Aprendizado de Máquina

Curso Tecnologia e Análise de Sistemas

Prof. Dr. Domingos Márcio Rodrigues Napolitano

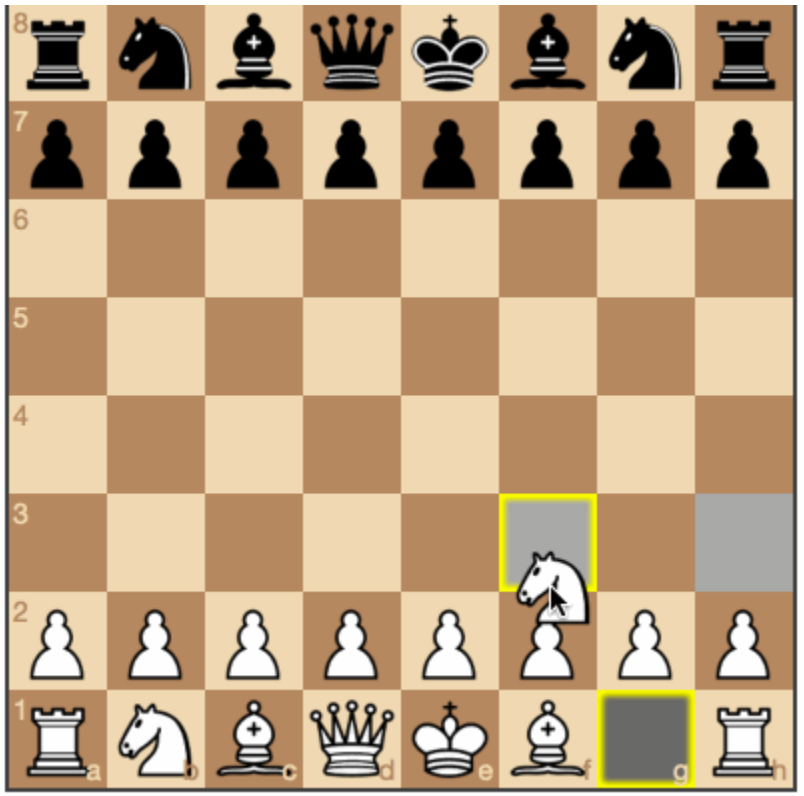
Sem 1 / 2020

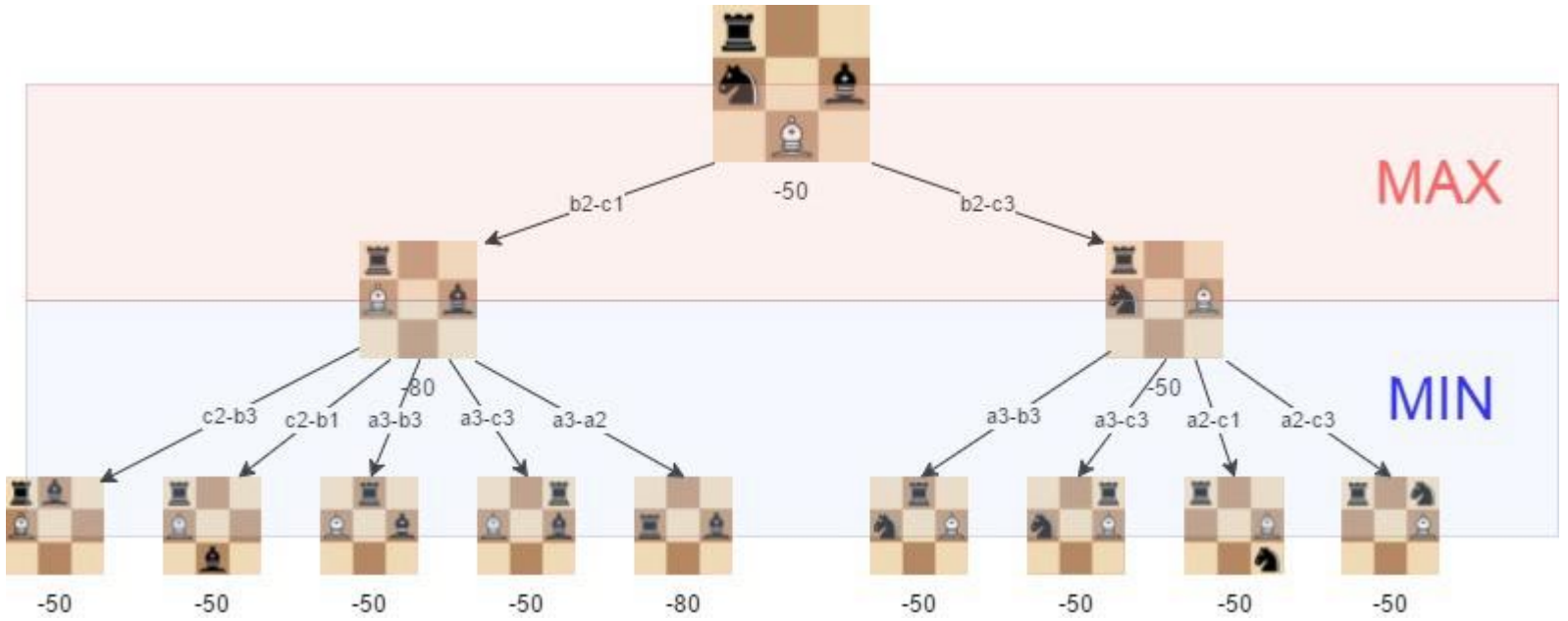
QUIZZ

- O Quizz Relativo ao material desta aula está disponível neste link:

- https://forms.gle/As4PmVtFuNyXAbJ47

Resolução de Problemas – Busca em Espaço de Estados

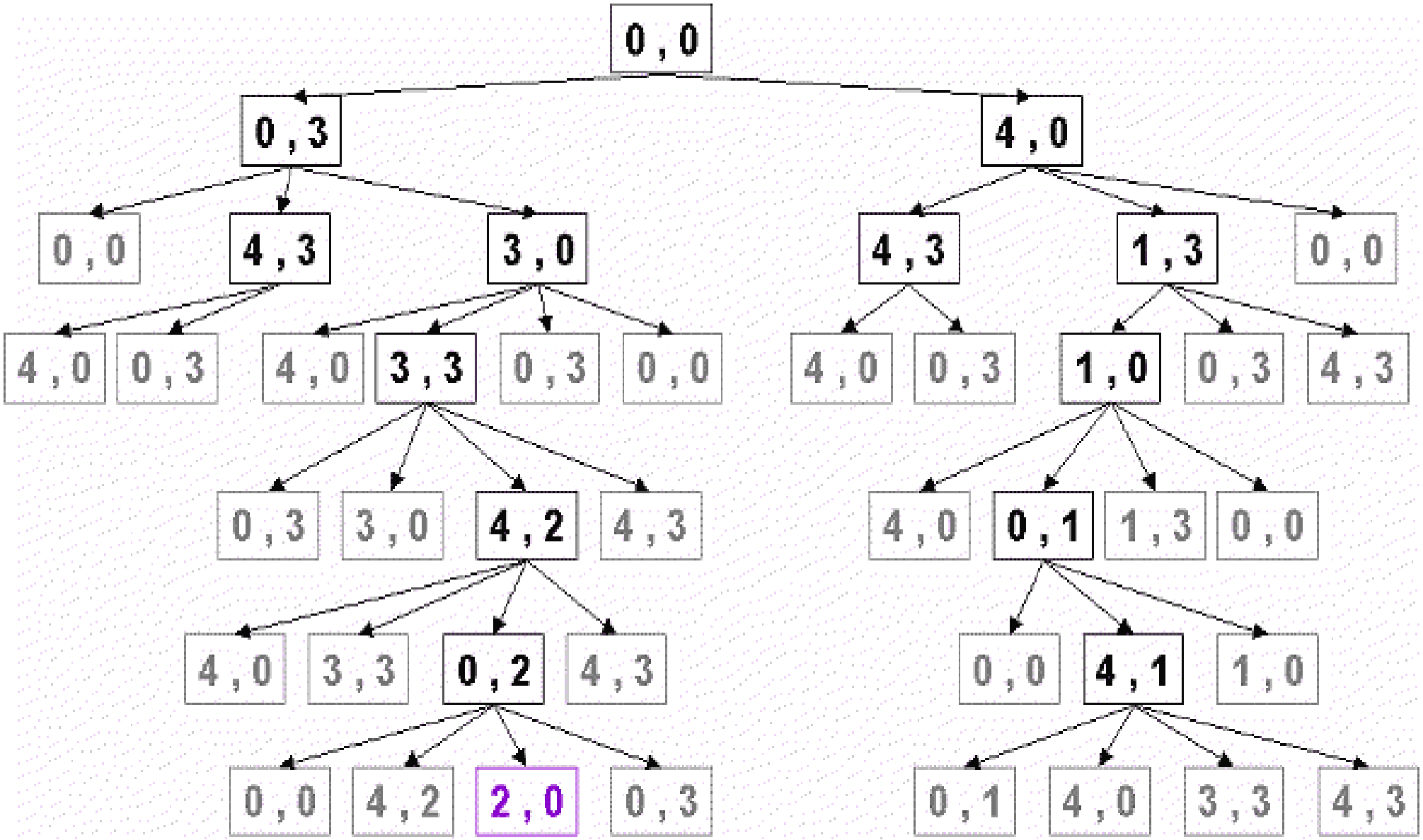




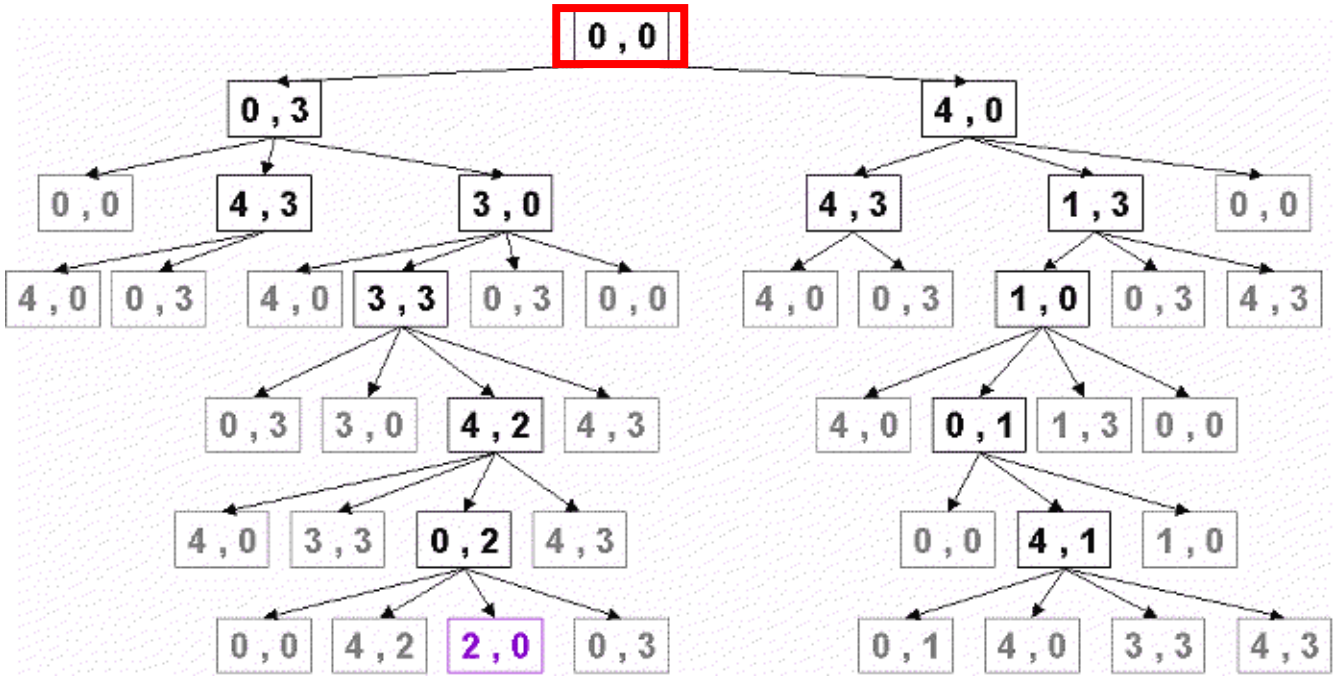
Problema: jarros

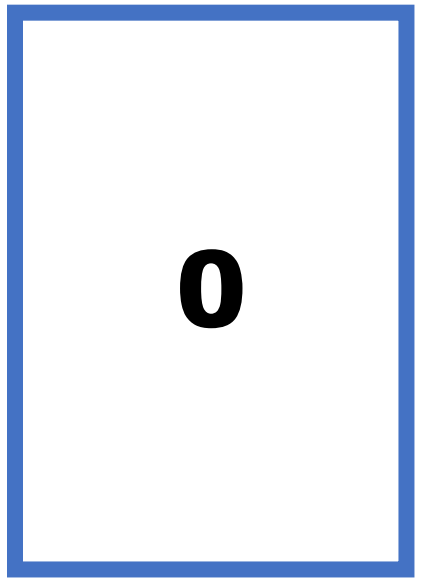
- Dados uma bica d`agua, um jarro de capacidade 3 litros e um jarro de capacidade 4 litros (ambos vazios). Como obter 2 litros?

Exemplo dos jarros

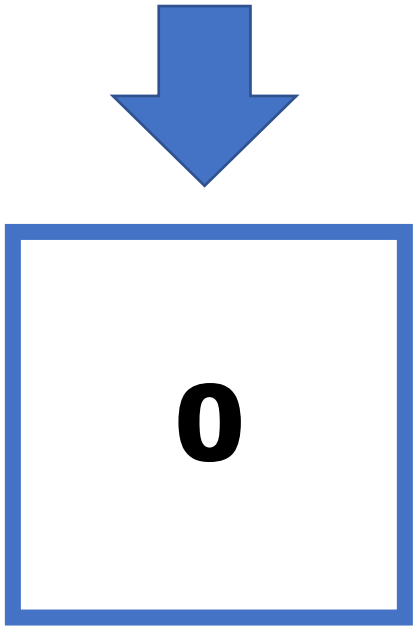


Exemplo dos jarros





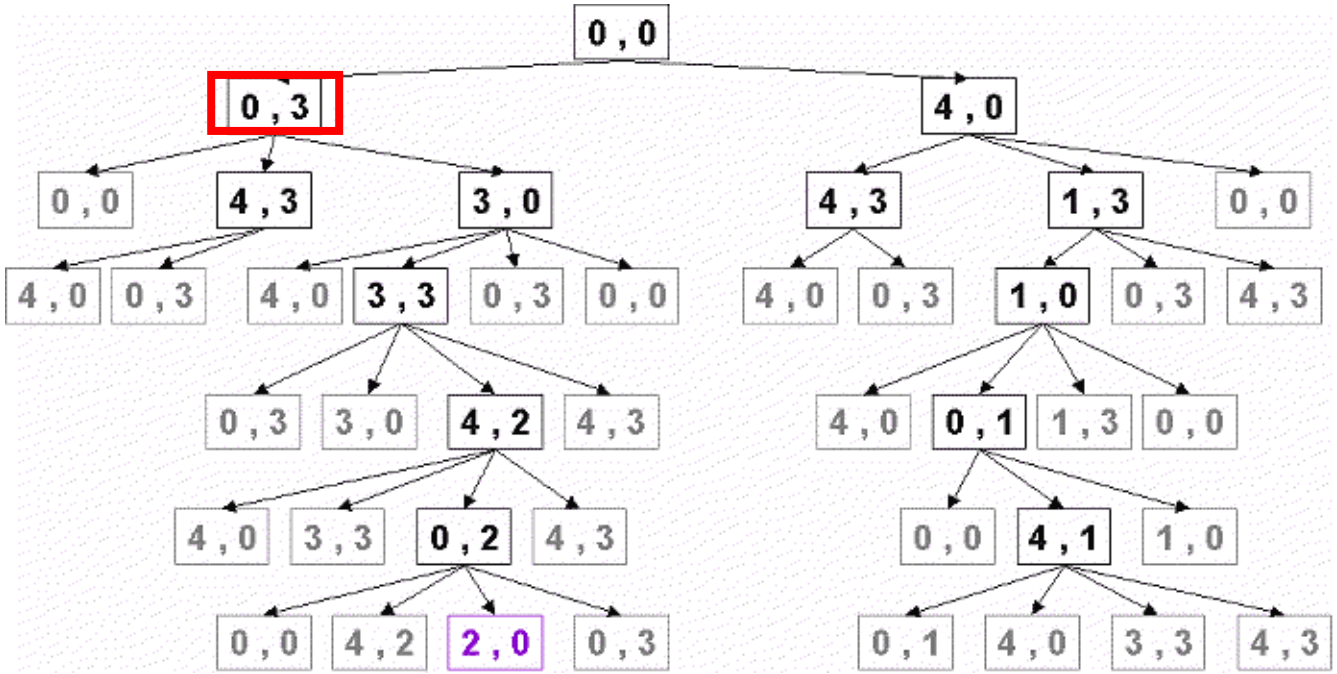
4L

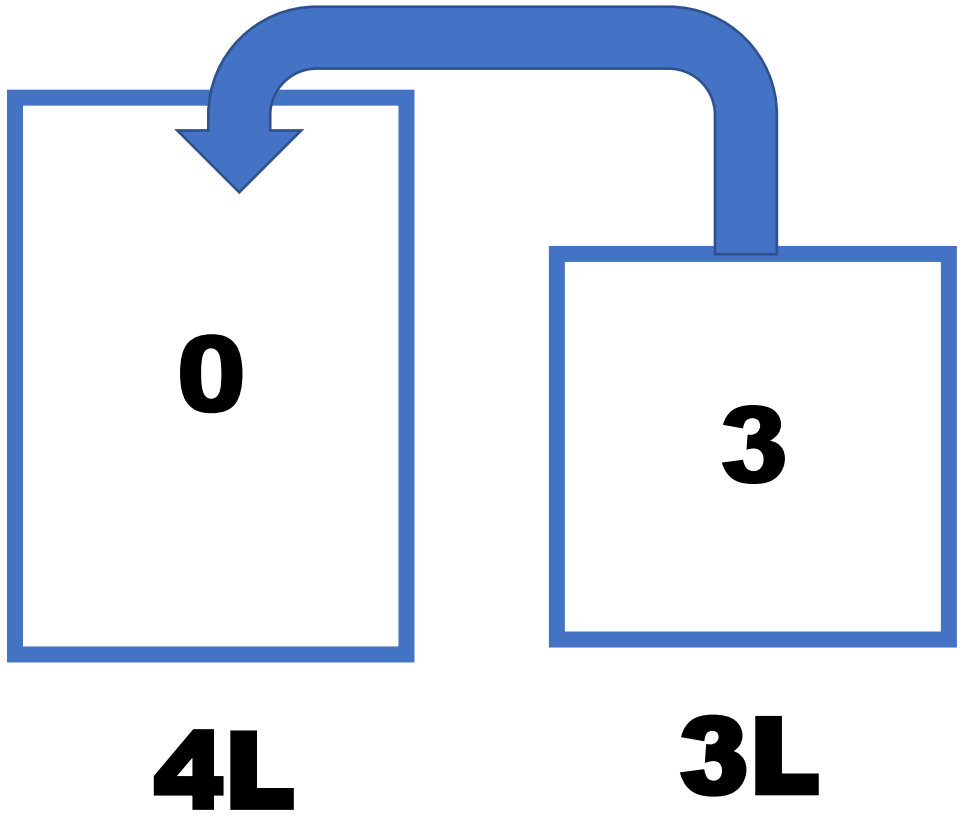


3L

Ação Encher Jarro 3L

Exemplo dos jarros

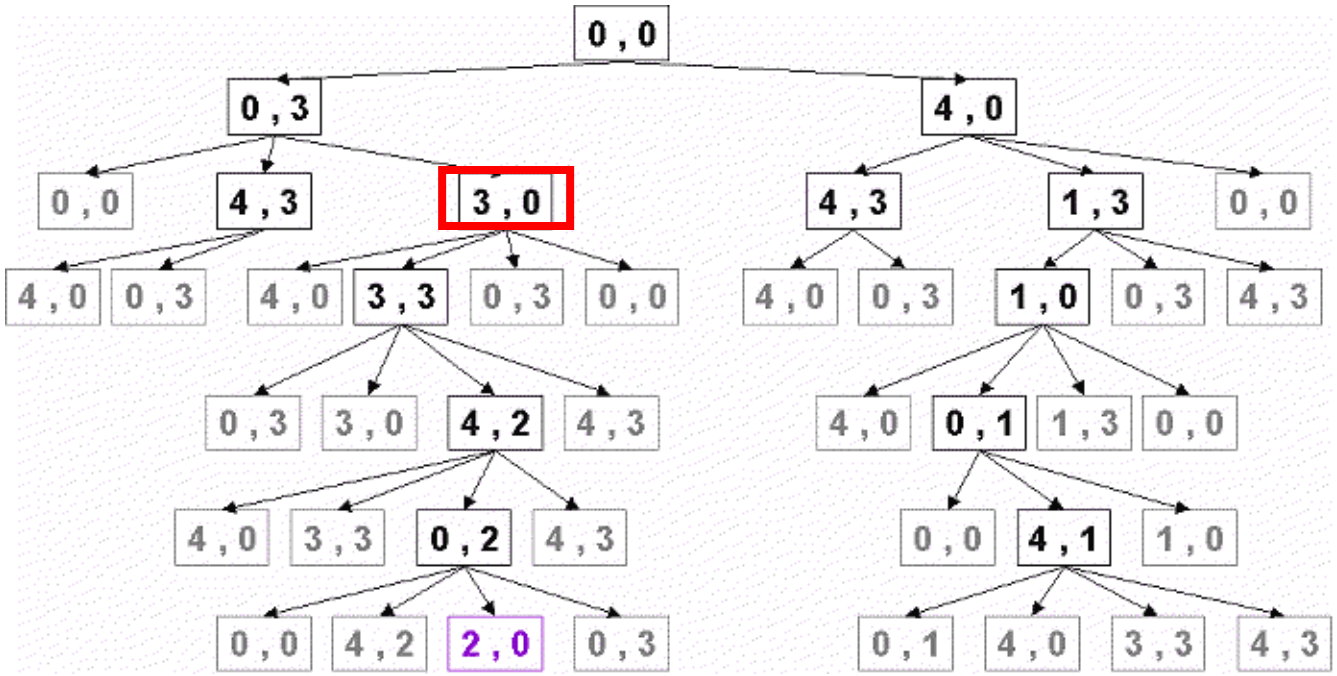


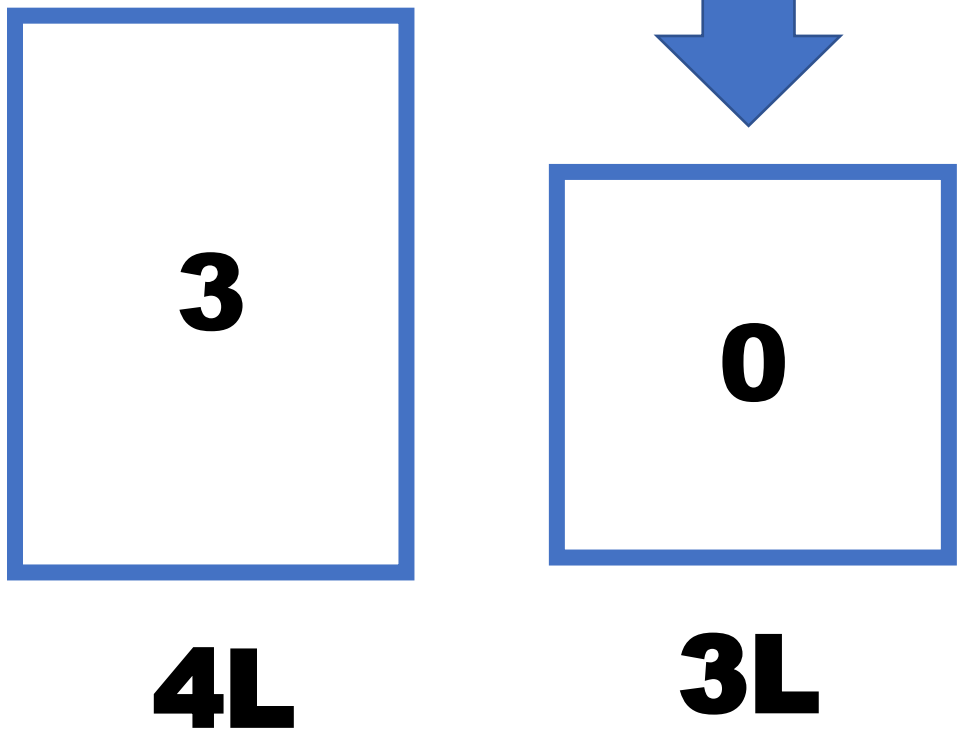


Despejar Conteúdo

Jarro 3L Jarro 4L

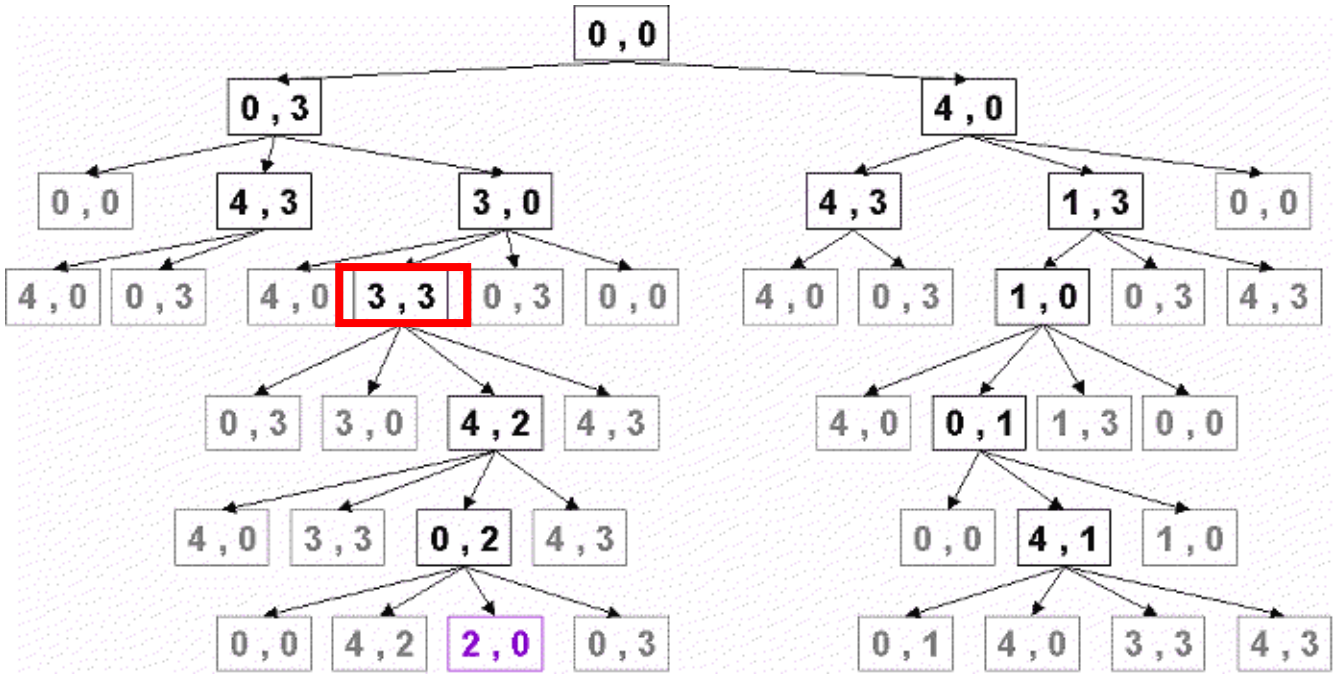
Exemplo dos jarros

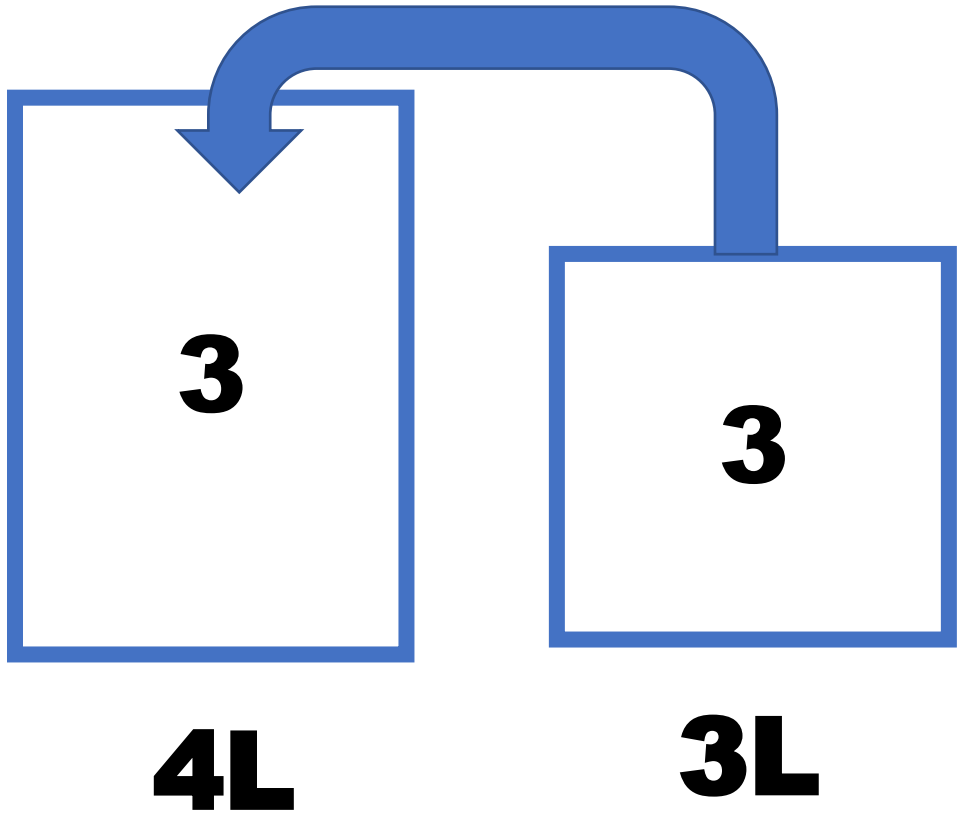




Ação Encher Jarro 3L

Exemplo dos jarros

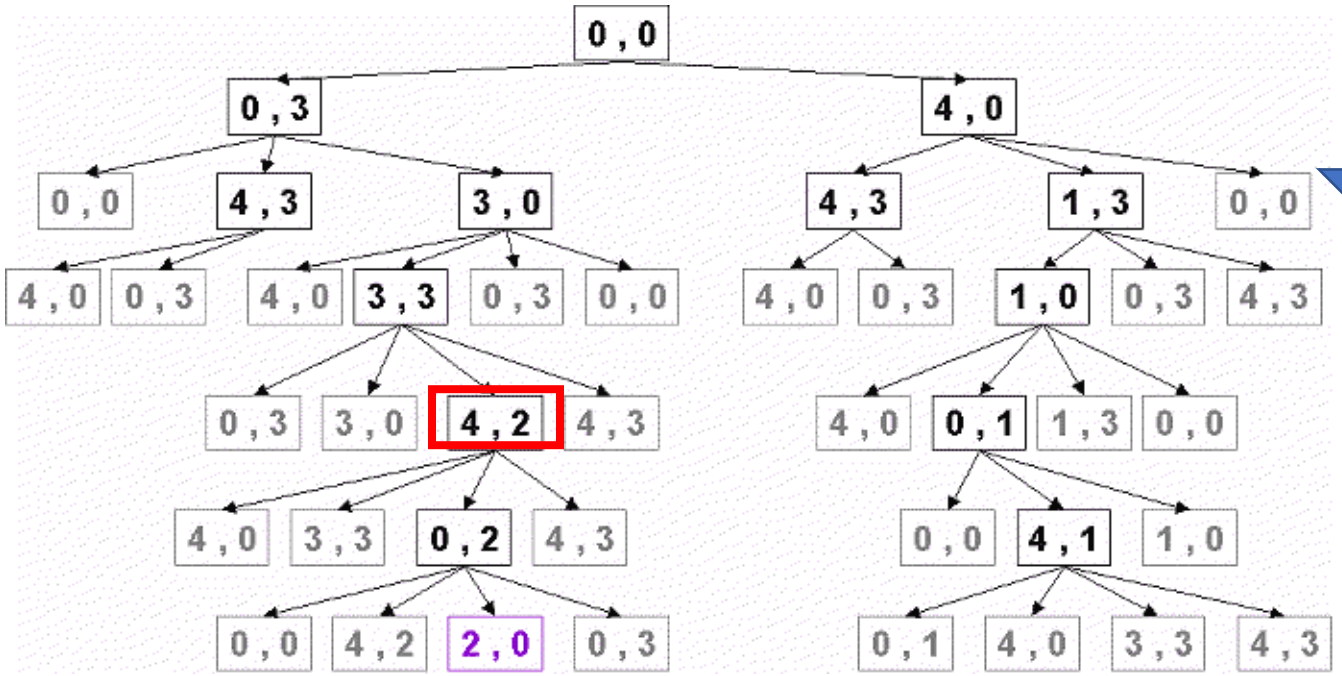


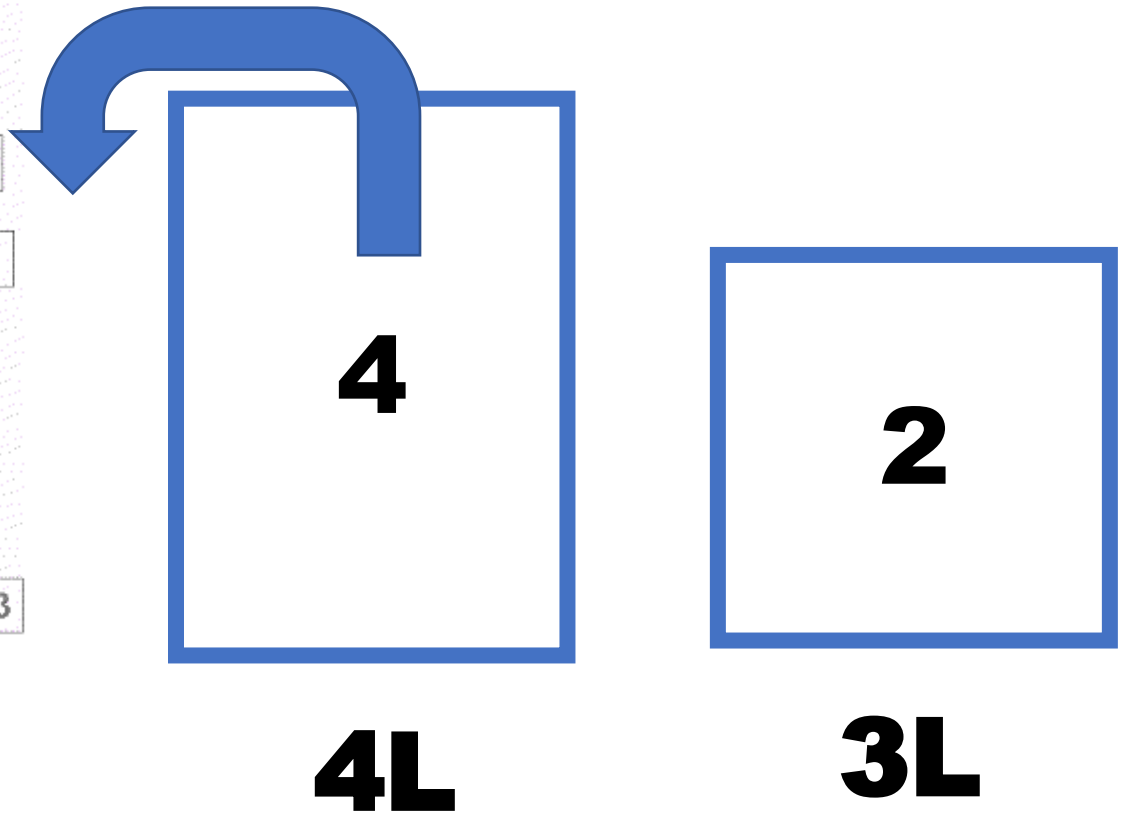


Despejar Conteúdo

Jarro 3L Jarro 4L

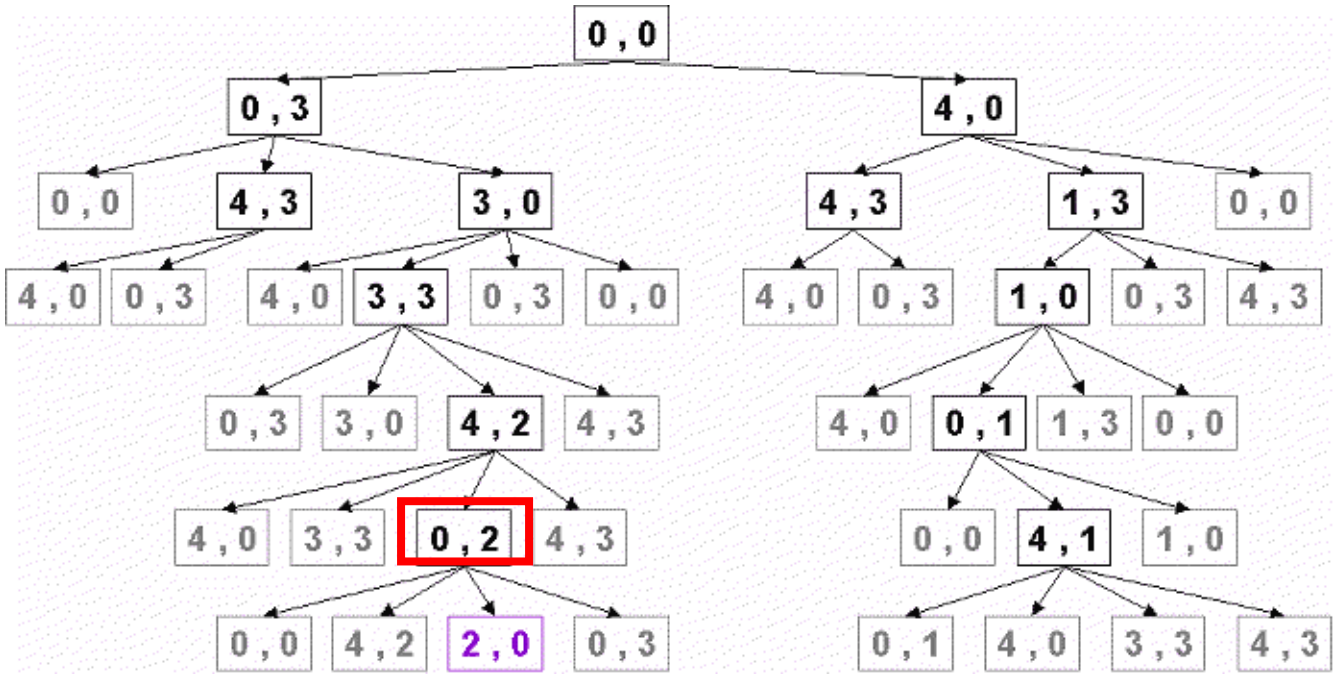
Exemplo dos jarros

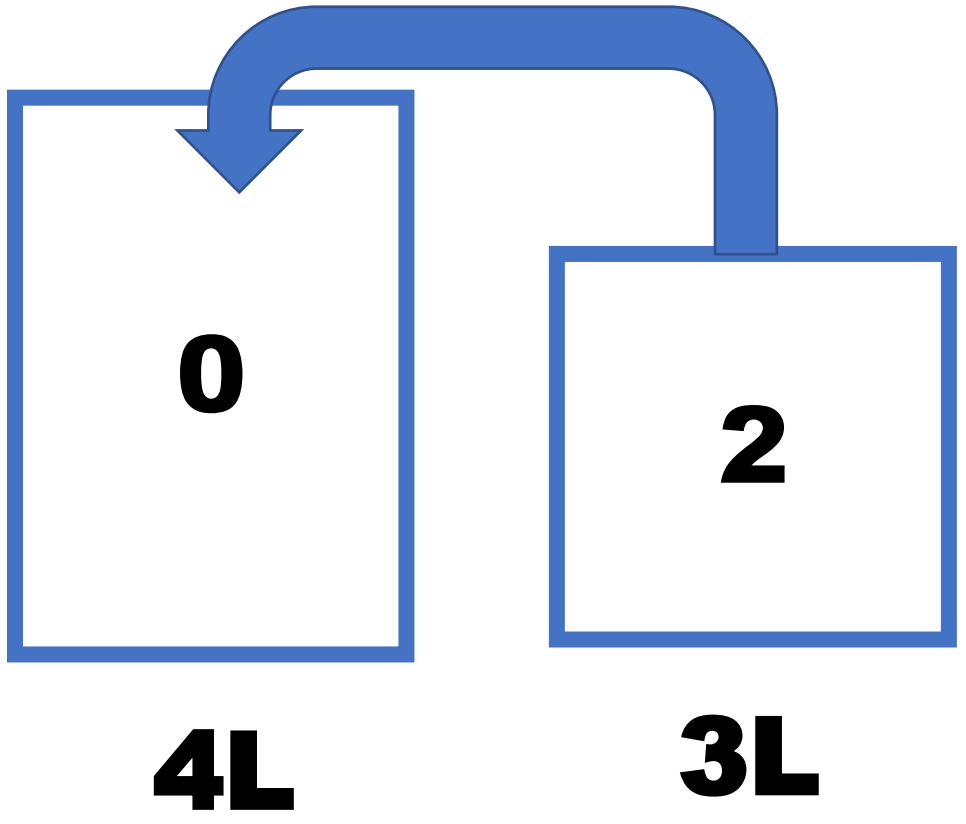




Despejar Conteúdo
Jarro 4L

Exemplo dos jarros

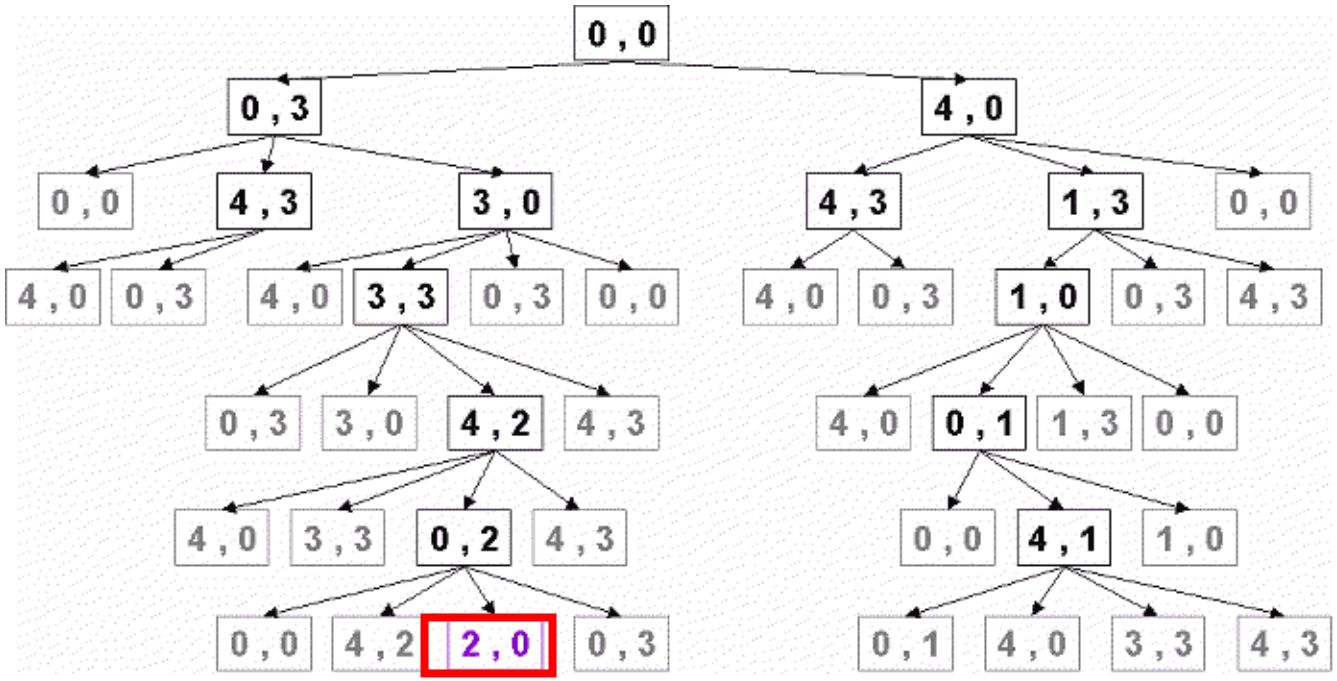


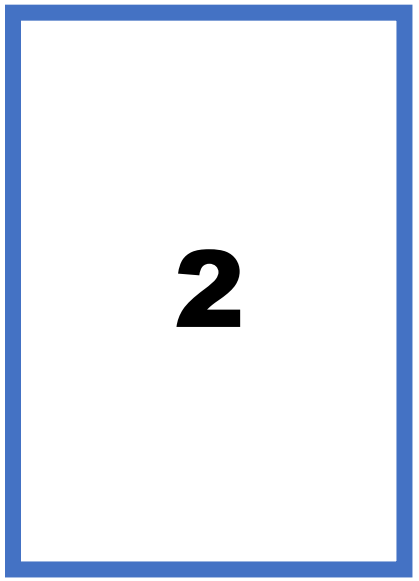


Despejar Conteúdo

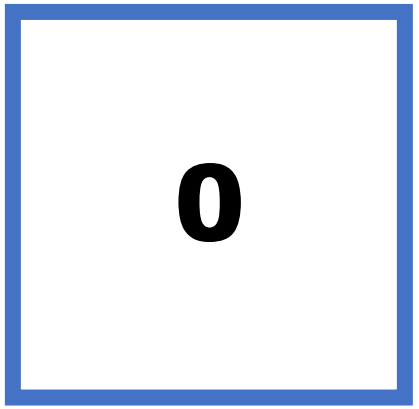
Jarro 3L Jarro 4L

Exemplo dos jarros





4L



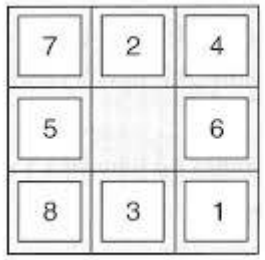
3L

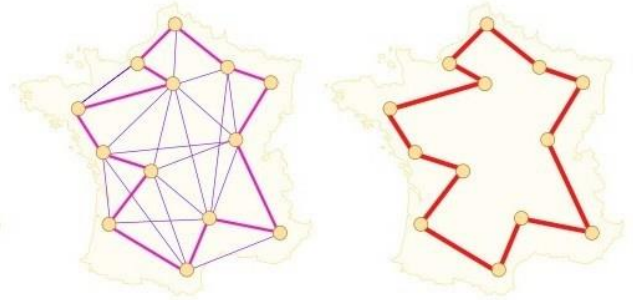
Resolução de Problemas – Busca em Espaço de Estados

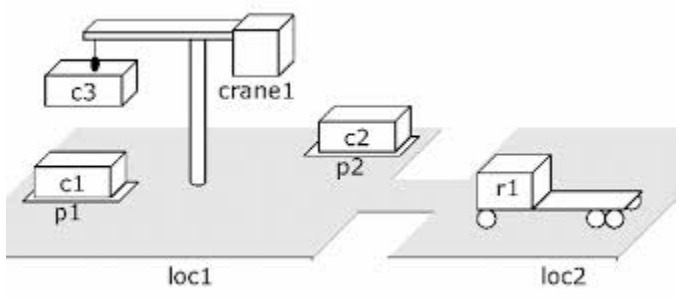
- Dedica-se ao estudo e elaboração de algoritmos, capazes de resolver, por exemplo, problemas considerados intratáveis do ponto de vista da computação convencional.
- Primeiros problemas por computador: prova automática de teoremas e jogos.
- Capacidade de cálculo e memória dos computadores: insuficientes perante o enorme número de caminhos de solução.
- **Exemplo**: jogo de xadrez
- **Um dos objetivos de IA**: resolver problemas que o homem não sabe resolver facilmente ou num tempo razoável, desde que sejam completamente formalizados.

