

INTELIGÊNCIA ARTIFICIAL

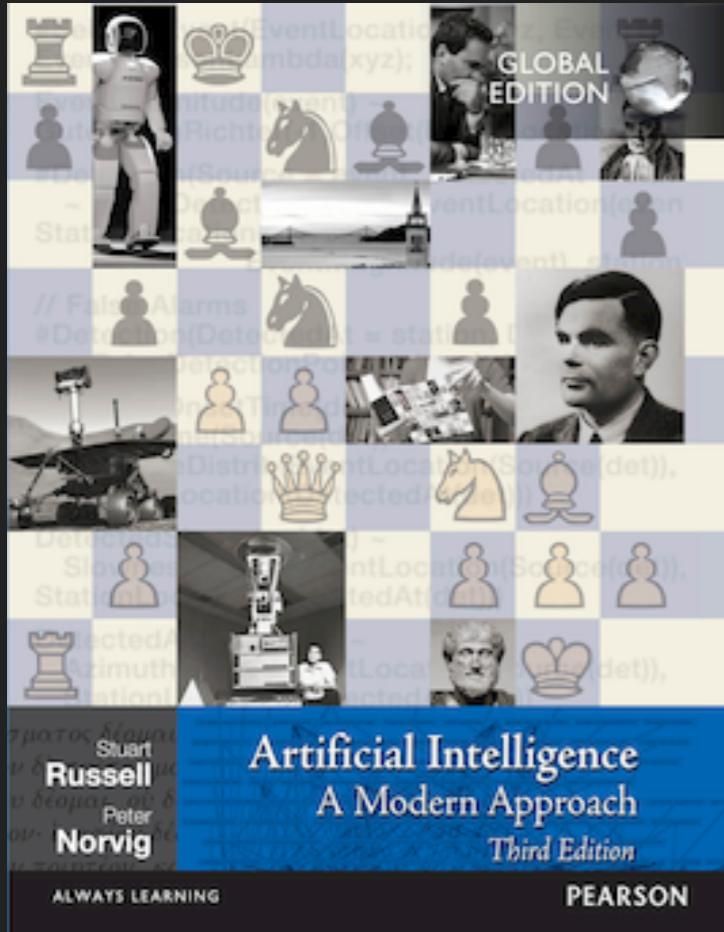
BUSCA EM ESPAÇO DE
ESTADOS

MATERIAL DE ESTUDO

Artificial Intelligence: A modern approach

Cap. 3

MATERIAL DE ESTUDO

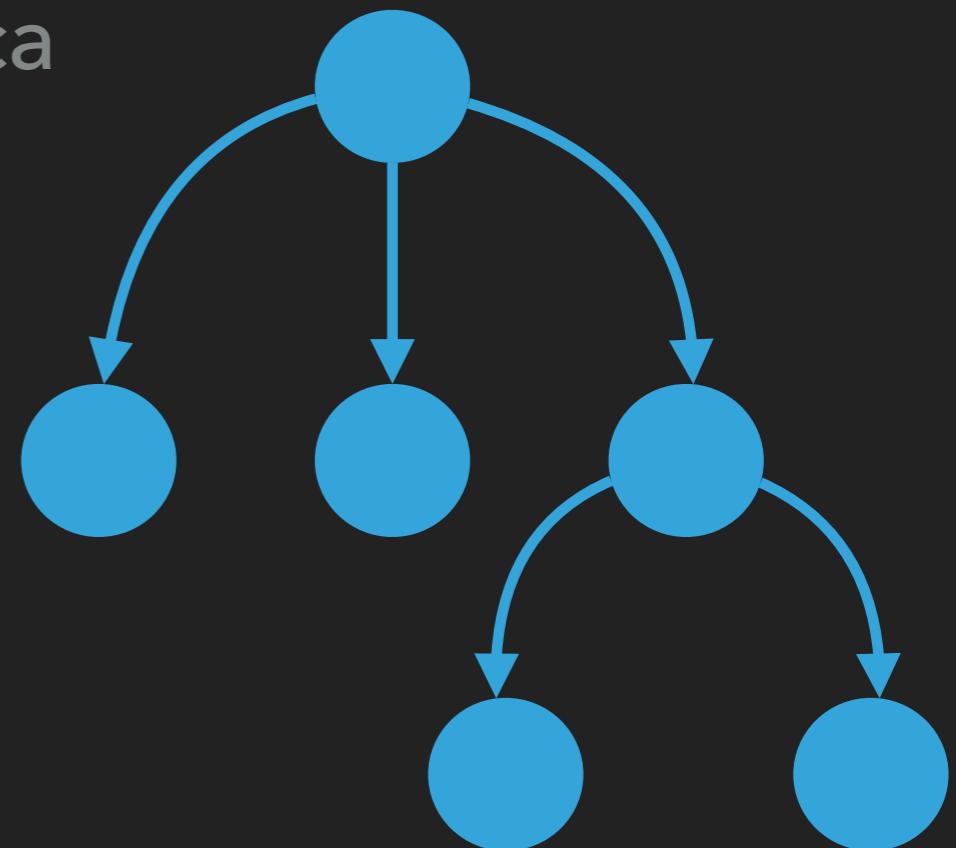


Artificial Intelligence: A modern approach

Cap. 3

SUMÁRIO

- ▶ Agentes para busca em espaços de estados
- ▶ Tipos de problemas de busca
- ▶ Formulação de problemas de busca
- ▶ Algoritmos e estratégias



