
  - Não

# Pesquisa por aprofundamento iterativo

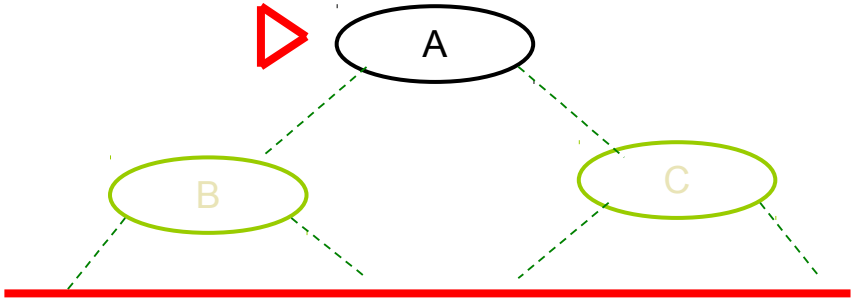
- Repete-se a pesquisa em profundidade limitada, para valores sucessivamente crescentes do limite l
- l = 0

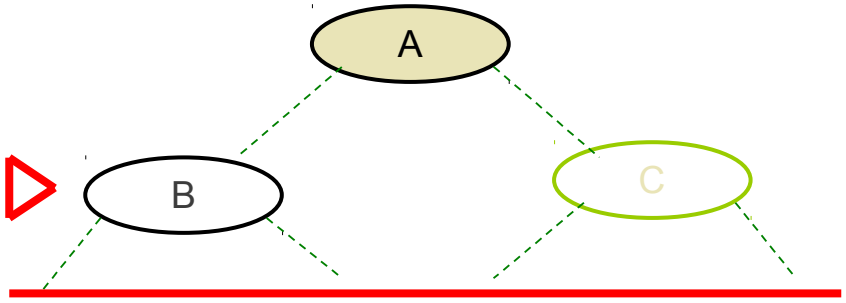


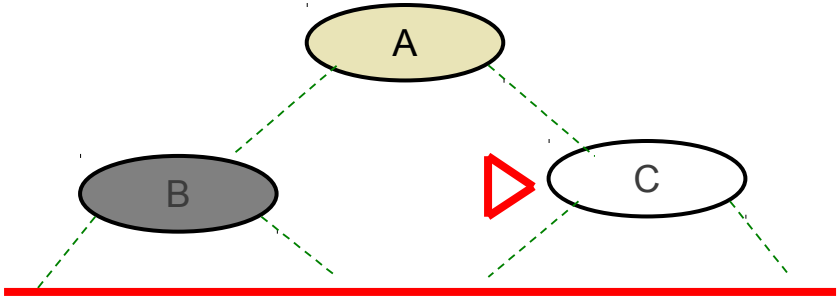



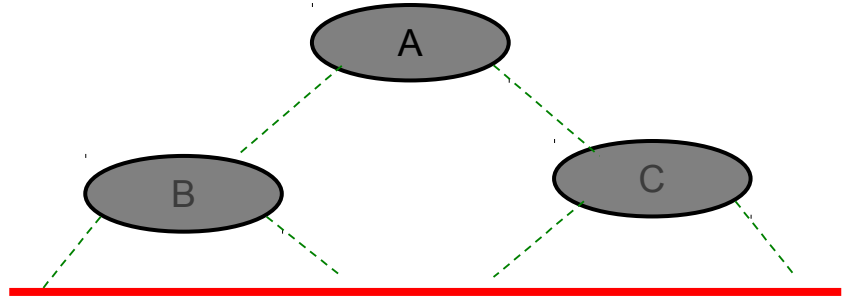
# Pesquisa por aprofundamento iterativo (cont.)