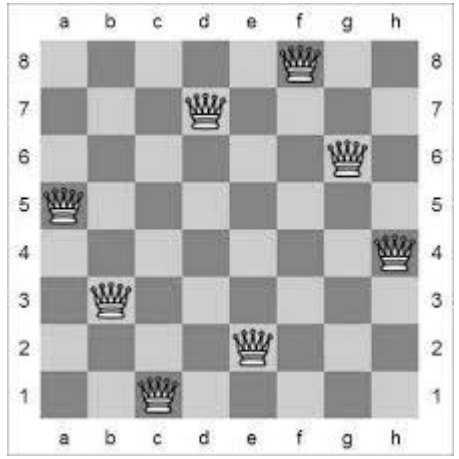
Exemplos de Problemas

- O quebra-cabeças 3x3
- O Caixeiro Viajante
- Cálculo Integral Formal
- Empilhamento de blocos: a partir de uma configuração de blocos iniciais, qual a seqüência de movimentos para se chegar a uma configuração final?
- As Oito Rainhas
- As Torres de Hanói









Torres de Hanói

- https://www.youtube.com/watch?v=CrBTg0nms2Y

Problemas e soluções bem definidos

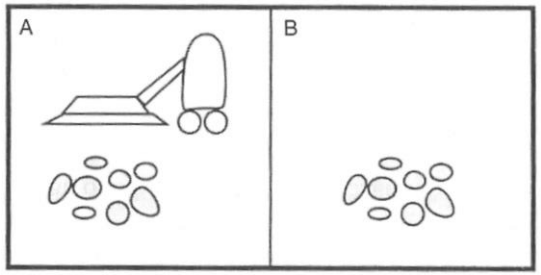
Quatro componentes para definir um problema:

1. O **estado inicial** em que o agente começa.
2. Uma descrição das **ações** possíveis que estão disponíveis para o agente.
   - Formulação mais comum: uso de uma **função sucessor**.
   - Estado inicial e função sucessor: definem o **espaço de estados** do problema.
   - **Caminho** no espaço de estados – seqüência de estados conectados por uma seqüência de ações.
3. O **teste de objetivo** – determina se um dado estado é um estado objetivo.
4. Função de **custo de caminho** – atribui um custo numérico a cada caminho.

Elementos:

- Estado Inicial
- Função Sucessor
- Teste de Objetivo
- Custo de Caminho

Miniproblemas



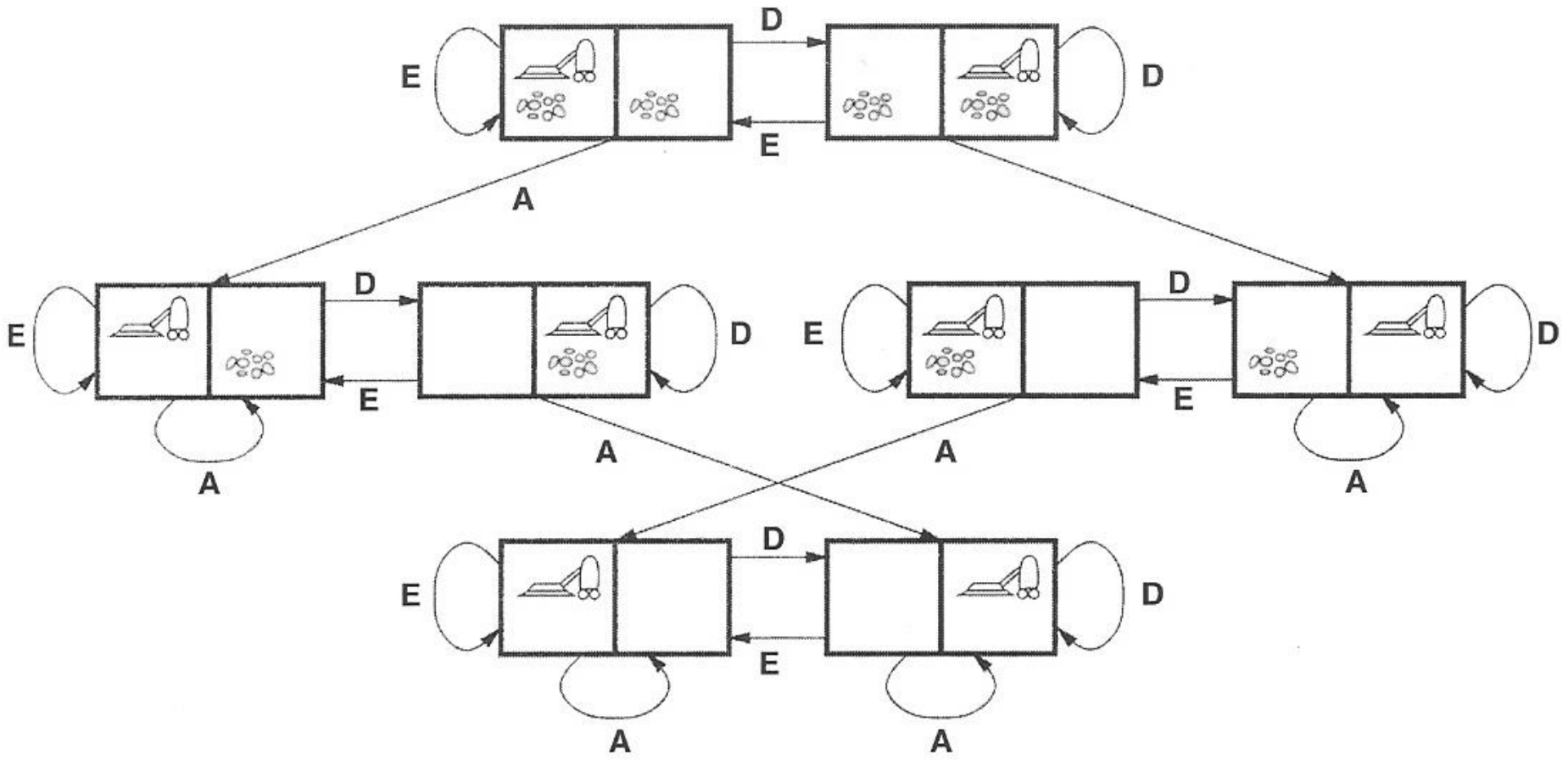
- **Exemplo 1:** Mundo do aspirador de pó com apenas 2 locais.

Problema do Mundo do Aspirador de Pó - Formulação

- **Estados**
  - O agente está em uma entre duas posições, cada uma das quais pode conter sujeira ou não.
  - Há 2 x 2^2 = 8 estados do mundo possíveis.
- **Estado inicial**
  - Qualquer estado pode ser designado como estado inicial.
- **Função Sucessor**
  - Gera os estados válidos que resultam da tentativa de executar as três ações (*Esquerda*, *Direita* e *Aspirar*).

Miniproblemas

Espaço de estados para o mundo do aspirador de pó.



Os arcos denotam ações: E = Esquerda, D = Direita, A = Aspirar

Miniproblemas

UNINOVE

Problema do Mundo do Aspirador de Pó - Formulação

- **Teste de objetivo**
  - Verifica se todos os quadrados estão limpos.
- **Custo de caminho**
  - Cada passo custa 1, e assim o custo do caminho é o número de passos do caminho.
- Esse miniproblema tem posições discretas, sujeira discreta, limpeza confiável e nunca é desorganizado depois de limpo.
- Ambiente com ***n*** posições: $n \times 2^n$ estados

Problemas do mundo real

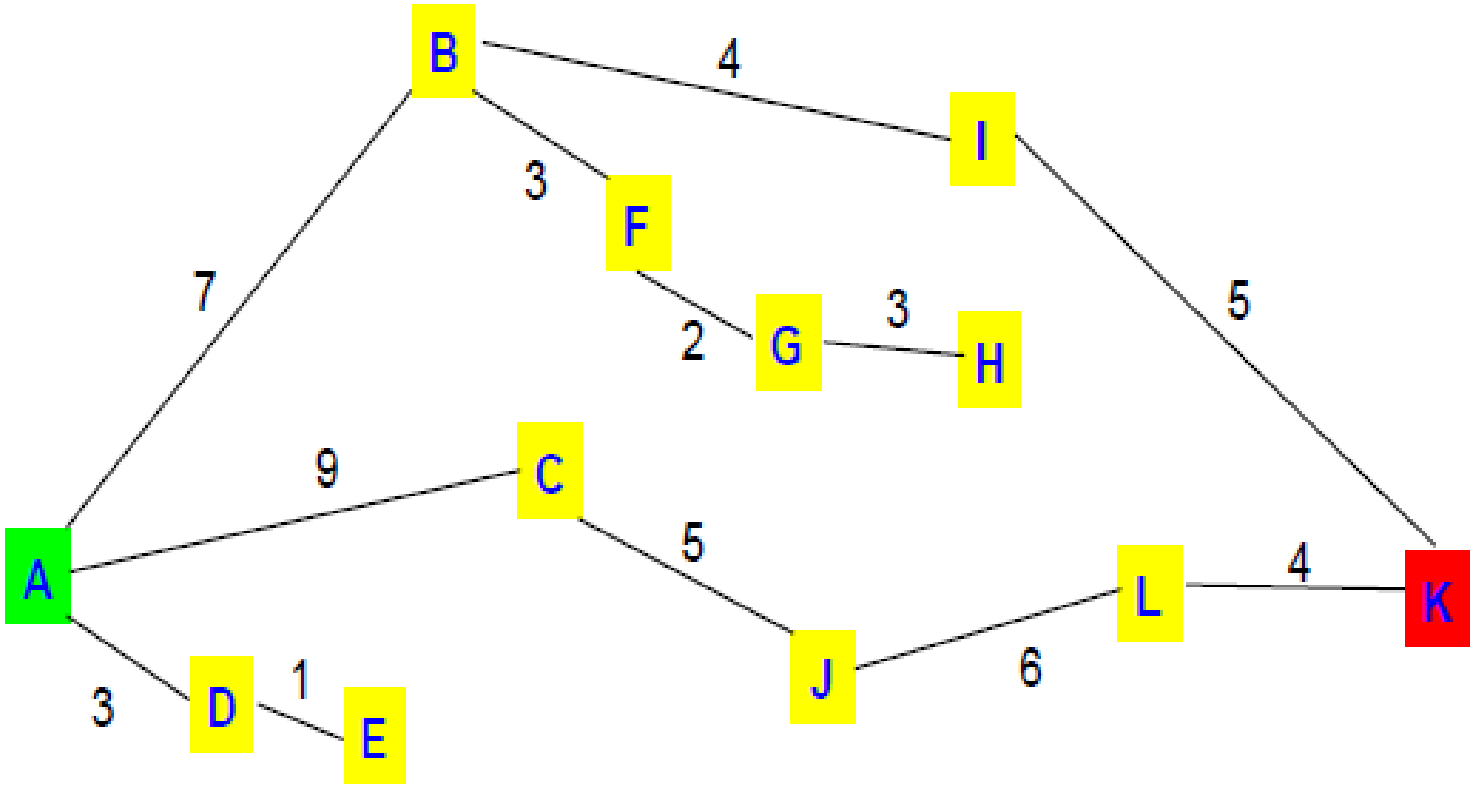
UNINOVE

Problema de Viagens Aéreas – Formulação

- **Estados**
  - Cada um é representado por uma posição (p.ex.: um aeroporto) e pela hora atual.
- **Estado inicial**
  - É especificado pelo problema.
- **Função Sucessor**
  - Retorna os estados resultantes de tomar qualquer vôo programado que parte depois da hora atual somada ao tempo de trânsito no aeroporto, desde o aeroporto atual até outro.
- **Teste de objetivo**
  - Estamos no destino após algum tempo especificado?

Problemas do mundo real

- Qual o menor percurso entre A e K ?



Problemas do mundo real

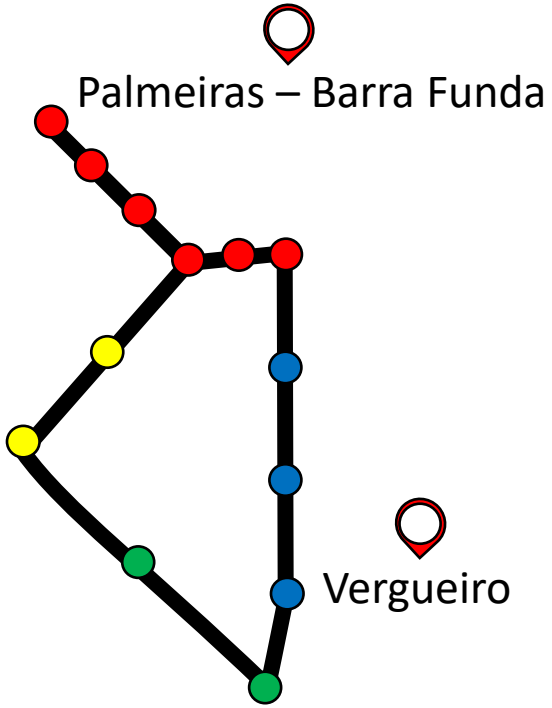
Problema de Viagens Aéreas – Formulação

- **Custo de caminho**
  - Depende do custo monetário, do tempo de espera, do tempo de vôo, dos procedimentos alfandegários, da hora do dia, ...

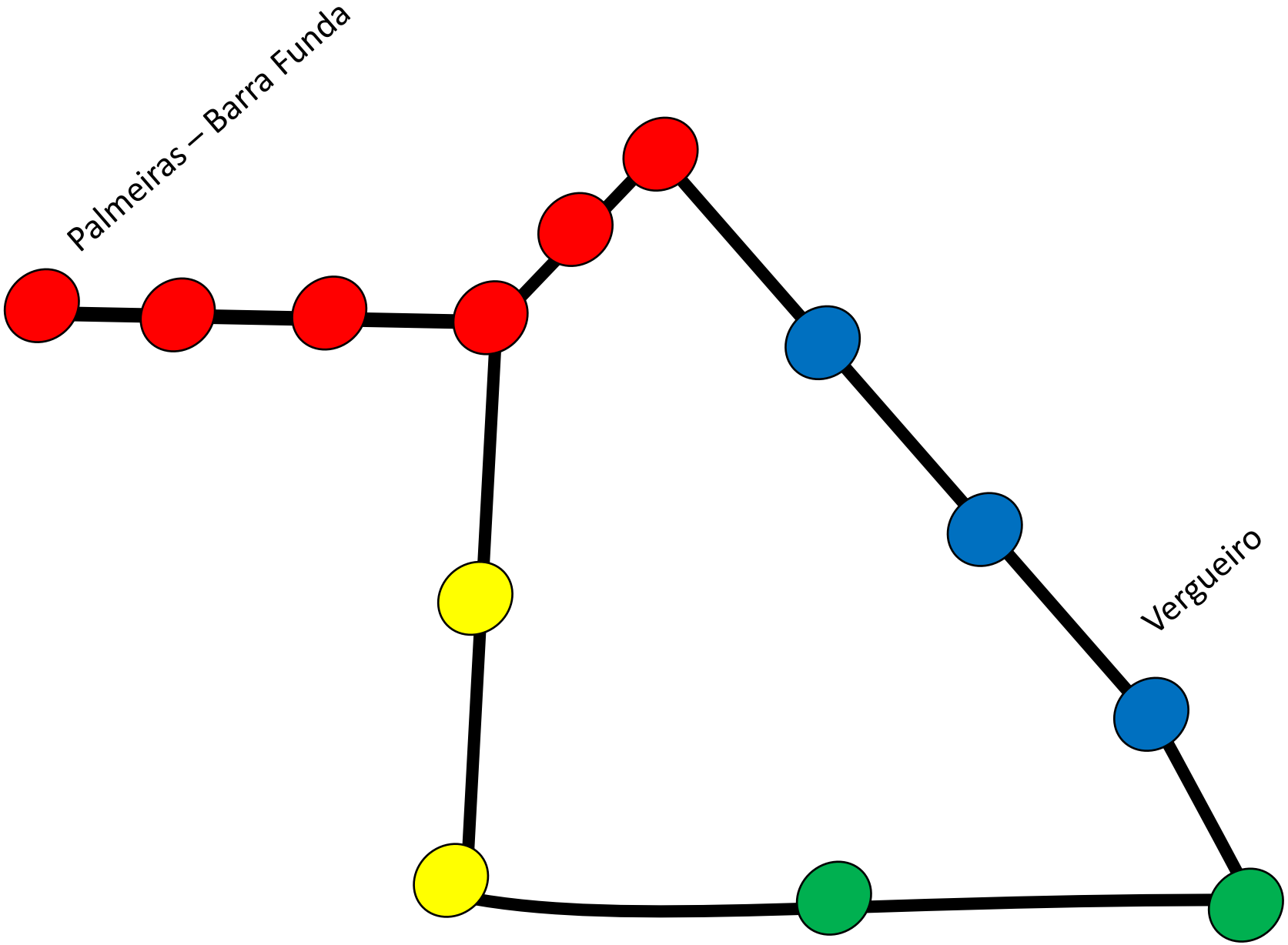
- Um sistema realmente bom deve incluir planos de contingência. **Exemplo**: reservas substitutas em vôos alternativos.

Um exemplo

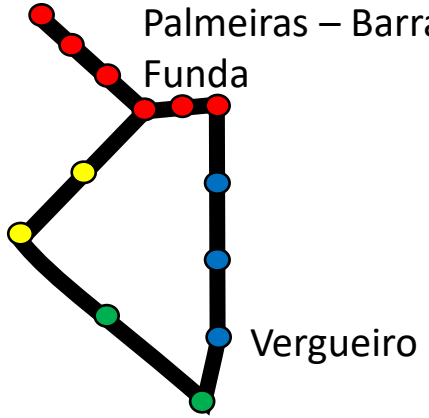
- Você está no Campus Memorial da Uninove e deseja ir para o Campus Vergueiro Quais as rotas



Mapeamento do Espaço de Estados



Mapeamento do Espaço de Estados



- **Primeiro Estado**
  - **Inicial – Palmeiras Barra Funda**
  - **Sucessor – Mal. Deodoro**
- **Segundo Estado**
  - **Inicial – Mal. Deodoro**
  - **Sucessor – Sta. Cecília**
- **Terceiro Estado**
  - **Inicial – Sta. Cecília**
  - **Sucessor – Republica**

- **Quarto Estado**
  - **Inicial – Republica**
  - **Sucessor – Higienópolis**
- **Quinto Estado**
  - **Inicial – Higienópolis**
  - **Sucessor – Paulista-Consolação**

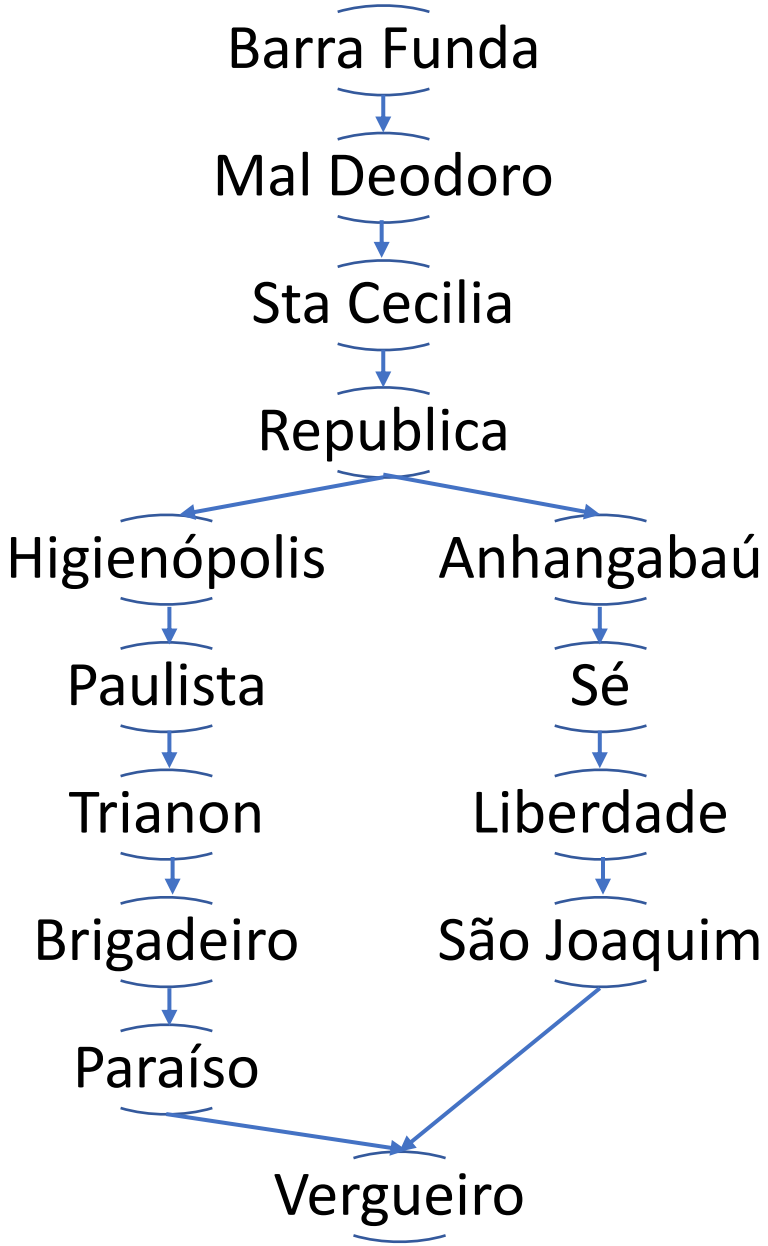
- **Quarto Estado**
  - **Inicial – Republica**
  - **Sucessor – Anhangabaú**
- **Quinto Estado**
  - **Inicial – Anhangabaú**
  - **Sucessor – Sé**

- **Sexto Estado Estado**
  - **Inicial – Paulista-Consolação**
  - **Sucessor – Trianon**
- **Sétimo Estado**
  - **Inicial – Trianon**
  - **Sucessor – Brigadeiro**
- **Oitavo Estado**
  - **Inicial – Brigadeiro**
  - **Sucessor – Paraíso**

- **Sexto Estado**
  - **Inicial – Sé**
  - **Sucessor – Liberdade**
- **Sétimo Estado**
  - **Inicial – Liberdade**
  - **Sucessor – São Joaquim**
- **Oitavo Estado**
  - **Inicial – São Joaquim**
  - **Sucessor – Vergeuiro**

- **Nono Estado**
  - **Inicial – Paraíso**
  - **Sucessor – Vergueiro**

Mapeamento do Espaço de Estados



Estado Inicial:

- Barra Funda

Função Sucessor

- Vai_de_para(Incial,Final) → Ligação(Inicial, Final)

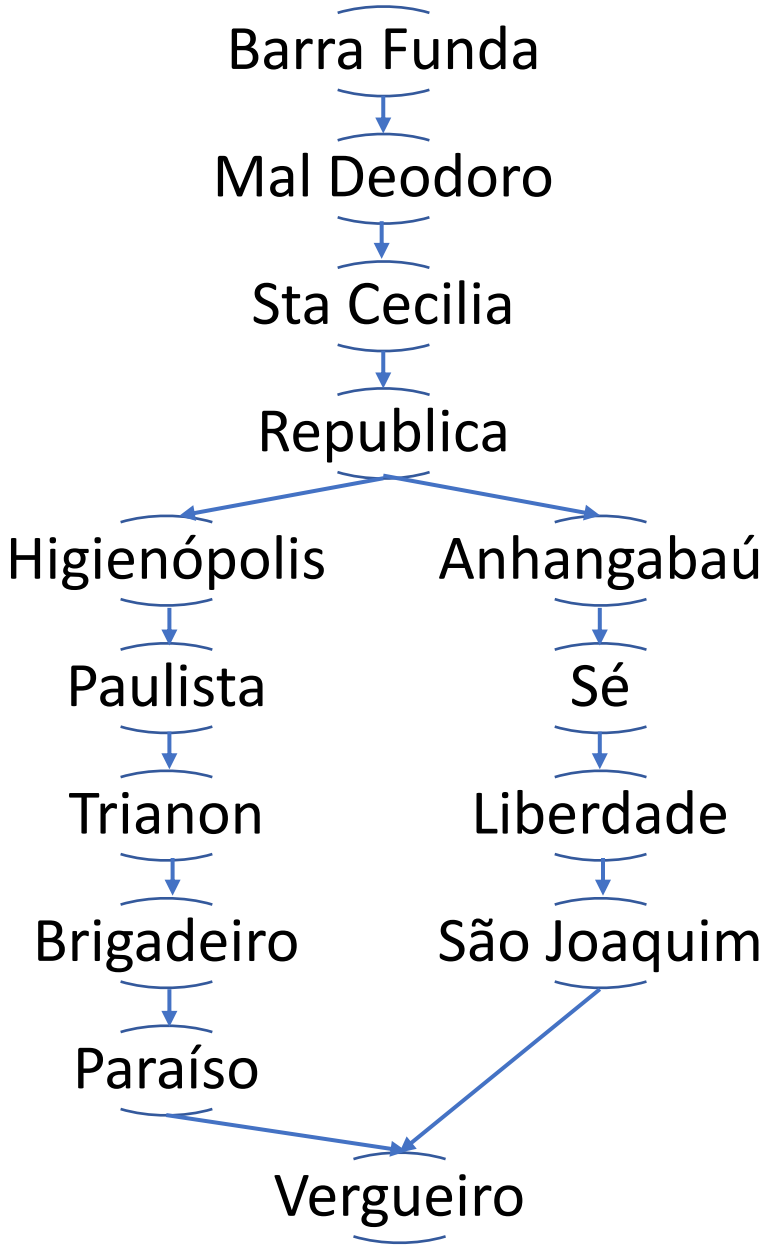
Teste de Objetivo

- Objetivo(Vergueiro)

Custo de Caminho

- Cada Estação tem Custo 1

Mapeamento do Espaço de Estados



Estado Inicial:

Em(Barra Funda)

Função Sucessor

Vai_de_para(Incial,Final) → Ligação(Inicial, Final)

Teste de Objetivo :objetivo(Vergueiro)

objetivo(Vergueiro)

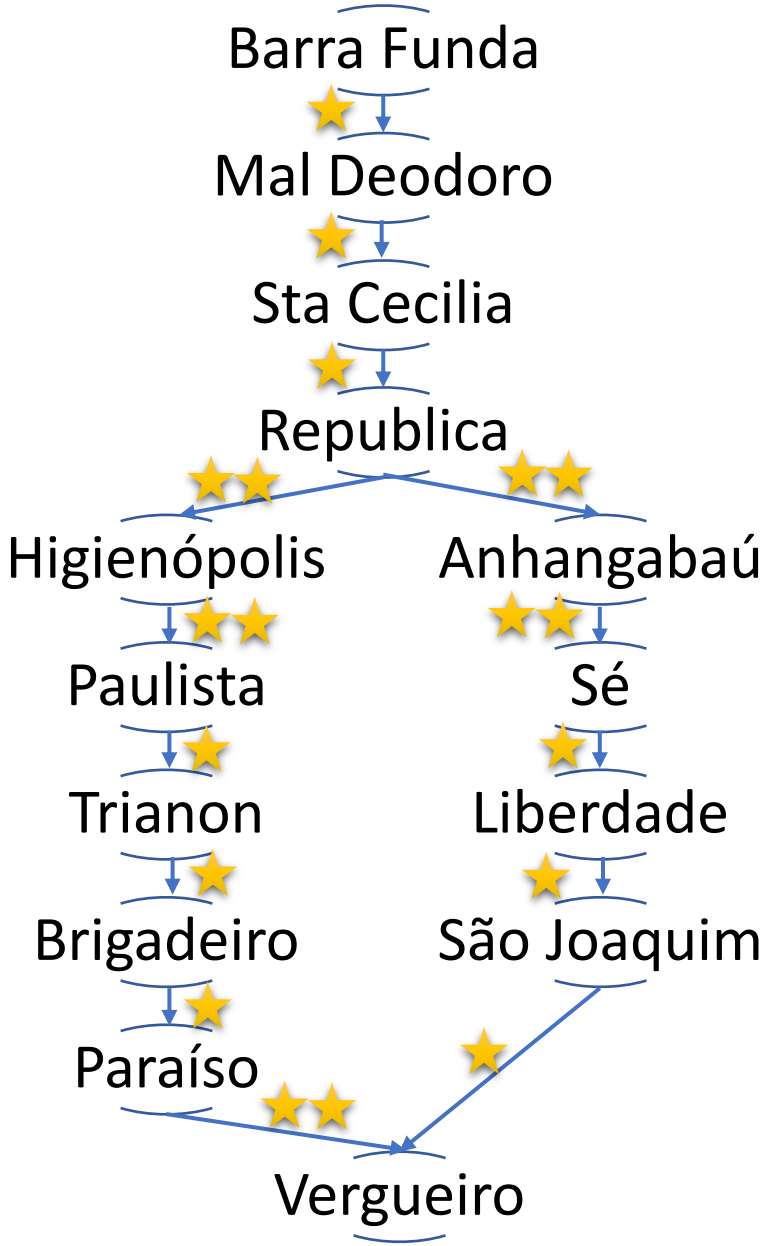
Custo de Caminho

Cada Estação tem Custo 1

Solução

- Existem 2 rotas
- Linha Vermelha, Amarela, Verde e Azul Custo 9
- Linha Vermelha, e Azul Custo 8

Mapeamento do Espaço de Estados



Estado Inicial:

Em(Barra Funda)

Função Sucessor

Vai_de_para(Incial,Final) → Ligação(Inicial, Final)

Teste de Objetivo :

objetivo(Vergueiro)

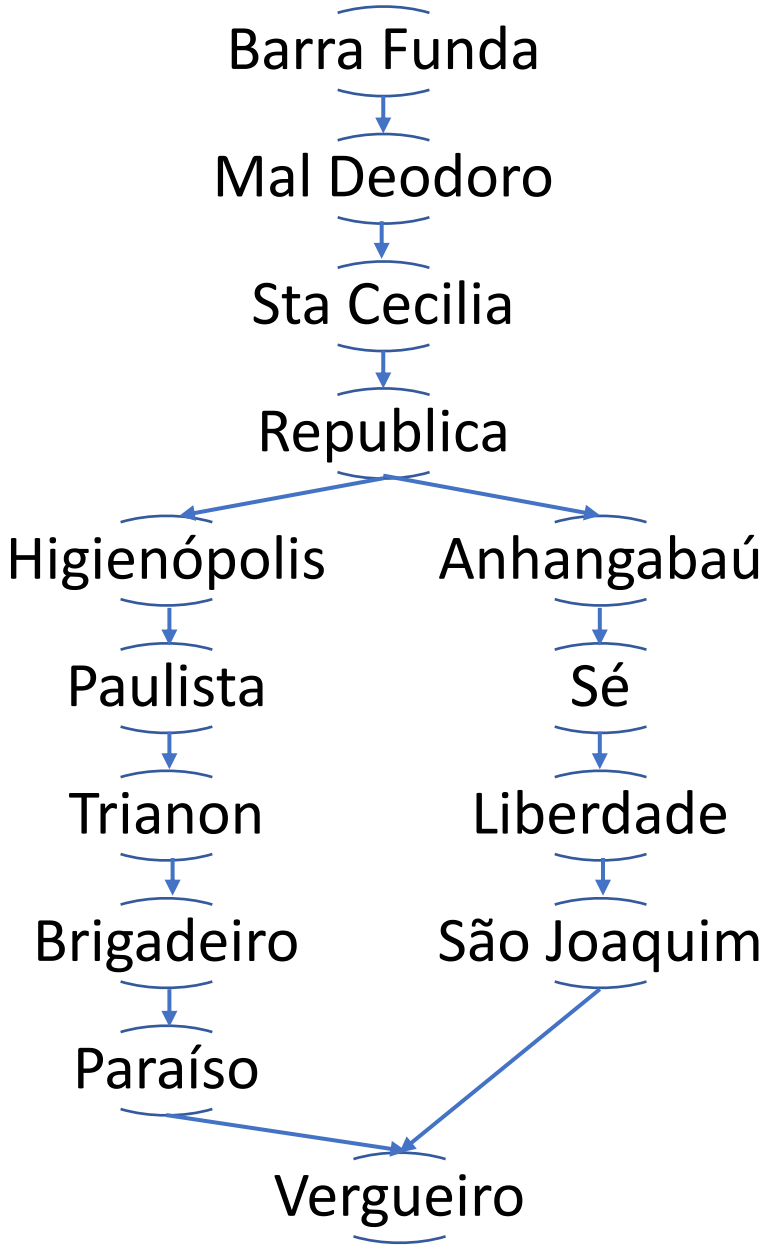
Custo de Caminho

Cada Estação tem Custo 1 e a **Transferência tem custo 2**

Solução

- Existem 2 rotas
- Linha Vermelha, Amarela, Verde e Azul Custo 12
- Linha Vermelha, e Azul Custo 10

Mapeamento do Espaço de Estados



Estado Inicial:

Em(**Vergueiro**)

Função Sucessor

Vai_de_para(Incial,Final) → Ligação(Inicial, Final)

Teste de Objetivo :objetivo(Vergueiro)

objetivo(**Higienópolis**)

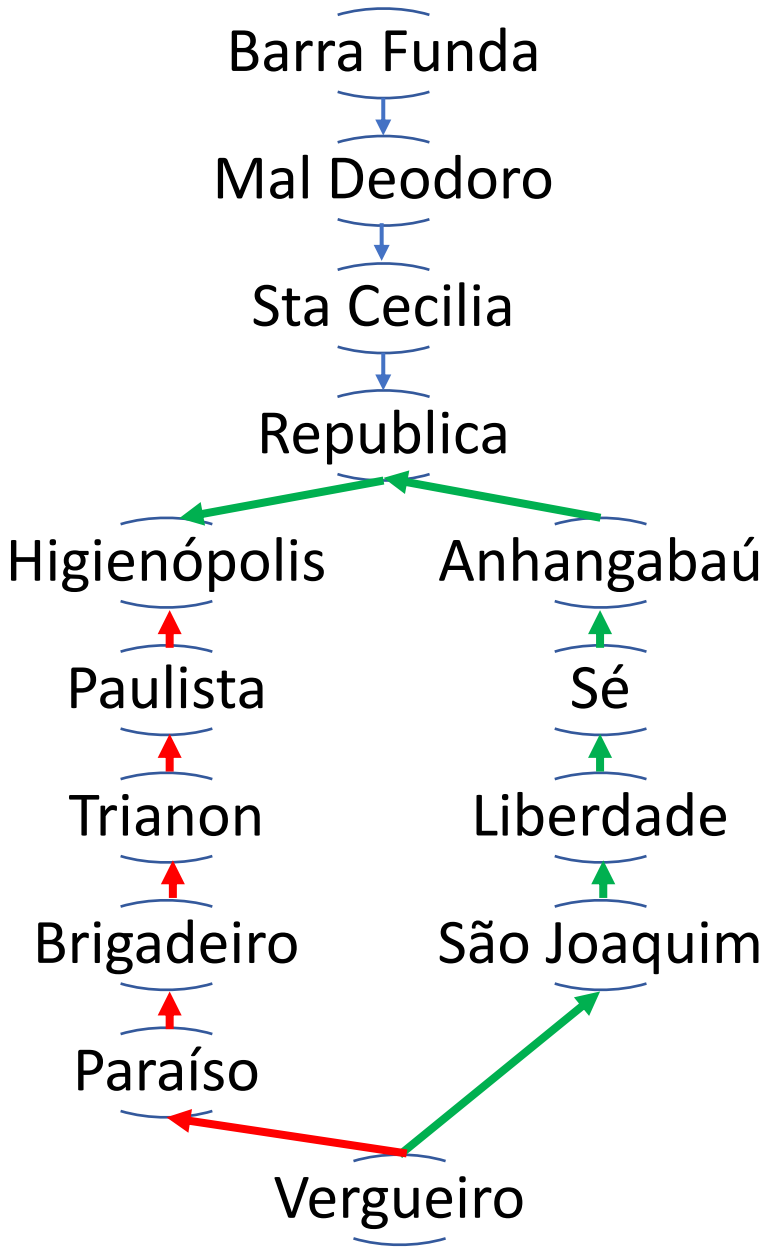
Custo de Caminho

Cada Estação tem Custo 1

Solução

- Quantas Rotas Existem ?

Mapeamento do Espaço de Estados



Estado Inicial:

Em(**Vergueiro**)

Função Sucessor

Vai_de_para(Incial,Final) → Ligação(Inicial, Final)

Teste de Objetivo :objetivo(Vergueiro)

objetivo(**Higienópolis**)

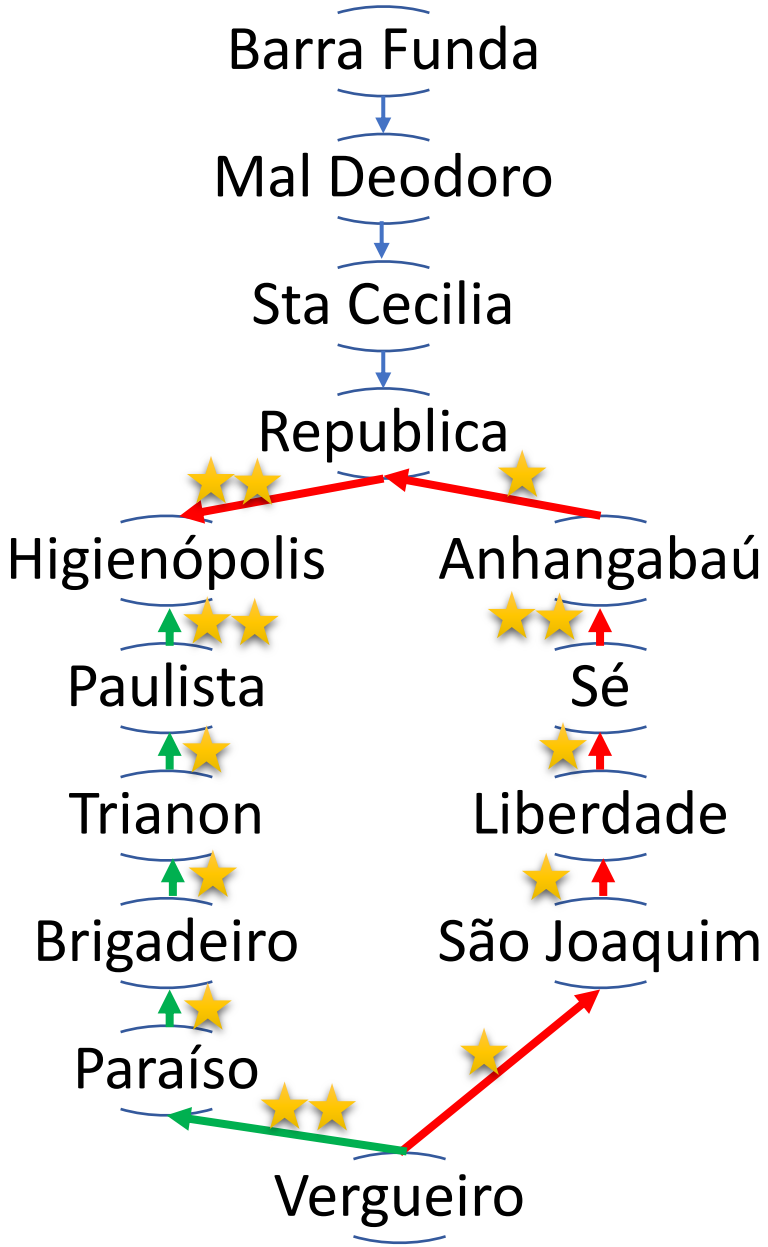
Custo de Caminho

Cada Estação tem Custo 1

Solução

- Quantas Rotas Existem ?
- **Linha Azul, Vermelha, Amarela Custo 6**
- **Linha Azul e Verde Custo 5**

Mapeamento do Espaço de Estados



Estado Inicial:

Em(**Vergueiro**)

Função Sucessor

Vai_de_para(Incial,Final) → Ligação(Inicial, Final)

Teste de Objetivo :objetivo(Vergueiro)

objetivo(**Higienópolis**)

Custo de Caminho

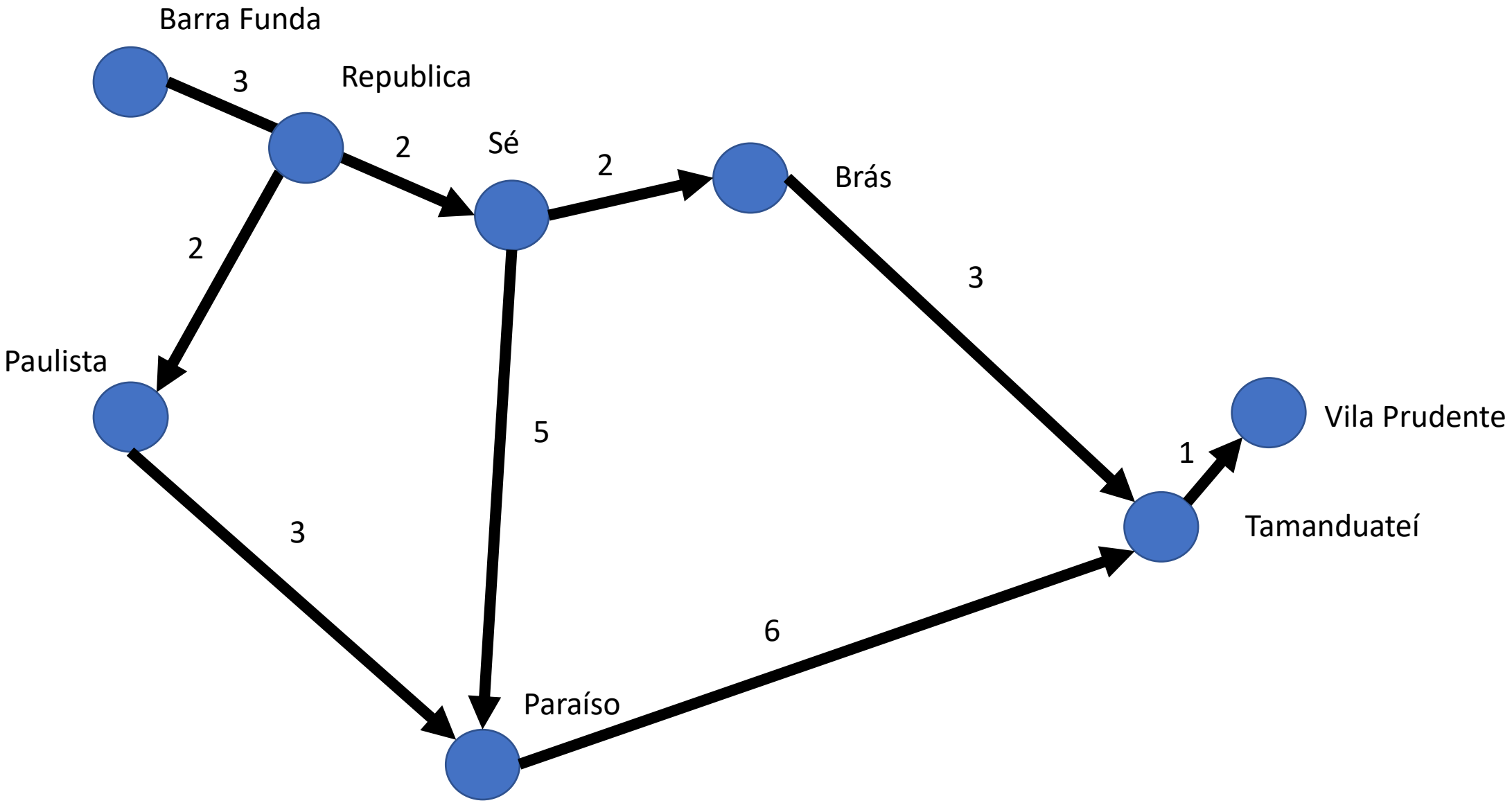
Cada Estação tem Custo 1

Solução

- Quantas Rotas Existem ?
- **Linha Azul, Vermelha, Amarela Custo 8**
- **Linha Azul e Verde Custo 7**