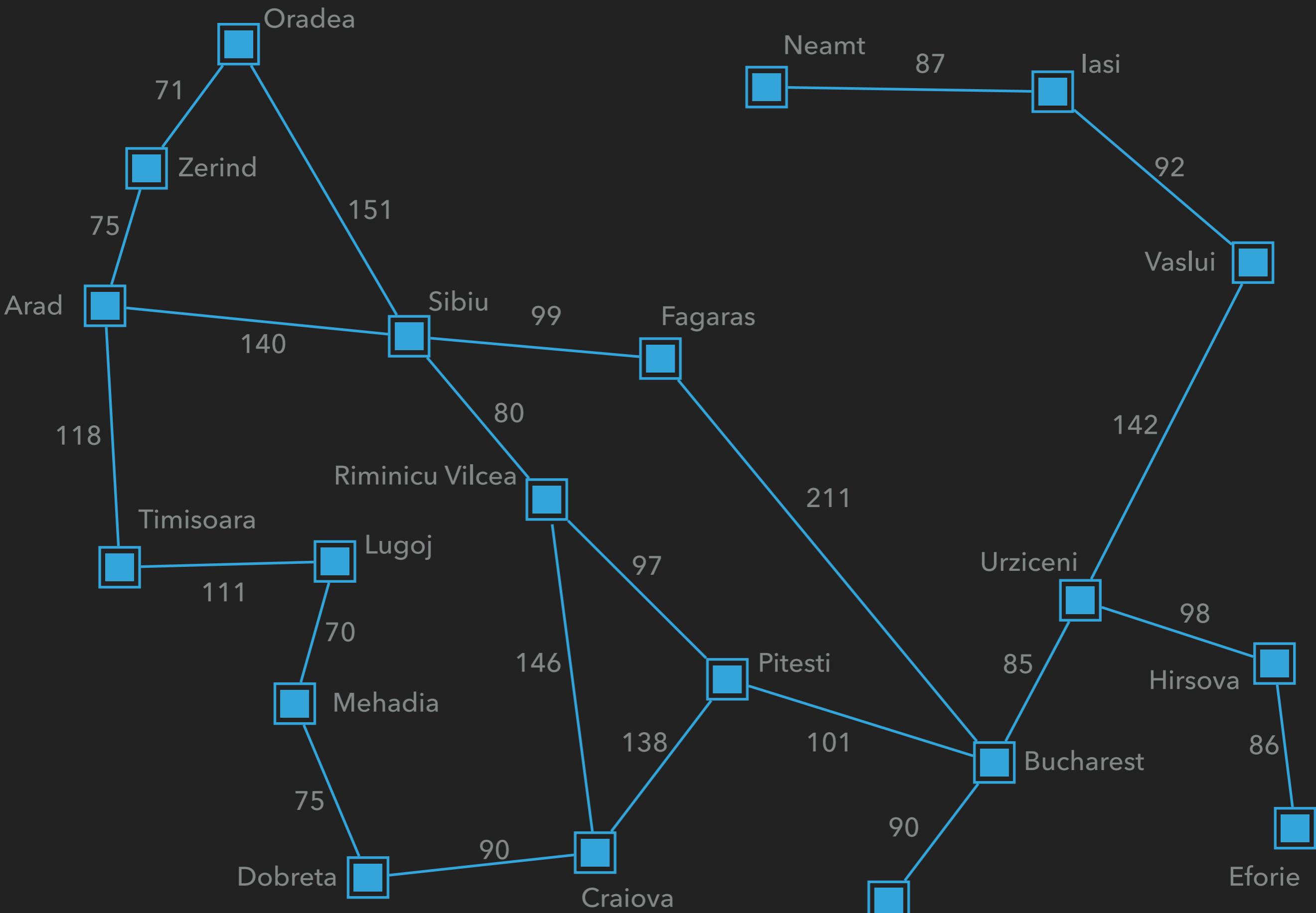


BUSCA EM ESPAÇO DE
ESTADOS

FÉRIAS NA
ROMÊNIA

Lucas Baggio Figueira [@lucasfigueira]

COMO DESCOPRIR UM CAMINHO ENTRE ARAD E BUCARESTE?



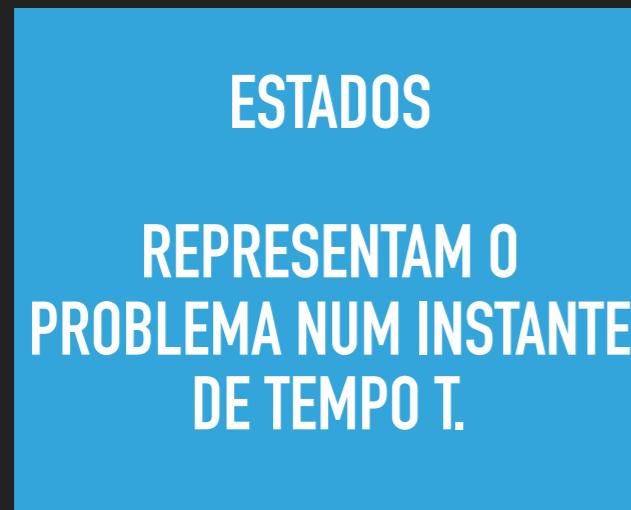
**QUAL A SEQUÊNCIA DE ESTADOS QUE
COMPÕE A SOLUÇÃO DO PROBLEMA?
(PROBLEM FORMULATION)**

PROBLEMA DE BUSCA

- ▶ Definido por 5 componentes:

PROBLEMA DE BUSCA

- ▶ Definido por 5 componentes:



PROBLEMA DE BUSCA

- Definido por 5 componentes:

ESTADOS

REPRESENTAM O
PROBLEMA NUM INSTANTE
DE TEMPO T.

CUSTO

QUANTO A SOLUÇÃO
(SEQUÊNCIA DE ESTADOS)
CUSTA PARA SER
EXECUTADA.

PROBLEMA DE BUSCA

- Definido por 5 componentes:

ESTADOS

REPRESENTAM O
PROBLEMA NUM INSTANTE
DE TEMPO T.

COLEÇÃO DE AÇÕES

CONJUNTO DE AÇÕES QUE
PODEM SER REALIZADAS
PELO AGENTE.

CUSTO

QUANTO A SOLUÇÃO
(SEQUÊNCIA DE ESTADOS)
CUSTA PARA SER
EXECUTADA.

PROBLEMA DE BUSCA

- Definido por 5 componentes:

ESTADOS

REPRESENTAM O
PROBLEMA NUM INSTANTE
DE TEMPO T.

FUNÇÃO SUCESSORA

DADO UM ESTADO GERAR
OS POSSÍVEIS ESTADOS
IMEDIATAMENTE
ALCANÇÁVEIS.

CUSTO

QUANTO A SOLUÇÃO
(SEQUÊNCIA DE ESTADOS)
CUSTA PARA SER
EXECUTADA.

COLEÇÃO DE AÇÕES

CONJUNTO DE AÇÕES QUE
PODEM SER REALIZADAS
PELO AGENTE.

PROBLEMA DE BUSCA

- Definido por 5 componentes:

ESTADOS

REPRESENTAM O
PROBLEMA NUM INSTANTE
DE TEMPO T.

FUNÇÃO SUCESSORA

DADO UM ESTADO GERAR
OS POSSÍVEIS ESTADOS
IMEDIATAMENTE
ALCANÇÁVEIS.

COLEÇÃO DE AÇÕES

CONJUNTO DE AÇÕES QUE
PODEM SER REALIZADAS
PELO AGENTE.

CUSTO

QUANTO A SOLUÇÃO
(SEQUÊNCIA DE ESTADOS)
CUSTA PARA SER
EXECUTADA.

TESTE DE OBJETIVO

DEFINIR COMO DETECTAR
SE O ESTADO FINAL
(OBJETIVO) FOI ATINGIDO.

PROBLEMA DE BUSCA

► De

A SOLUÇÃO É A SEQUÊNCIA
DE ESTADOS A PARTIR DO
ESTADO INICIAL ATÉ O
ESTADO FINAL.

OS ESTADOS
IMEDIATAMENTE
ALCANÇÁVEIS.

SE O ESTADO FINAL
(OBJETIVO) FOI ATINGIDO.

**DEFINA O PROBLEMA
DO ASPIRADOR DE PÓ**

ESTADOS?

DE
DO

MA
PÓ

ESTADOS?

DE
DO

MA
PÓ

ESTADOS?

AÇÕES?

DE
DO

MA
PÓ

ESTADOS?

AÇÕES?

DE
DO

MA
PÓ

ESTADOS?

AÇÕES?

FUNÇÃO SUCESSORA?

DE
DO

MA
PÓ

ESTADOS?

AÇÕES?

FUNÇÃO SUCESSORA?

DE
DO

MA
PÓ

ESTADOS?

AÇÕES?

FUNÇÃO SUCESSORA?

TESTE DE OBJETIVO?

DE
DO

MA
PÓ

ESTADOS?

AÇÕES?

FUNÇÃO SUCESSORA?