- l = 1



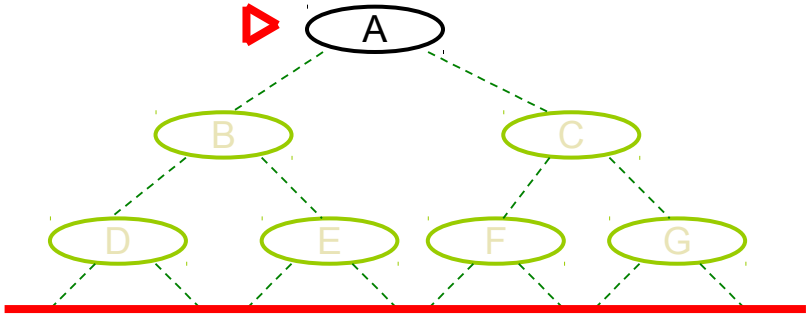



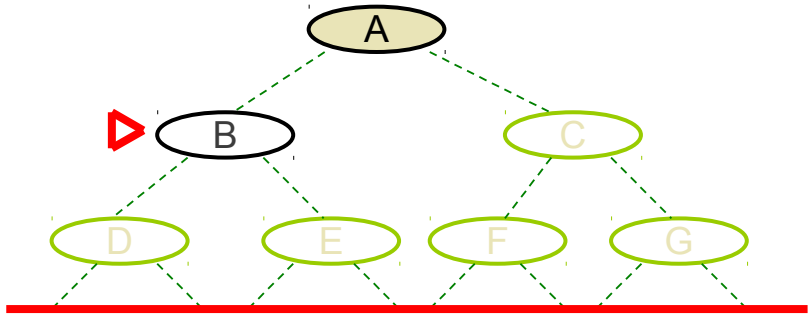



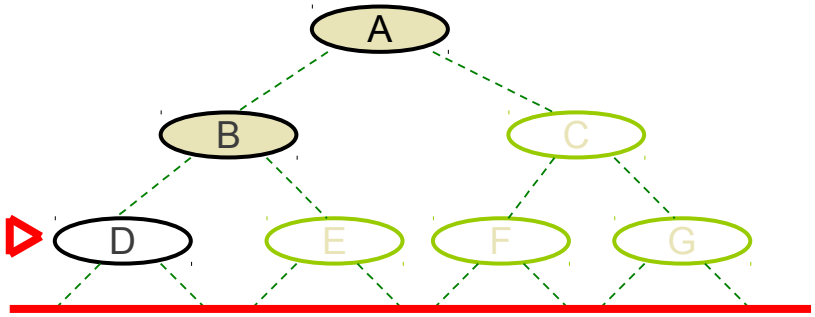



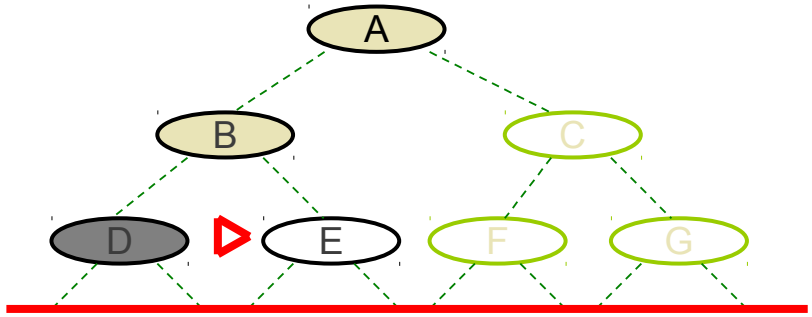
# Pesquisa por aprofundamento iterativo (cont.)