Você está na Barra Funda e quer ir na Vila Prudente

3 Rotas


Barra Funda
3
Republica
2
Sé
2
Brás
2
3
Paulista
5
Vila Prudente
1
3
Tamanduateí
6
Paraíso

Rota 1

Estado Inicial:

Em(_____)

Função Sucessor

Vai_de_para(Incial,Final) → Ligação(Inicial, Final)

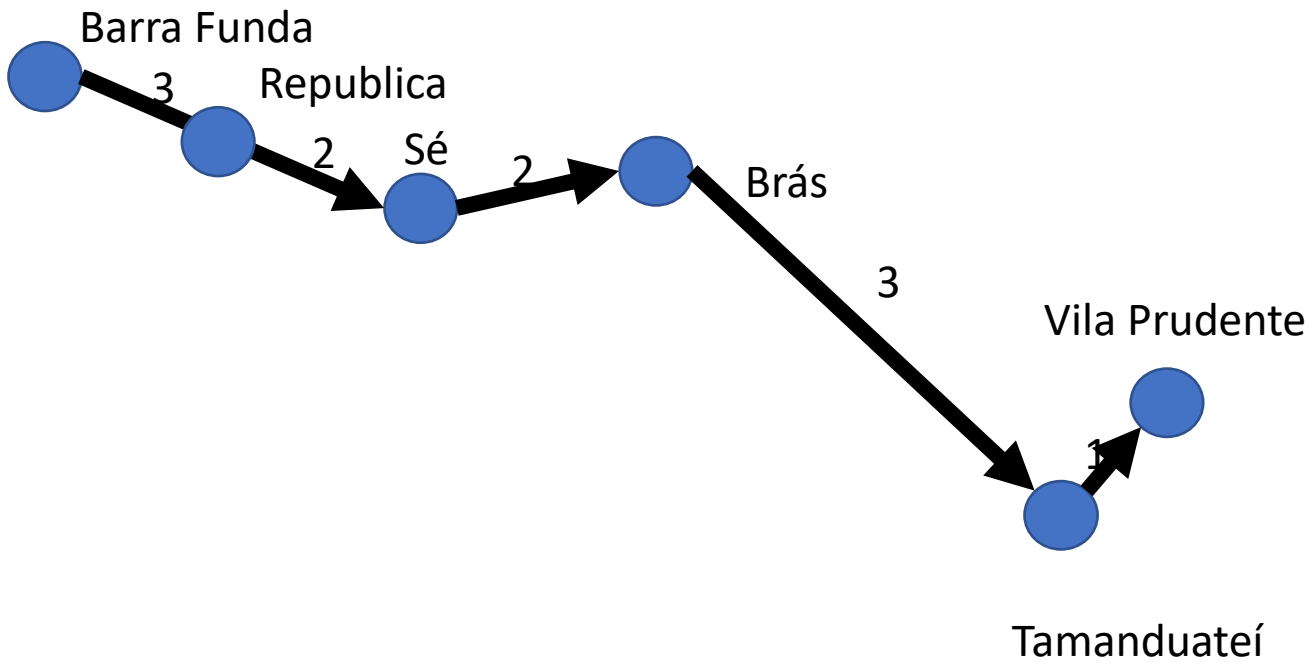
Teste de Objetivo :objetivo(Vergueiro)

objetivo(_____)

Custo de Caminho

Cada Estação tem Custo 1

Qual o custo



Rota 1

Estado Inicial:

Em(_____)

Função Sucessor

Vai_de_para(Incial,Final) → Ligação(Inicial, Final)

Teste de Objetivo :objetivo(Vergueiro)

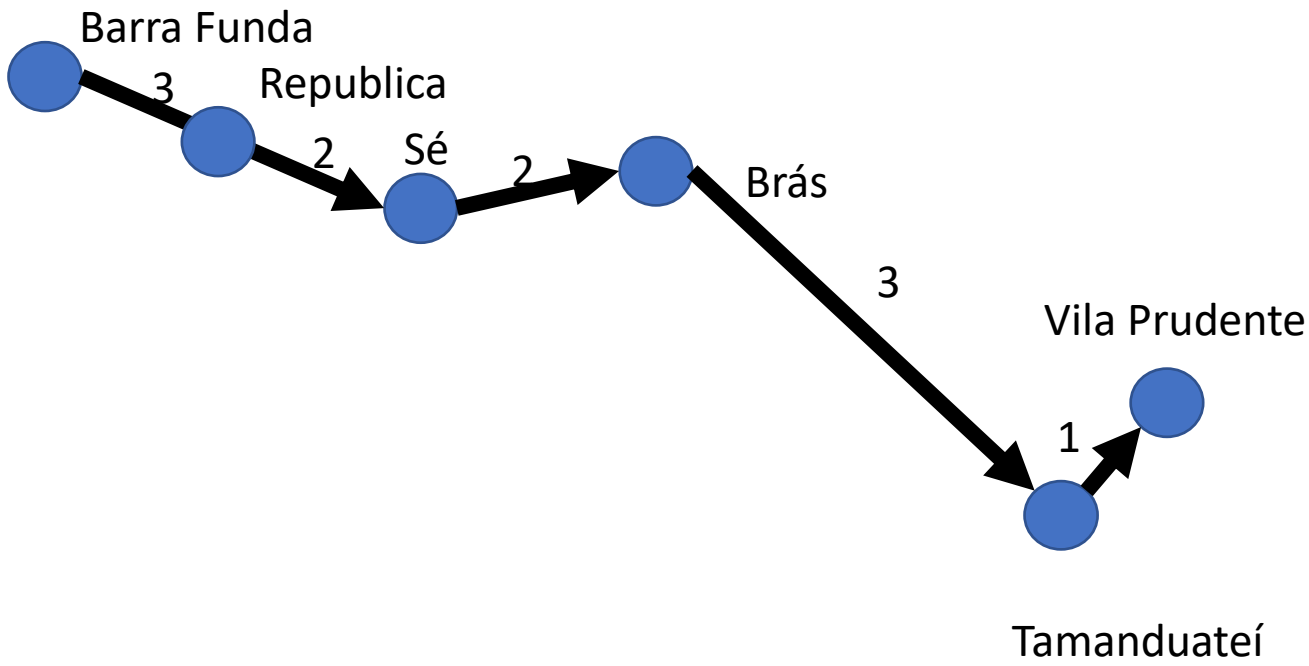
objetivo(_____)

Custo de Caminho

Cada Estação tem Custo 1

Qual o custo

8 sem translado
12 com translado



Rota 1

Estado Inicial:

Em(**Barra Funda**)

Função Sucessor

Vai_de_para(Incial,Final) → Ligação(Inicial, Final)

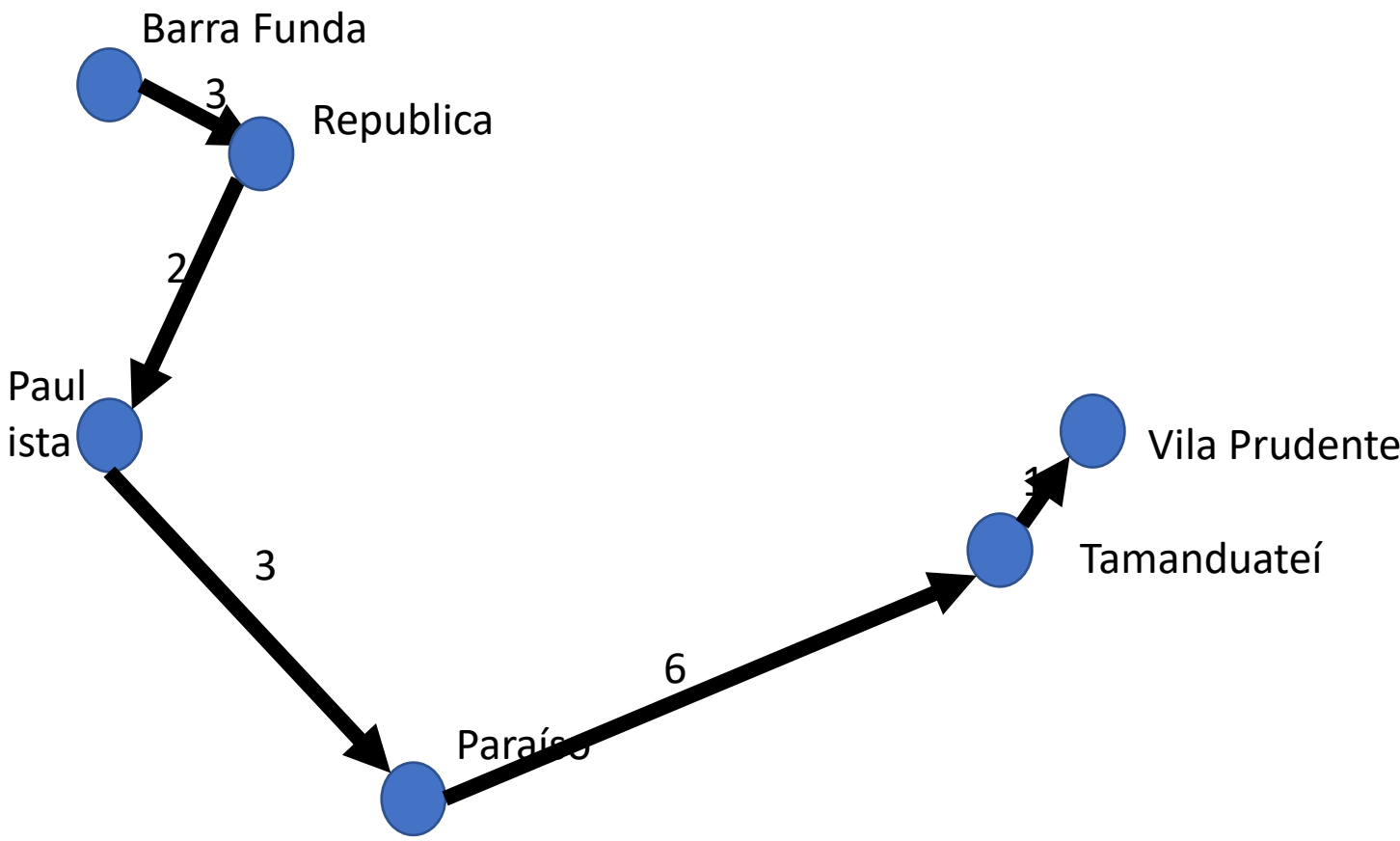
Teste de Objetivo :objetivo(Vergueiro)

objetivo(**Vila Prudente**)

Custo de Caminho

Cada Estação tem Custo 1

Qual o custo



Rota 1

Estado Inicial:

Em(**Barra Funda**)

Função Sucessor

Vai_de_para(Incial,Final) → Ligação(Inicial, Final)

Teste de Objetivo :objetivo(Vergueiro)

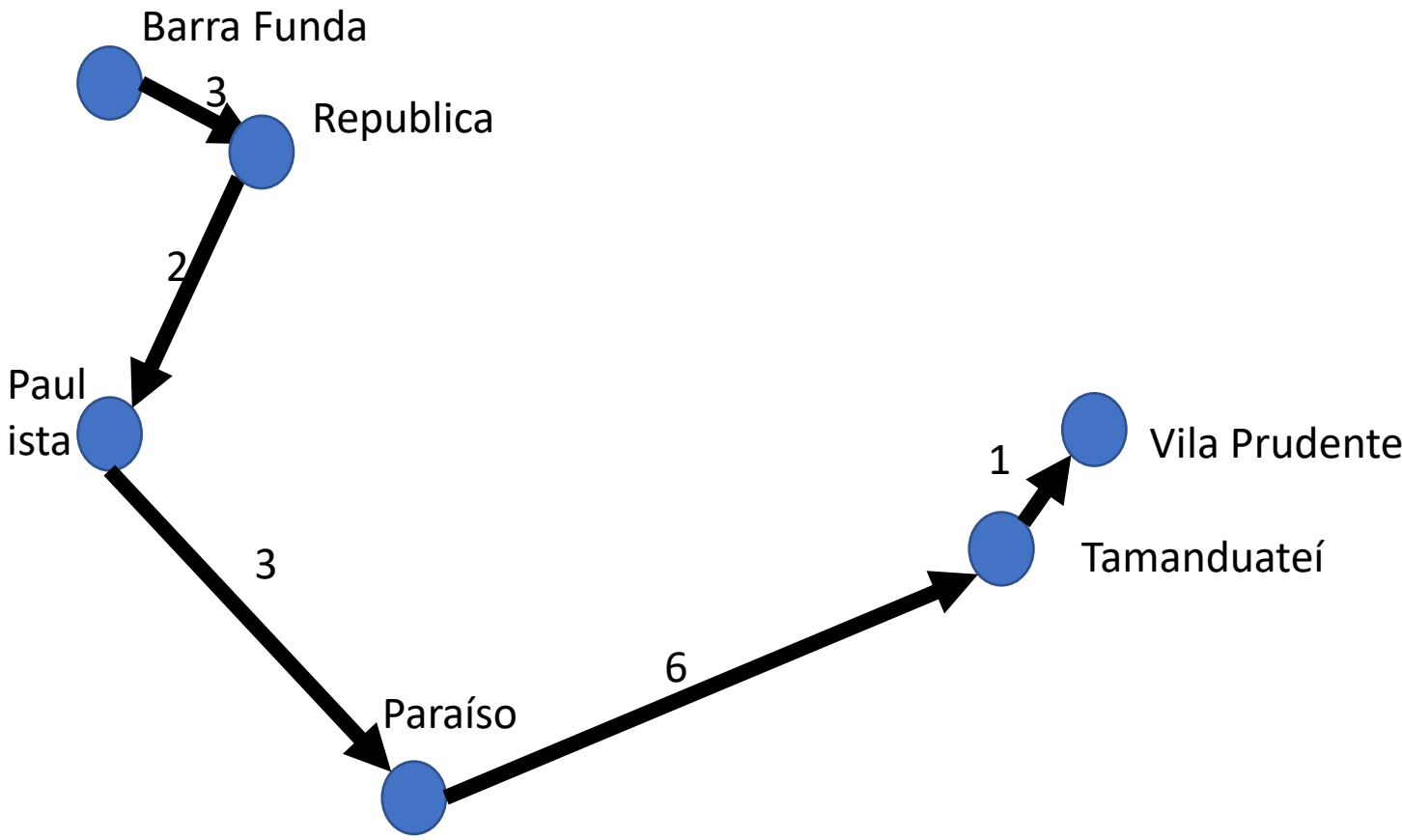
objetivo(**Vila Prudente**)

Custo de Caminho

Cada Estação tem Custo 1

Qual o custo

15 sem translado
19 com translado



Rota 2

Estado Inicial:

Em(_____)

Função Sucessor

Vai_de_para(Incial,Final) → Ligação(Inicial, Final)

Teste de Objetivo :objetivo(Vergueiro)

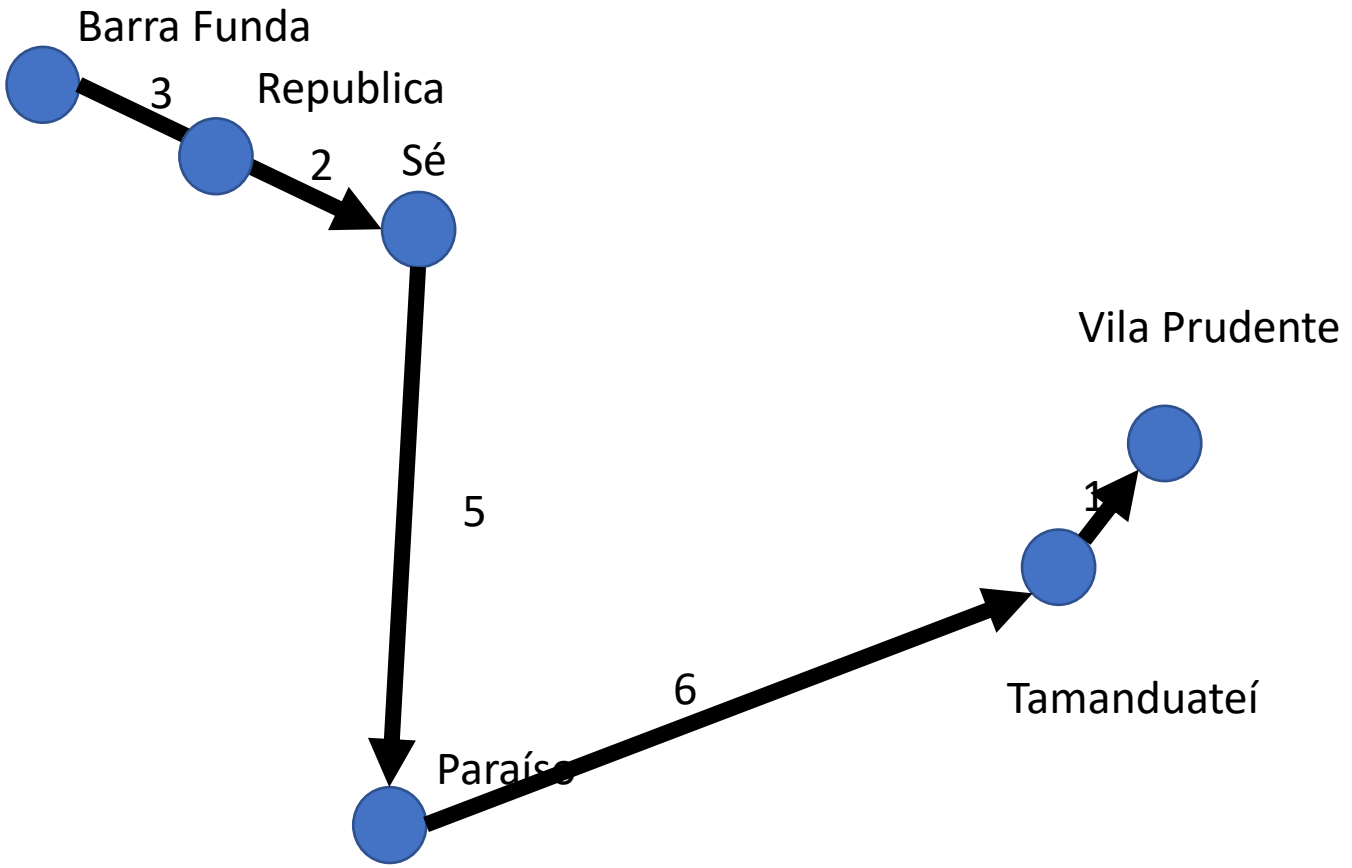
objetivo(_____)

Custo de Caminho

Cada Estação tem Custo 1

Qual o custo

- 17 sem translados
- 19 com transalados



Rota 2

Estado Inicial:

Em(_____)

Função Sucessor

Vai_de_para(Incial,Final) → Ligação(Inicial, Final)

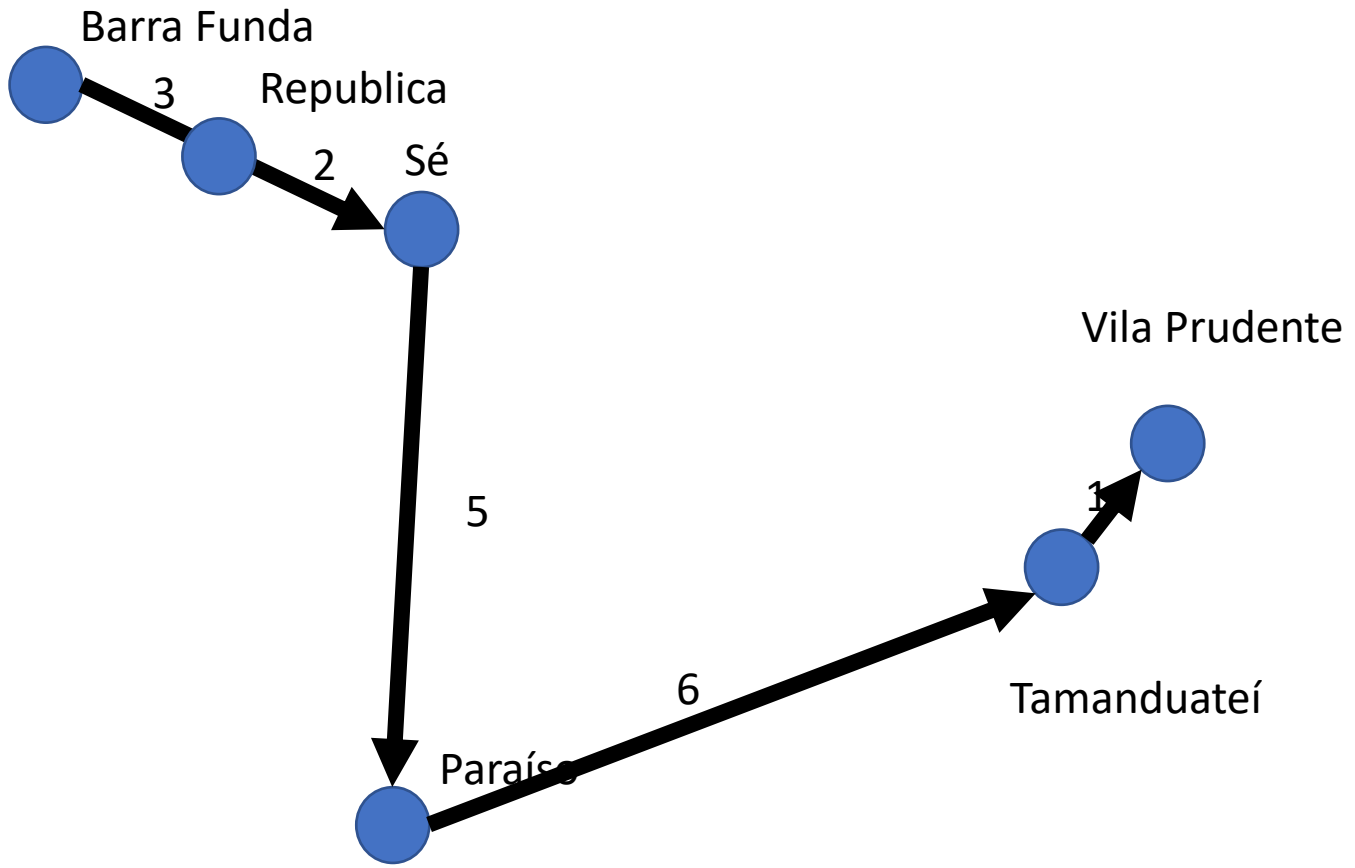
Teste de Objetivo :objetivo(Vergueiro)

objetivo(_____)

Custo de Caminho

Cada Estação tem Custo 1

Qual o custo



Em Busca de Soluções

Busca em todo o espaço de estados

- Uso de uma ***Árvore* de busca** explícita – gerada pelo estado inicial e pela função sucessor.

- Uso de um ***grafo* de busca** (substituindo a árvore de busca) – o mesmo estado pode ser alcançado a partir de vários caminhos.

Espaço de Estados do Problema

- Um problema pode ser visto como uma tripla: {***I***,***O***,***B***}
  - ***I*** = estados iniciais
  - ***O*** = conjunto de operações
  - ***B*** = estados objetivo

- Uma solução para o problema é uma seqüência finita de operações que permite sair de um elemento em ***I*** e chegar a um elemento em ***B***.

Espaço de Estados do Problema

- **Um sistema de resolução de problemas comporta**:
  - Um conjunto de estruturas de dados organizada em um grafo;
  - Um conjunto de operadores caracterizados por suas condições de aplicação e sua ação;
  - Uma estrutura de controle implementando a estratégia de resolução.

Estratégias de Busca

Abordagens de busca básicas num espaço de estados:

- **Busca Cega** (Sem informação/Não informada)
  - Não tem informação sobre qual sucessor é mais promissor para atingir a meta.

- **Busca Heurística** (Busca Com Informação/Informada)
  - Possui informação (estimativa) de qual sucessor é mais promissor para atingir a meta.
  - É uma busca cega com algum guia ou orientação.

- **Todas as estratégias de busca se distinguem pela ordem em que os nós são expandidos.**

Estratégias de Busca Cega

- **Busca em Largura**
- Busca de Custo Uniforme
- **Busca em Profundidade**
- Busca em Profundidade Limitada
- Busca em Profundidade com Aprofundamento Iterativo
- Busca Bidirecional
- Evitando Estados Repetidos
- Busca com Conhecimento Incompleto

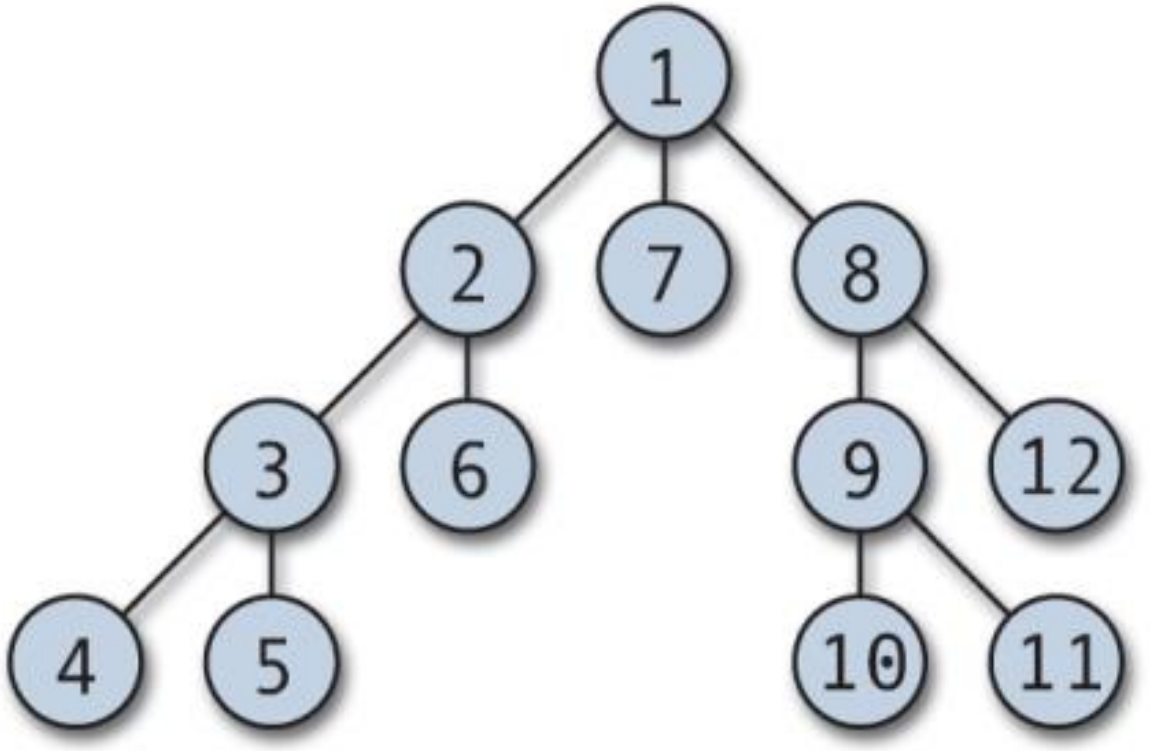
Busca em Profundidade ...

- **Ordem de expansão dos nós**:

  1. Nó raiz
  2. Primeiro nó de profundidade 1
  3. Primeiro nó de profundidade 2, etc …

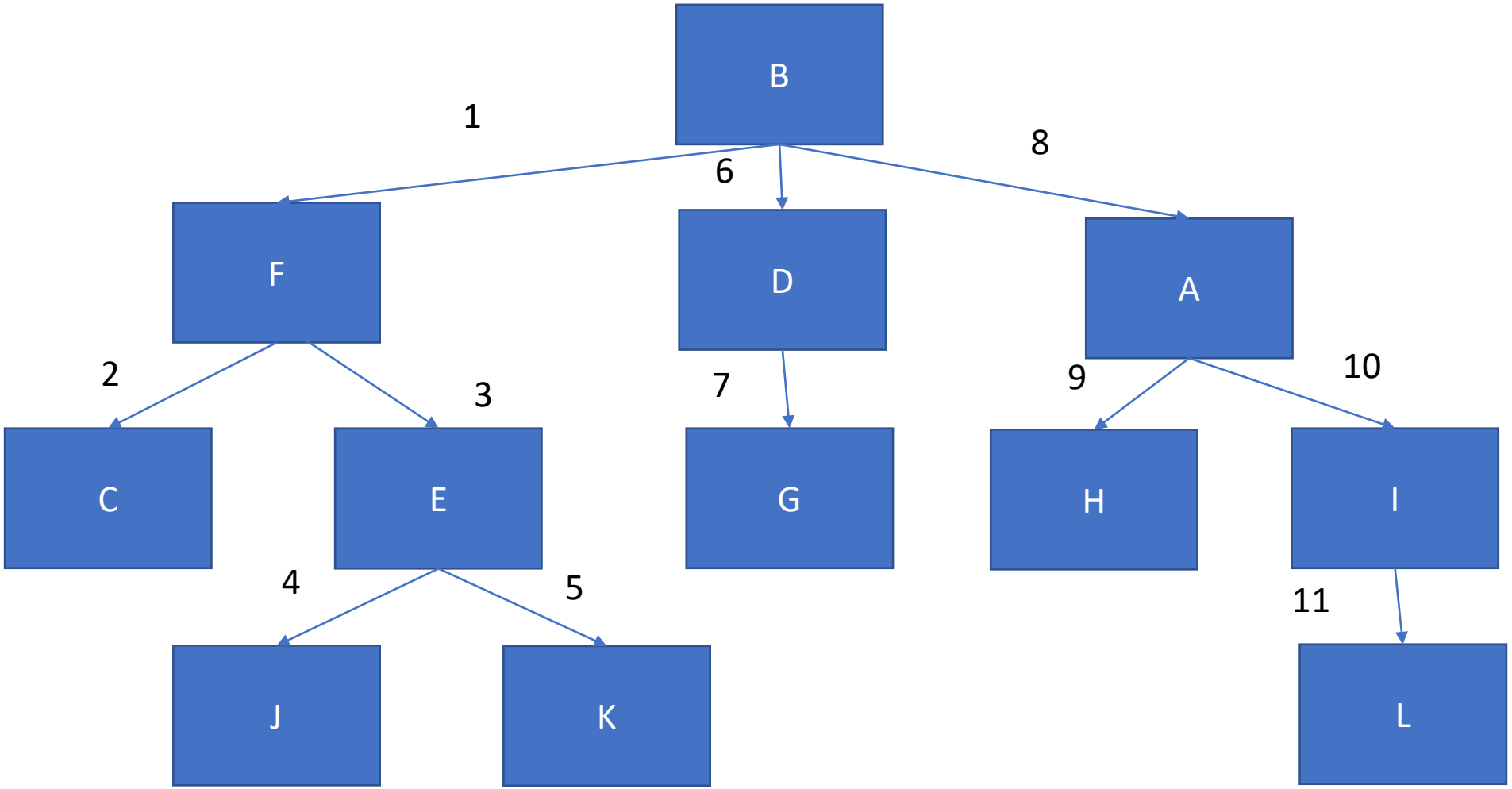
Busca em Profundidade ...

- Começa na raiz e avança para baixo em níveis cada vez mais profundos;
- Um operador é aplicado a um nó para gerar o próximo nó mais profundo na sequência;
- O processo continua até que uma solução é encontrada ou um retrocesso é forçado ao atingir-se um nó terminal que não é solução.



Busca em Profundidade ...

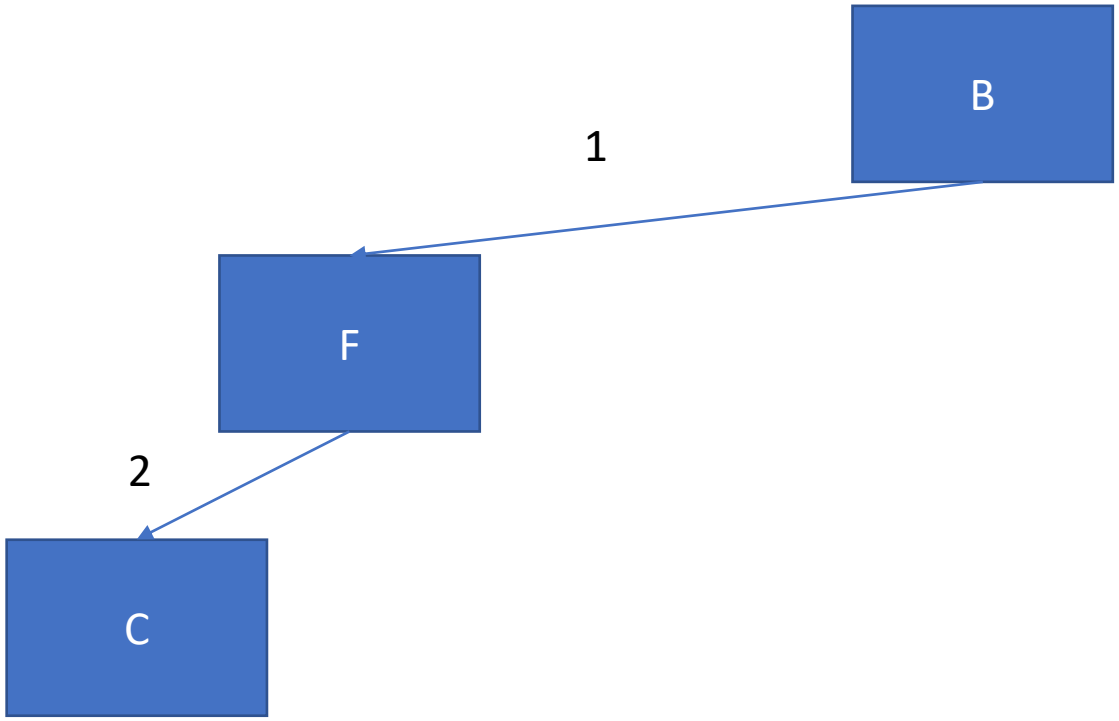
- Faz uma busca sistemática em cada filho de um nó até encontrar a meta.
- **Exemplo**: O caminho para se chegar ao nó **G**, usando busca em profundidade: **Caminho = { B, F, C, E, J, K, D, G }**



Busca em Profundidade ...

- Faz uma busca sistemática em cada filho de um nó até encontrar a meta.
- **Exemplo**: O caminho para se chegar ao nó **G**, usando busca em profundidade: **Caminho = { B, F, C, E, J, K, D, G }**

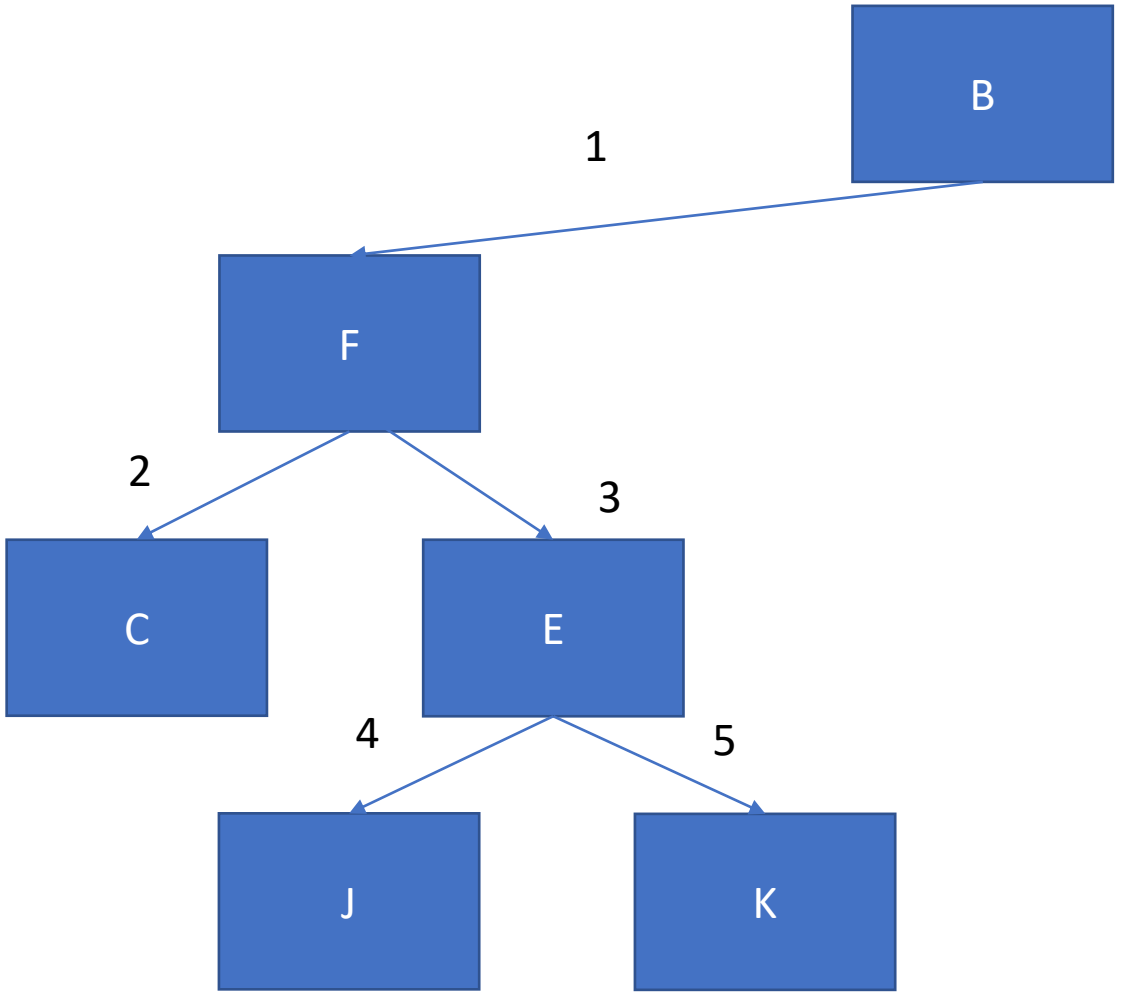
Caminho:
B
F
C



Busca em Profundidade ...

- Faz uma busca sistemática em cada filho de um nó até encontrar a meta.
- **Exemplo**: O caminho para se chegar ao nó **G**, usando busca em profundidade: **Caminho = { B, F, C, E, J, K, D, G }**

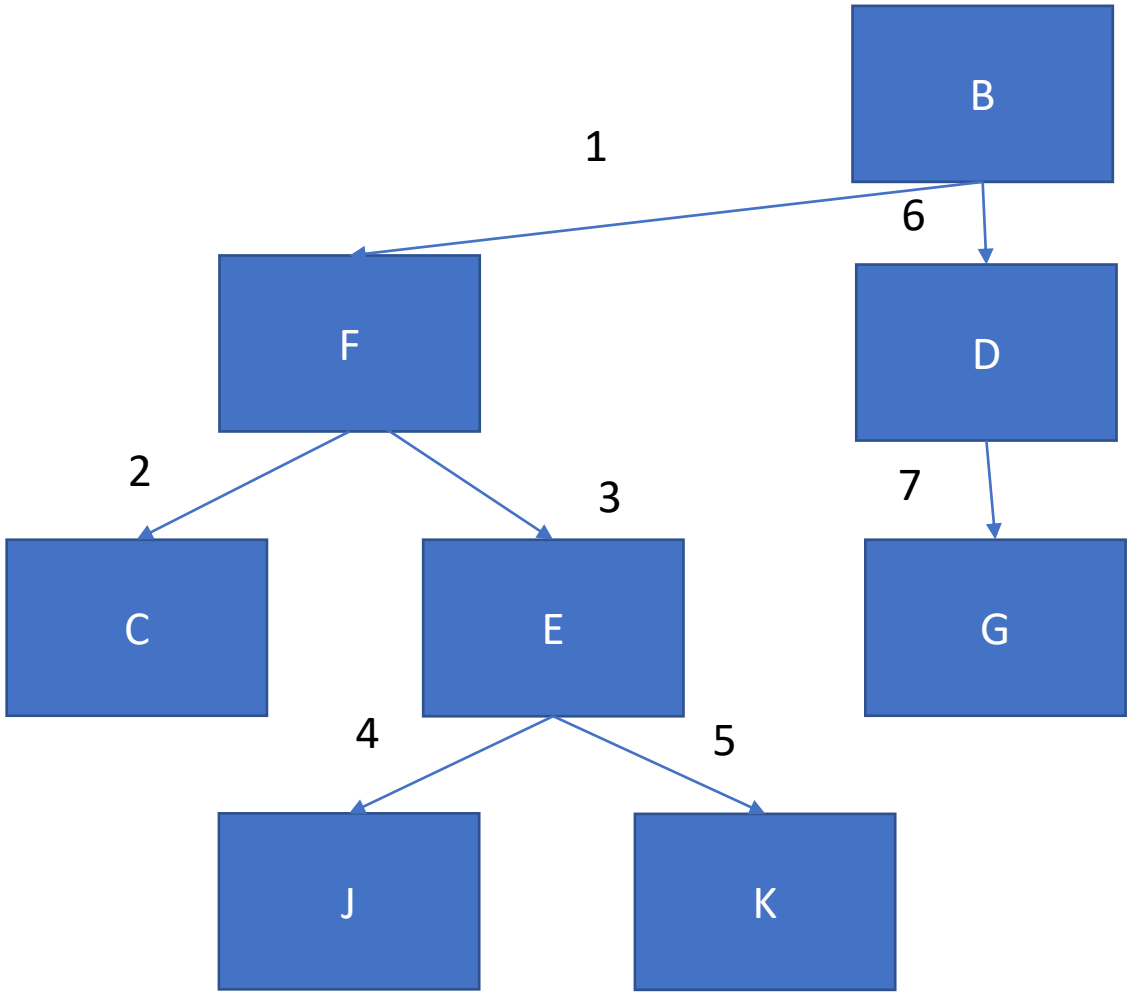
Caminho:
B
F
C
E
J
K



Busca em Profundidade ...

- Faz uma busca sistemática em cada filho de um nó até encontrar a meta.
- **Exemplo**: O caminho para se chegar ao nó **G**, usando busca em profundidade: **Caminho = { B, F, C, E, J, K, D, G }**

Caminho:
B
F
D
E
J
K
D
G



Busca em Profundidade ...

Problema

- Garante uma solução, mas a busca pode ser muito demorada.

- **Motivo**: muitas ramificações diferentes podem ter que ser consideradas até o nível mais profundo antes de uma solução ser atingida.

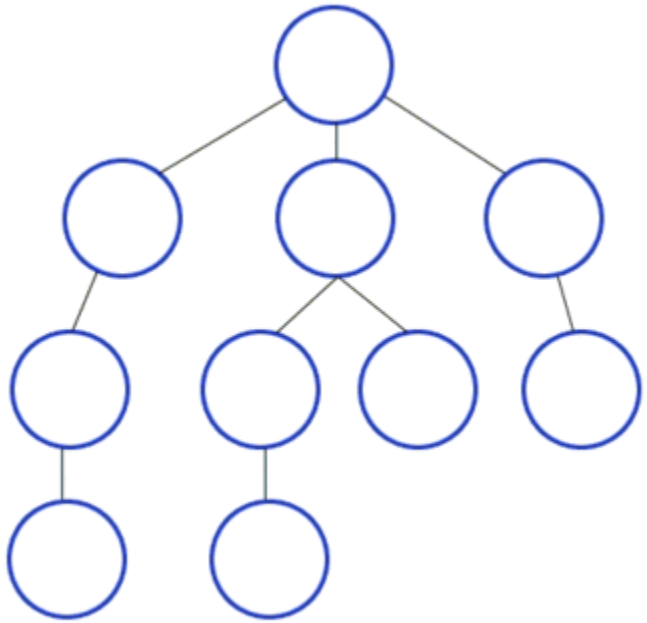
Busca em Largura ...

- **Ordem de expansão dos nós**:

1. Nó raiz

2. Todos os nós de profundidade 1

3. Todos os nós de profundidade 2, etc …



Busca em Largura ...