TESTE DE OBJETIVO?

DE
DO

MA
PÓ

ESTADOS?

AÇÕES?

FUNÇÃO SUCESSORA?

TESTE DE OBJETIVO?

CUSTO?

DE
DO

MA
PÓ

**DEFINA O TESTE
DE EINSTEIN**

	1 ^a Casa	2 ^a Casa	3 ^a Casa	4 ^a Casa	5 ^a Casa
Cor	<input type="text"/>				
Nacionalidade	<input type="text"/>				
Bebida	<input type="text"/>				
Cigarro	<input type="text"/>				
Animal	<input type="text"/>				

Dicas:

- O Norueguês vive na primeira casa.
- O Inglês vive na casa Vermelha.
- O Sueco tem Cachorros como animais de estimação.
- O Dinamarquês bebe Chá.
- A casa Verde fica do lado esquerdo da casa Branca.
- O homem que vive na casa Verde bebe Café.
- O homem que fuma Pall Mall cria Pássaros.
- O homem que vive na casa Amarela fuma Dunhill.
- O homem que vive na casa do meio bebe Leite.
- O homem que fuma Blends vive ao lado do que tem Gatos.
- O homem que cria Cavalos vive ao lado do que fuma Dunhill.
- O homem que fuma BlueMaster bebe Cerveja.
- O Alemão fuma Prince.
- O Norueguês vive ao lado da casa Azul.
- O homem que fuma Blends é vizinho do que bebe Água.



ESTADOS?

AÇÕES?

FUNÇÃO SUCESSORA?

TESTE DE OBJETIVO?

CUSTO?

Cor
Nacionali
Bebid
Cigar
Anim

Dicas:

- O Norueguês vive no cão.
- O Inglês vive na casa amarela.
- O Sueco tem Cachorro.
- O Dinamarquês fuma Dunhill.
- A casa Verde fica ao lado da casa Vermelha.
- O homem que vira o leite vive no fundo.
- O homem que fuma Dunhill vive no topo.
- O homem que vira o leite bebe Água.



5^a Casa

	▼
	▼
	▼
	▼
	▼

Leite.
o que tem Gatos.
o que fuma Dunhill.
veja.

que bebe Água.

	1 ^a Casa	2 ^a Casa	3 ^a Casa	4 ^a Casa	5 ^a Casa
Cor	<input type="button" value="▼"/>				
Nacionalidade	<input type="button" value="▼"/>				
Bebida	<input type="button" value="▼"/>				
Cigarro	<input type="button" value="▼"/>				
Animal	<input type="button" value="▼"/>				

Dicas:

- O Norueguês vive na primeira casa.
- O Inglês vive na casa Vermelha.
- O Sueco tem Cachorros como animais de estimação.
- O Dinamarquês bebe Chá.
- A casa Verde fica do lado esquerdo da casa Branca.
- O homem que vive na casa Verde bebe Café.
- O homem que fuma Pall Mall cria Pássaros.
- O homem que vive na casa Amarela fuma Dunhill.
- O homem que vive na casa do meio bebe Leite.
- O homem que fuma Blends vive ao lado do que tem Gatos.
- O homem que cria Cavalos vive ao lado do que fuma Dunhill.
- O homem que fuma BlueMaster bebe Cerveja.
- O Alemão fuma Prince.
- O Norueguês vive ao lado da casa Azul.
- O homem que fuma Blends é vizinho do que bebe Água.

DEFINA OS ESTADOS?

DE EINSTEIN

ESTADOS?

AÇÕES?

FUNÇÃO SUCESSORA?

TESTE DE OBJETIVO?

CUSTO?

DEFINA O PROBLEMA DO PUZZLE-8

Estado Inicial

6	2	8
3		5
1	4	7

Estado Final

1	2	3
4	5	6
7		8

ESTADOS?

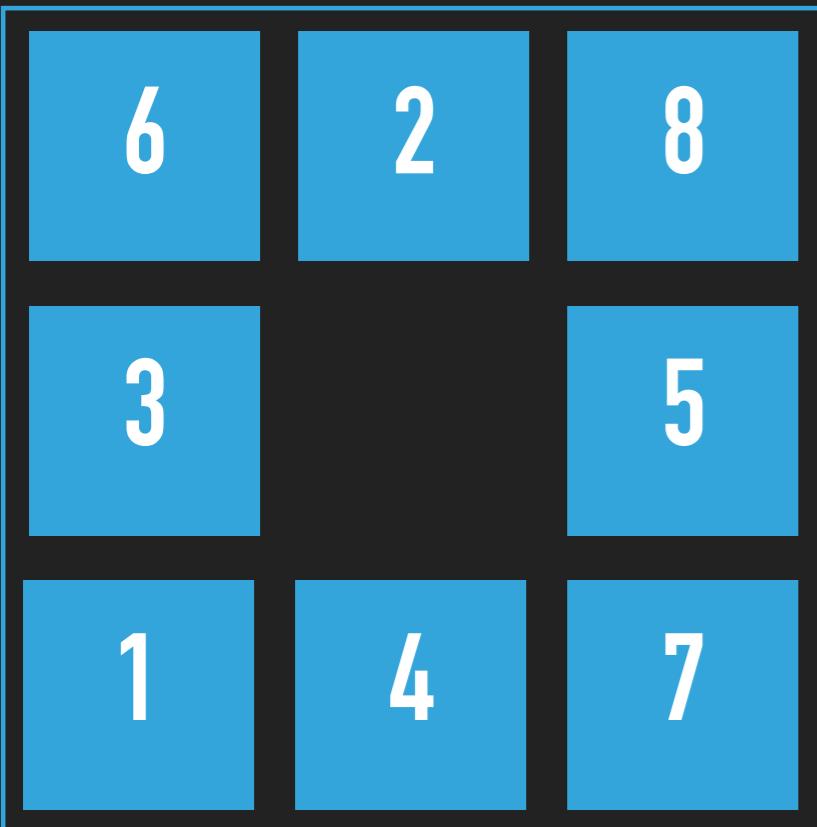
AÇÕES?

FUNÇÃO SUCESSORA?

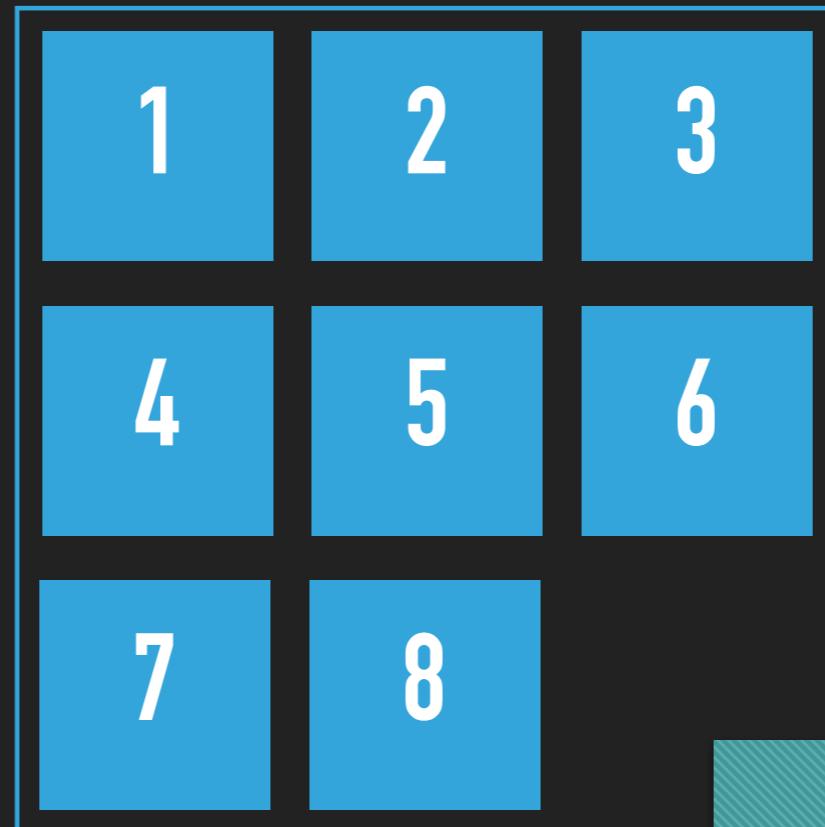
TESTE DE OBJETIVO?