- l = 2



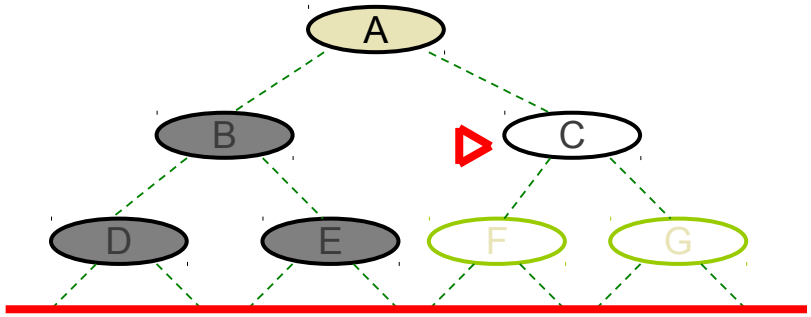



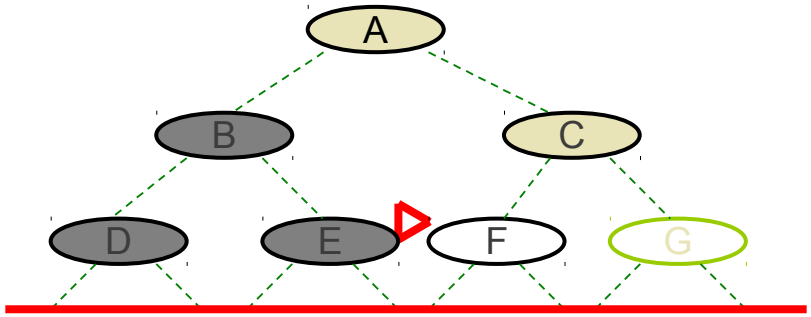



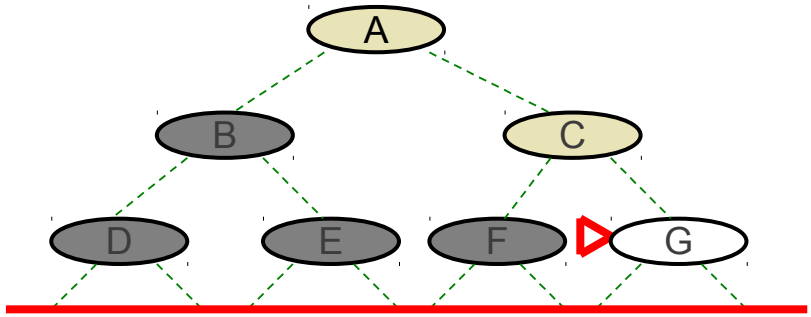



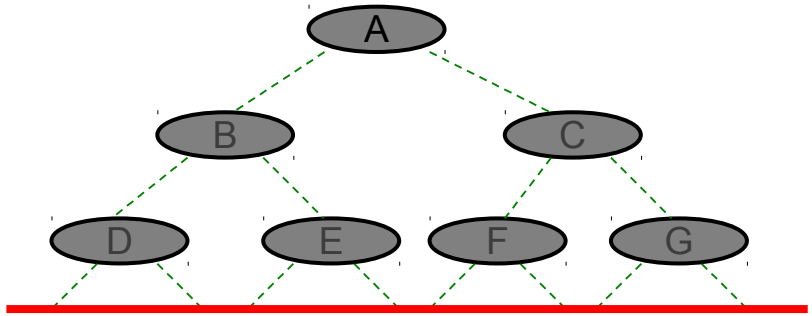
# Pesquisa por aprofundamento iterativo (cont.)

- l = 2 (cont.)




- Continua com l = 3, 4, ... até encontrar uma solução ou esgotar as alternativas

# Pesquisa por aprofundamento iterativo (cont.)

- Completo ??
  - Sim
- Tempo ??
  - $(d+1)\ b^0 + (d)\ b^1 + (d-1)\ b^2 + (d-2)\ b^3 + \ldots + b^d = O(b^d)$
  - Exemplo com b = 10 e d = 5:

    **Número de expansões:**

    $$(d+1)\ 1 + (d)\ b + (d-1)\ b^2 + \ldots + 3\ b^{d-1} + 1\ b^d$$

    **6 + 50 + 400 + 3 000 + 20 000 + 100 000 = 123 456**
- Espaço ??
  - O( b d)

    **Número de nós, com b=10 e d=5:**

    **1 + 10 + 10 + 10 + 10 + 10 = 51**
- Ótimo ??
  - Sim, se o custo de cada passo for 1

# Comparação

| Critério | Largura | Custo uniforme | Profund. | Profund. limitada | Aprofund. iterativo |
|---|---|---|---|---|---|
| Completo ? | Sim | Sim | Não | Sim, se l>=d | Sim |
| Tempo | $b^{d+1}$ | $O(b^{1+C^*/\varepsilon})$ | $b^{m}$ | $b^{l}$ | $b^{d}$ |
| Espaço | $b^{d+1}$ | $O(b^{1+C^*/\varepsilon})$ | b m | b l | b d |
| Óptimo ? | Sim | Sim | Não | Não | Sim |