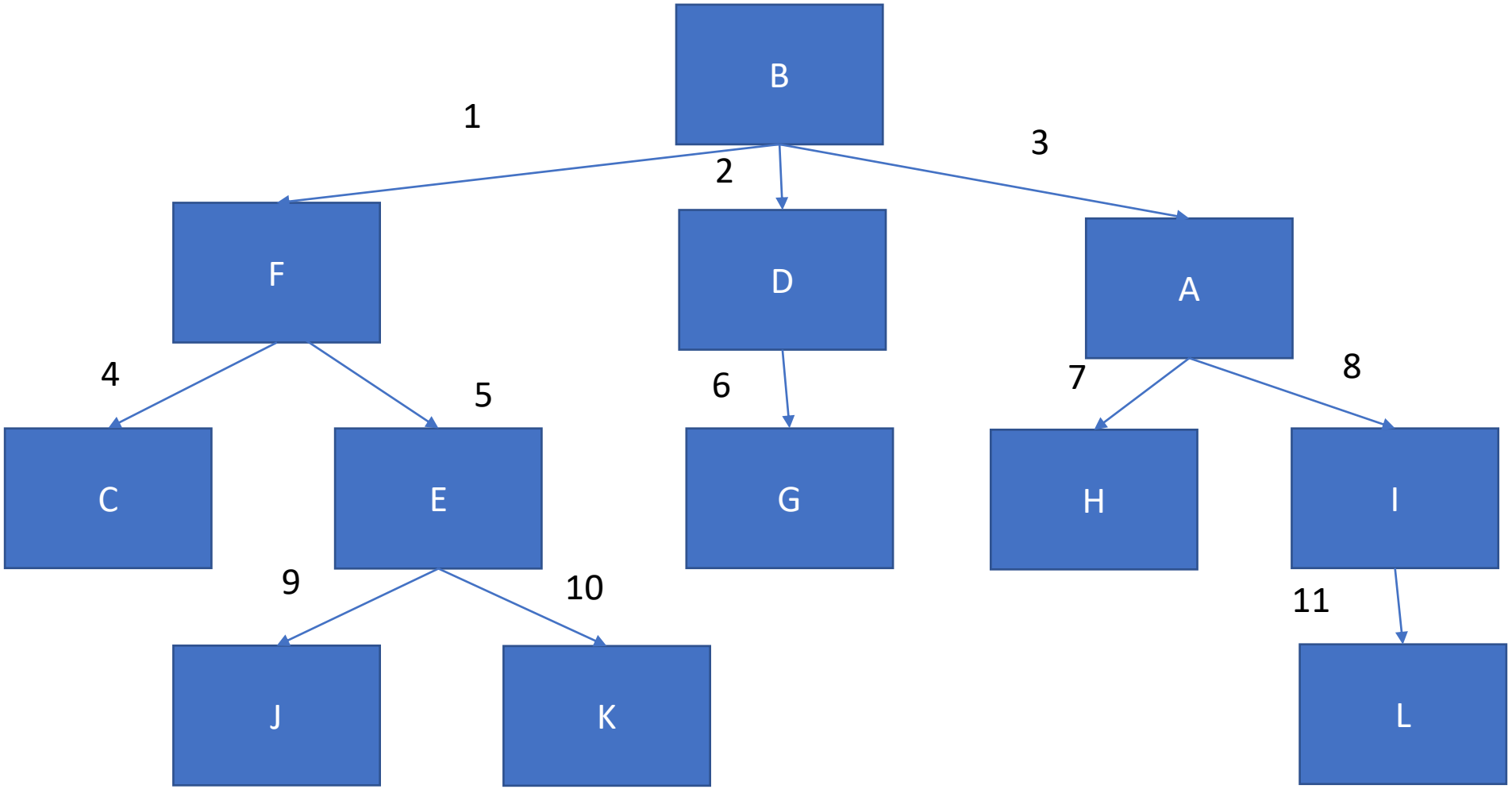
- Os nós em cada nível da árvore são completamente examinados antes de se mover para o próximo nível.

- Uma busca em largura sempre encontrará o menor caminho entre o estado inicial e o estado-objetivo.

- O menor caminho é o caminho com o menor número de passos (não confundir com o caminho de menor custo).

Busca em Largura ...

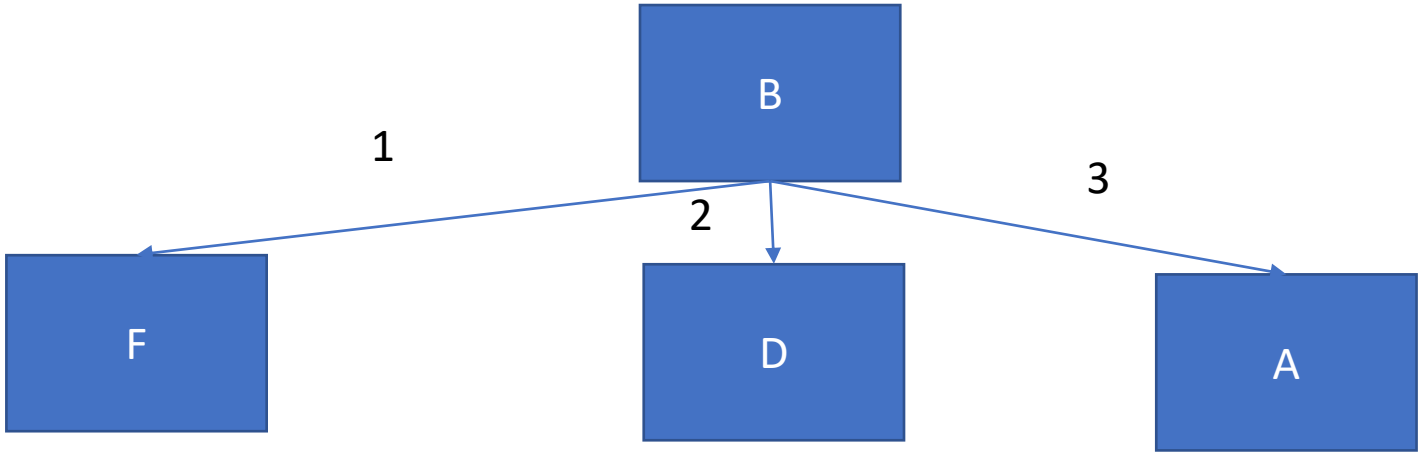
- Faz uma busca sistemática examinando primeiro os módulos próximos à raiz.
- **Exemplo**: Caminho para encontrar o nó **G**, usando a Busca em Largura: **Caminho = { B, F, D, A, C, E, G }**



Busca em Largura ...

- Faz uma busca sistemática examinando primeiro os módulos próximos à raiz.
- **Exemplo**: Caminho para encontrar o nó **G**, usando a Busca em Largura: **Caminho = { B, F, D, A, C, E, G }**

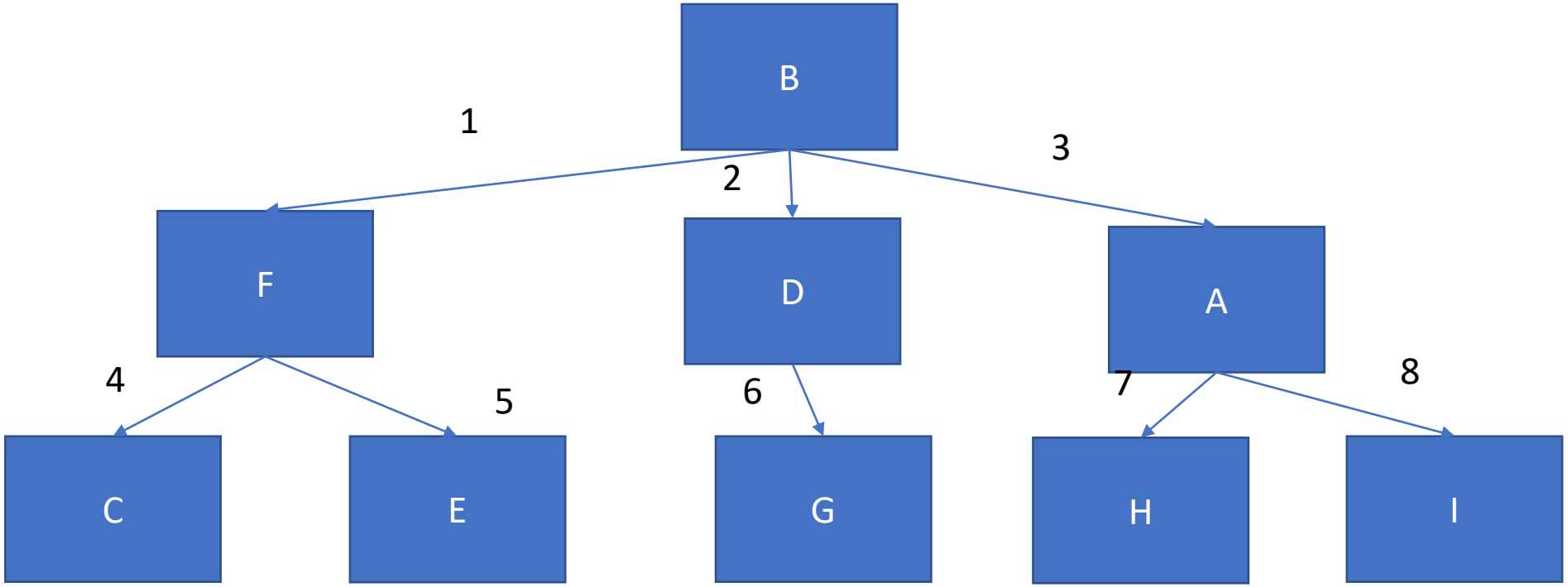
Caminho:
B
F
D
A



Busca em Largura ...

- Faz uma busca sistemática examinando primeiro os módulos próximos à raiz.
- **Exemplo**: Caminho para encontrar o nó **G**, usando a Busca em Largura: **Caminho = { B, F, D, A, C, E, G }**

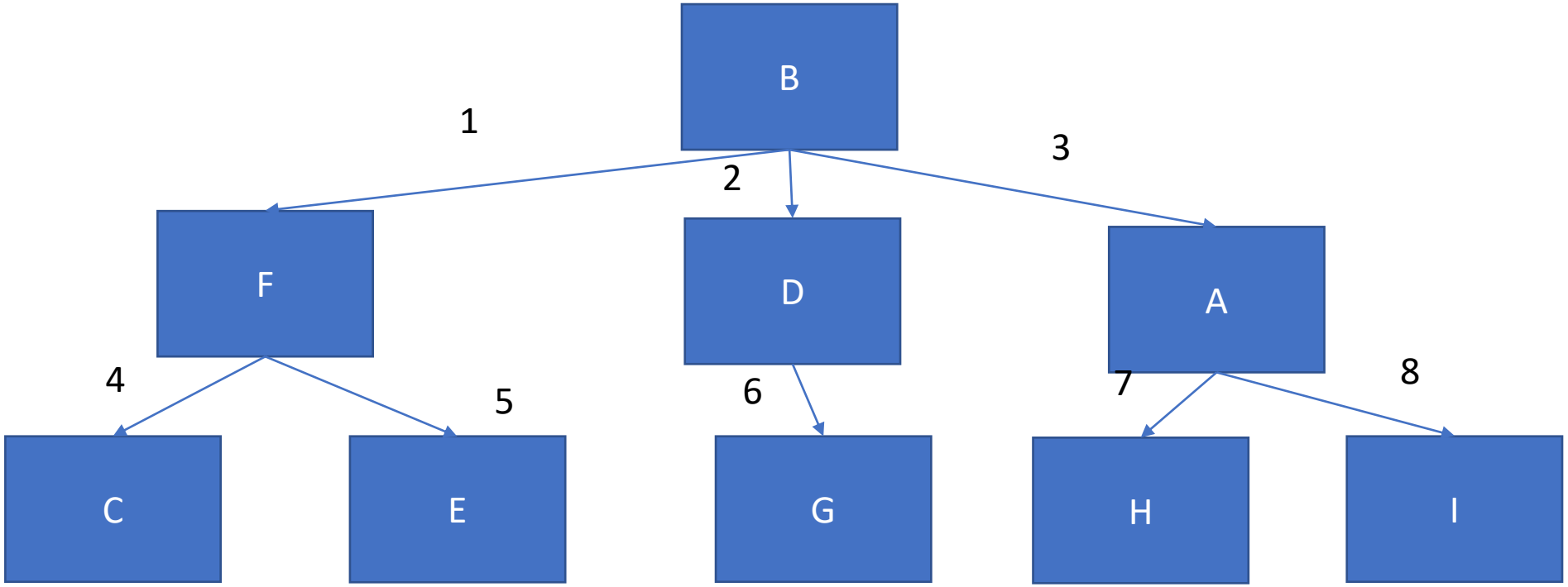
Caminho:
B
F
D
A
C
E
G



Busca em Largura ...

- Faz uma busca sistemática examinando primeiro os módulos próximos à raiz.
- **Exemplo**: Caminho para encontrar o nó **G**, usando a Busca em Largura: **Caminho = { B, F, D, A, C, E, G }**

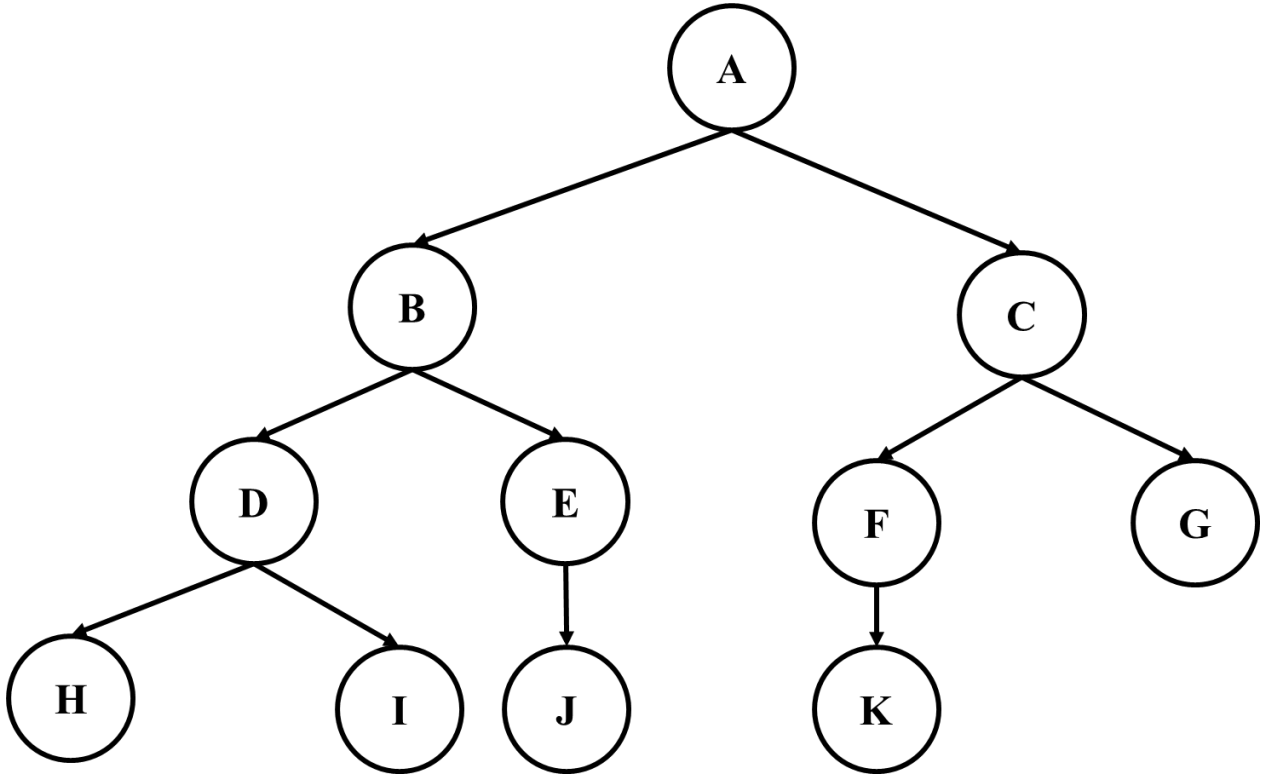
Caminho:
B
F
D
A



Exercicio

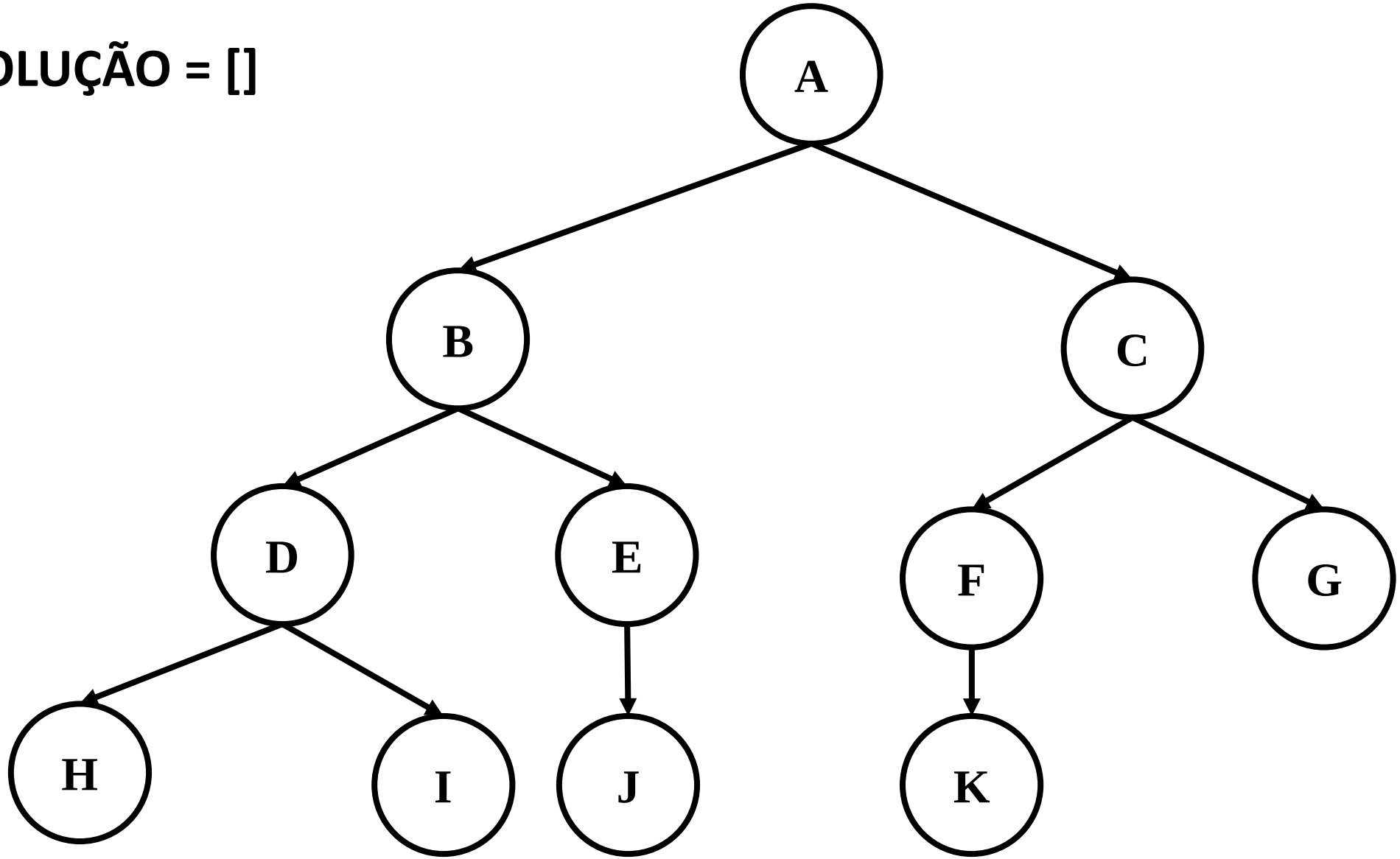
Dado o espaço de busca ao lado determine:

1. Qual o caminho tendo como objetivo J realizando uma busca em PROFUNDIDADE
2. Qual o caminho tendo como objetivo J realizando uma busca em LARGURA
3. Qual o caminho tendo como objetivo K realizando uma busca em PROFUNDIDADE
4. Qual o caminho tendo como objetivo K realizando uma busca em LARGURA



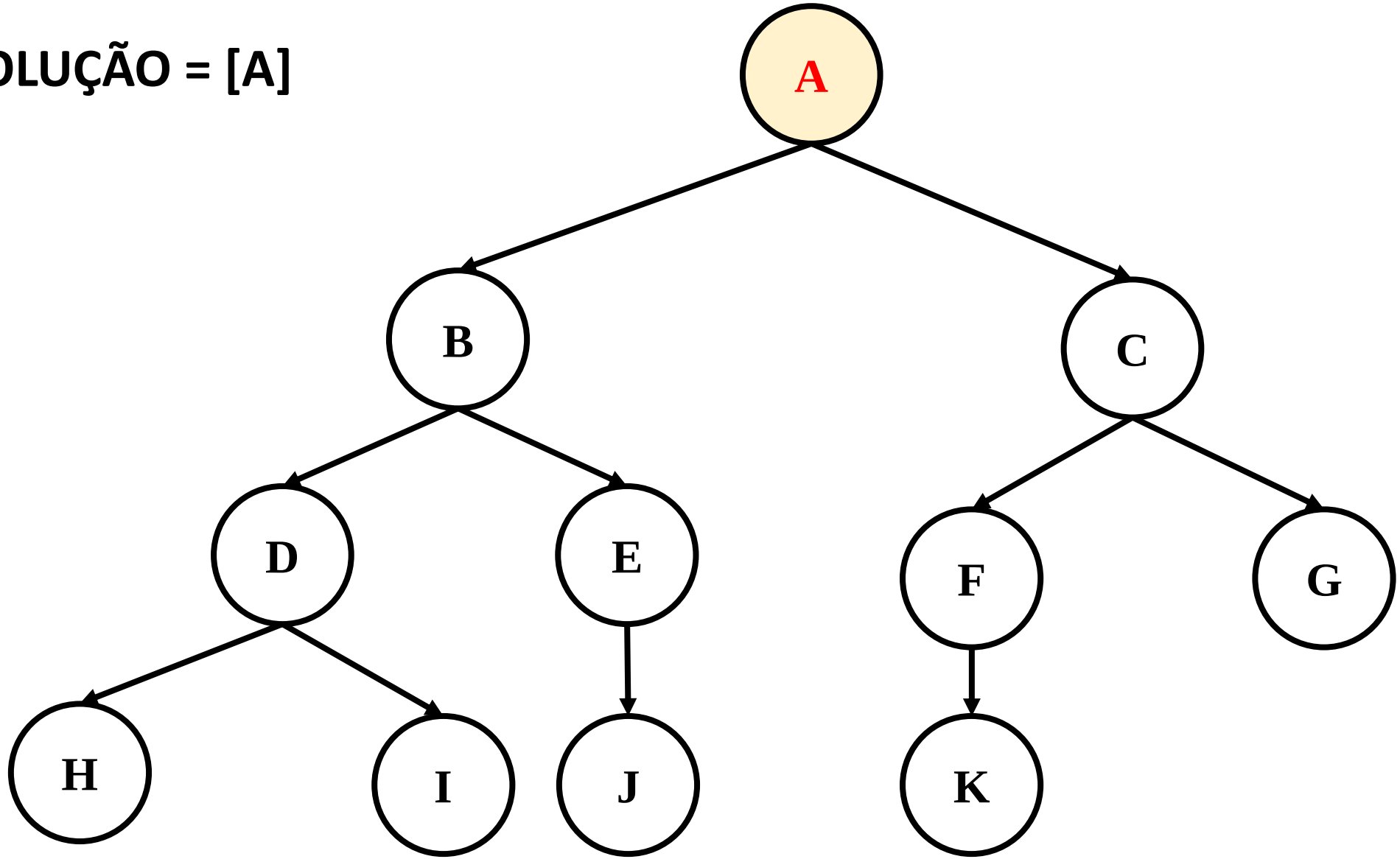
ORIGEM A OBJETIVO J PROFUNDIDADE

SOLUÇÃO = []



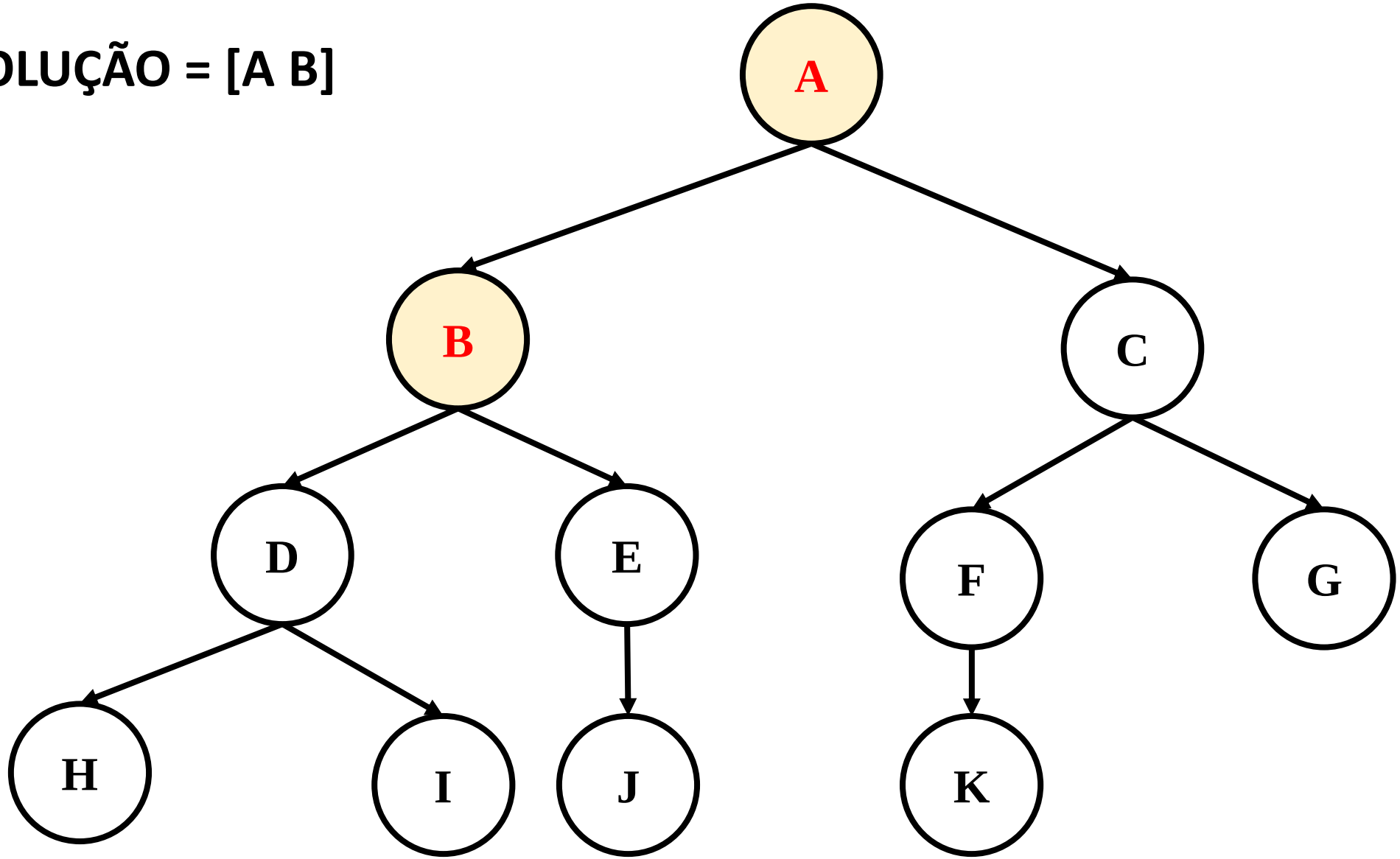
ORIGEM A OBJETIVO J PROFUNDIDADE

SOLUÇÃO = [A]



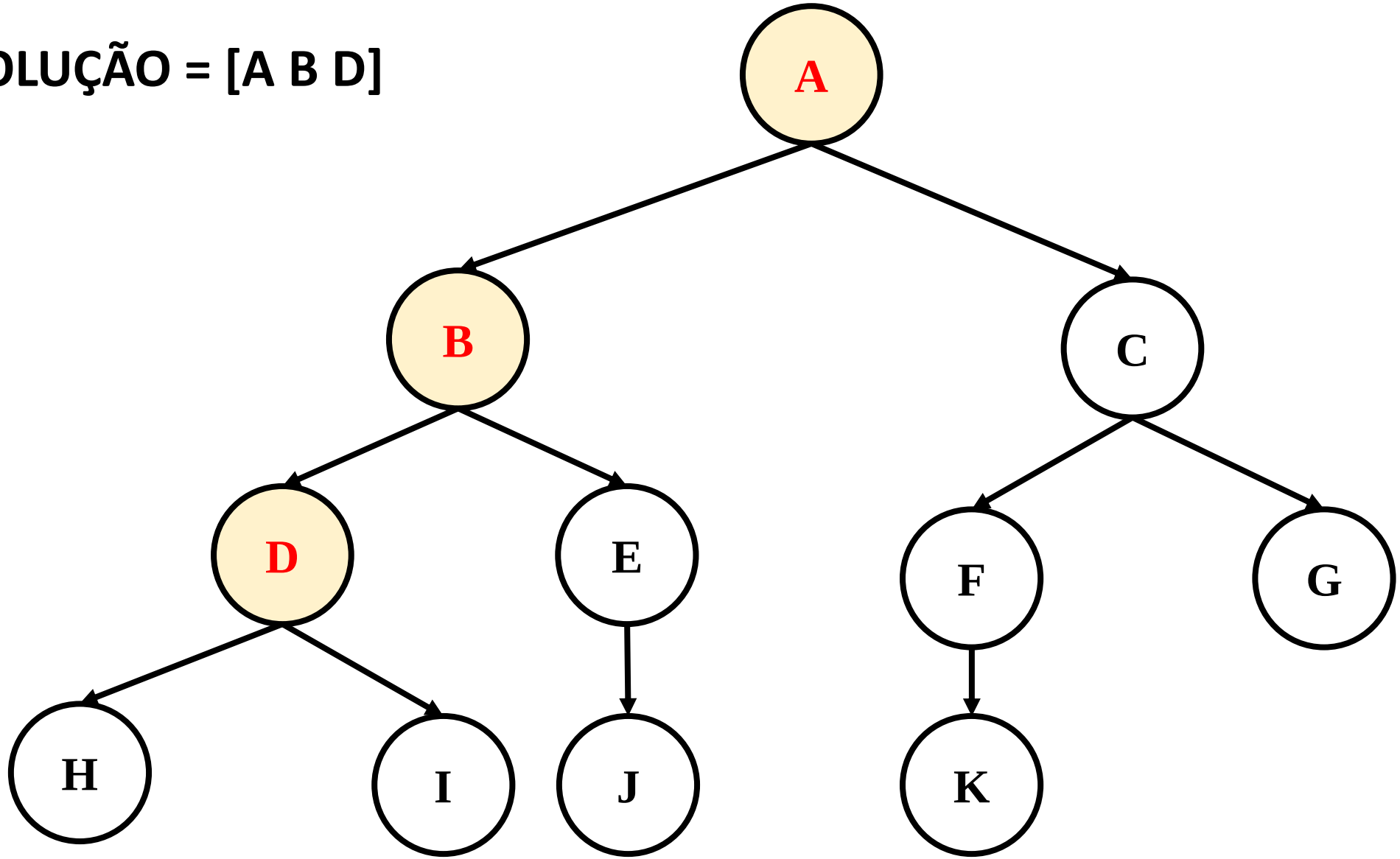
ORIGEM A OBJETIVO J PROFUNDIDADE

SOLUÇÃO = [A B]



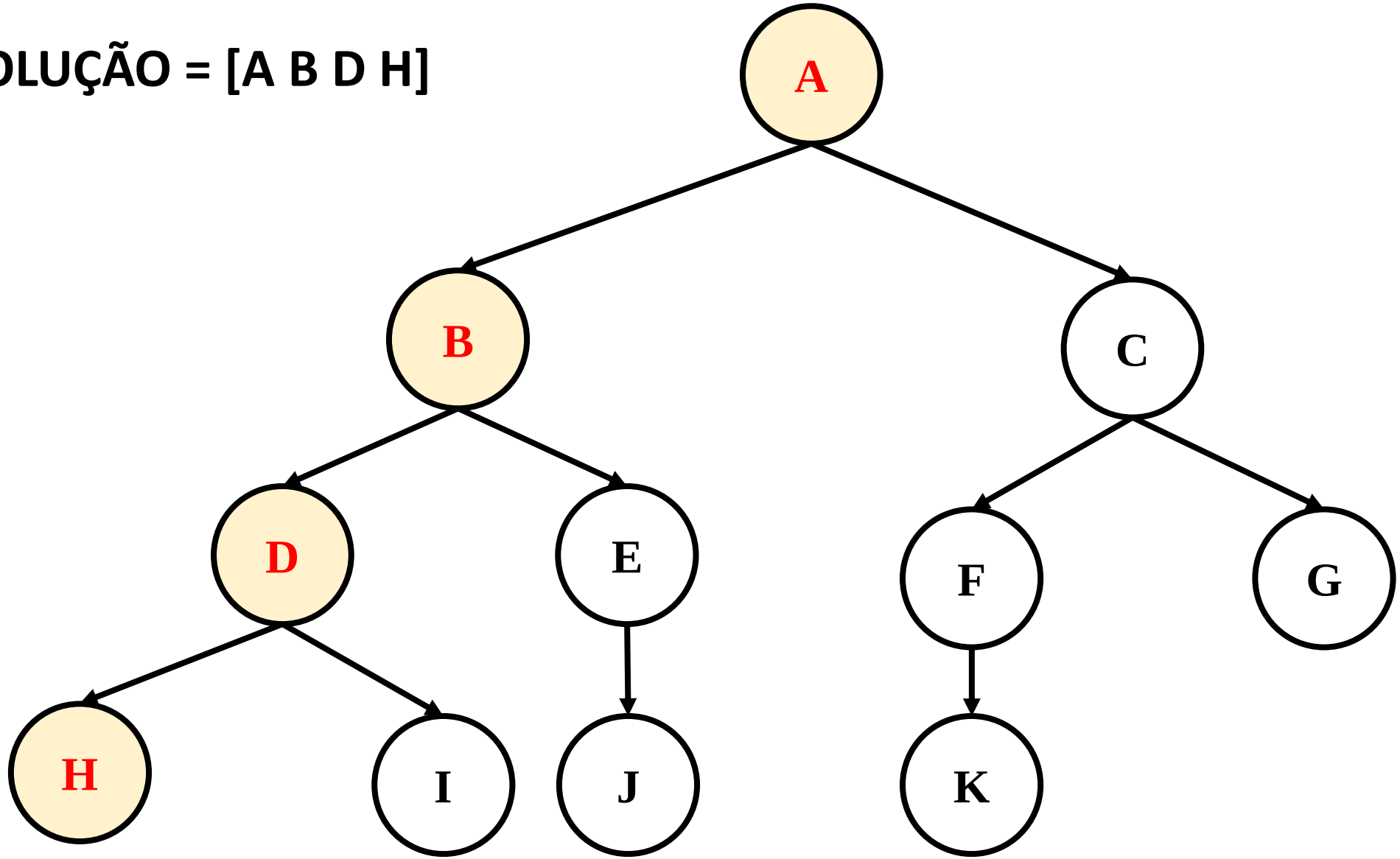
ORIGEM A OBJETIVO J PROFUNDIDADE

SOLUÇÃO = [A B D]



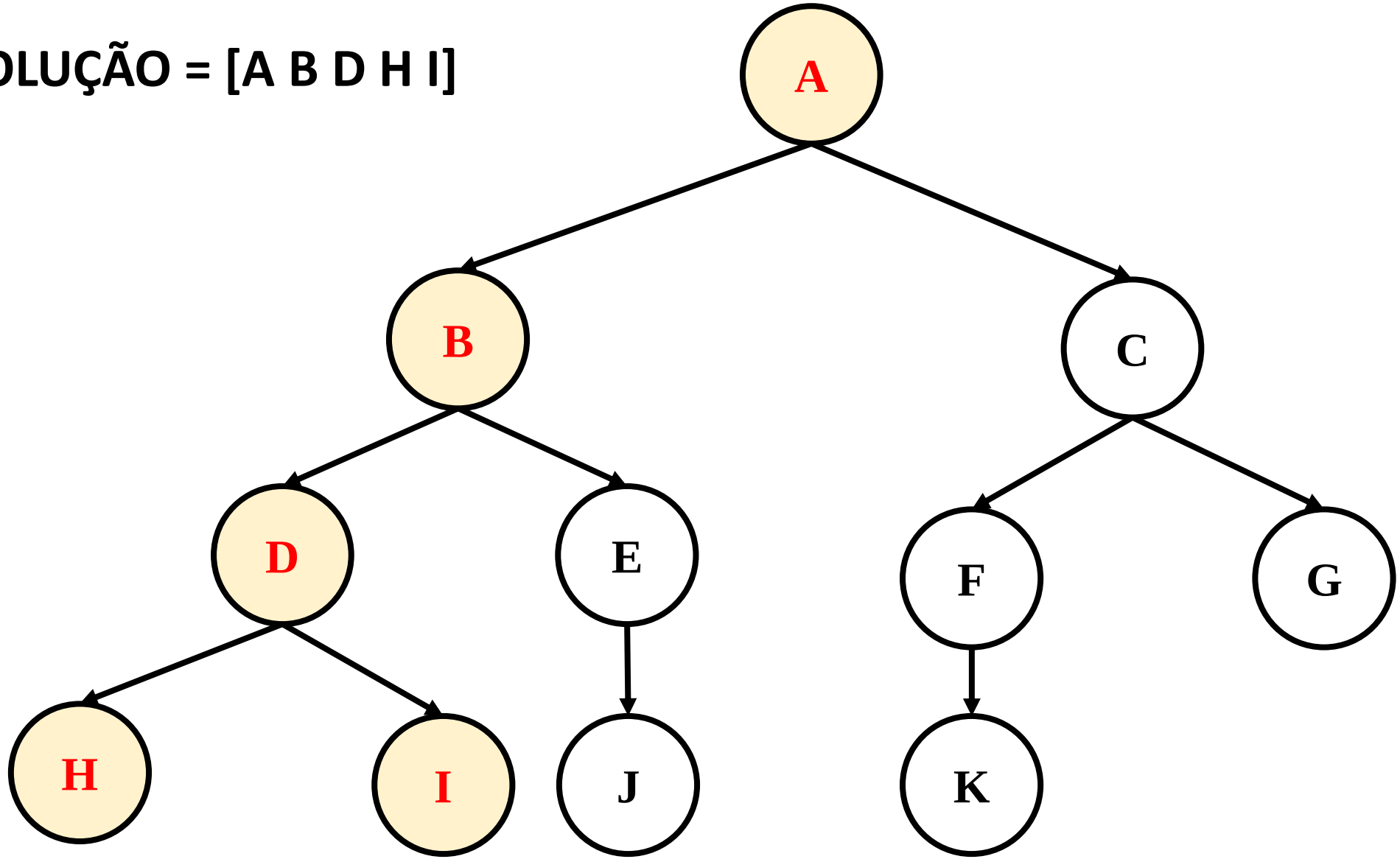
ORIGEM A OBJETIVO J PROFUNDIDADE

SOLUÇÃO = [A B D H]



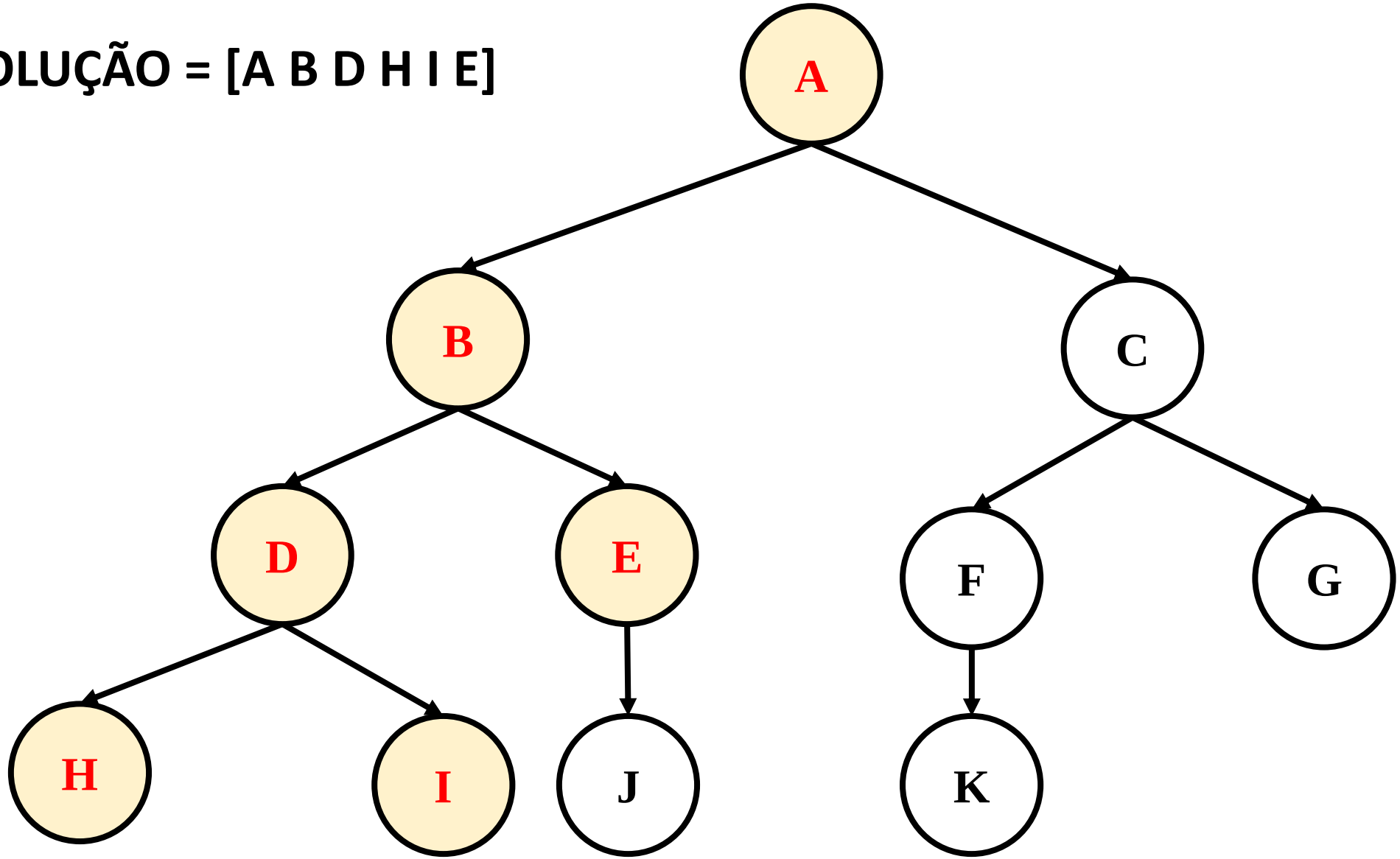
ORIGEM A OBJETIVO J PROFUNDIDADE

SOLUÇÃO = [A B D H I]



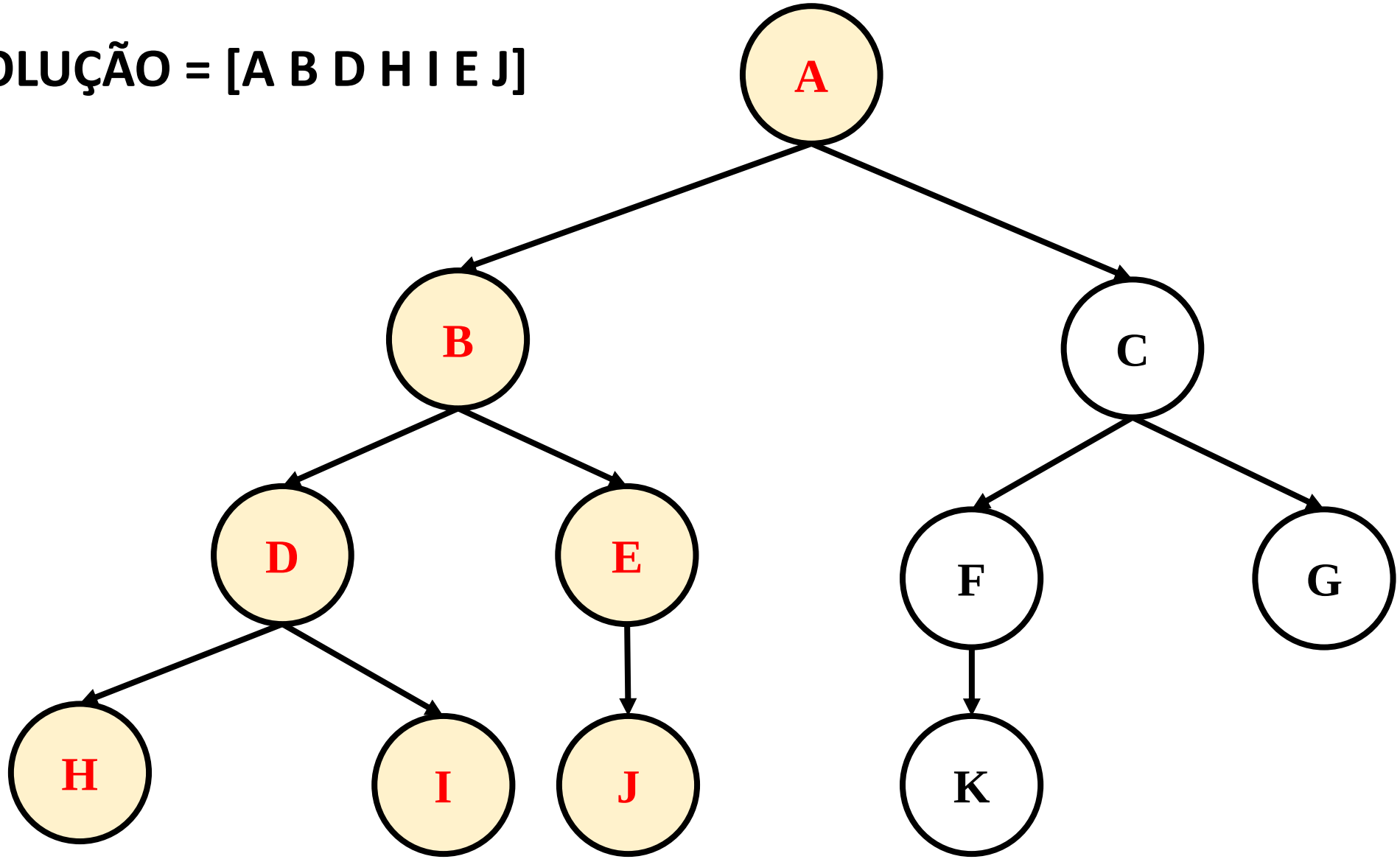
ORIGEM A OBJETIVO J PROFUNDIDADE

SOLUÇÃO = [A B D H I E]