CUSTO?

Estado Inicial



Estado Final



ESTADOS?

AÇÕES?

FUNÇÃO SUCESSORA?

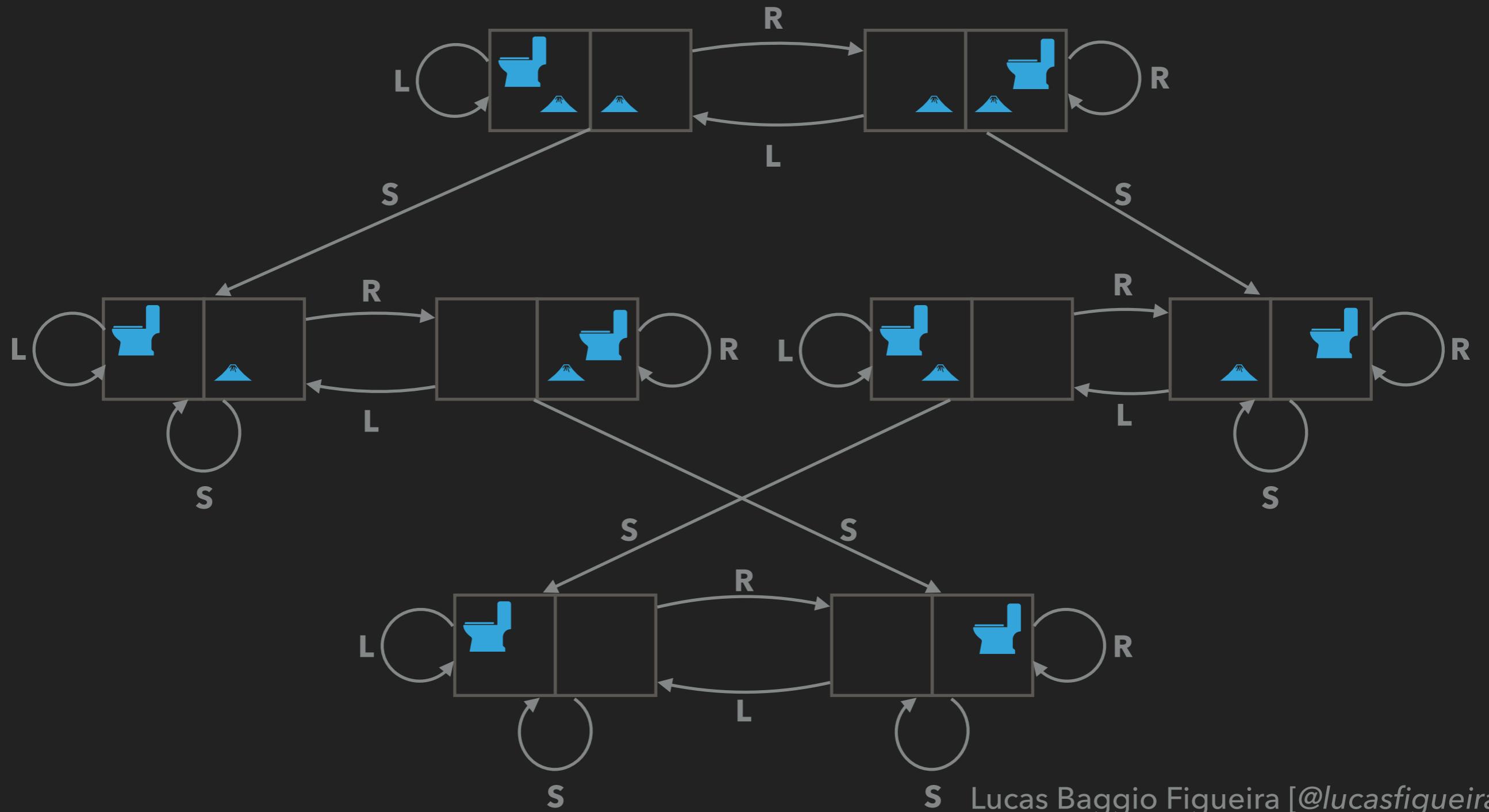
TESTE DE OBJETIVO?

CUSTO?

AGENTE DE BUSCA

```
function Simple-Problem-Solving-Agent(percept) returns an action
    static: seq, an action sequence, initially empty
            state, some description of the current world state
            goal, a goal, initially null
            problem, a problem formulation
    state ← Update-State(state, percept)
    if seq is empty then
        goal ← Formulate-Goal(state)
        problem ← Formulate-Problem(state, goal)
        seq ← Search(problem)
        action ← Recommendation(seq, state)
    seq ← Remainder(seq, state)
    return action
```

MUNDO DO ASPIRADOR DE PÓ





**ESPAÇO DE
ESTADOS**

QUAL SEU TAMANHO?

**A COMPLEXIDADE DO PROBLEMA
ESTÁ RELACIONADA AO TAMANHO?**

ESPAÇO DE ESTADOS VISTO COMO UMA ÁRVORE N-ÁRIA

```
function Tree-Search(problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains a goal state then
            return the corresponding solution
        else expand the node and add the resulting nodes to the search tree
    end
```