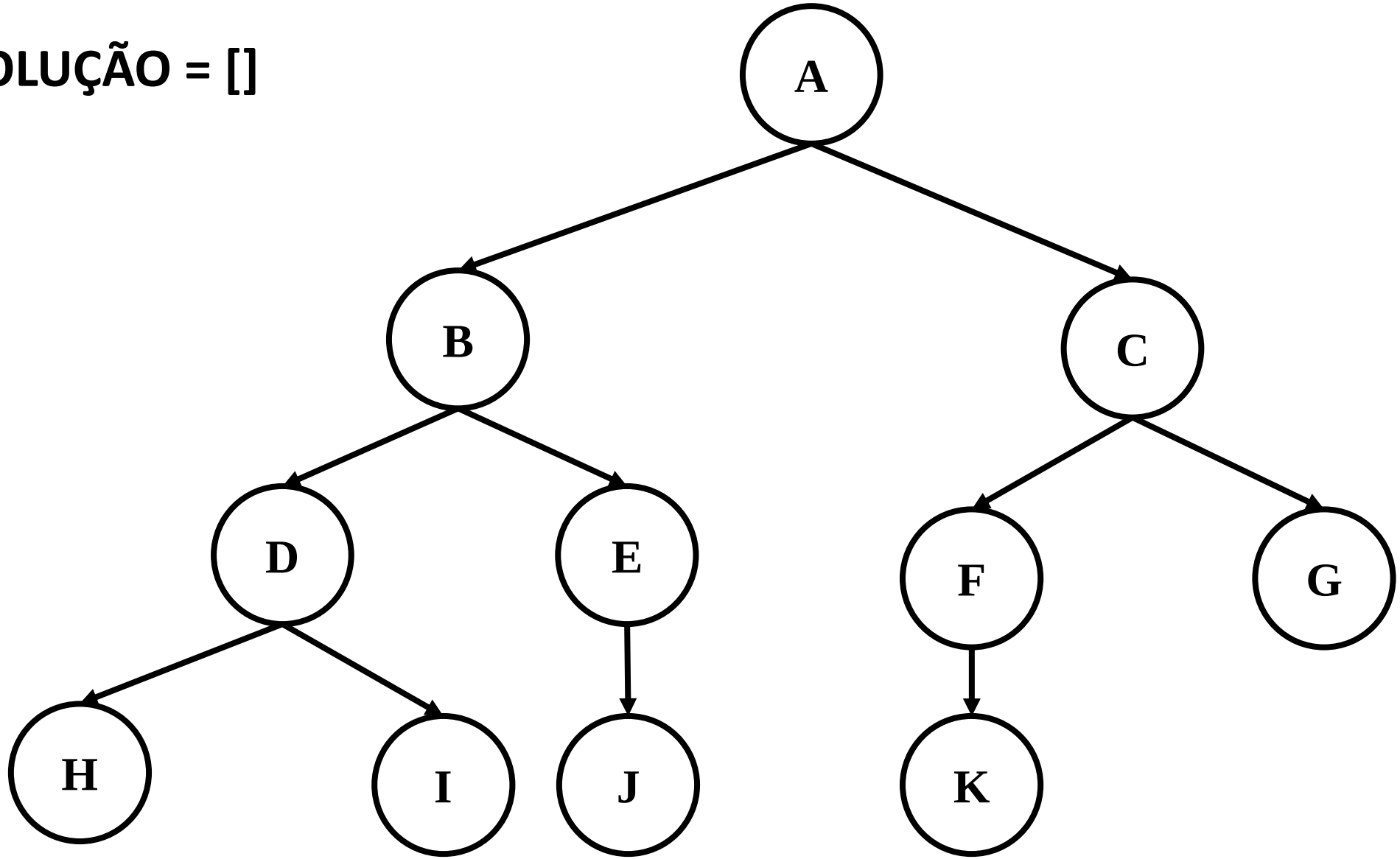
ORIGEM A OBJETIVO J PROFUNDIDADE

SOLUÇÃO = [A B D H I E J]



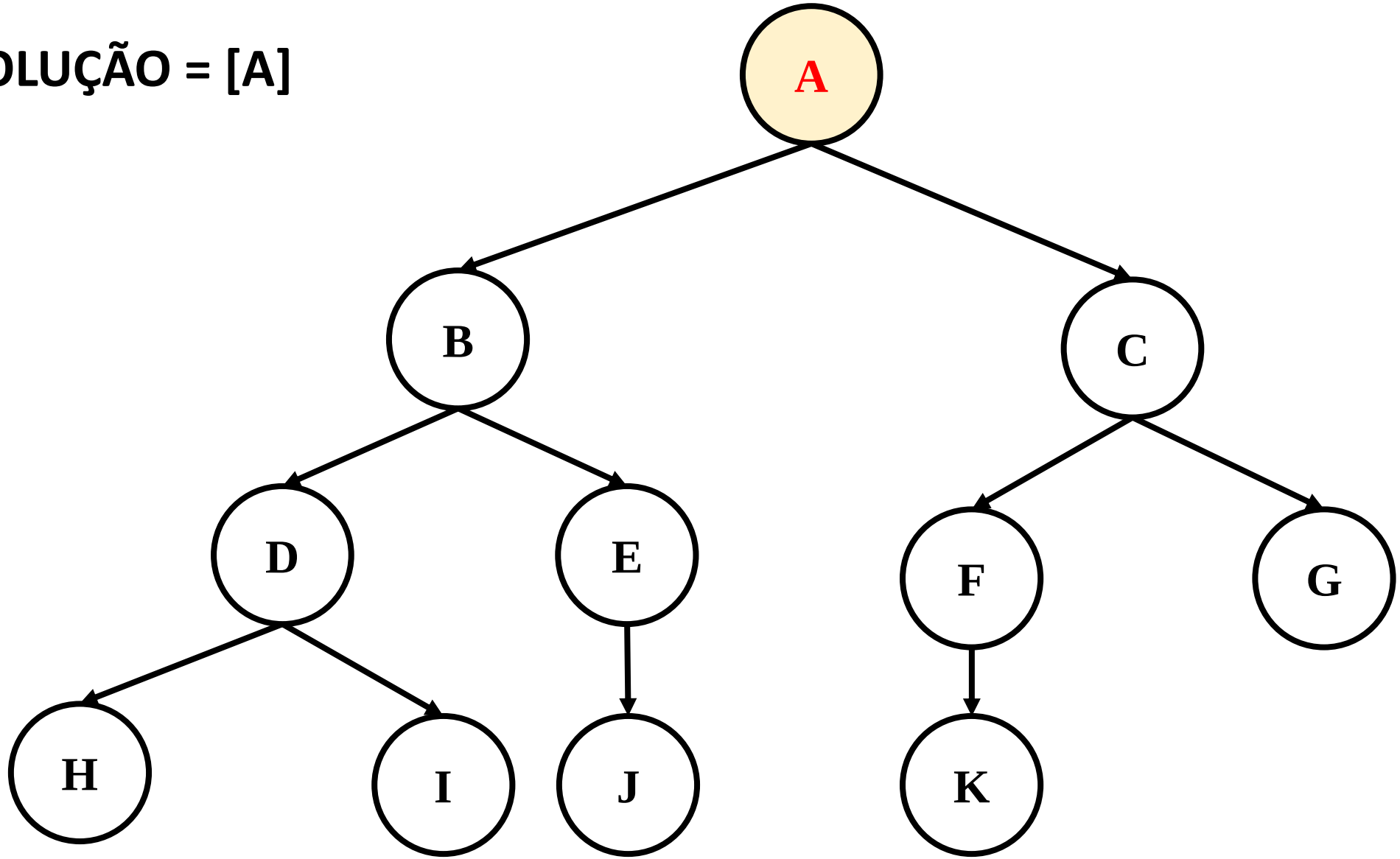
ORIGEM A OBJETIVO J LARGURA

SOLUÇÃO = []



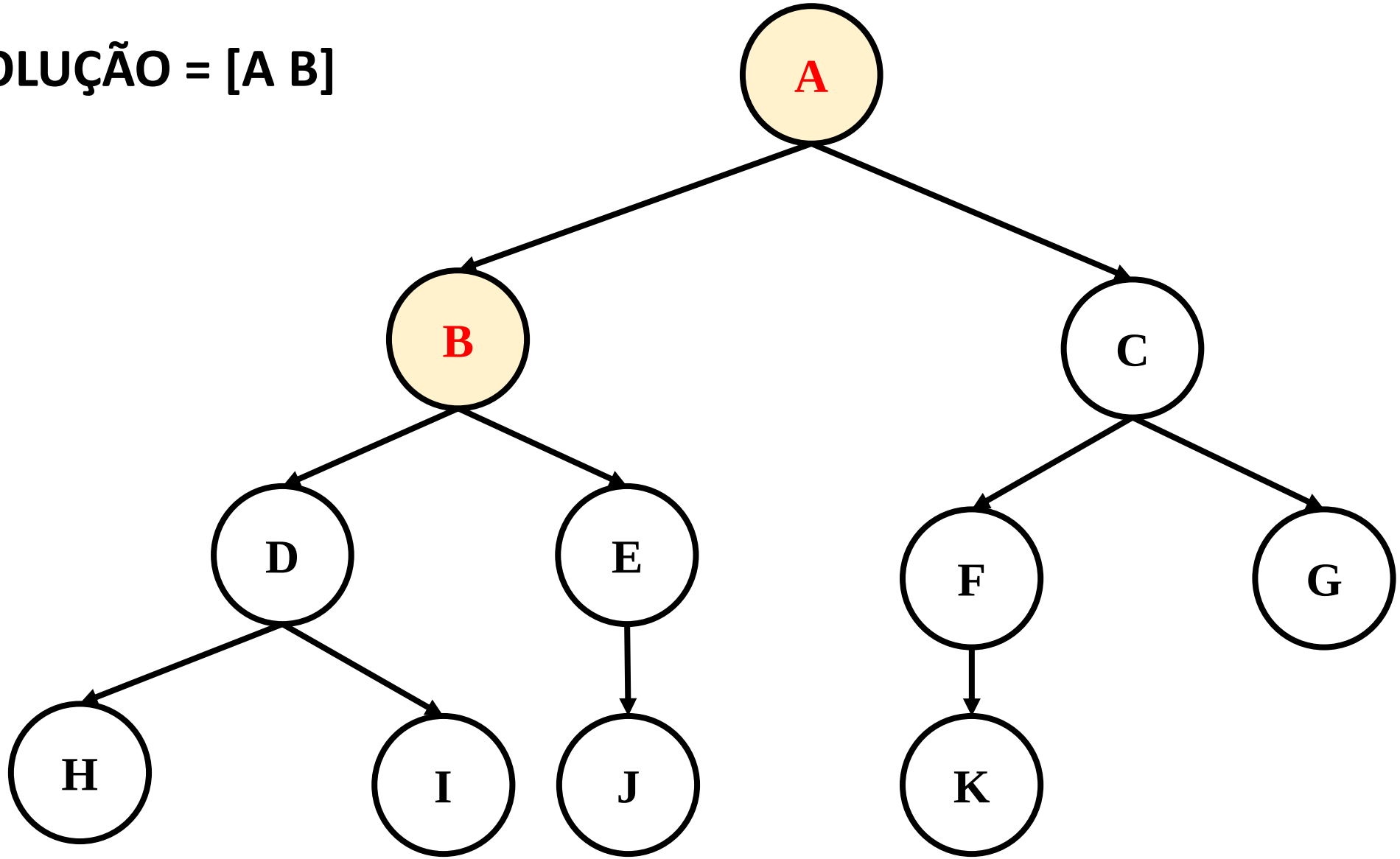
ORIGEM A OBJETIVO J LARGURA

SOLUÇÃO = [A]



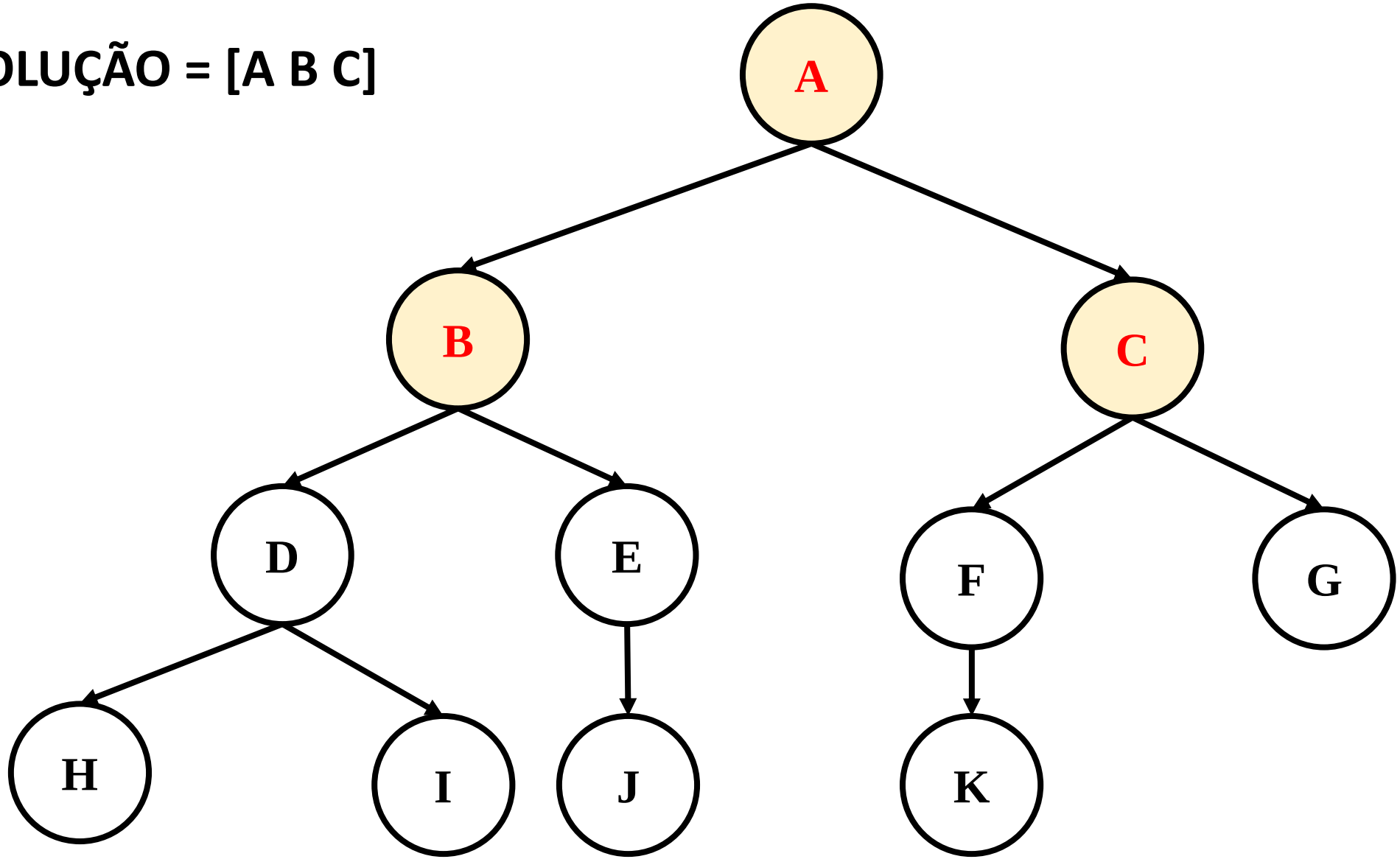
ORIGEM A OBJETIVO J LARGURA

SOLUÇÃO = [A B]



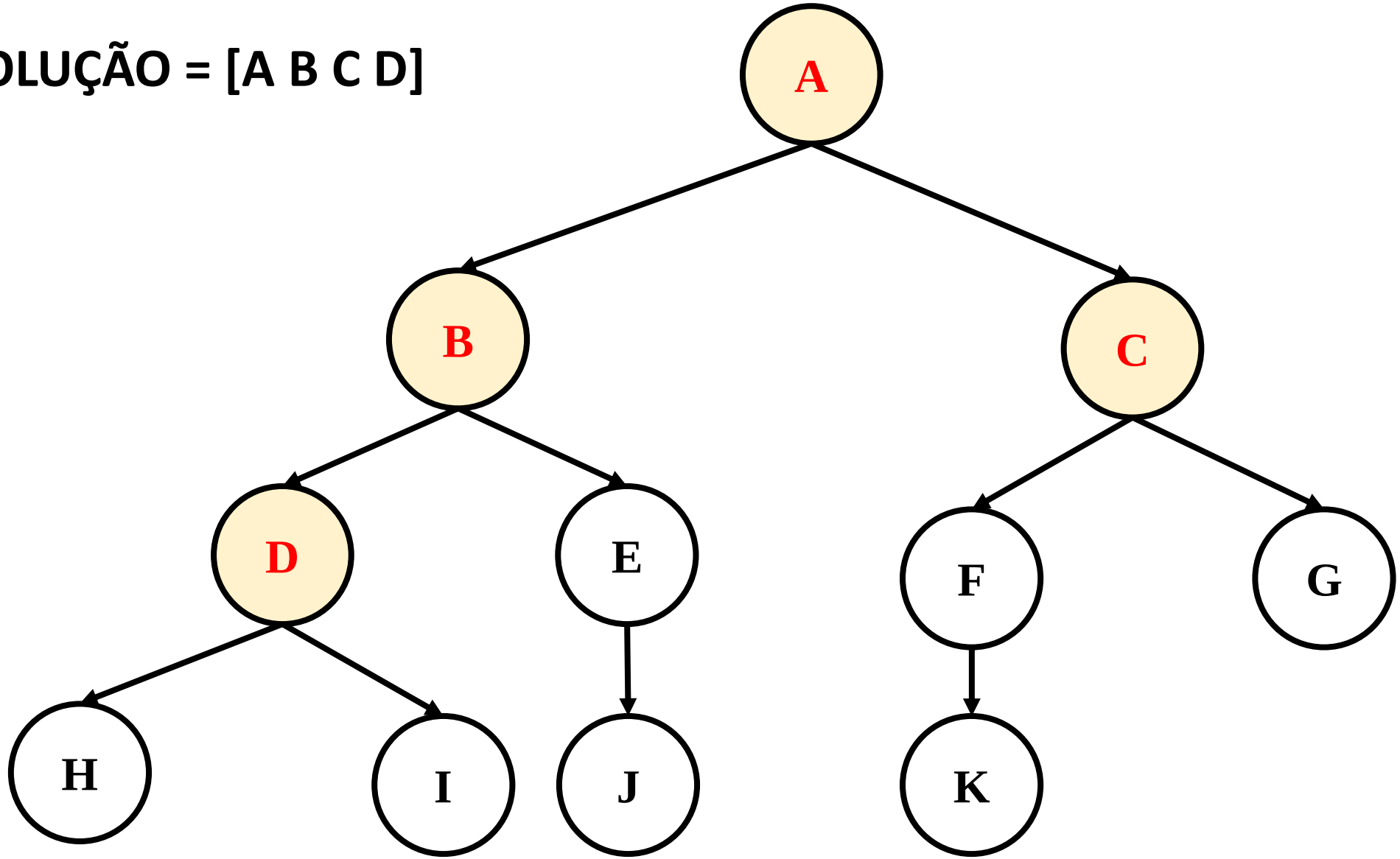
ORIGEM A OBJETIVO J LARGURA

SOLUÇÃO = [A B C]



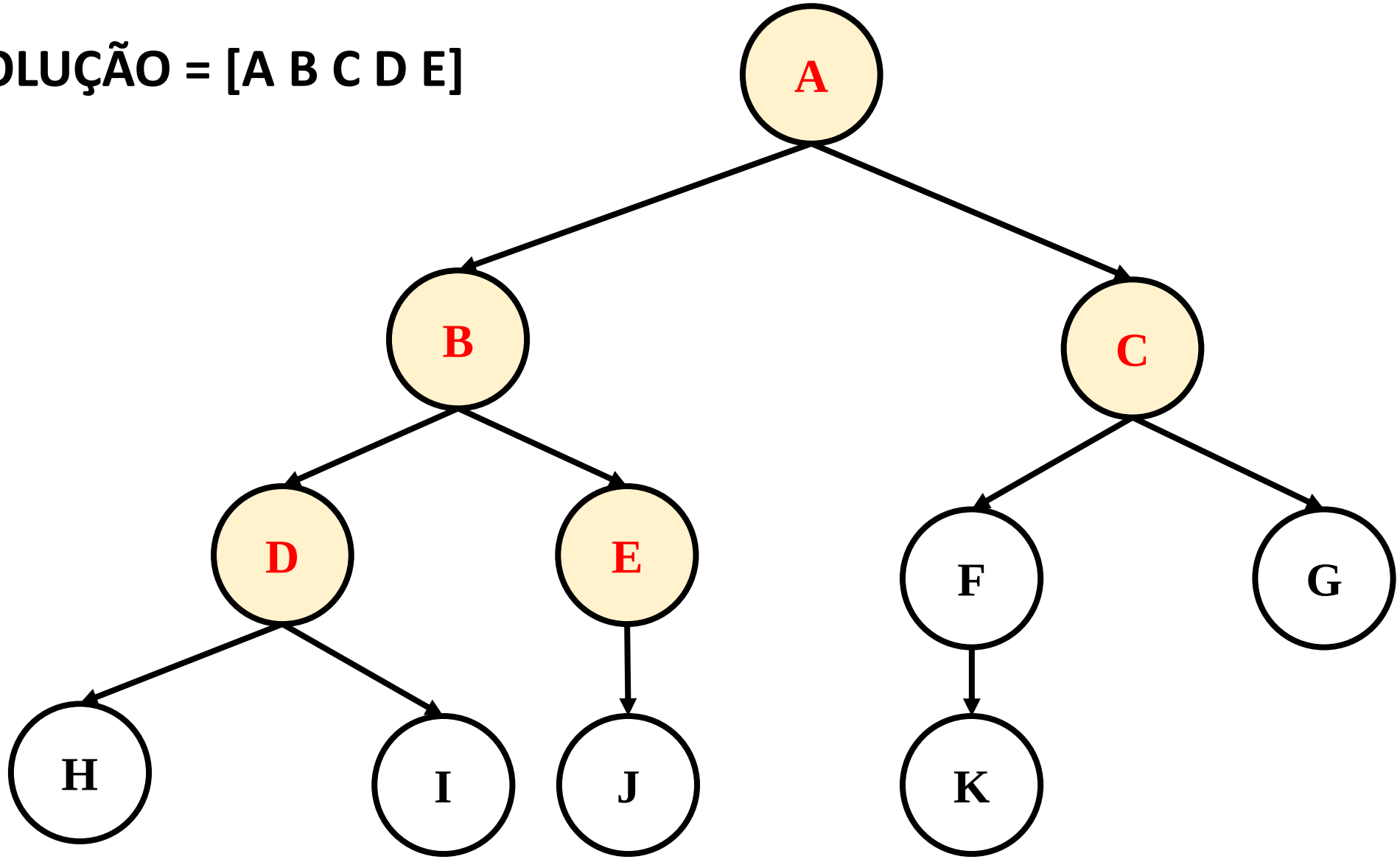
ORIGEM A OBJETIVO J LARGURA

SOLUÇÃO = [A B C D]



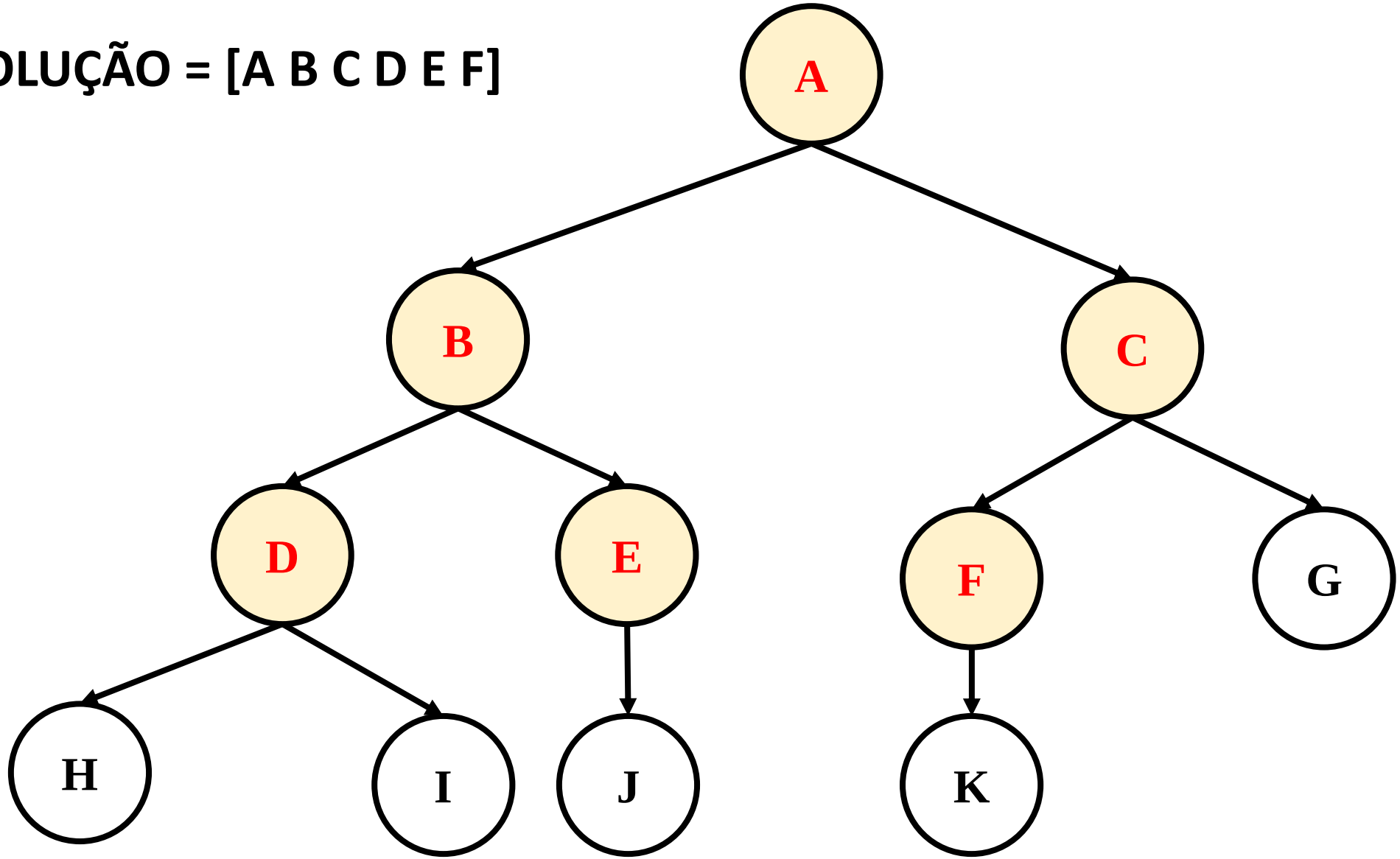
ORIGEM A OBJETIVO J LARGURA

SOLUÇÃO = [A B C D E]



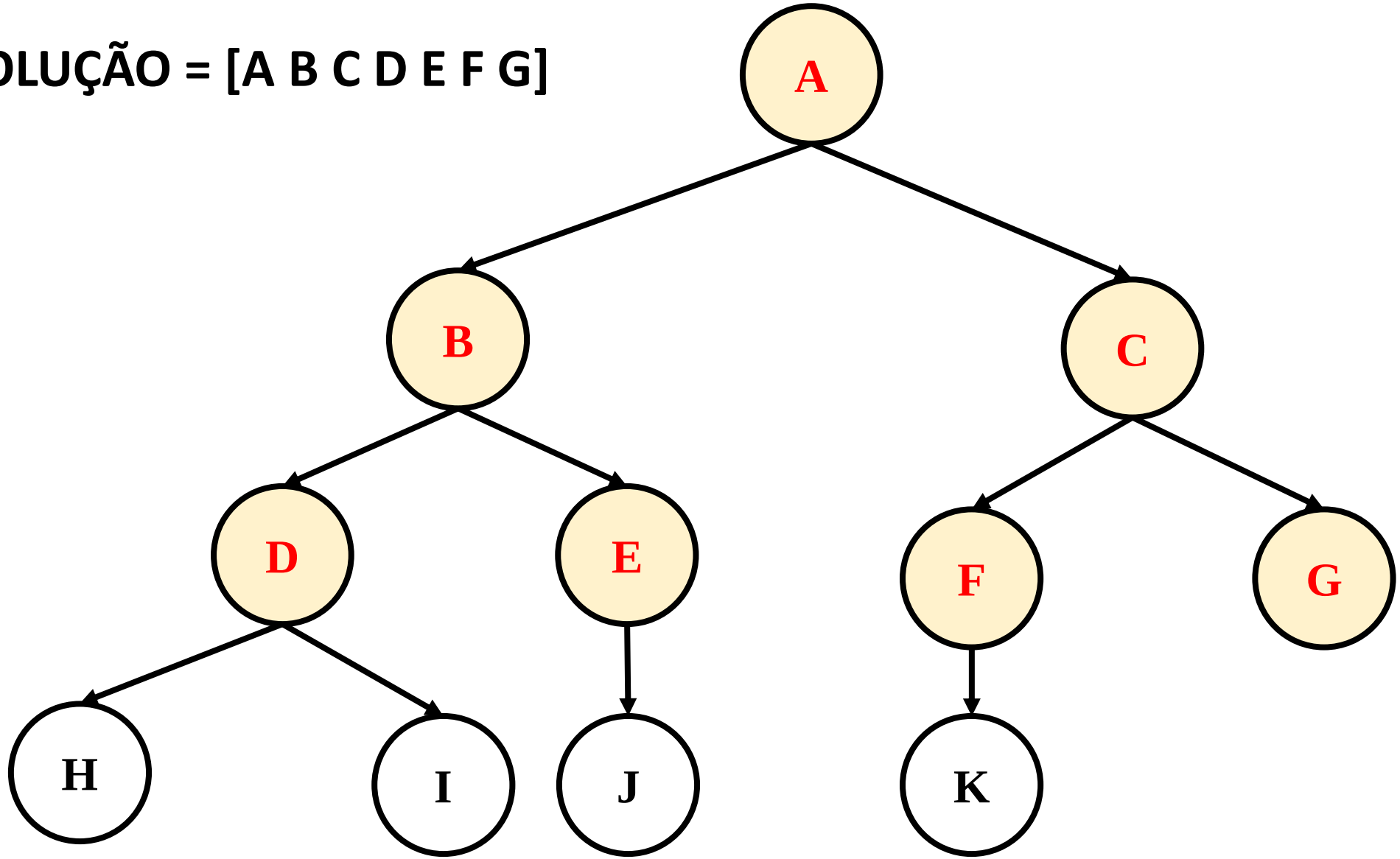
ORIGEM A OBJETIVO J LARGURA

SOLUÇÃO = [A B C D E F]



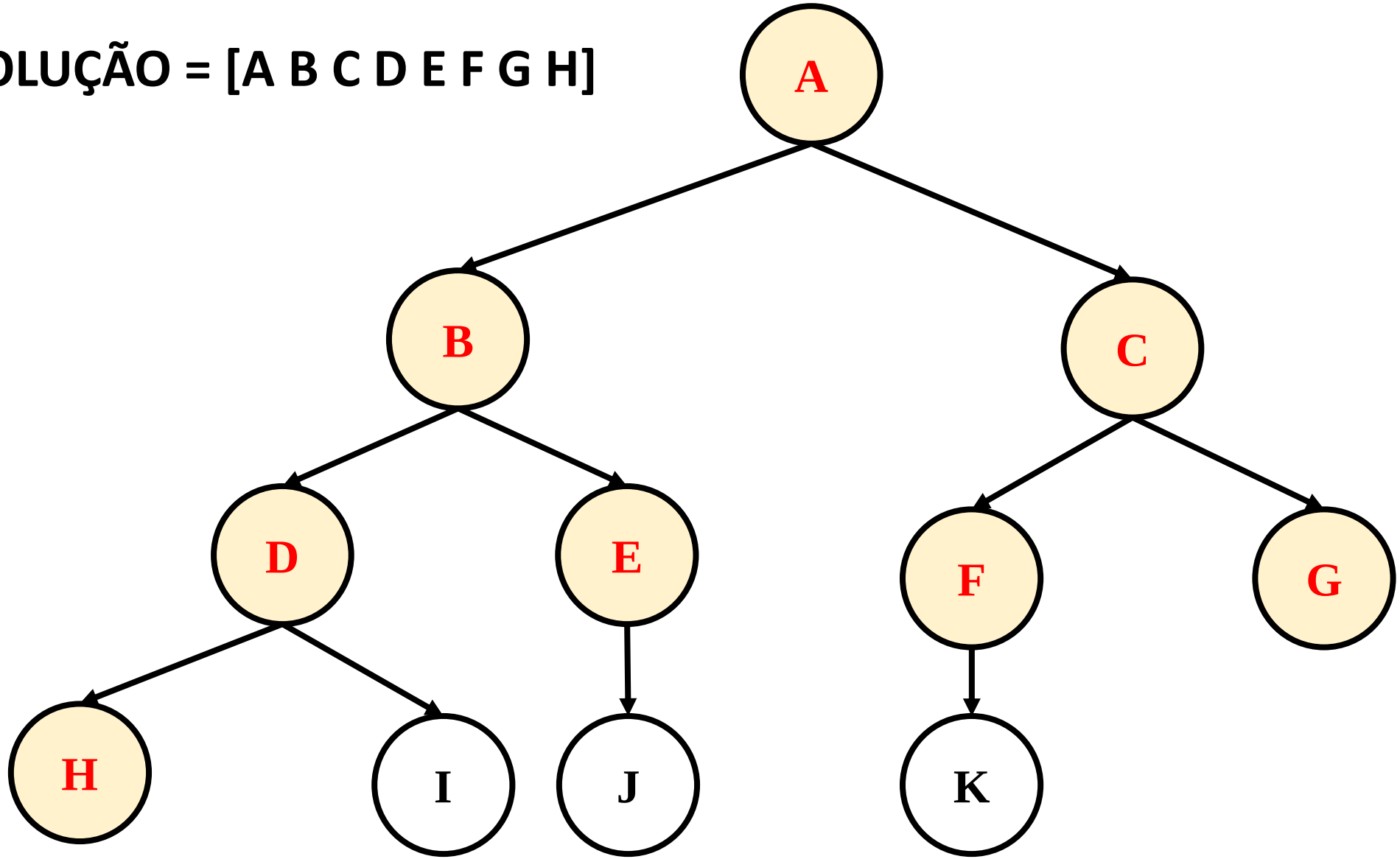
ORIGEM A OBJETIVO J LARGURA

SOLUÇÃO = [A B C D E F G]



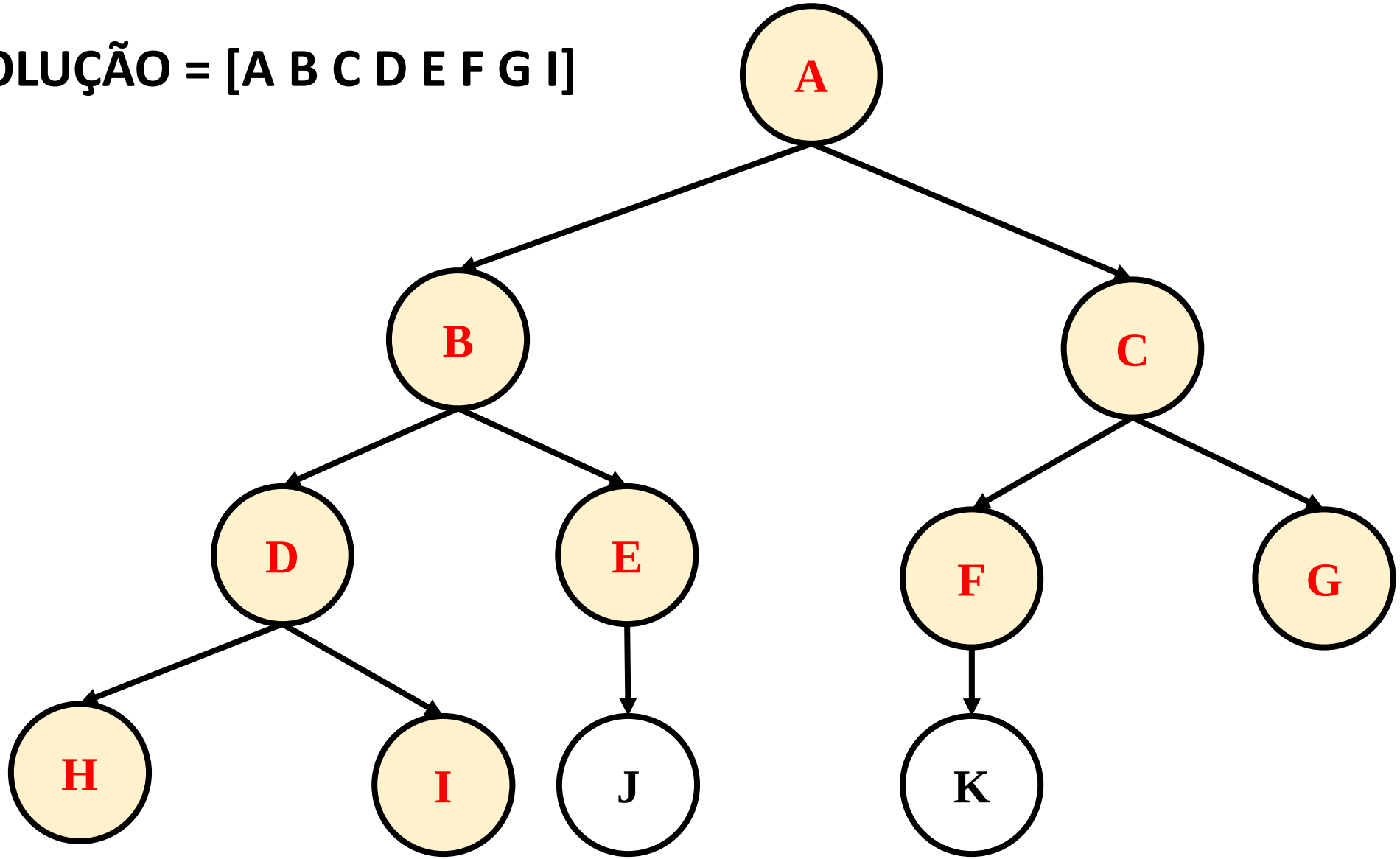
ORIGEM A OBJETIVO J LARGURA

SOLUÇÃO = [A B C D E F G H]



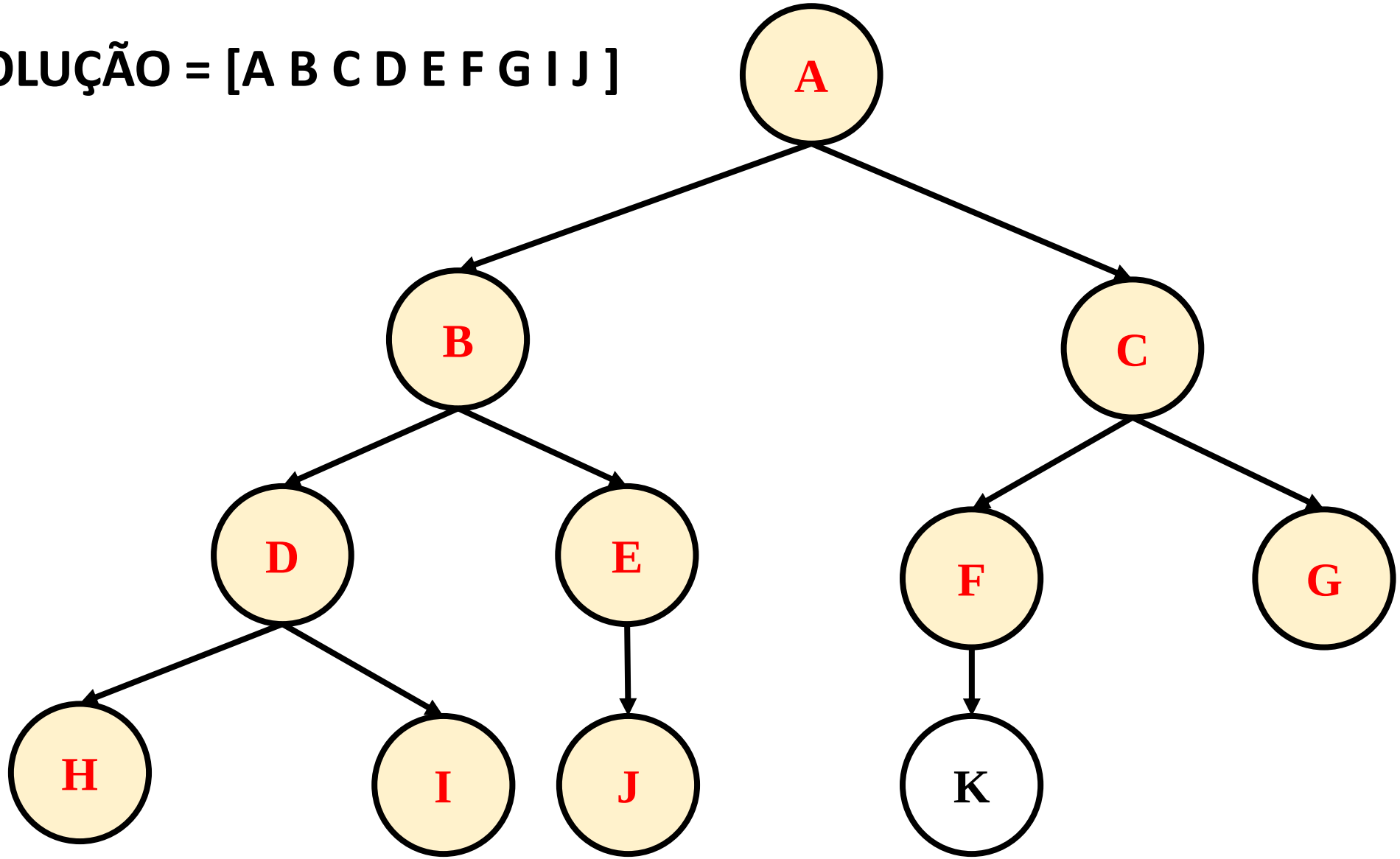
ORIGEM A OBJETIVO J LARGURA

SOLUÇÃO = [A B C D E F G I]



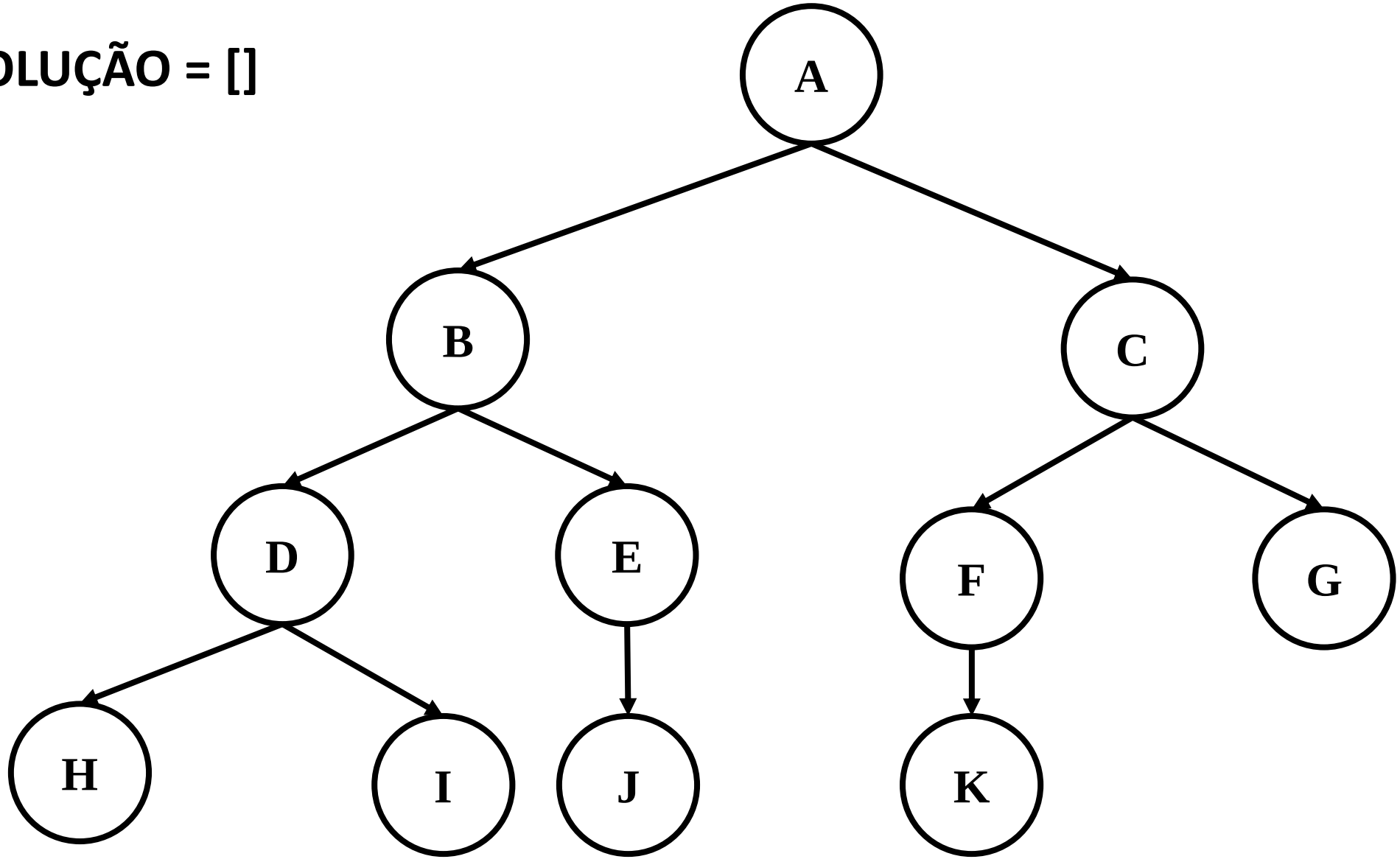
ORIGEM A OBJETIVO J LARGURA

SOLUÇÃO = [A B C D E F G I J]



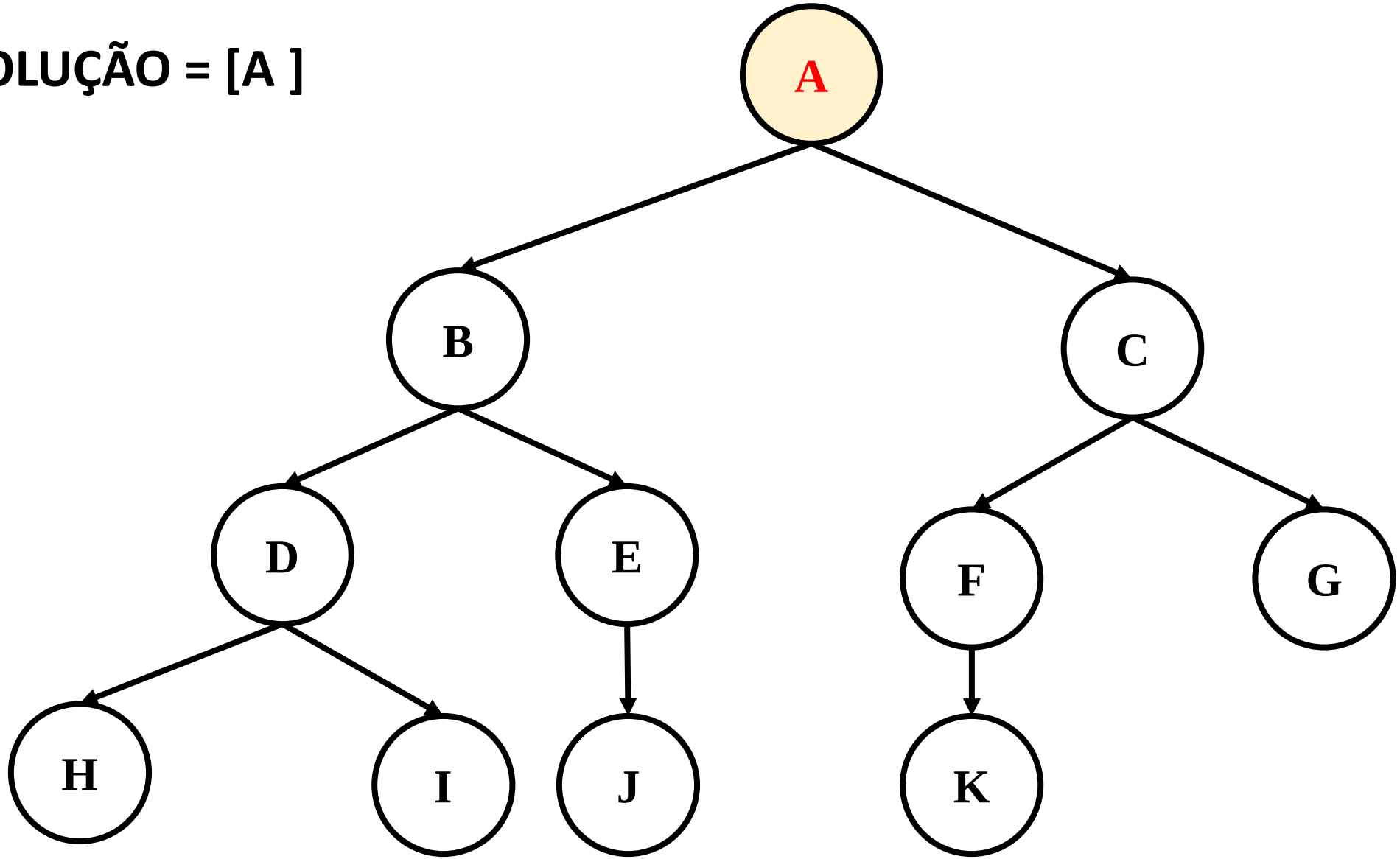
ORIGEM A OBJETIVO K PROFUNDIDADE

SOLUÇÃO = []



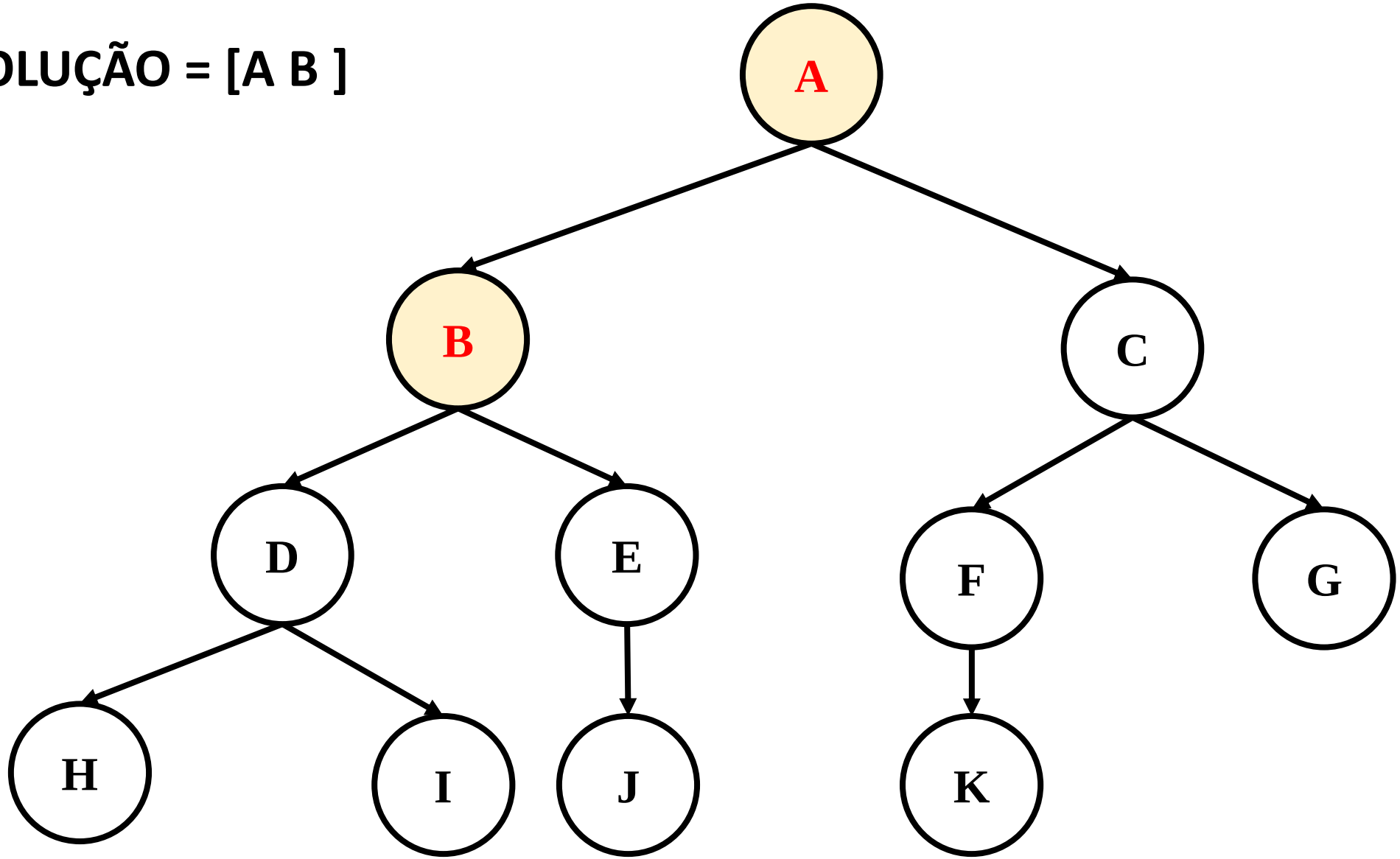
ORIGEM A OBJETIVO K PROFUNDIDADE

SOLUÇÃO = [A]



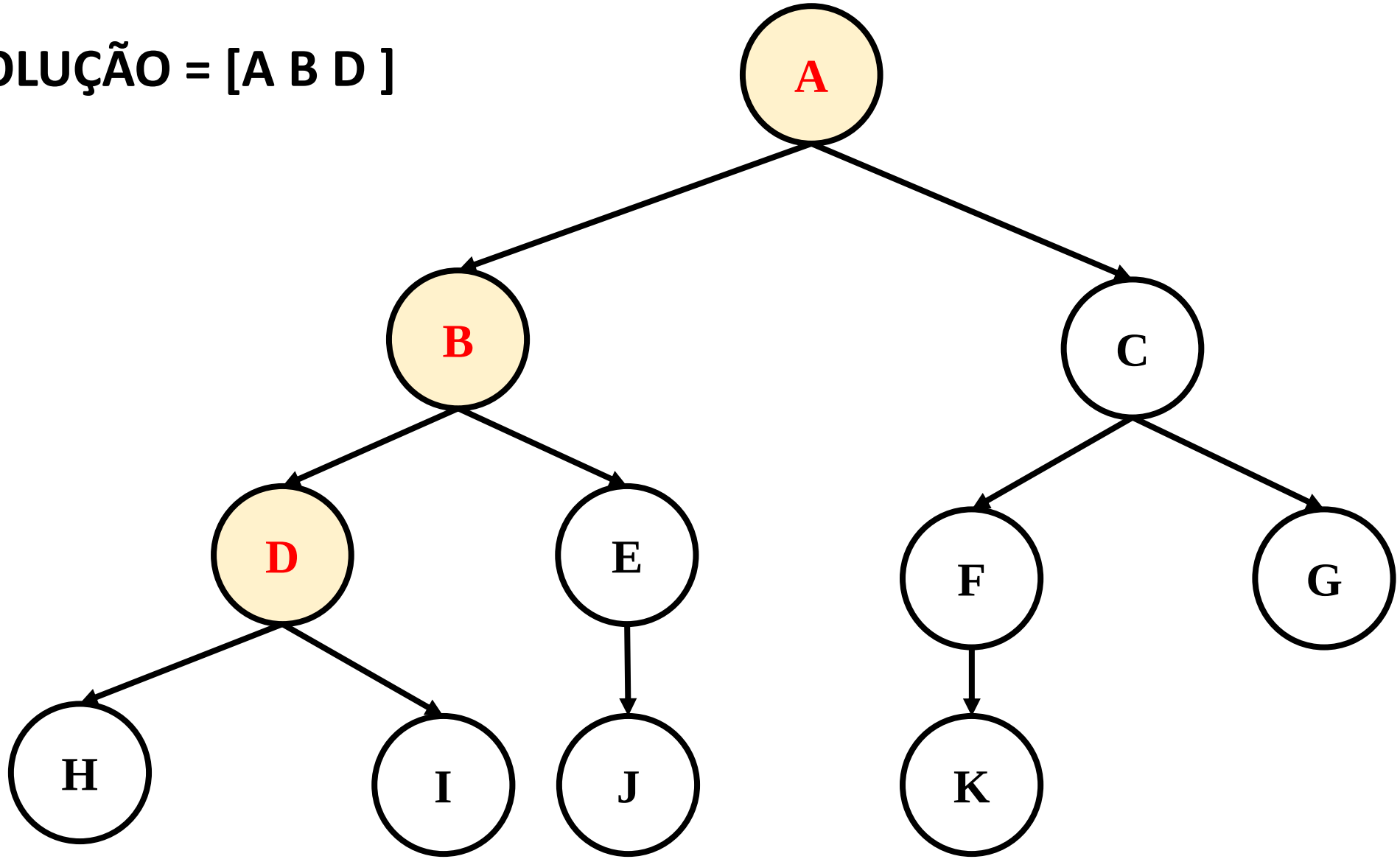
ORIGEM A OBJETIVO K PROFUNDIDADE

SOLUÇÃO = [A B]



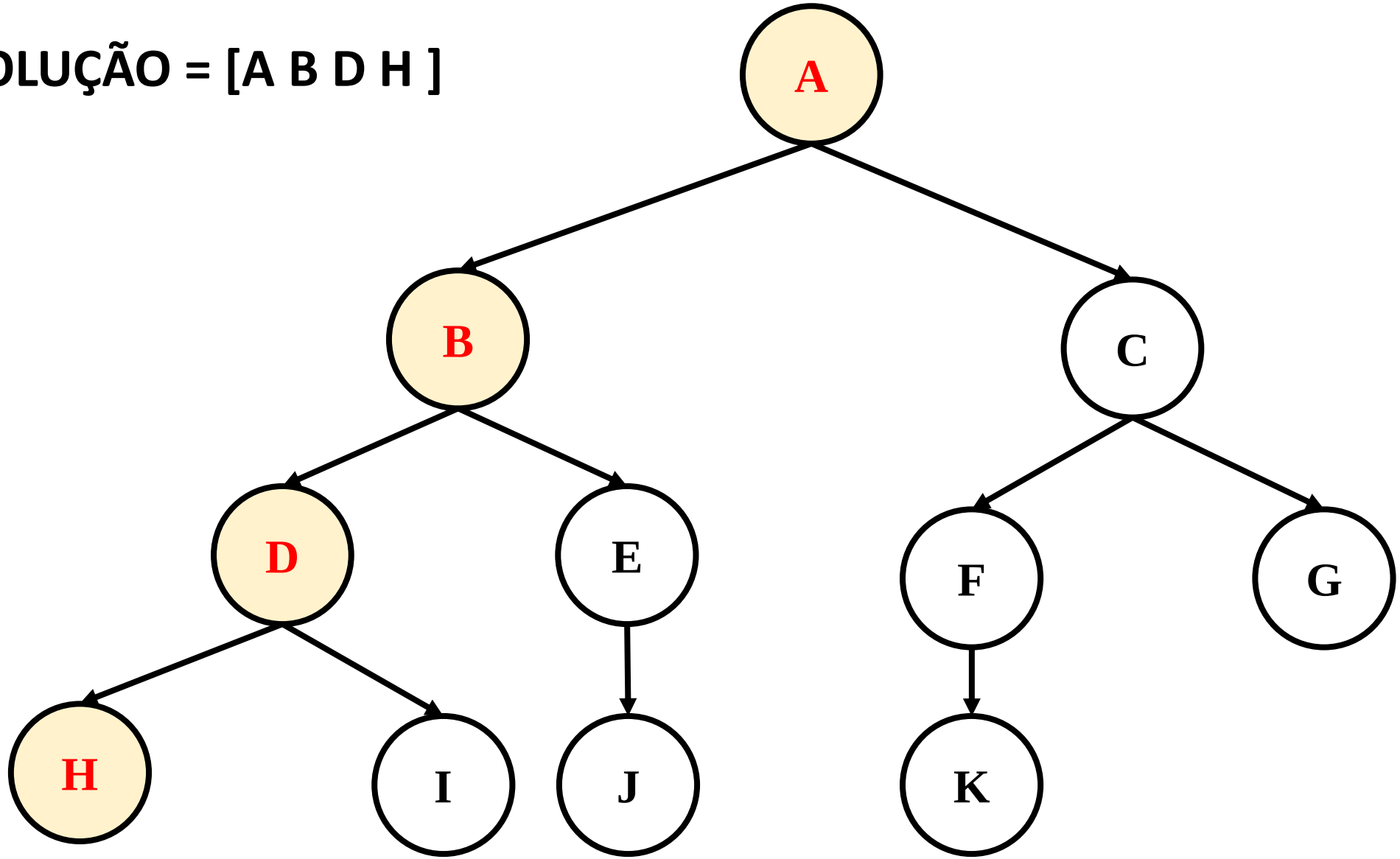
ORIGEM A OBJETIVO K PROFUNDIDADE

SOLUÇÃO = [A B D]



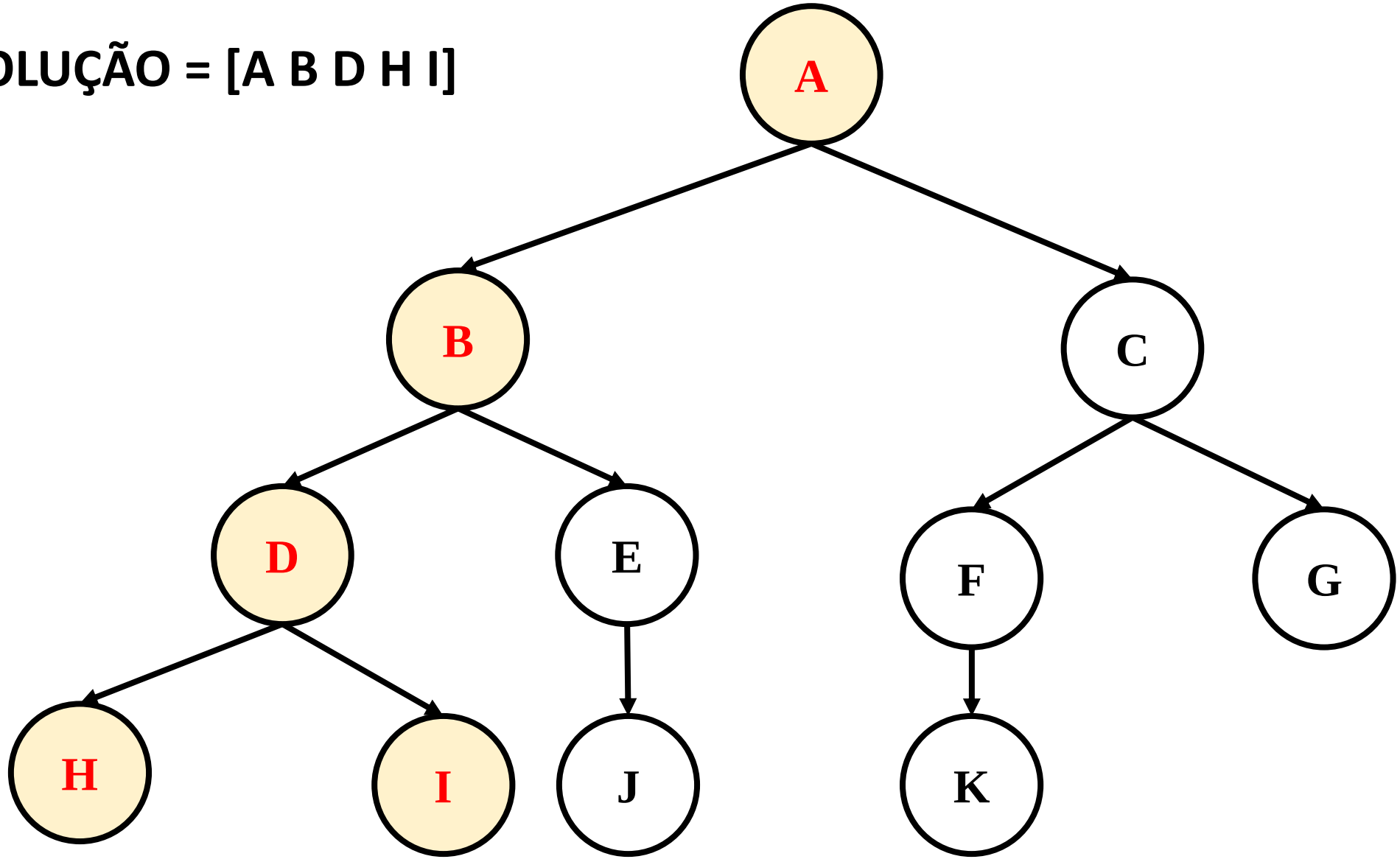
ORIGEM A OBJETIVO K PROFUNDIDADE

SOLUÇÃO = [A B D H]



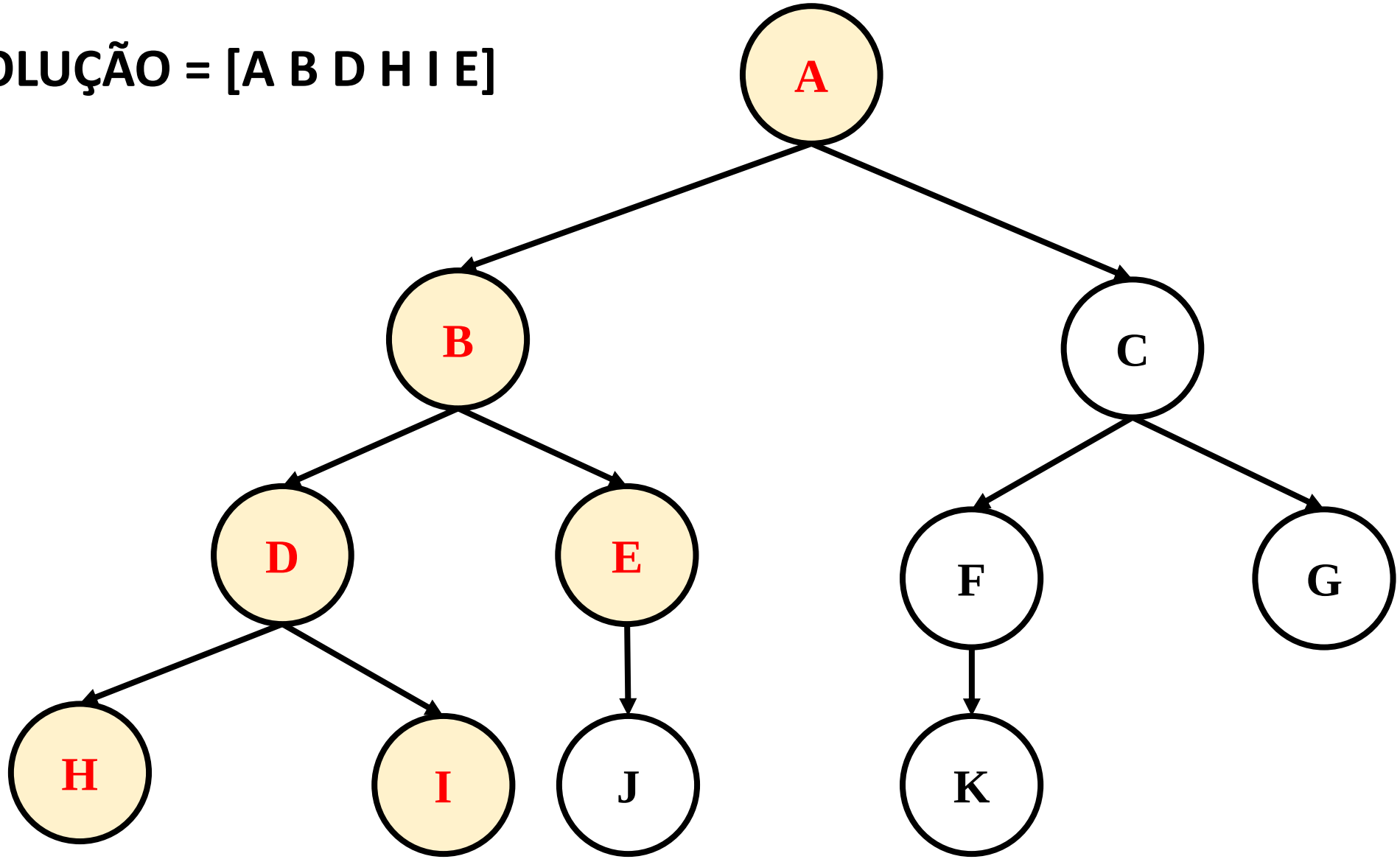
ORIGEM A OBJETIVO K PROFUNDIDADE

SOLUÇÃO = [A B D H I]



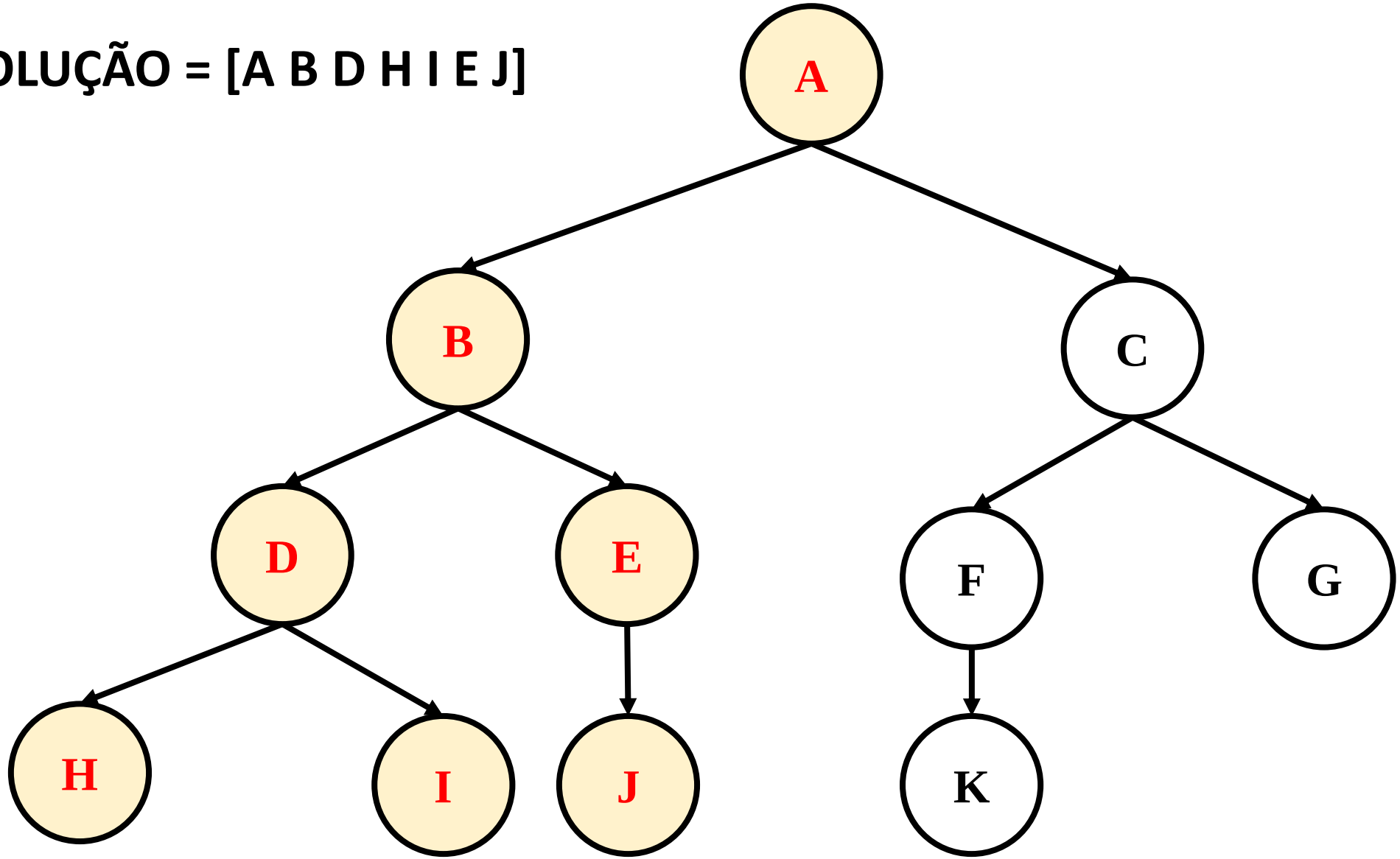
ORIGEM A OBJETIVO K PROFUNDIDADE

SOLUÇÃO = [A B D H I E]



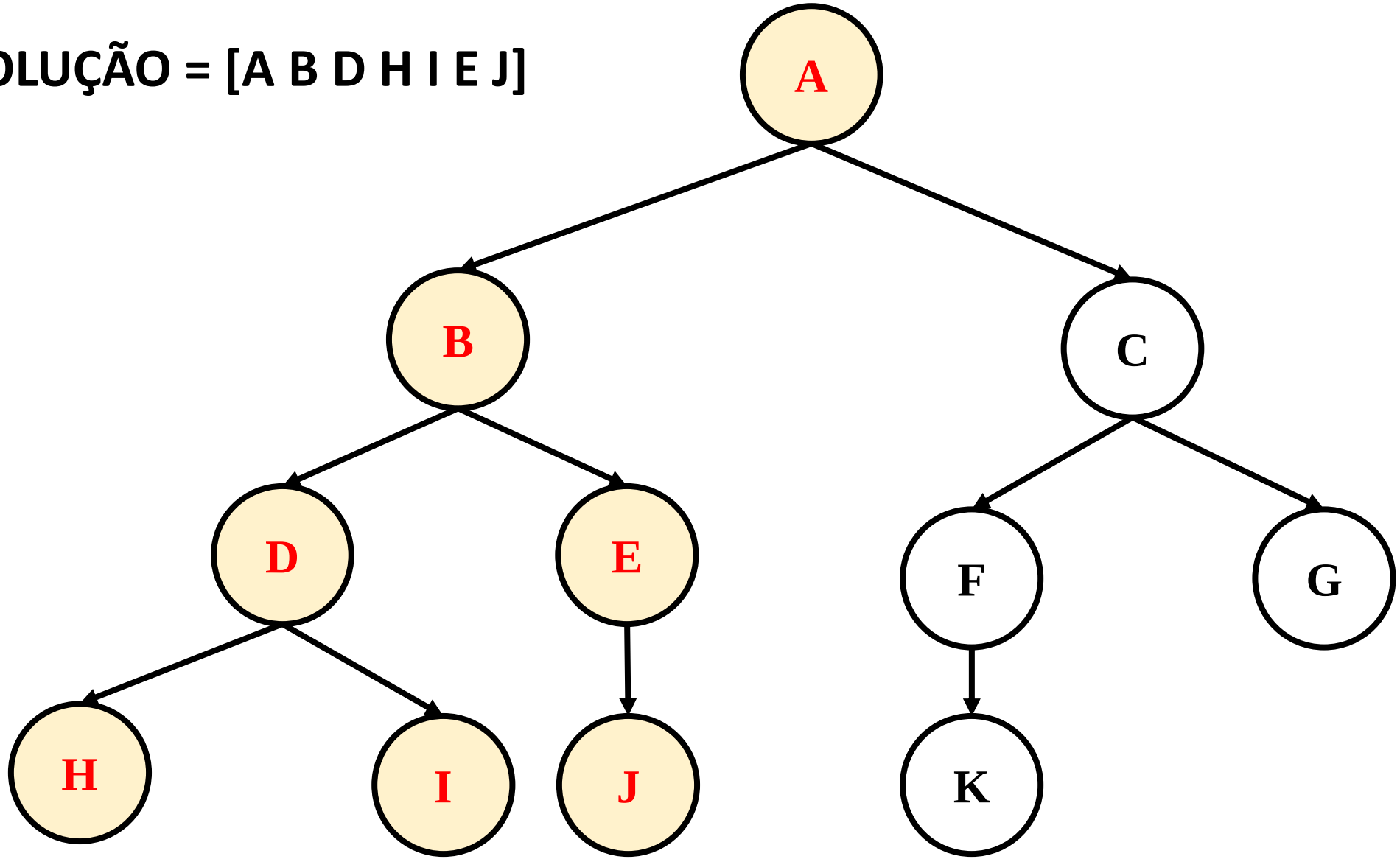
ORIGEM A OBJETIVO K PROFUNDIDADE

SOLUÇÃO = [A B D H I E J]



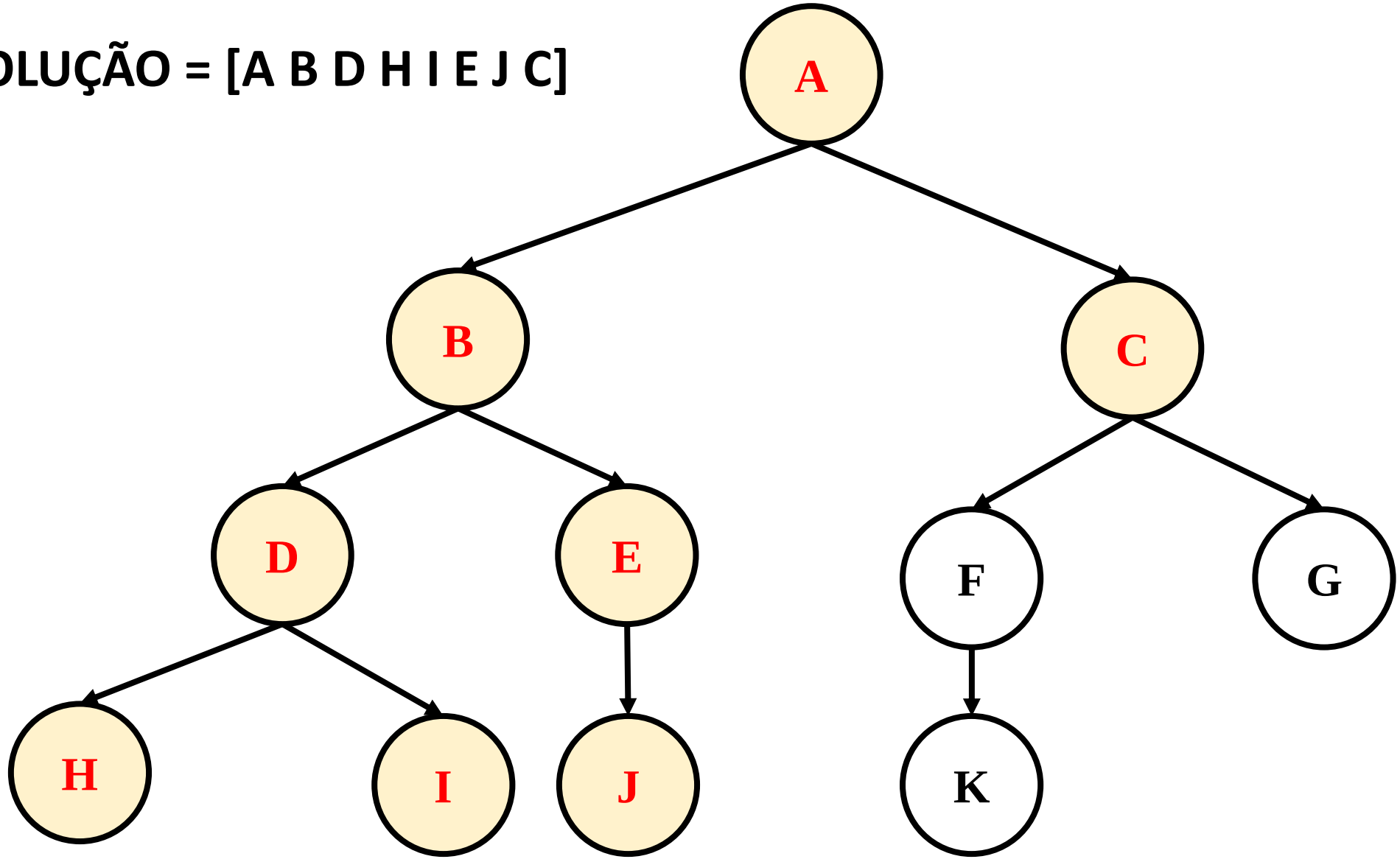
ORIGEM A OBJETIVO K PROFUNDIDADE

SOLUÇÃO = [A B D H I E J]



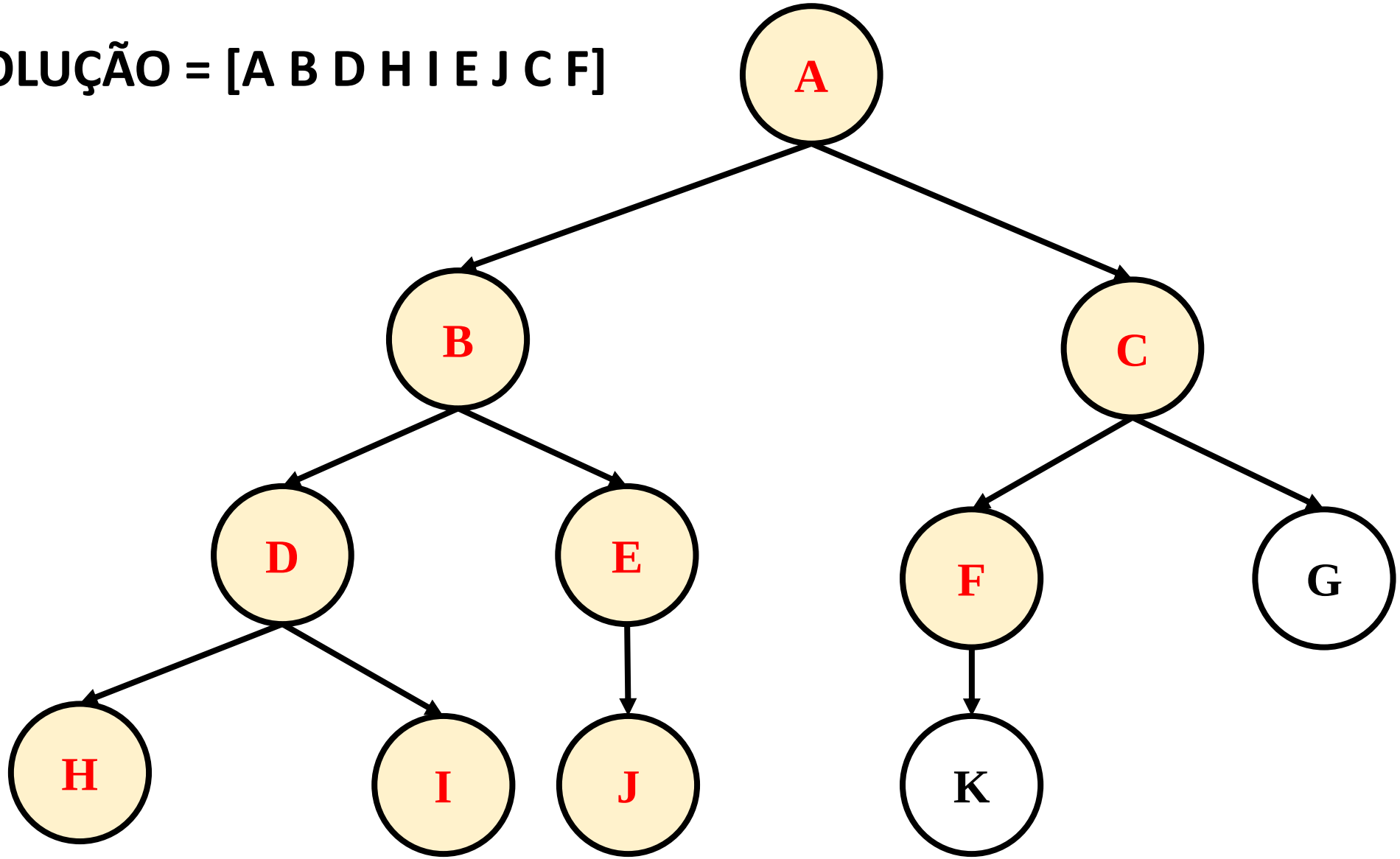
ORIGEM A OBJETIVO K PROFUNDIDADE

SOLUÇÃO = [A B D H I E J C]



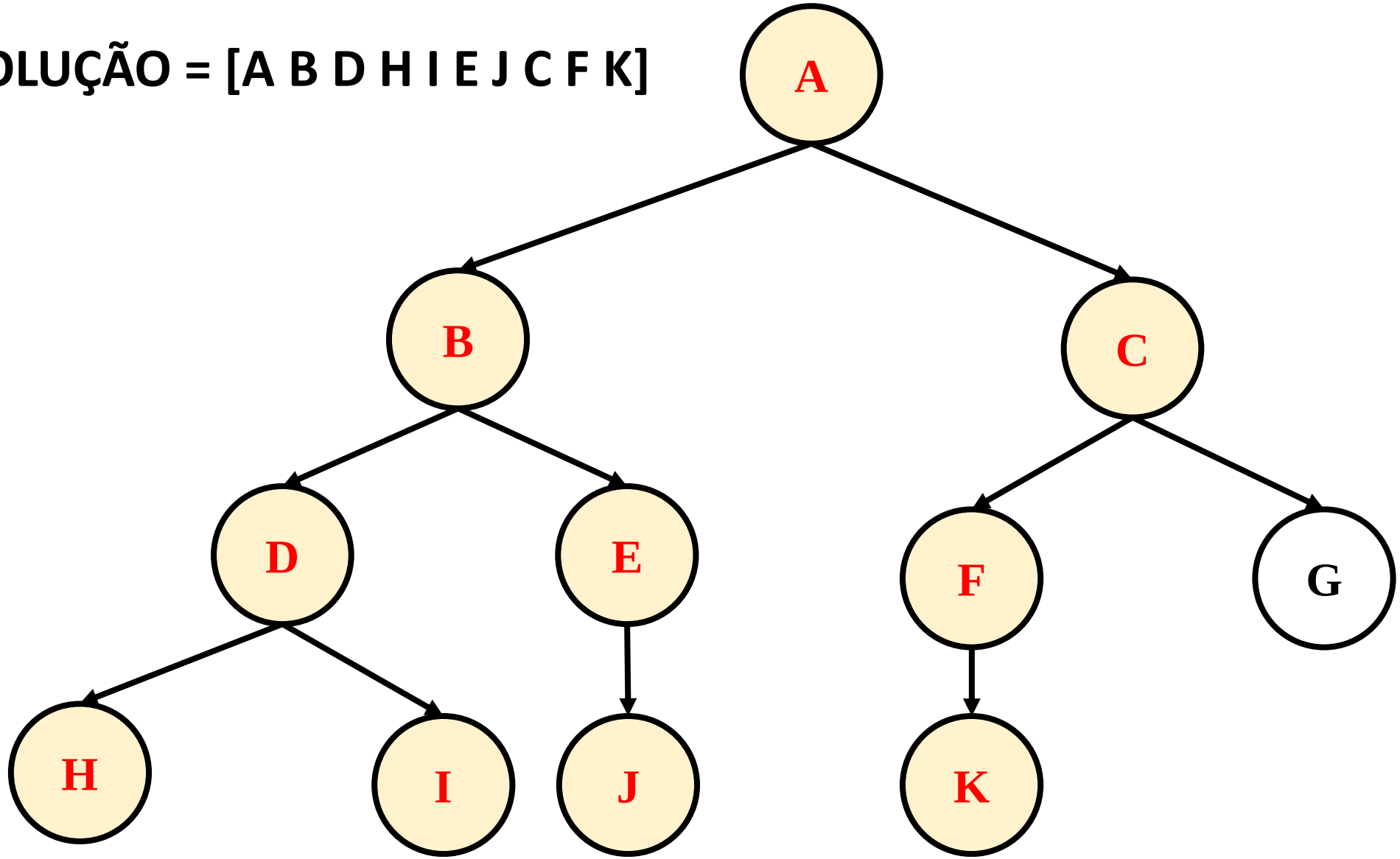
ORIGEM A OBJETIVO K PROFUNDIDADE

SOLUÇÃO = [A B D H I E J C F]



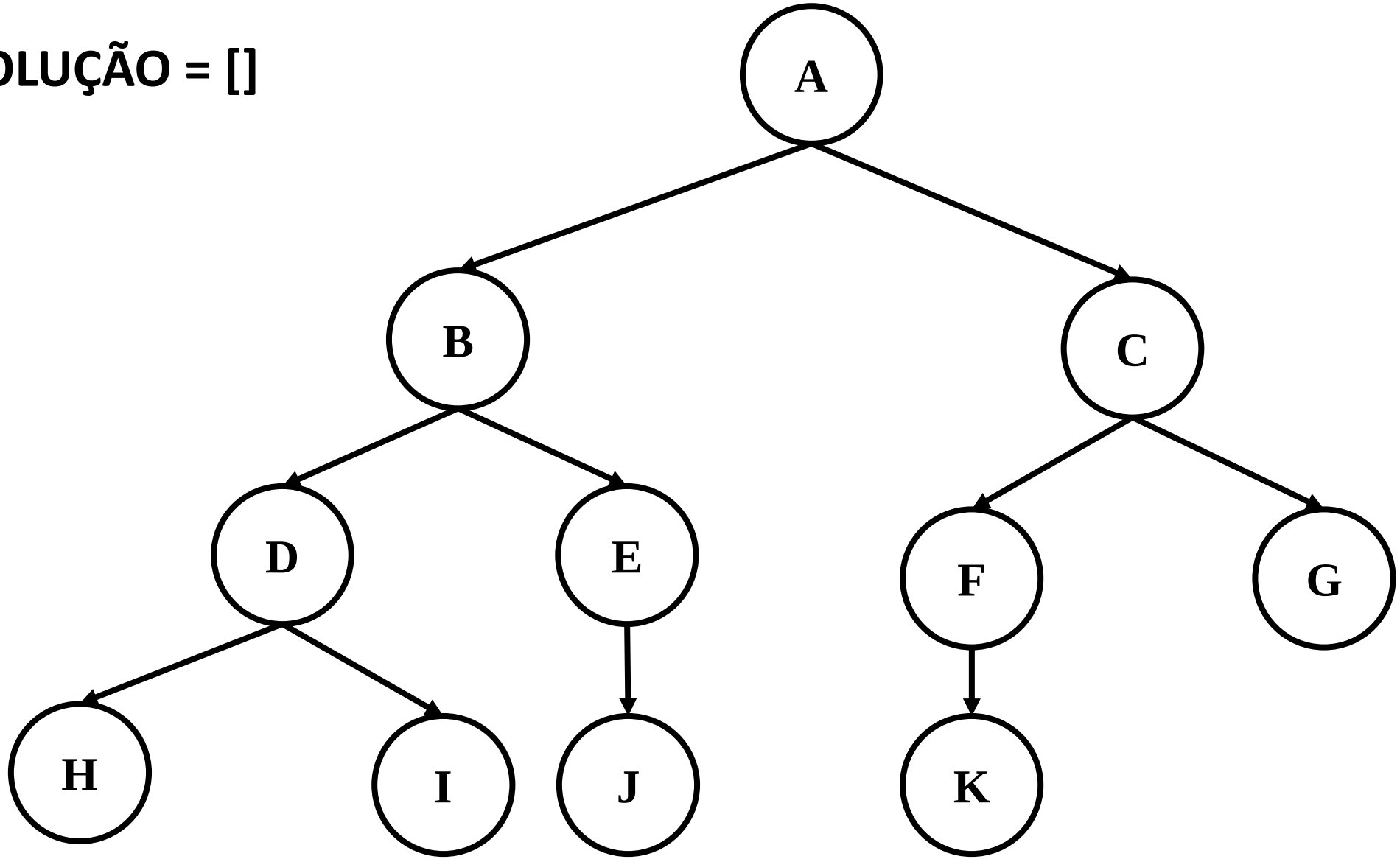
ORIGEM A OBJETIVO K PROFUNDIDADE

SOLUÇÃO = [A B D H I E J C F K]