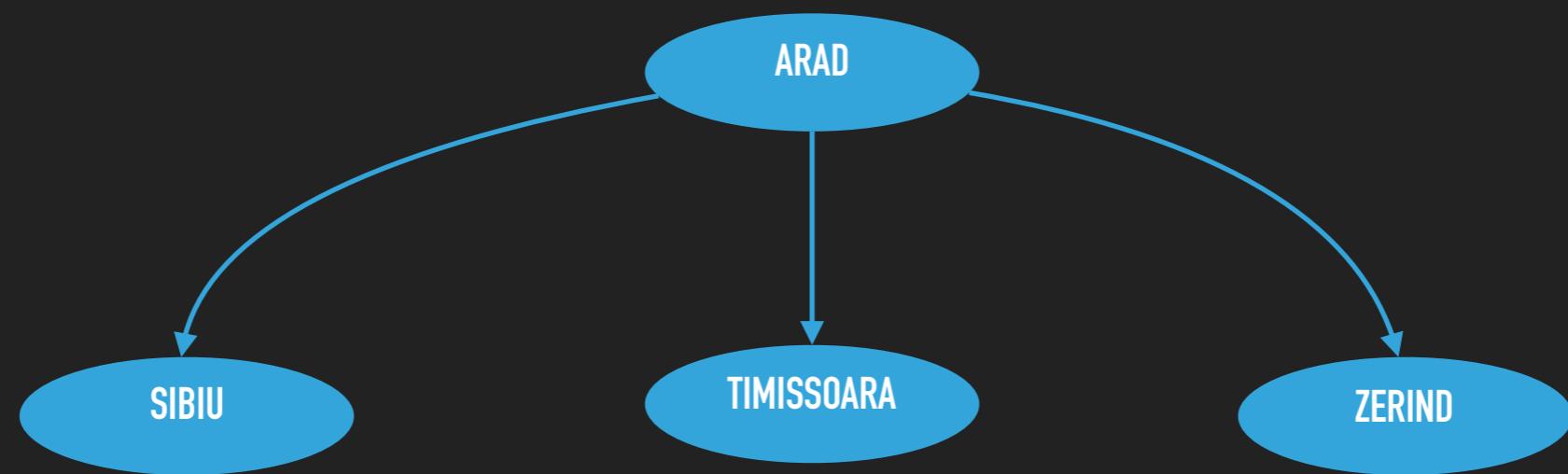
ÁRVORE DE BUSCA PARA O PROBLEMA DAS FÉRIAS NA ROMÊNIA

ÁRVORE DE BUSCA PARA O PROBLEMA DAS FÉRIAS NA ROMÊNIA

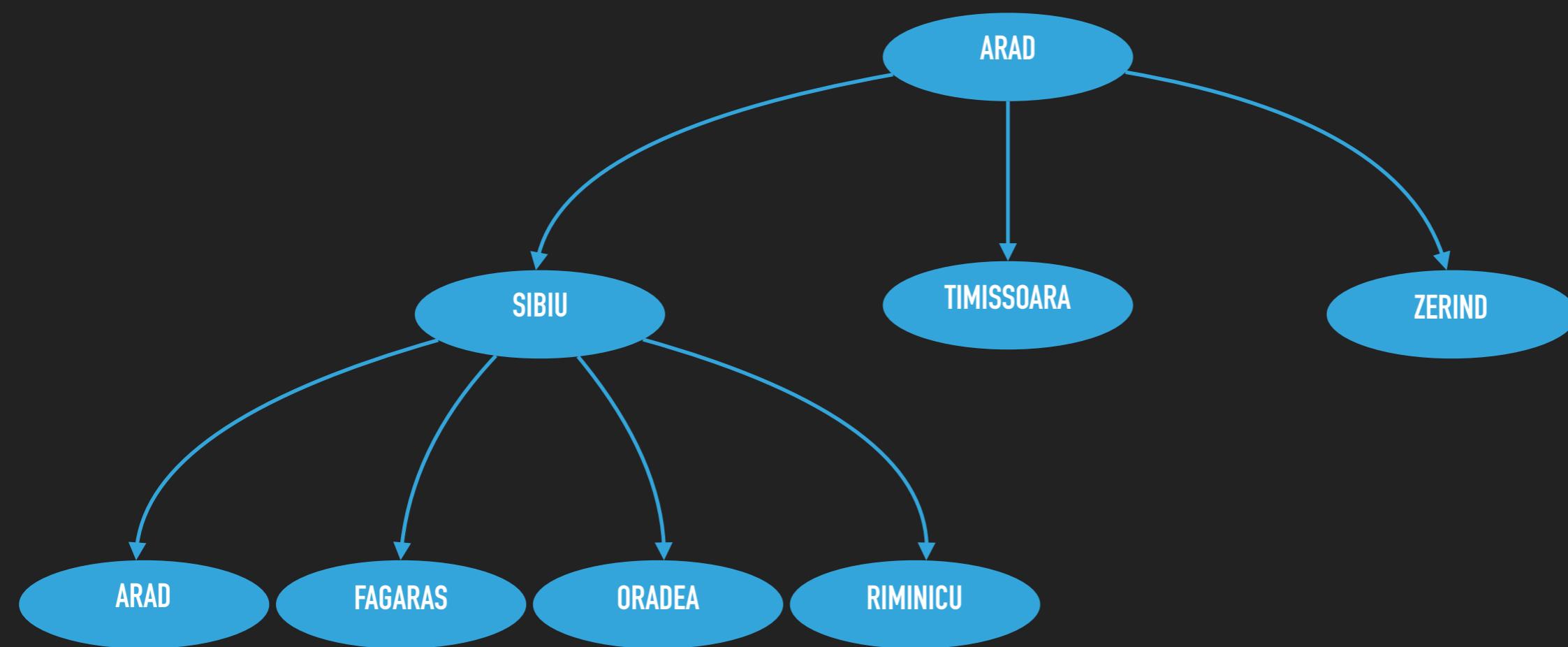


ARAD

ÁRVORE DE BUSCA PARA O PROBLEMA DAS FÉRIAS NA ROMÊNIA



ÁRVORE DE BUSCA PARA O PROBLEMA DAS FÉRIAS NA ROMÊNIA



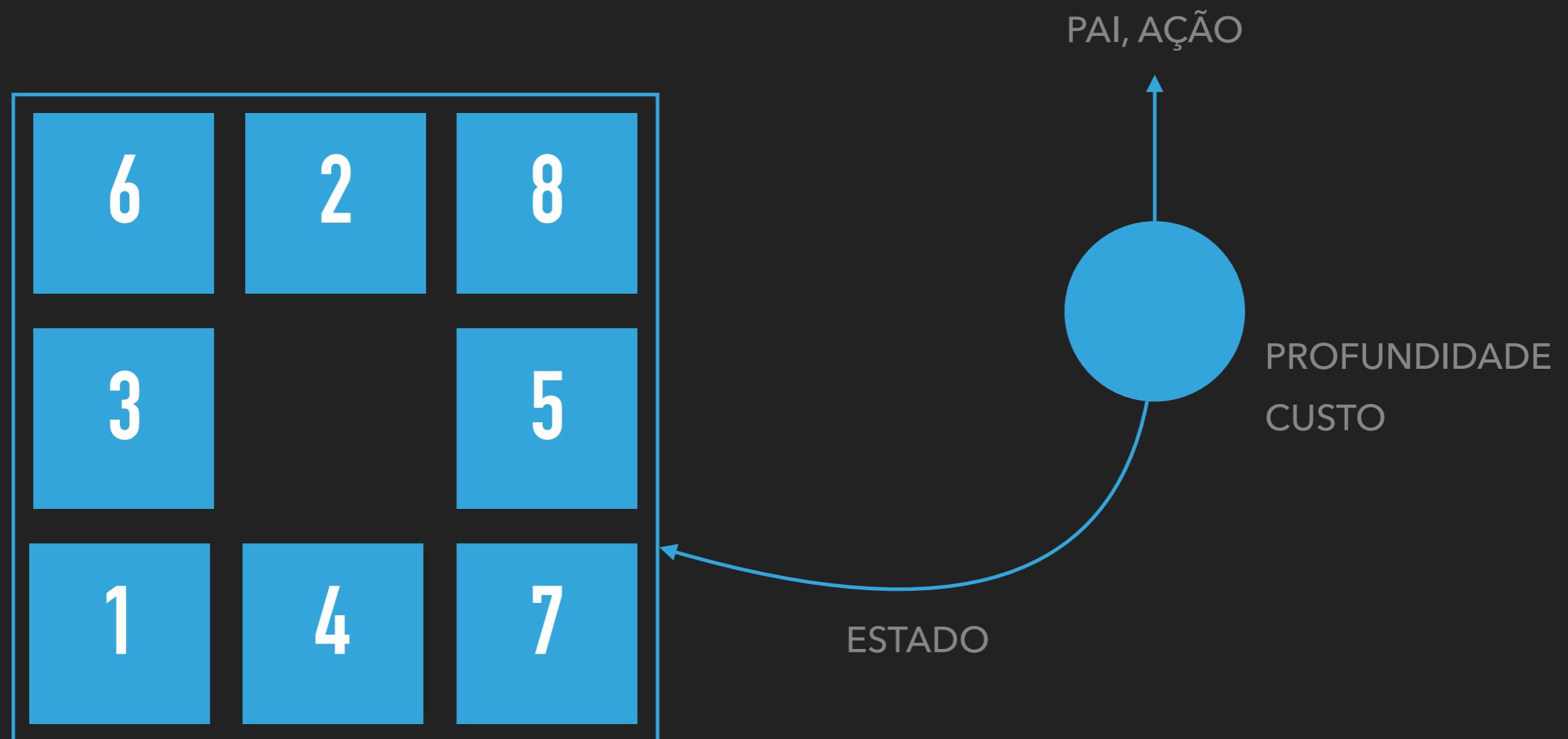
ESTADOS X NÓS

- ▶ **Estados** são representações (abstrações) de situações físicas
- ▶ **Nós** são estruturas de dados que fazem parte da árvore de busca.
- ▶ Inclui: nó pai, nós-filhos, profundidade, custo do trajeto

ESTADOS X NÓS

ESTADOS X NÓS

ESTADOS X NÓS



ALGORITMO GERAL DE BUSCA EM ESPAÇO DE ESTADOS

```

function Tree-Search(problem, fringe) returns a solution, or failure
    fringe  $\leftarrow$  Insert(Make-Node(Initial-State[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node  $\leftarrow$  Remove-Front(fringe)
        if Goal-Test(problem, State(node)) then return node
        fringe  $\leftarrow$  InsertAll(Expand(node, problem), fringe)

```

```

function Expand(node, problem) returns a set of nodes
    successors  $\leftarrow$  the empty set
    for each action, result in Successor-FN(problem, State[node]) do
        s  $\leftarrow$  a new Node
        Parent-Node[s]  $\leftarrow$  node
        Action[s]  $\leftarrow$  action
        State[s]  $\leftarrow$  result
        Path-Cost[s]  $\leftarrow$  Path-Cost[node] + Step-Cost(node, action, s)
        Depth[s]  $\leftarrow$  Depth[node] + 1
        add s to successors
    return successors

```

```
function Tree-Search(problem, fringe) returns a solution, or failure
    fringe  $\leftarrow$  Insert(Make-Node(Initial-State[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node  $\leftarrow$  Remove-Front(fringe)
        if Goal-Test(problem, State(node)) then return node