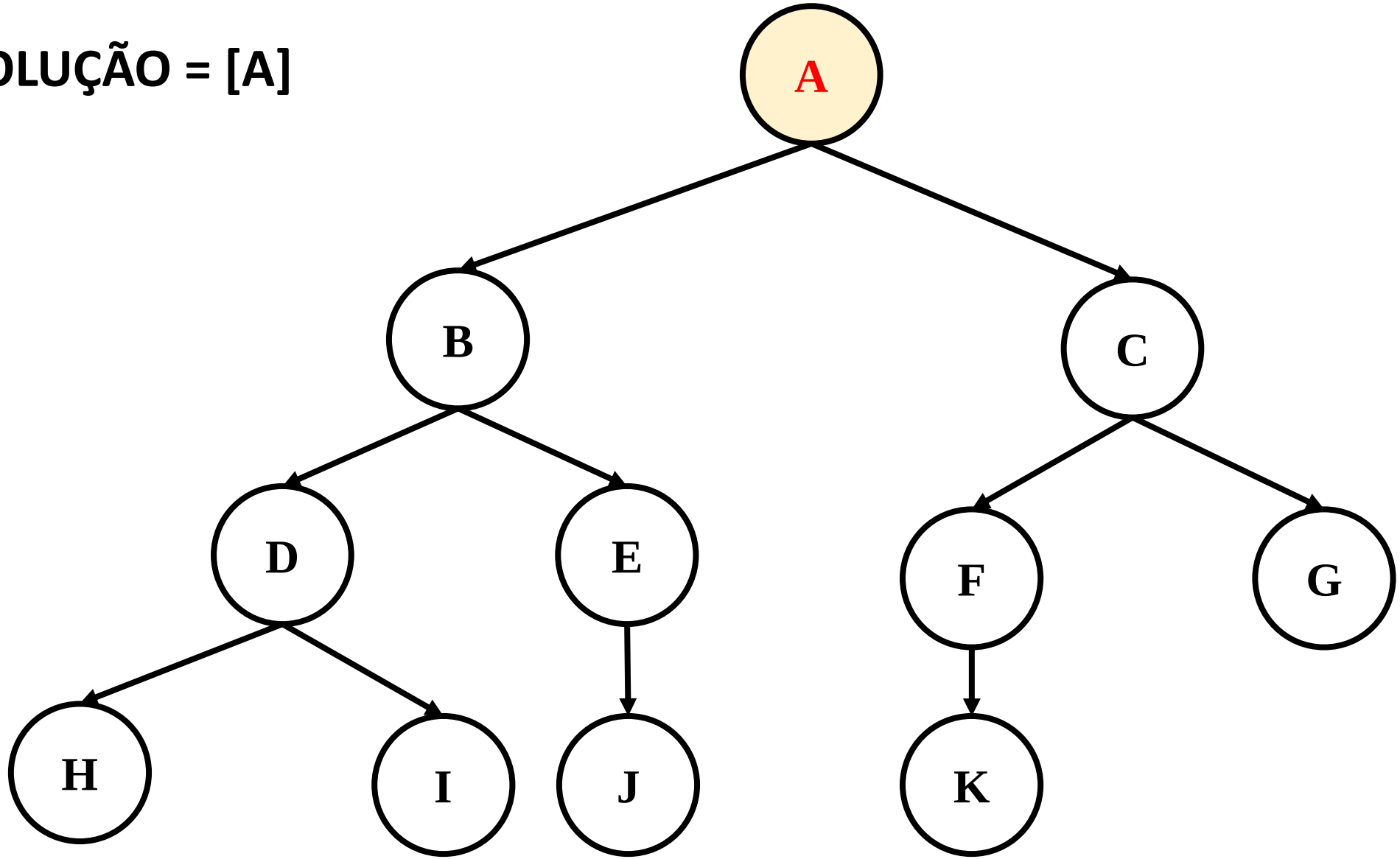
ORIGEM A OBJETIVO K LARGURA

SOLUÇÃO = []



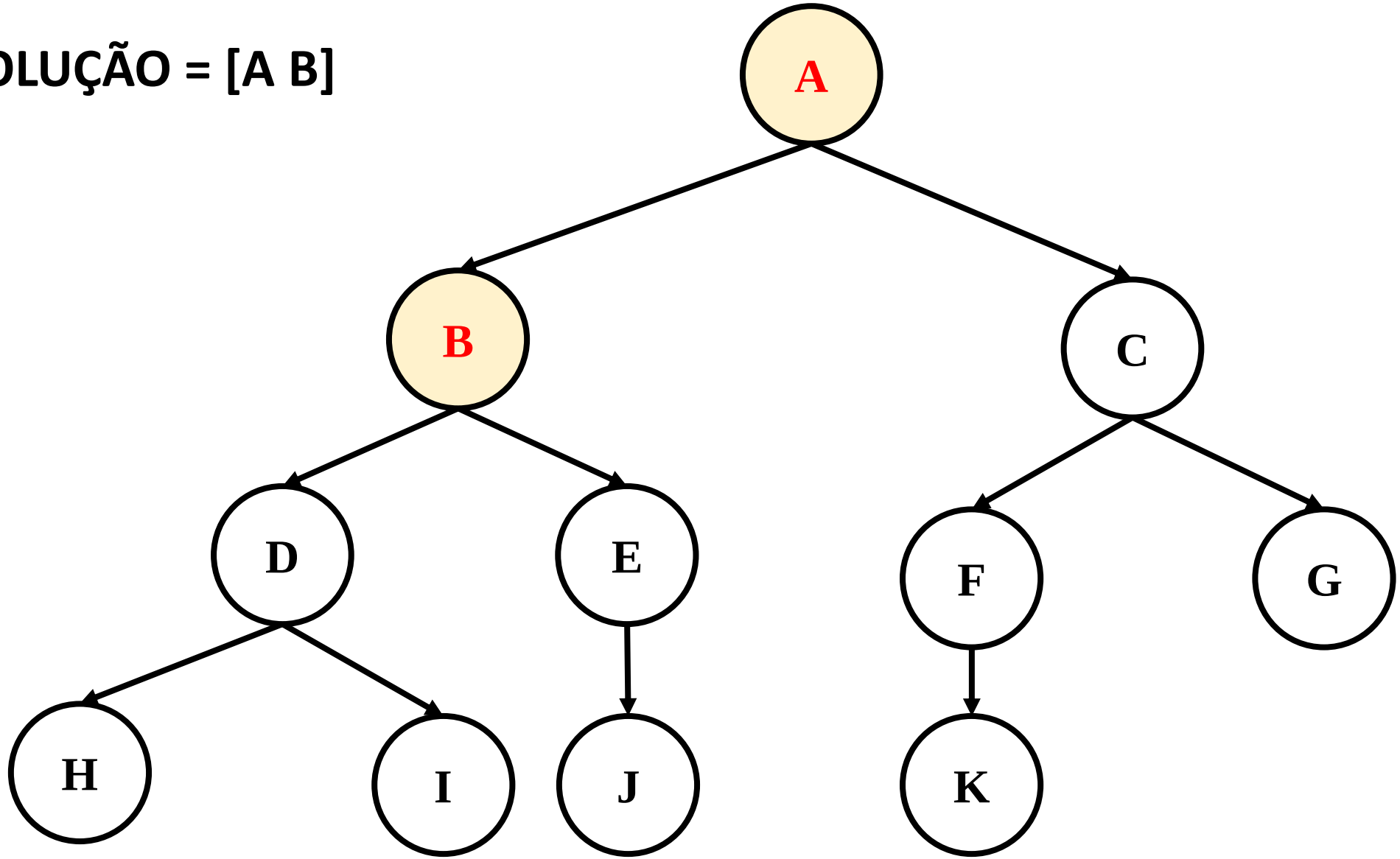
ORIGEM A OBJETIVO K LARGURA

SOLUÇÃO = [A]



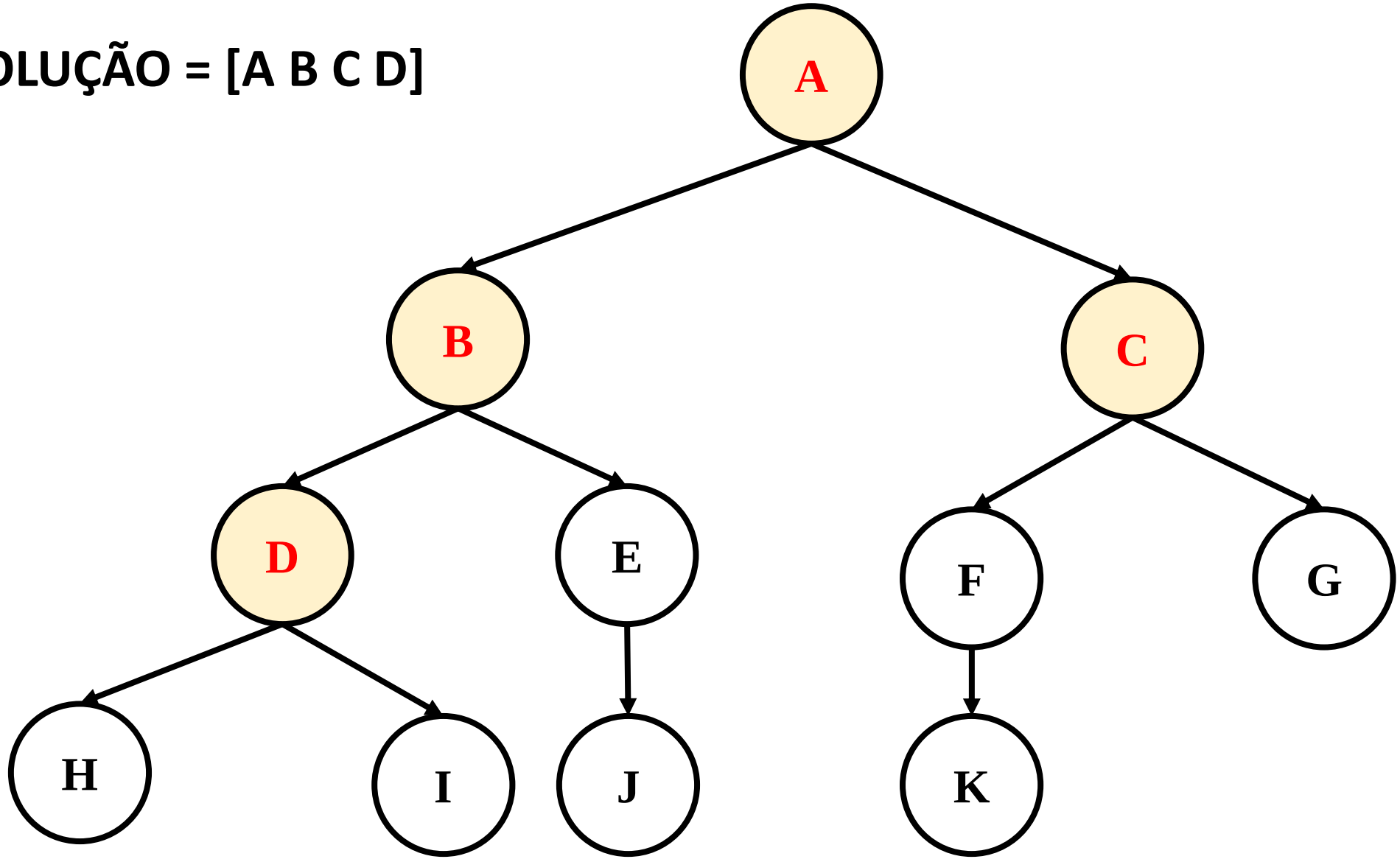
ORIGEM A OBJETIVO K LARGURA

SOLUÇÃO = [A B]



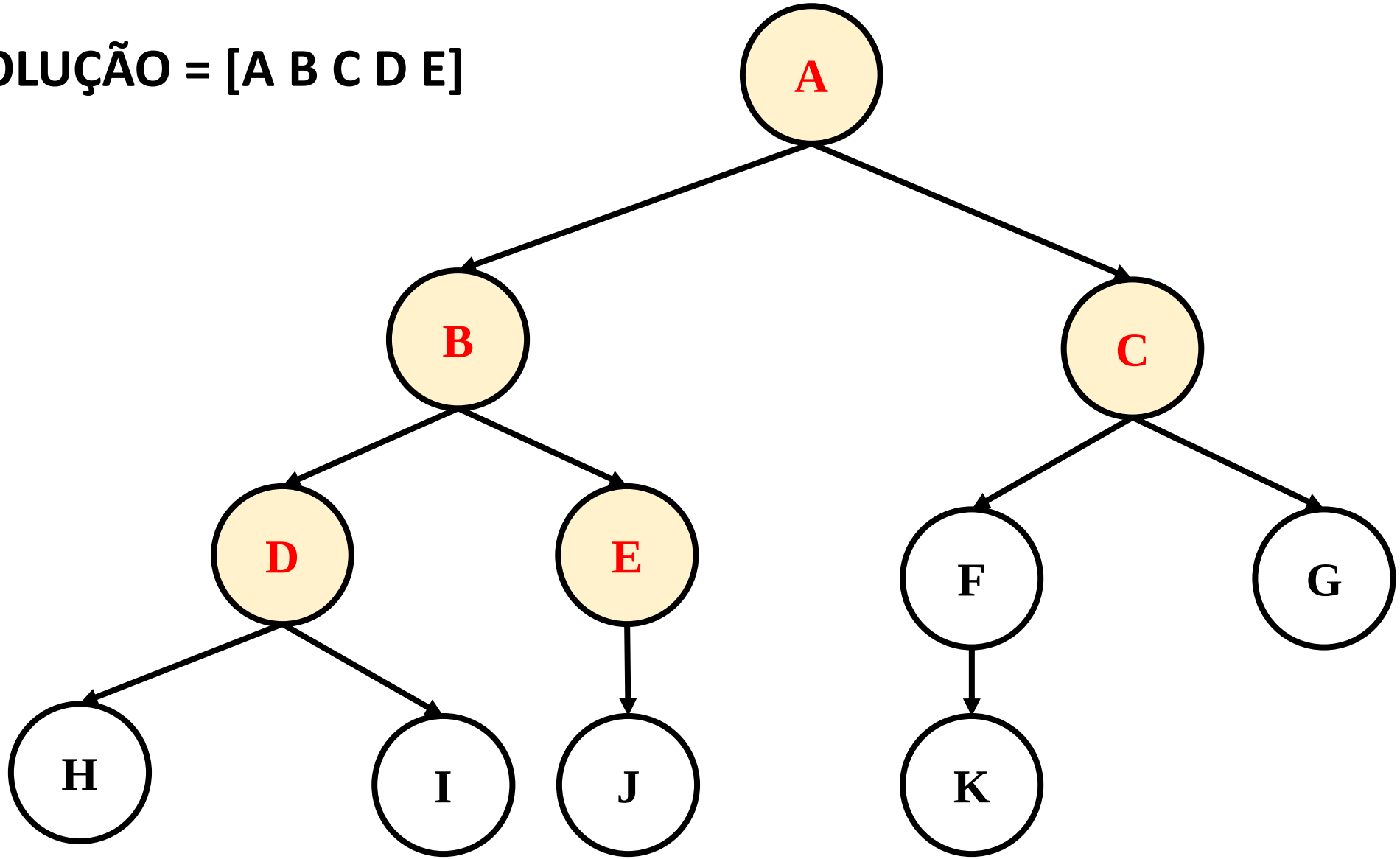
ORIGEM A OBJETIVO K LARGURA

SOLUÇÃO = [A B C D]



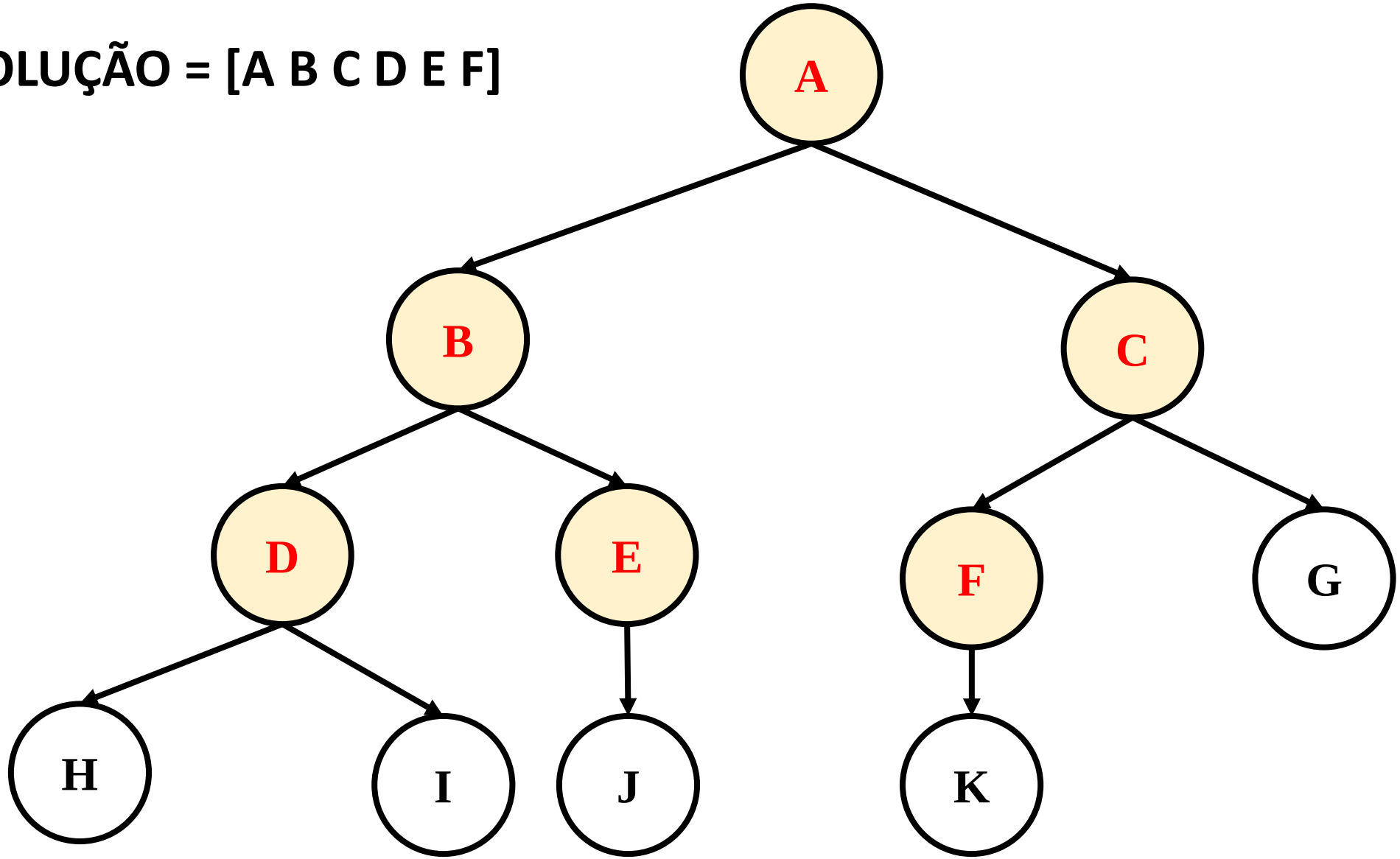
ORIGEM A OBJETIVO K LARGURA

SOLUÇÃO = [A B C D E]



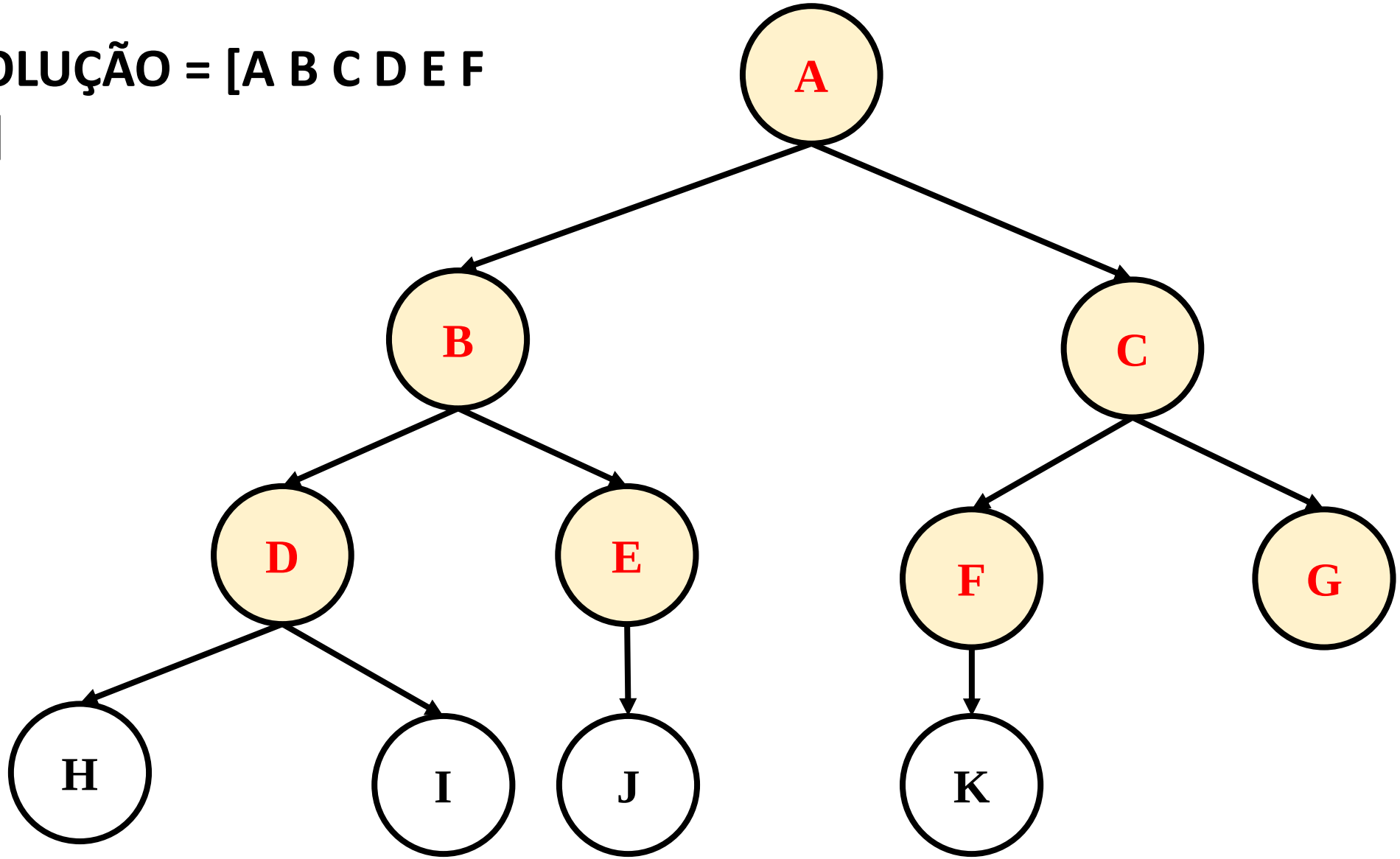
ORIGEM A OBJETIVO K LARGURA

SOLUÇÃO = [A B C D E F]



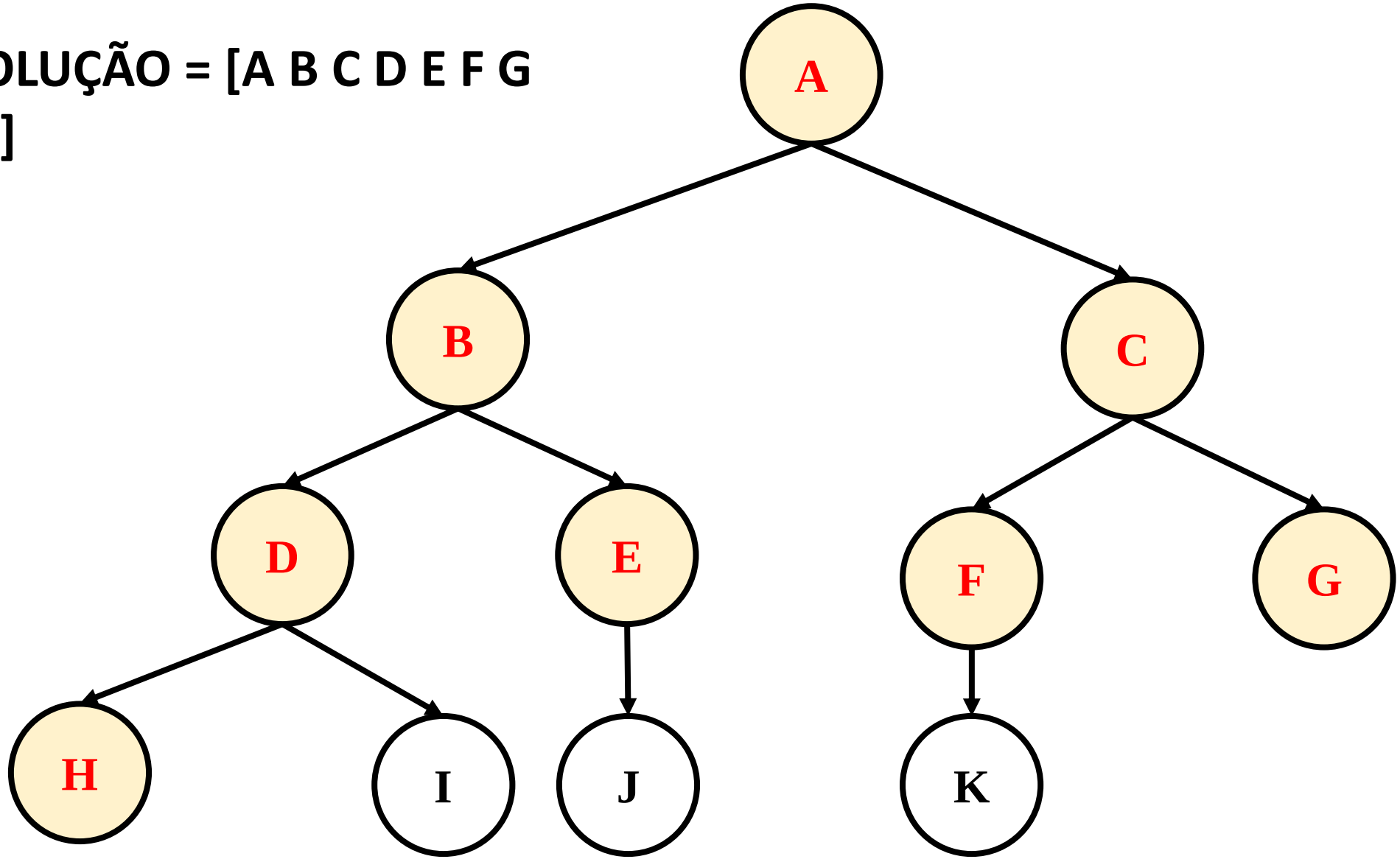
ORIGEM A OBJETIVO K LARGURA

SOLUÇÃO = [A B C D E F G]



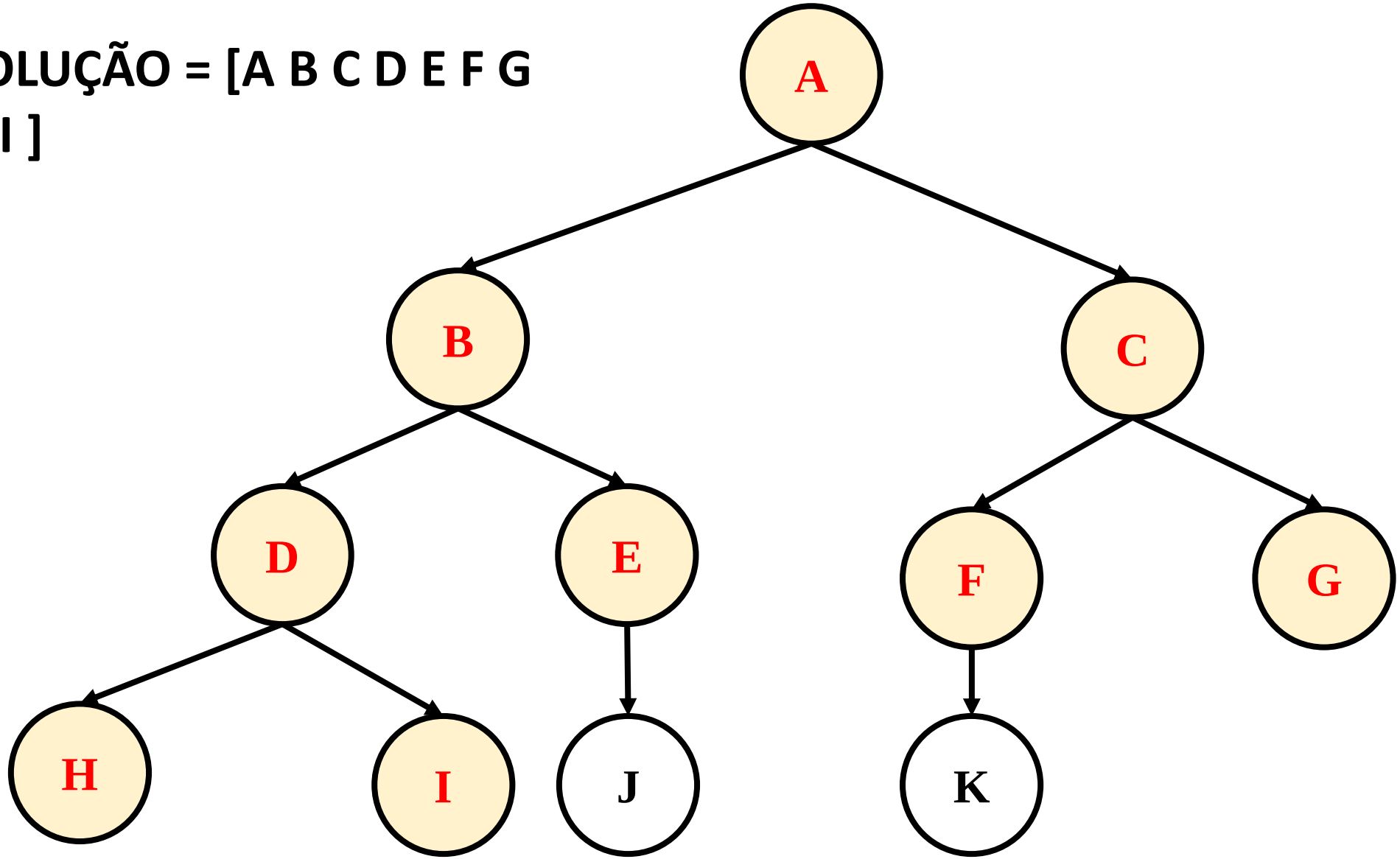
ORIGEM A OBJETIVO K LARGURA

SOLUÇÃO = [A B C D E F G H]



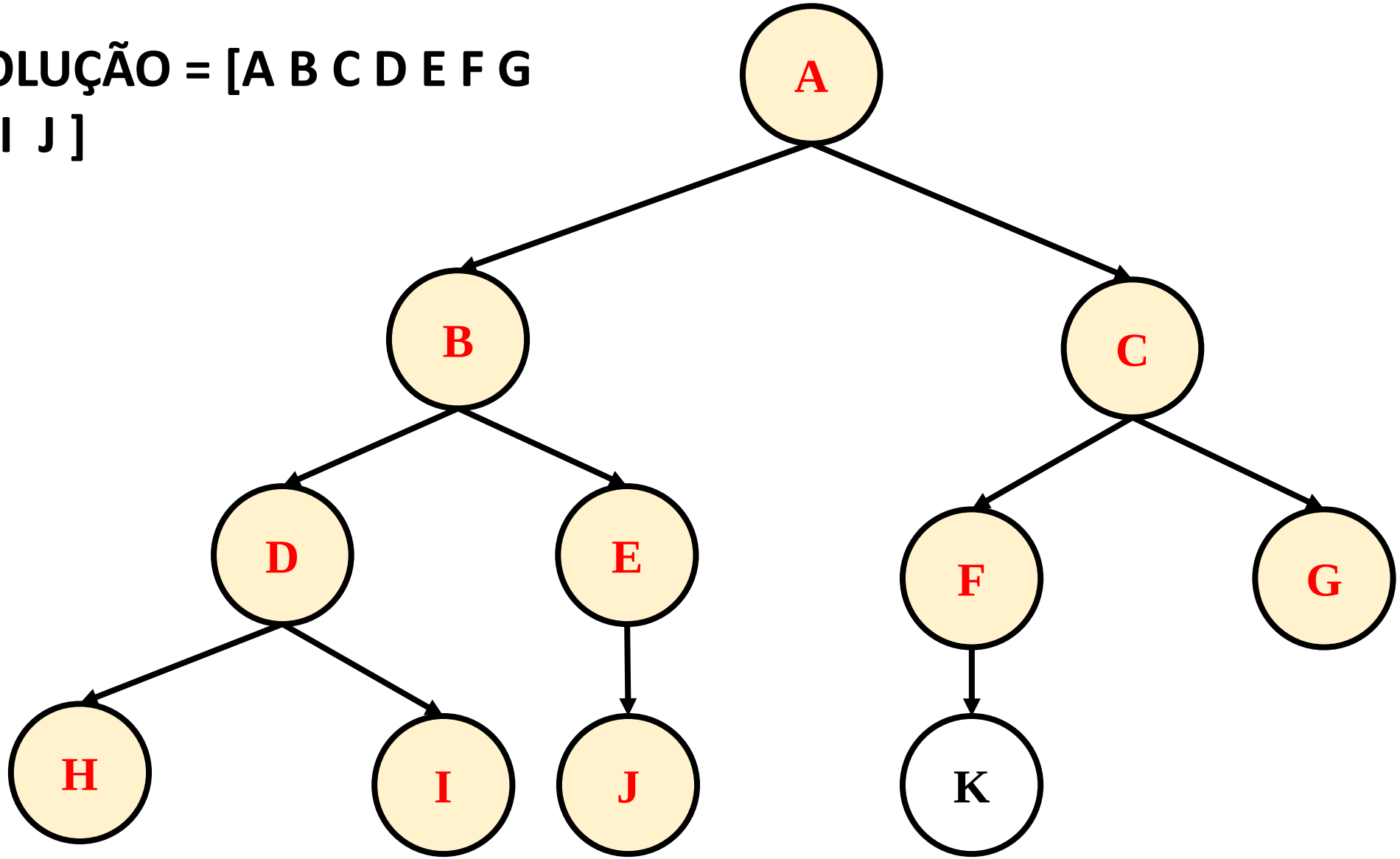
ORIGEM A OBJETIVO K LARGURA

SOLUÇÃO = [A B C D E F G H I]



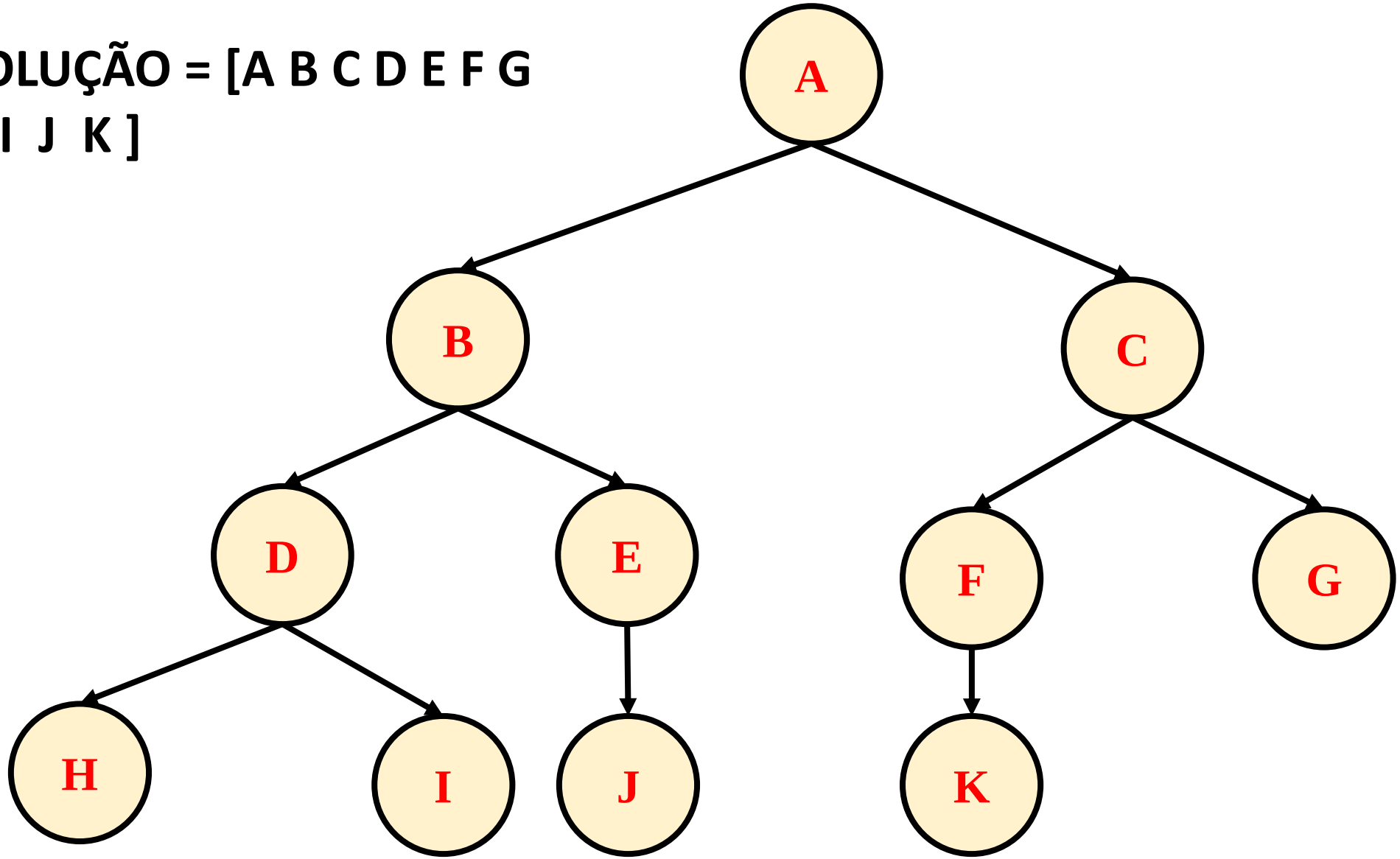
ORIGEM A OBJETIVO K LARGURA

SOLUÇÃO = [A B C D E F G H I J]



ORIGEM A OBJETIVO K LARGURA

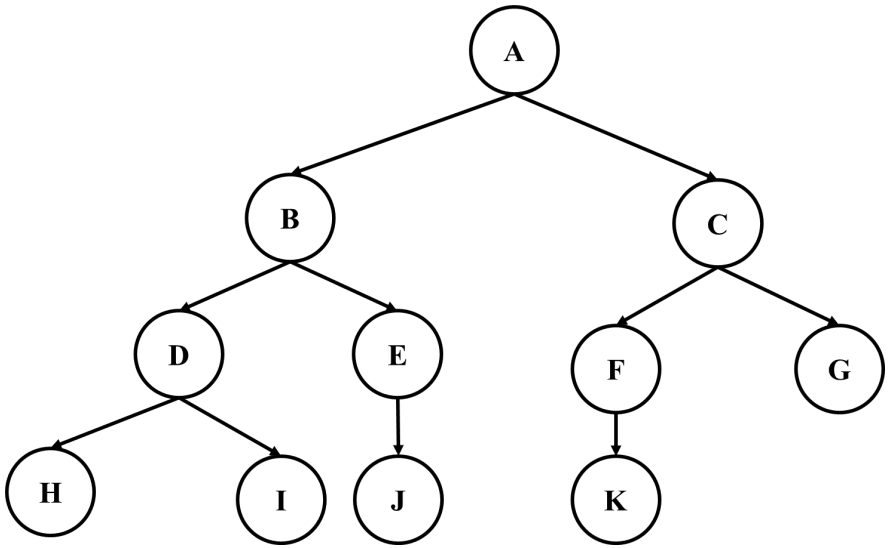
SOLUÇÃO = [A B C D E F G H I J K]



Exercicio

Dado o espaço de busca ao lado determine:

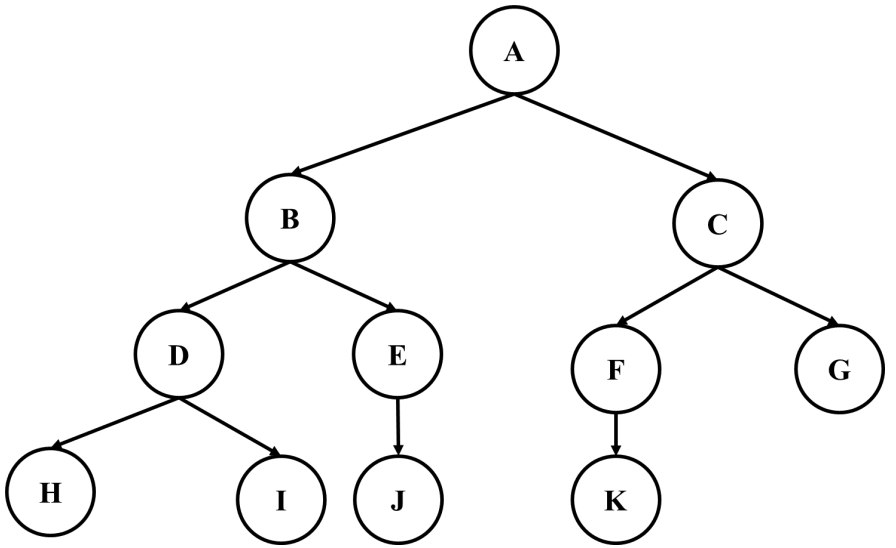
- Qual o caminho tendo como objetivo J realizando uma busca em PROFUNDIDADE
- Qual o caminho tendo como objetivo J realizando uma busca em LARGURA
- Qual o caminho tendo como objetivo K realizando uma busca em PROFUNDIDADE
- Qual o caminho tendo como objetivo K realizando uma busca em LARGURA



Exercicio

Dado o espaço de busca ao lado determine:

- Qual o caminho tendo como objetivo J realizando uma busca em PROFUNDIDADE
- **SOLUÇÃO(A,J, LARG) = [A B D H I E J] 7 ETAPAS**
- Qual o caminho tendo como objetivo J realizando uma busca em LARGURA
- **SOLUÇÃO(A,J, LARG) = [A B C D E F G I J] 9 ETAPAS**
- Qual o caminho tendo como objetivo K realizando uma busca em PROFUNDIDADE
- **SOLUÇÃO(A,K, PROF) = SOLUÇÃO = [A B D H I E J C F K] 10 ETAPAS**
- Qual o caminho tendo como objetivo K realizando uma busca em LARGURA **SOLUÇÃO(A,K, LARG) = [A B C D E F G H I J K] 10 ETAPAS**



Observações ...

- Critérios importantes na análise de um algoritmo de busca:
  - Completeza: O algoritmo oferece a garantia de encontrar uma solução quando ela existir?
  - Otimização: A estratégia encontra a solução ótima (tem o menor custo de caminho entre todas as soluções)?
  - Complexidade de tempo: Quanto tempo ele leva para encontrar uma solução?
  - Complexidade de espaço: Quanto de memória é necessário para executar a busca?

Estratégias de Busca Heurística

- Usam *conhecimento específico* do problema na busca da solução.
- Mais eficientes do que a busca não informada.

- Algoritmo geral: Busca pela Melhor Escolha - BME (*Best-first search*)
  - Seleciona para expansão o nó que tiver o menor custo estimado até a meta (objetivo), segundo uma *função de avaliação f(n).*
  - Tipicamente *f(n)* usa uma *função heurística h(n) =* custo estimado do caminho mais econômico do nó *n* até um nó objetivo (Restrição inicial: se *n* é um nó objetivo, *h*(*n*)=0).

Busca Heurística

- Uma forma de uso da informação heurística sobre um problema consiste em computar estimativas numéricas para os nós no espaço de estados;
- Uma estimativa indica o quanto um nó é promissor com relação ao alcance de um nó-objetivo;
- A idéia é continuar a busca sempre a partir do nó mais promissor no conjunto de candidatos;
- O programa de busca do melhor caminho (escolha) é baseado neste princípio.

Busca Heurística

- **Busca do melhor caminho** - pode ser derivada de um refinamento da busca em largura.

- **Busca em largura** - sempre escolhe para expansão os menores caminhos-candidatos (isto é, os nós extremos menos profundos da busca).

- **Busca do melhor caminho** - refina este princípio calculando uma estimativa heurística para cada candidato e escolhe para expansão o melhor candidato de acordo com esta estimativa.

Busca Heurística

Greedy best-first search

(*Busca* gulosa pela melhor escolha)

- Tenta expandir o nó mais próximo à meta, na suposição de que isso provavelmente levará a uma solução rápida.

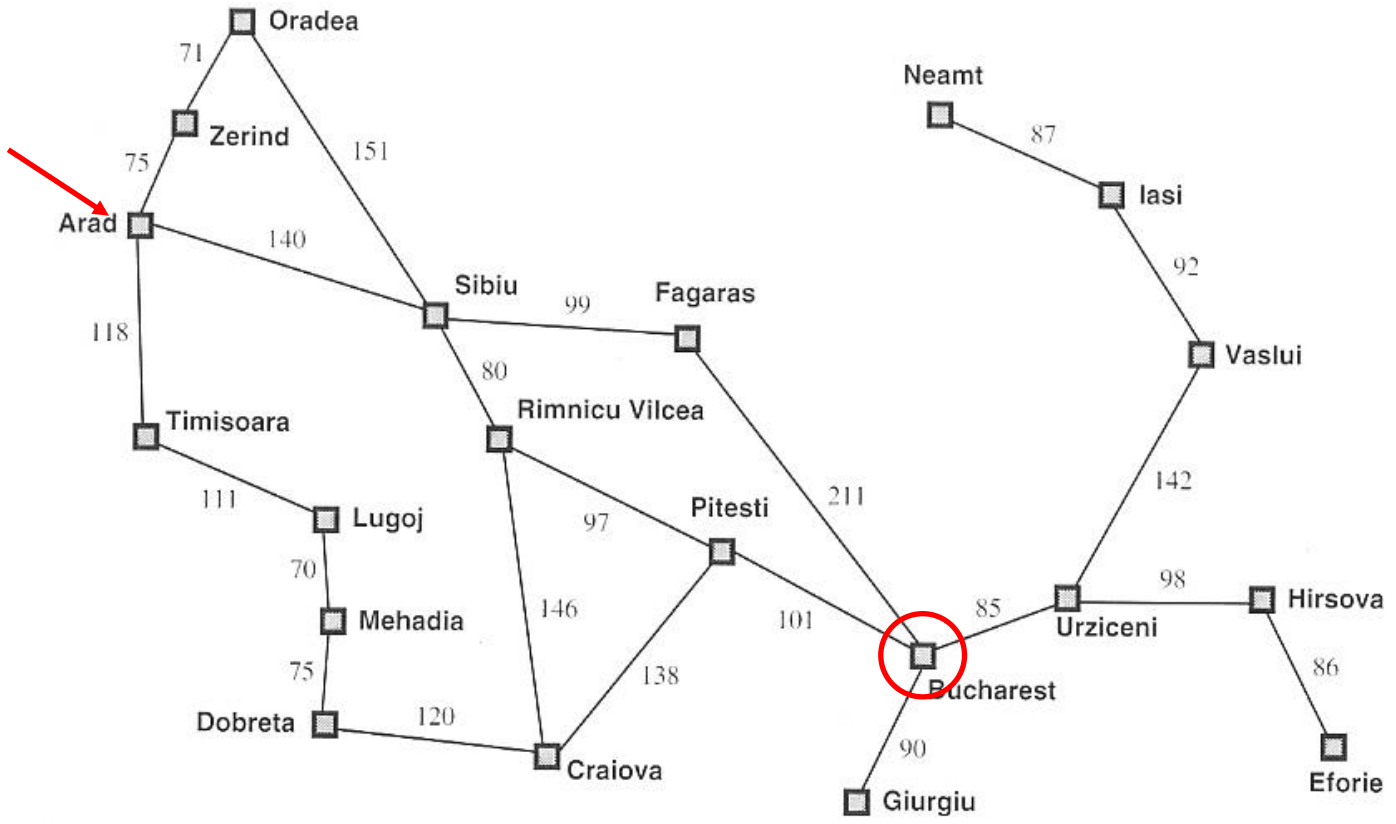
- Avalia nós para expandir com base unicamente na função heurística: $f(n) = h(n)$

- **Exemplo**: encontrar a melhor rota (rota mais curta) de uma cidade a outra, num mapa.
  - $h(n)$ = distância em linha reta entre as cidades e a cidade-meta.

Busca Heurística

Exemplo: Localização de rotas na Romênia, usando a heurística de **distância em linha reta (h_{DLR})**

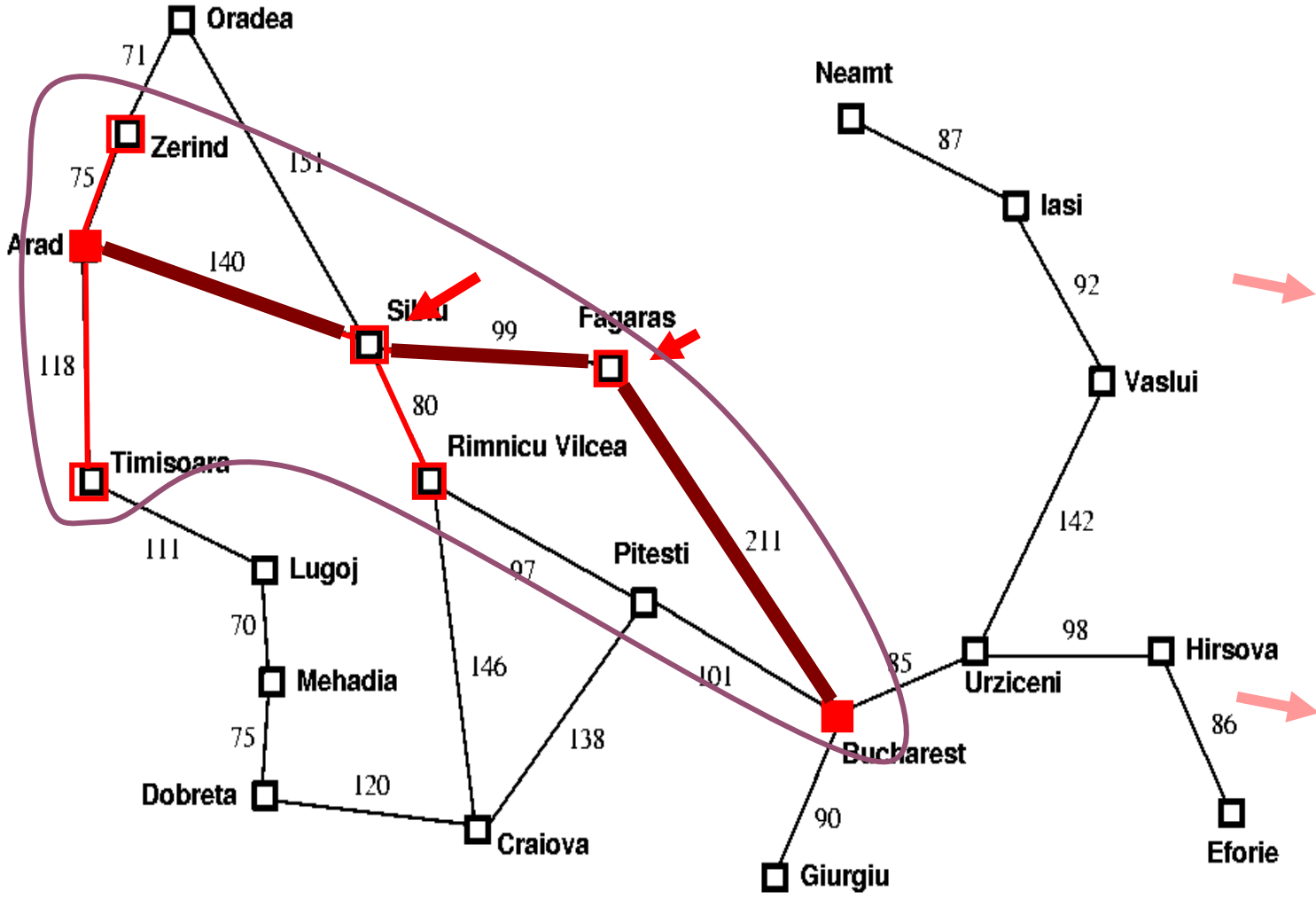
Objetivo: Bucharest (Bucareste)



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Um mapa rodoviário simplificado de parte da Romênia.

Busca Heurística Busca pela melhor escolha - Busca Gulosa

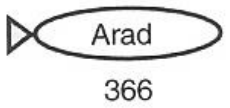


Distância em linha reta para Bucharest:

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

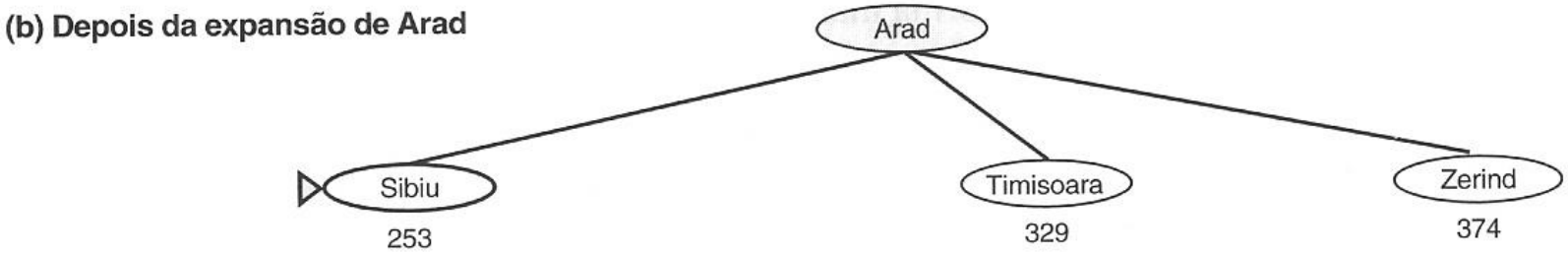
Exemplo – Passo a Passo ...

(a) O estado inicial



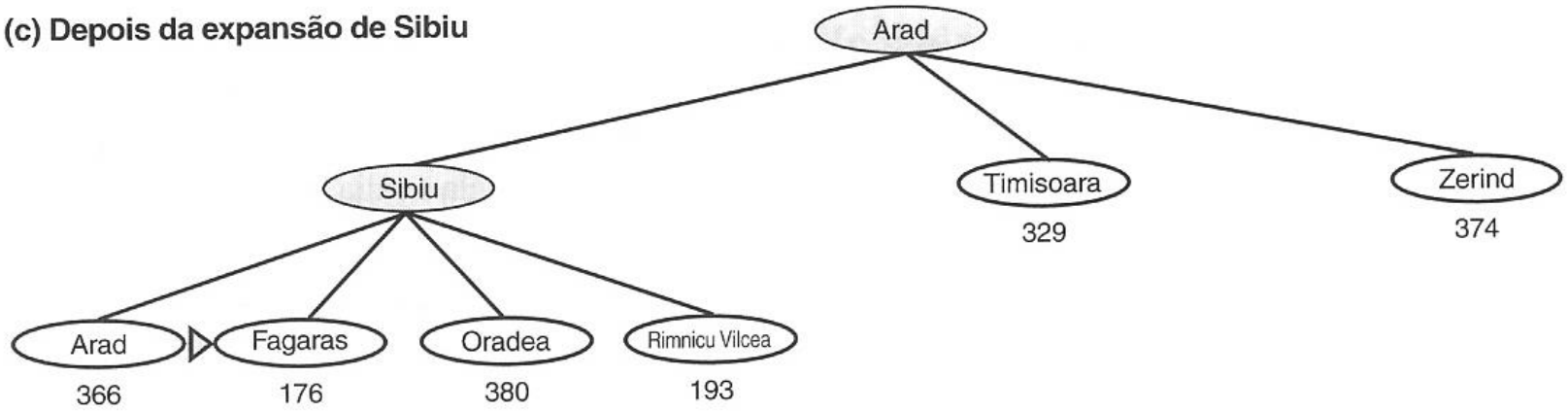
Fases de uma busca gulosa pela melhor escolha para Bucareste, usando-se a heurística de distância em linha reta $\mathbf{h_{DLR}}$. Os nós são identificados por seus valores de h.

Exemplo – Passo a Passo ...



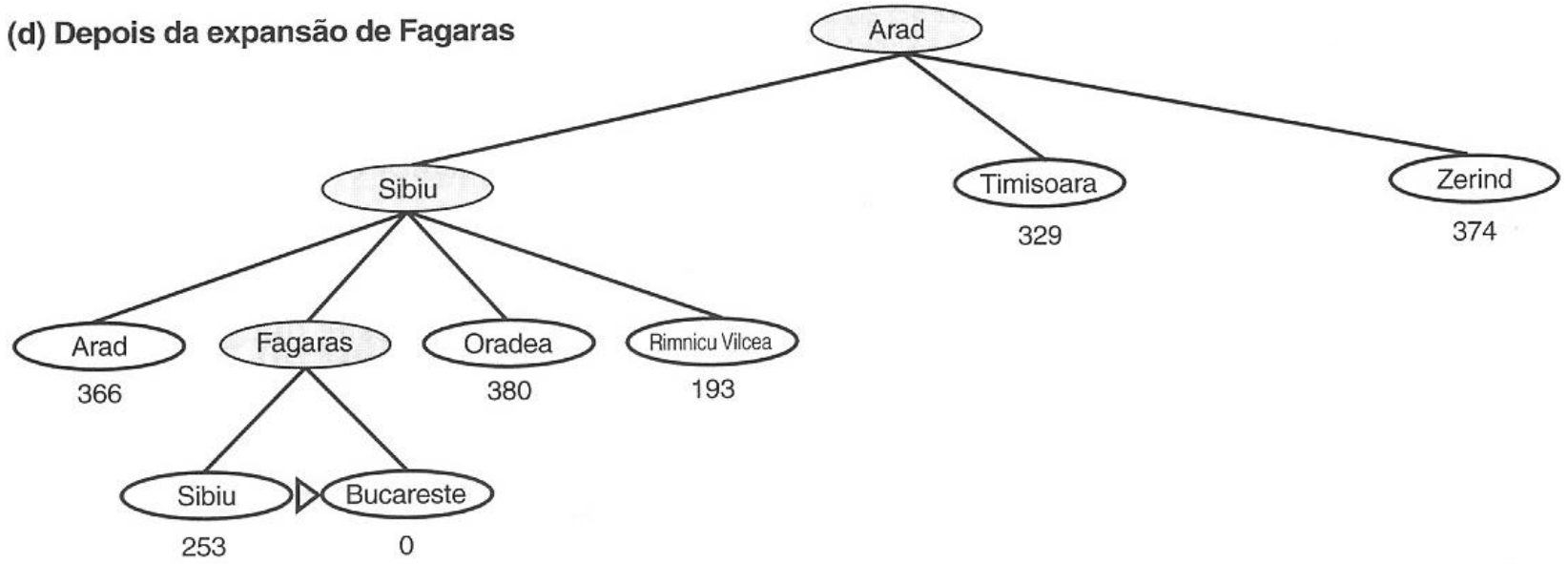
Fases de uma busca gulosa pela melhor escolha para Bucareste, usando-se a heurística de distância em linha reta $\mathbf{h_{DLR}}$. Os nós são identificados por seus valores de h.

Exemplo – Passo a Passo ...

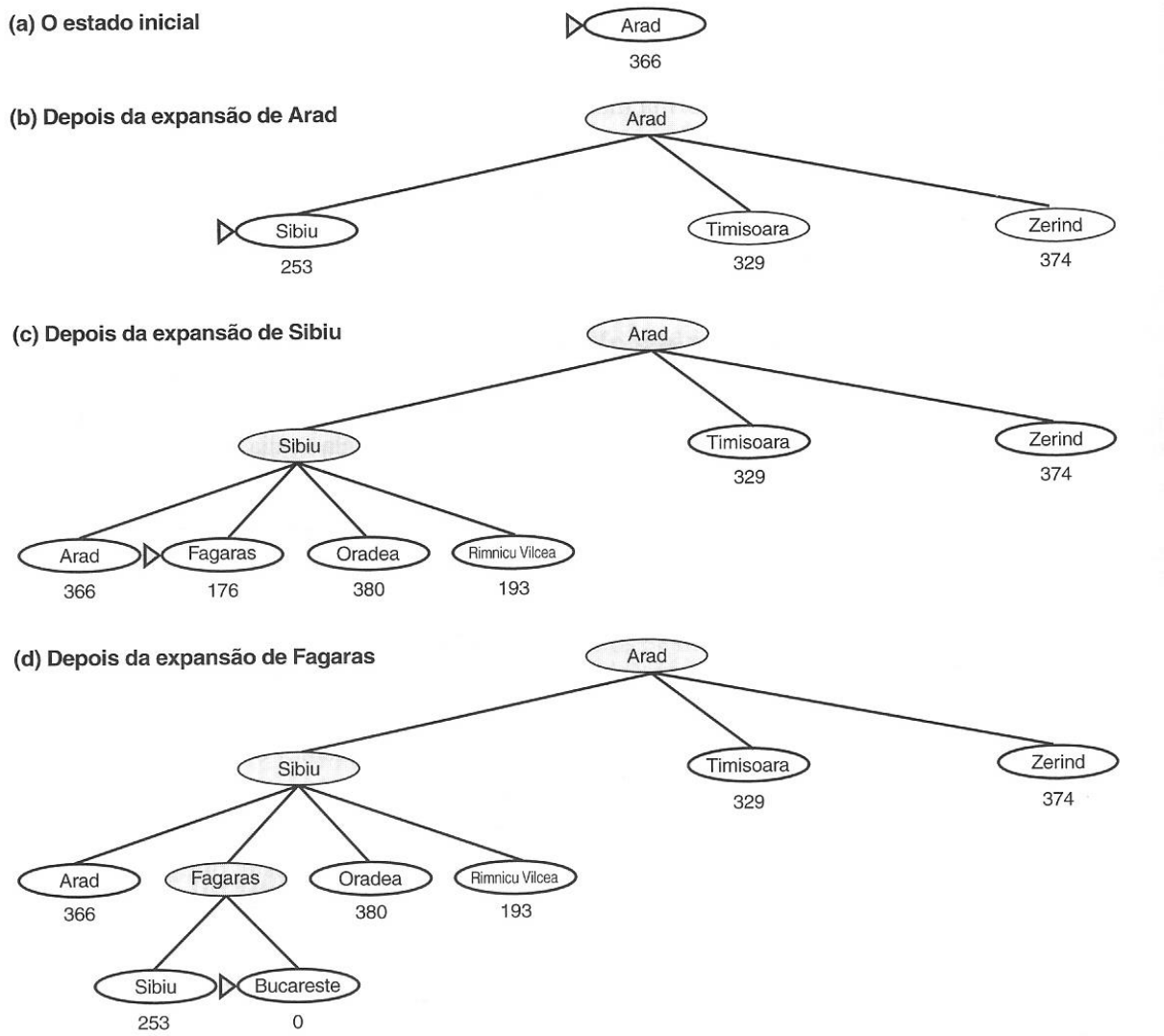


Fases de uma busca gulosa pela melhor escolha para Bucareste, usando-se a heurística de distância em linha reta $\mathbf{h_{DLR}}$. Os nós são identificados por seus valores de h.

Exemplo – Passo a Passo ...



Fases de uma busca gulosa pela melhor escolha para Bucareste, usando-se a heurística de distância em linha reta $\mathbf{h_{DLR}}$. Os nós são identificados por seus valores de h.



Fases de uma busca gulosa pela melhor escolha para Bucareste, usando-se a heurística de distância em linha reta $\mathbf{h_{DLR}}$. Os nós são identificados por seus valores de h.

Busca Heurística

Não é completa

- pode entrar em ciclos e não encontrar a solução se não detectar estados repetidos;
- pode se perder em um caminho infinito e nunca retroceder para tentar outras opções.

Não é ótima

- No exemplo encontrou caminho (Arad, Sibiu, Fagaras, Bucharest) que é 32km maior que (Arad, Sibiu, Rimnicu Vilcea, Pitesti, Bucharest)

- Dependendo do problema e da qualidade da heurística a complexidade pode ter uma redução substancial.

Busca Heurística

BME mais "famoso": Busca A*

- **Objetivo:** Minimizar o custo total estimado da solução.
- **Função de avaliação:** $f(n) = g(n) + h(n)$
  - $g(n)$ = distância (custo) do nó inicial ao nó n
  - $h(n)$ = distância (custo) **estimada** de n ao nó final
  - Assim, $f(n)$ **estima** o custo da melhor solução que passa por n.
- **A* expande o nó de menor valor de f na fronteira do espaço de estados.**

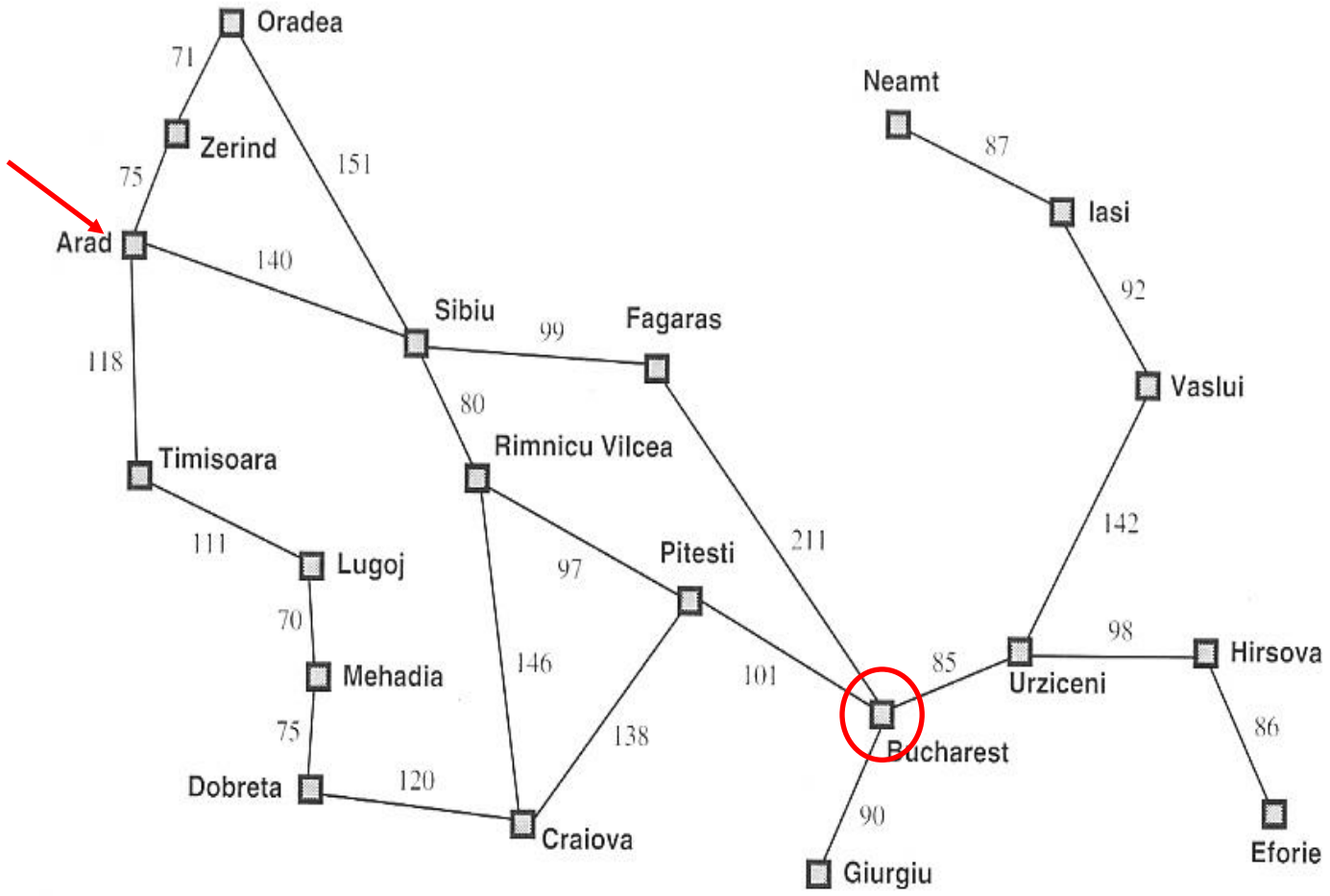
Busca Heurística

- Quando ***n*** é encontrado pelo processo de busca, tem-se a seguinte situação:
  - Um caminho de ***i*** para ***n*** já deve ter sido encontrado e o seu custo pode ser calculado como a soma dos custos dos arcos no caminho, e pode servir como uma estimativa ***g*(*n*)** do custo mínimo de ***i*** para ***n***.
  - ***h*(*n*)** é mais problemático porque o espaço entre ***n*** e ***t*** ainda não foi explorado, e portanto ***h*(*n*)** é meramente um palpite baseado no conhecimento geral do algoritmo sobre o problema particular.
  - Não existe um método universal para construção de ***h***, pois depende do domínio do problema.

Busca Heurística

Exemplo: Localização de rotas na Romênia, usando a **Busca *A**

Objetivo: Bucharest (Bucareste)

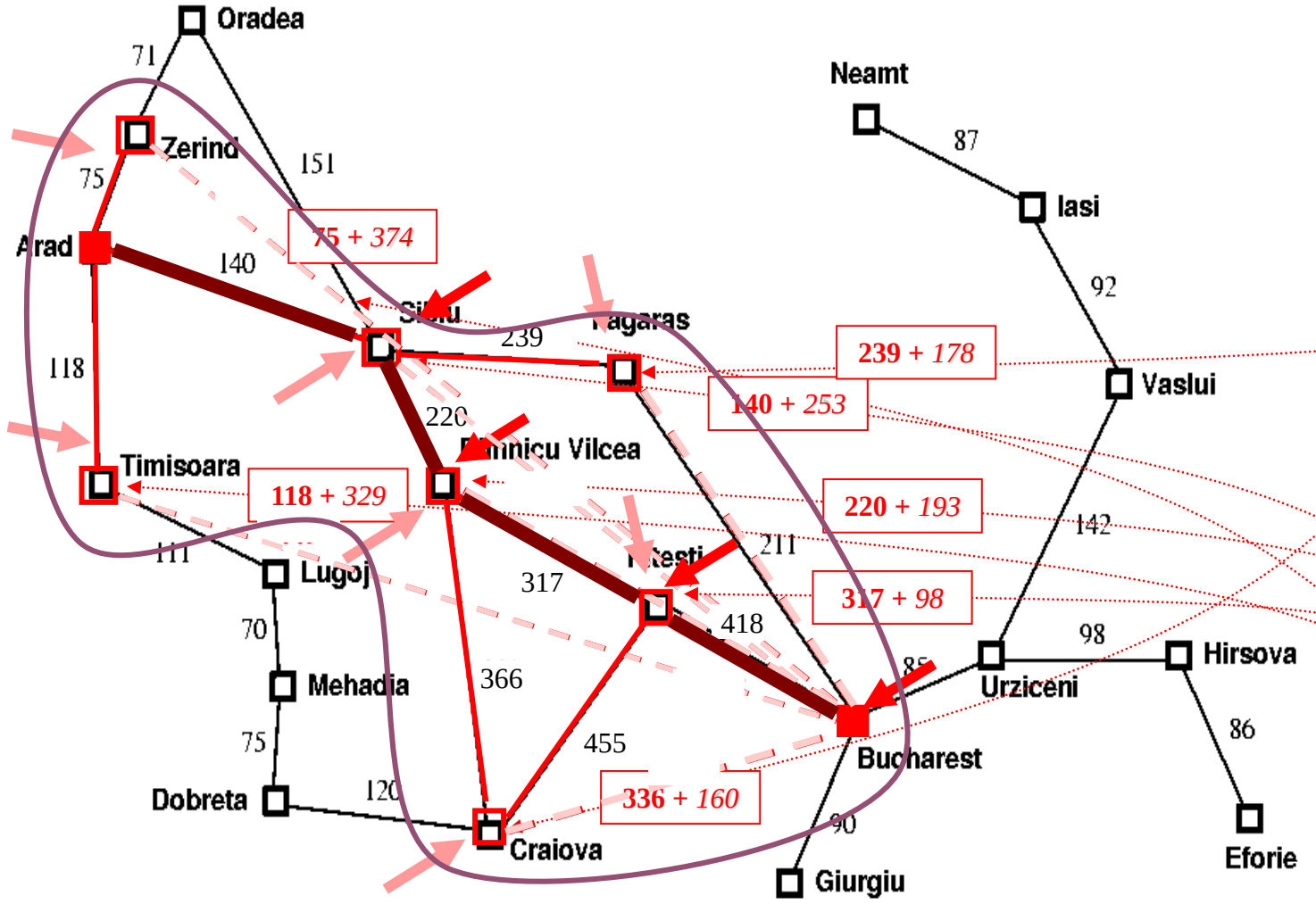


Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Um mapa rodoviário simplificado de parte da Romênia.

Busca Heurística

Busca pela melhor escolha - Algoritmo A*

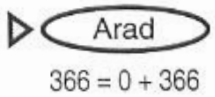


Distância em linha reta para Bucharest:

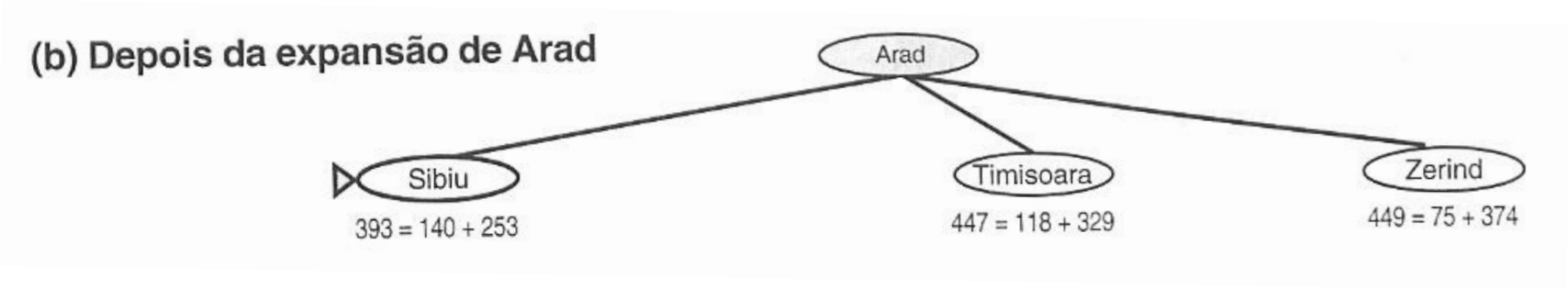
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Exemplo – Passo a Passo ...

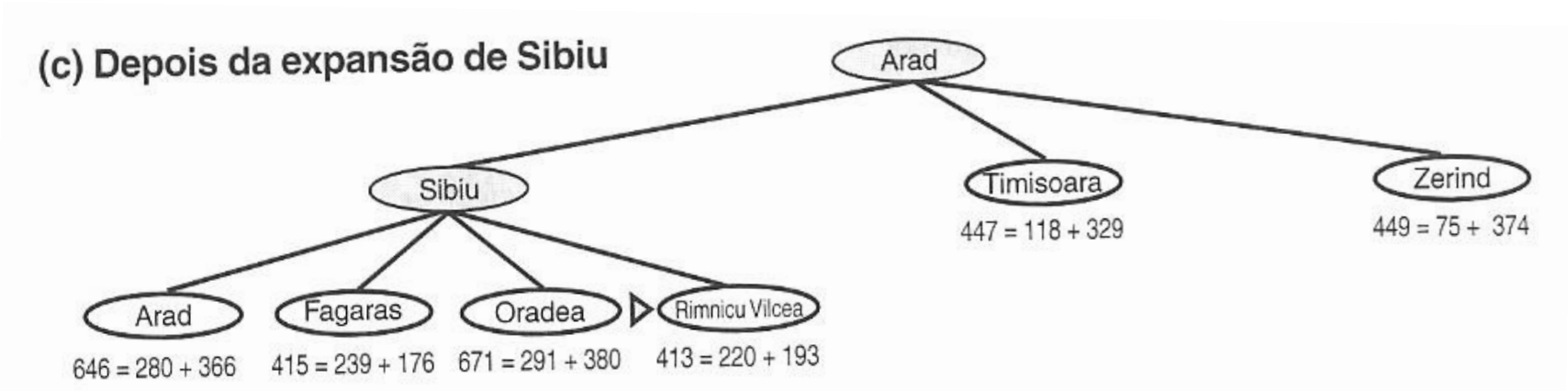
(a) O estado inicial



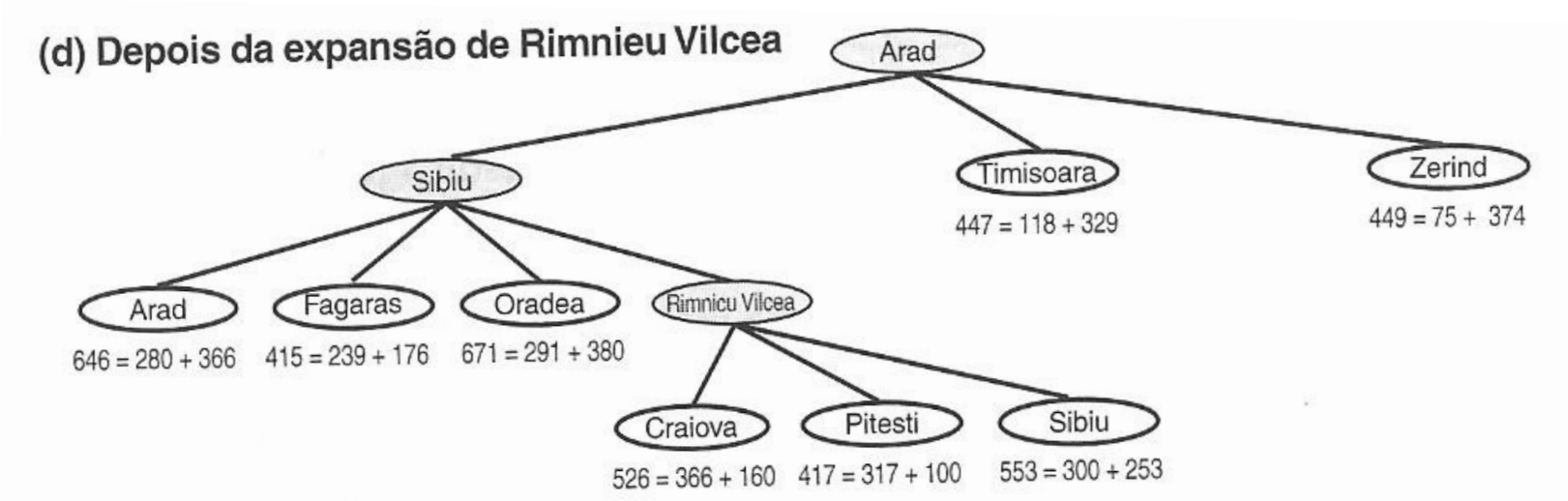
Exemplo – Passo a Passo ...



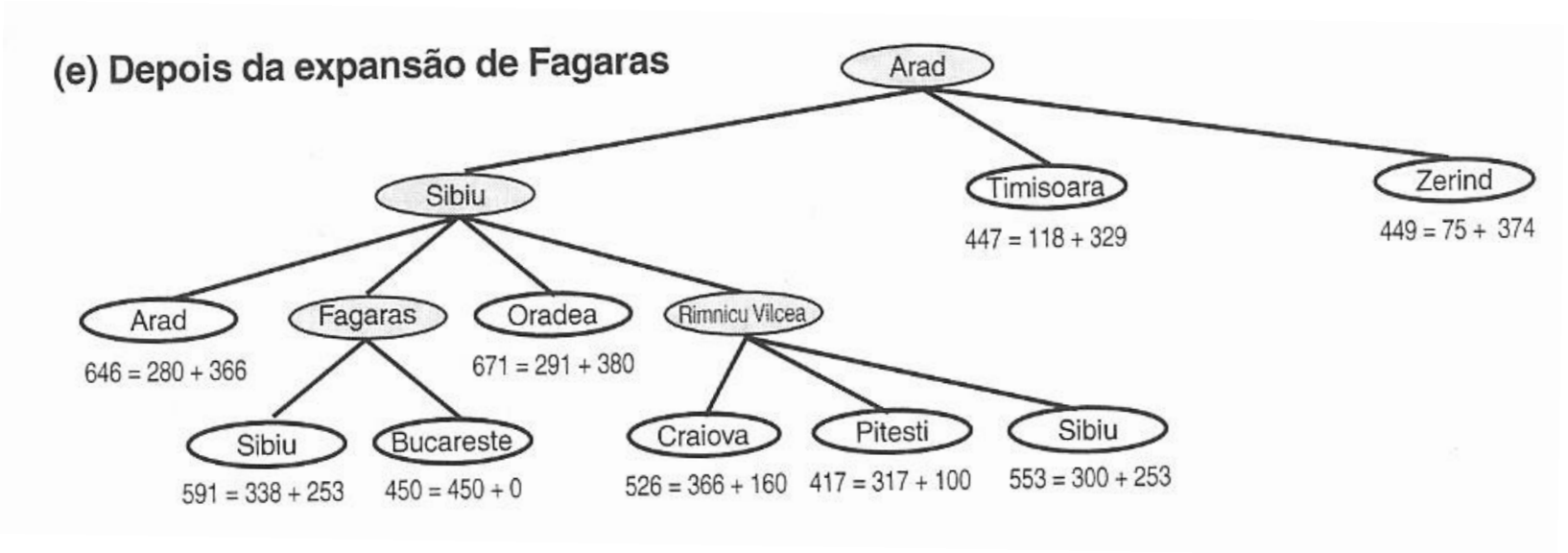
Exemplo – Passo a Passo ...



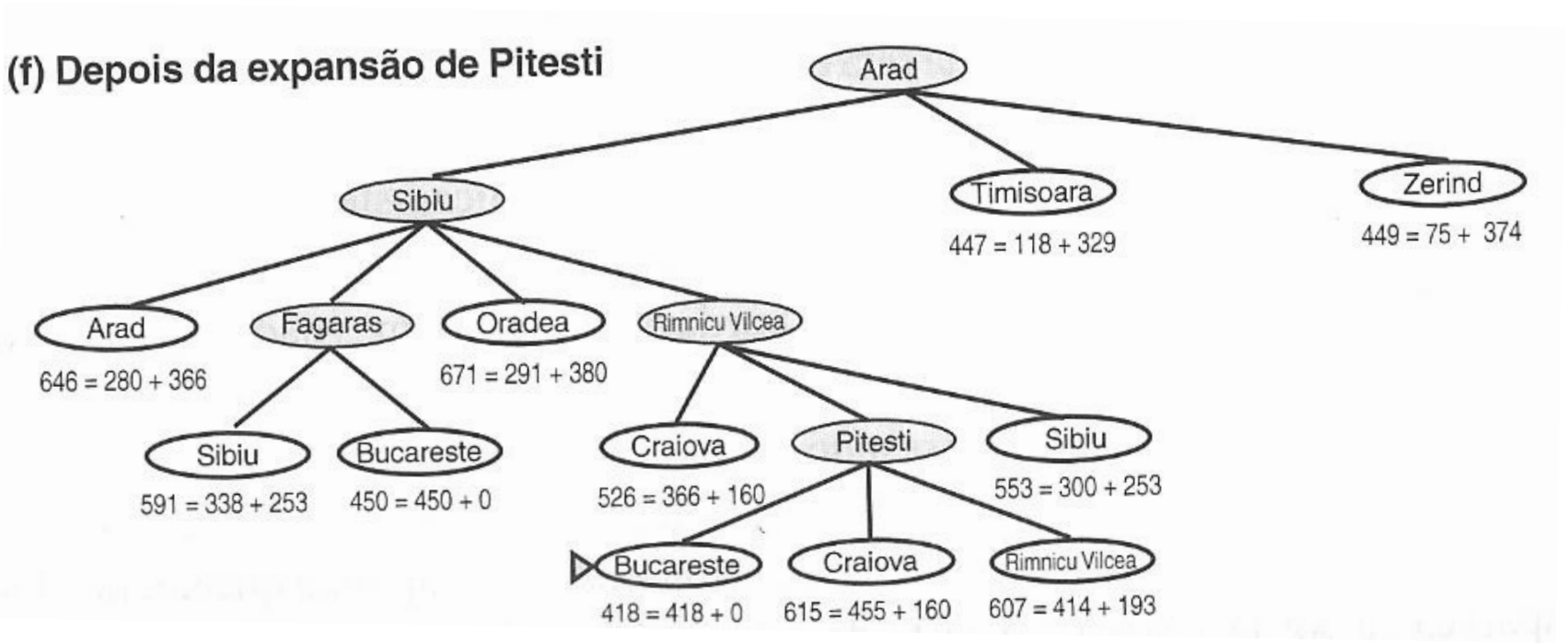
Exemplo – Passo a Passo ...



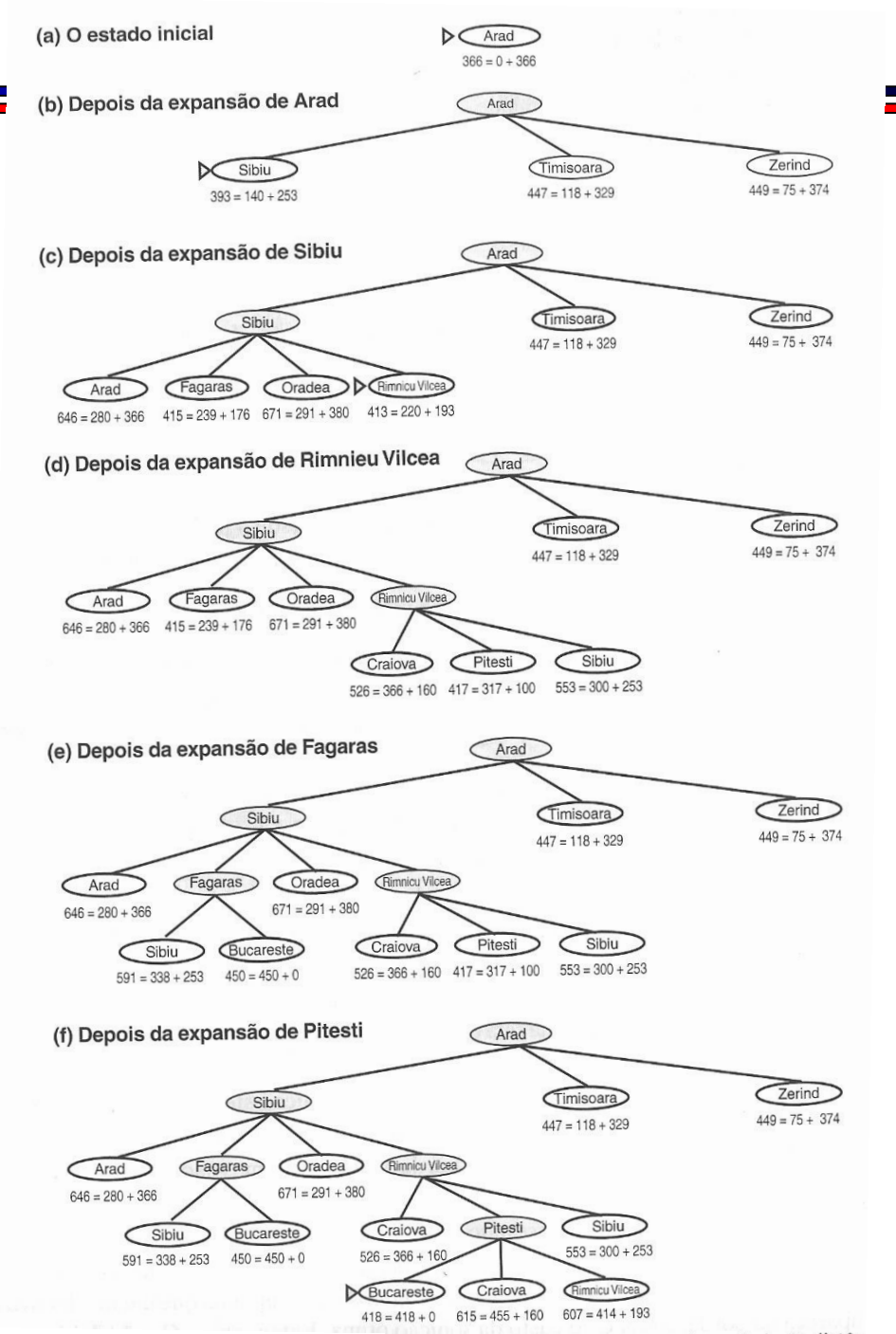
Exemplo – Passo a Passo ...



Exemplo – Passo a Passo



Estágios em uma busca A* por Bucareste. Os nós estão rotulados ***f = g + h***. Os valores de *h* são distâncias em linha reta para Bucareste.



Desempenho do A*

- A análise do caráter ótimo de ***A**** é direta se for usada com BUSCA-EM-ÁRVORE: ***A**** *será ótima se h(n) for uma* ***heurística admissível***.

- Conseqüência mais importante da **consistência** (também chamada **monotonicidade**) é: ***A**** *usando BUSCA-EM-GRAFO é ótima se h(n) é* ***consistente***.