        fringe  $\leftarrow$  InsertAll(Expand(node, problem), fringe)
```

QUAL A ESTRATÉGIA ADOTADA PARA A EXPANSÃO DO NÓ ATUAL?

```
function Expand(node, problem, successors)
    for each action, result in Successors-In(problem, state[node]), do
        s  $\leftarrow$  a new Node
        Parent-Node[s]  $\leftarrow$  node
        Action[s]  $\leftarrow$  action
        State[s]  $\leftarrow$  result
        Path-Cost[s]  $\leftarrow$  Path-Cost[node] + Step-Cost(node, action, s)
        Depth[s]  $\leftarrow$  Depth[node] + 1
        add s to successors
    return successors
```

ESTRATÉGIAS

- ▶ Considerações:
 - ▶ **Completude** - a solução é encontrada se a mesma existir.
 - ▶ **Complexidade de tempo** - número de nós gerados/expandidos para encontrar a solução.
 - ▶ **Complexidade espacial** - número de nós máximo em memória.
 - ▶ **Otimalidade** - encontra a solução com menor custo.

ESTRATÉGIAS

► ATENÇÃO:

B - FATOR DA RAMIFICAÇÃO DA ÁRVORE (NÚMERO MÁXIMO DE SUCESSORES DE QUALQUER NÓ)

D - PROFUNDIDADE MÁXIMA DO MENOR CUSTO

M - PROFUNDIDADE MÁXIMA ABSOLUTA

memória.

► Otimalidade - encontra a solução com menor custo.

ESTRATÉGIAS - ABORDAGENS

BUSCA SEM
INFORMAÇÃO

BUSCA COM
INFORMAÇÃO

ESTRATÉGIAS - ABORDAGENS

BUSCA SEM
INFORMAÇÃO

BUSCA COM
INFORMAÇÃO



ESTRATÉGIAS - ABORDAGENS

BUSCA SEM
INFORMAÇÃO



BUSCA COM
INFORMAÇÃO



Lucas Duggio Figueira [@lucasfigueira]

BUSCA SEM INFORMAÇÃO

- ▶ Também chamada de busca cega, por quê?
- ▶ Usa apenas o conhecimento contido na definição do problema.
- ▶ Não assume ou infere conhecimento a partir das informações obtidas.
- ▶ Tenta "varrer" todo o espaço de estados. Abordagem tátil.

**BUSCA EM
LARGURA**

BUSCA EM LARGURA

Expande o nó raíz, e em sequência avalia e expande cada um de seus sucessores.

BUSCA EM LARGURA

- ▶ Expandir o mais raso nó não expandido.
- ▶ **Fringe** nada mais é do que uma:

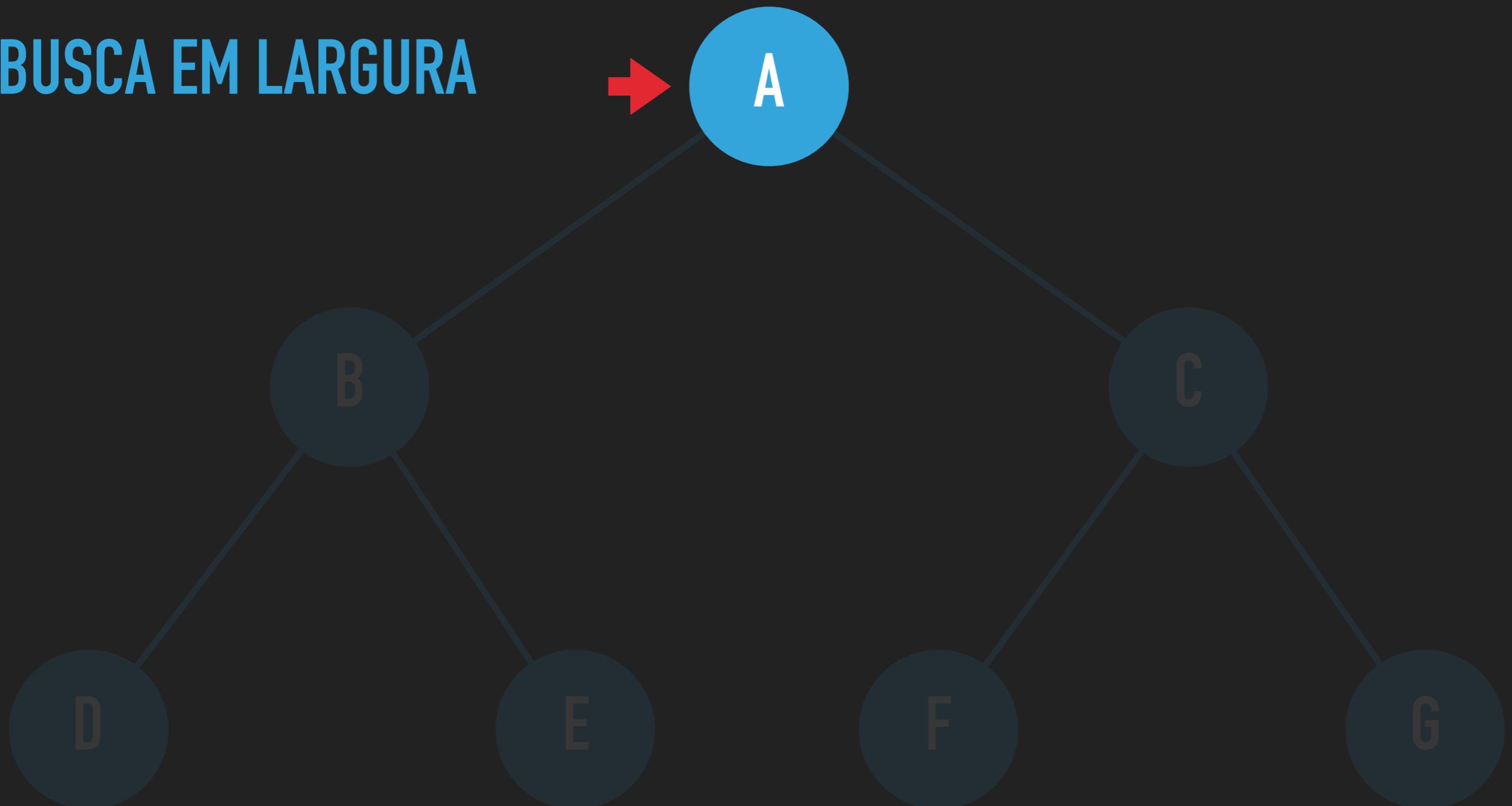
BUSCA EM LARGURA

- ▶ Expandir o mais raso nó não expandido.
- ▶ Fringe nada mais é do que uma:



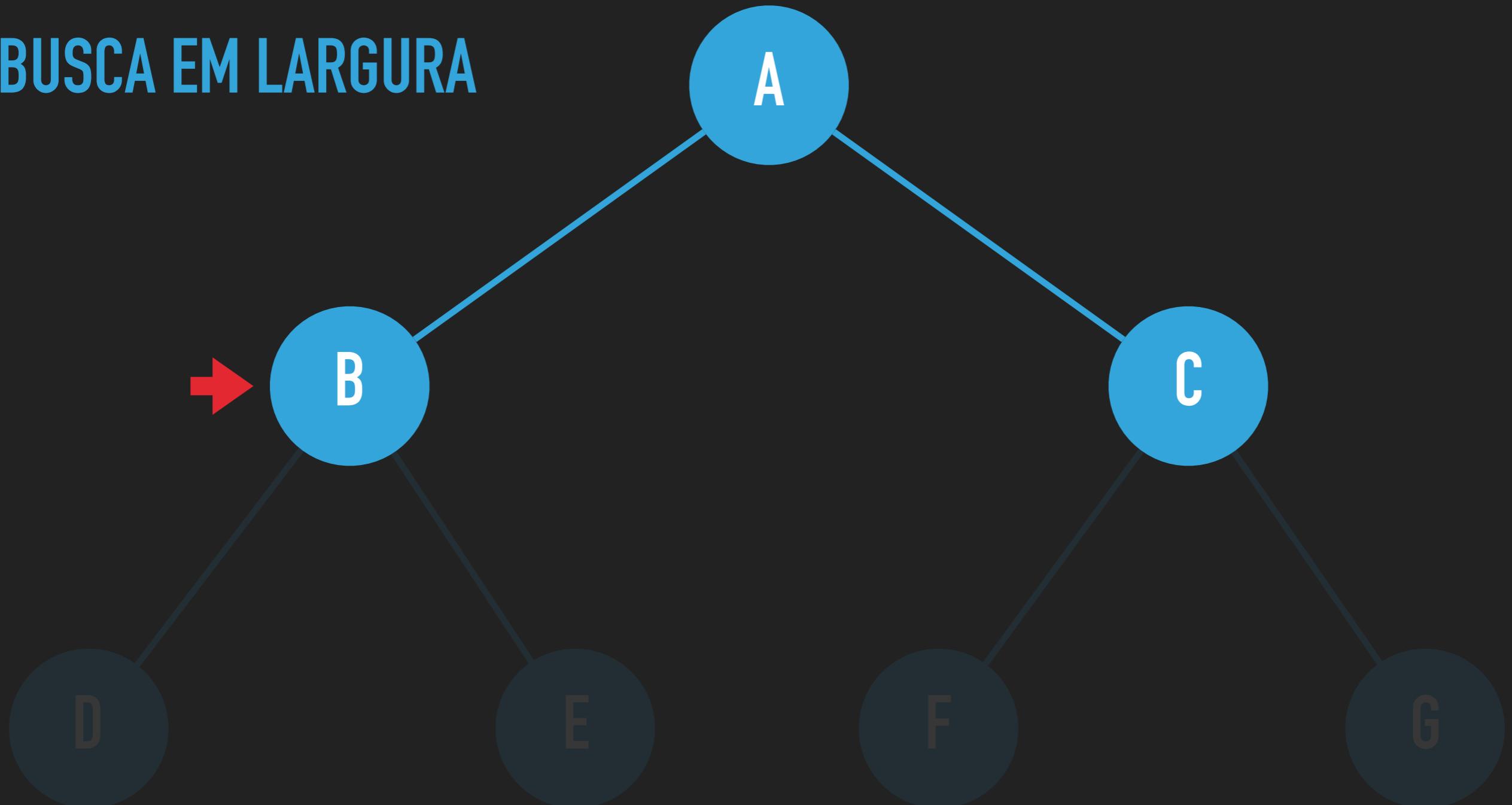
FIFO

BUSCA EM LARGURA



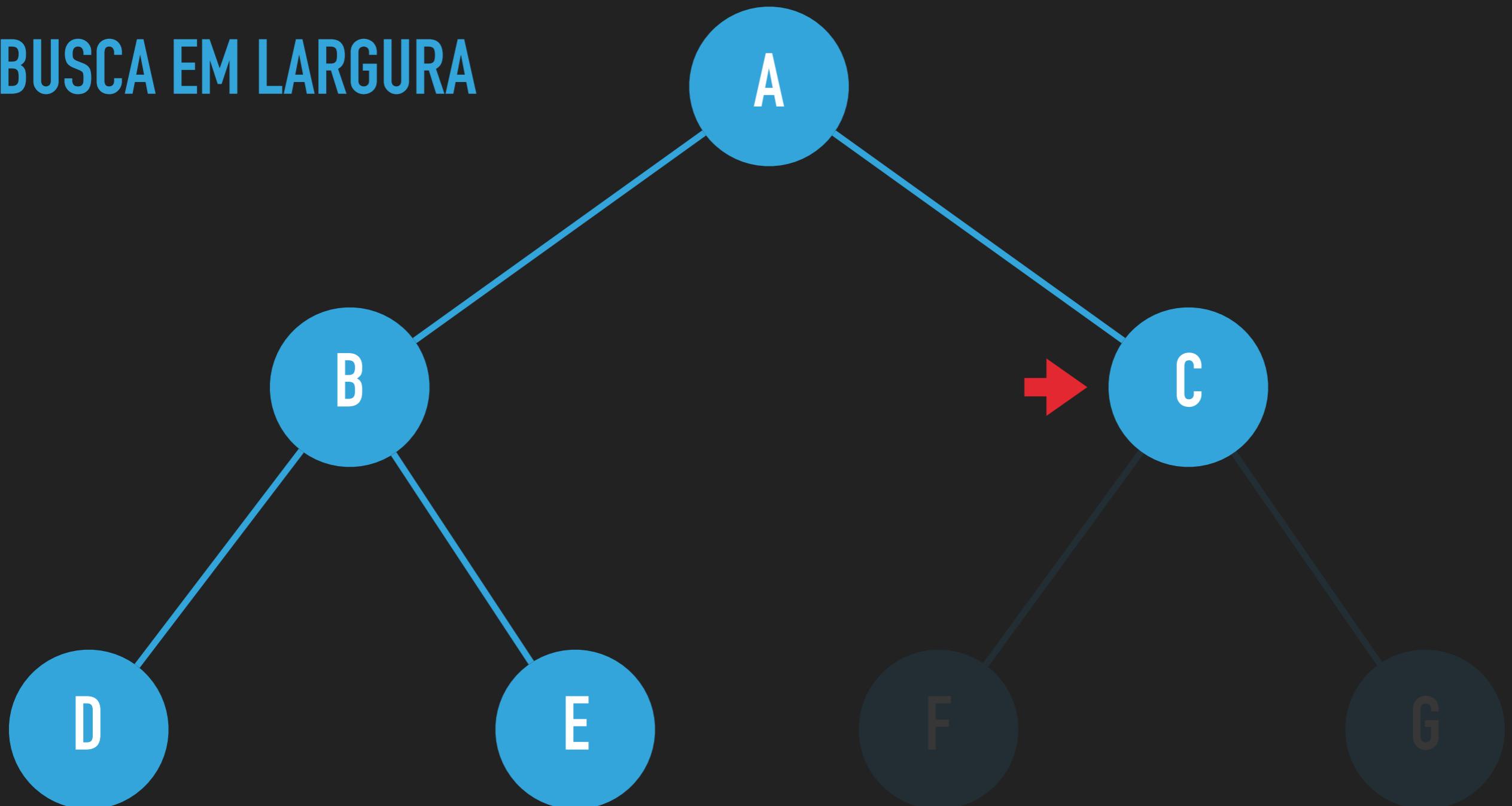
FRINGE: [A]

BUSCA EM LARGURA



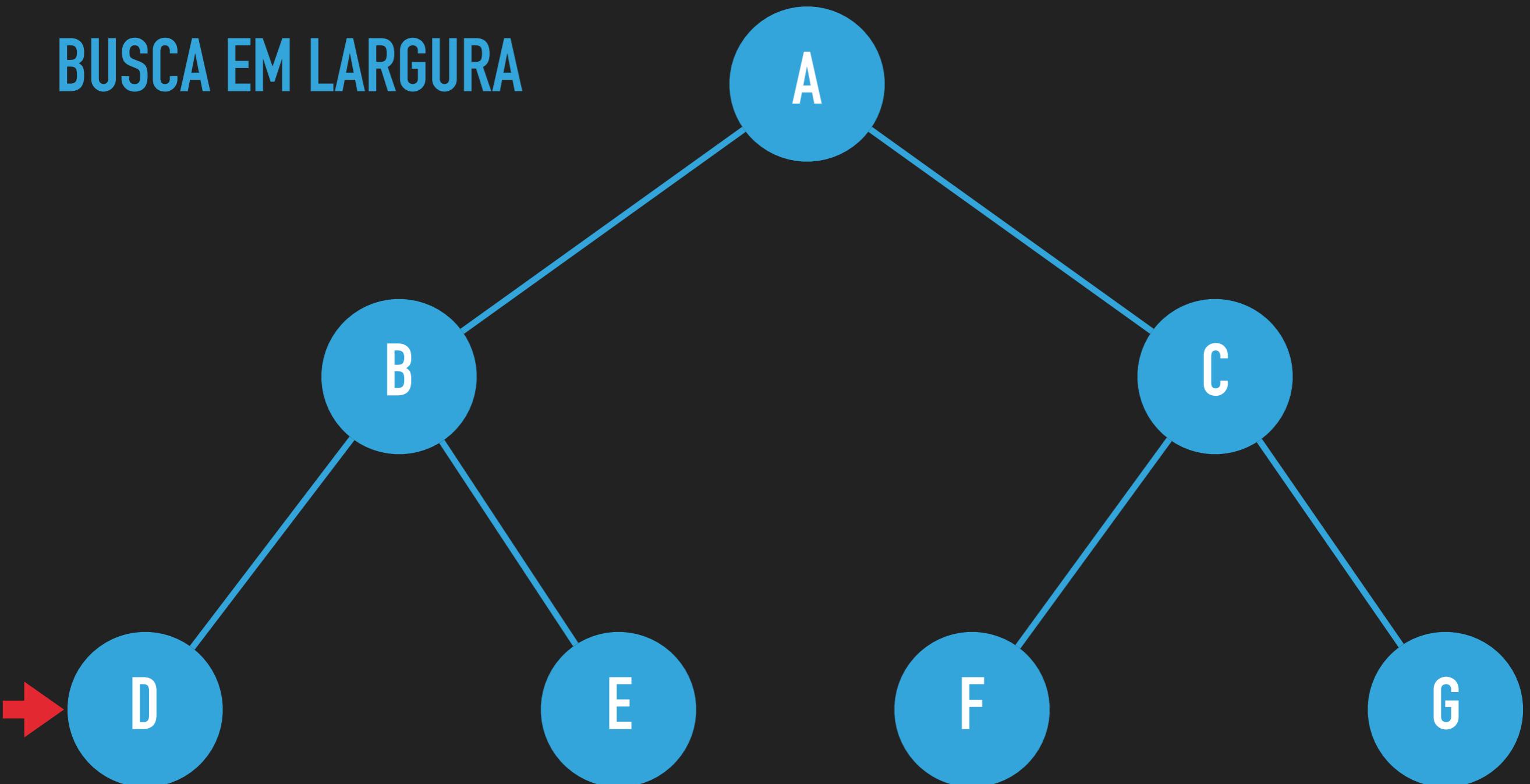
FRINGE: [B, C]

BUSCA EM LARGURA



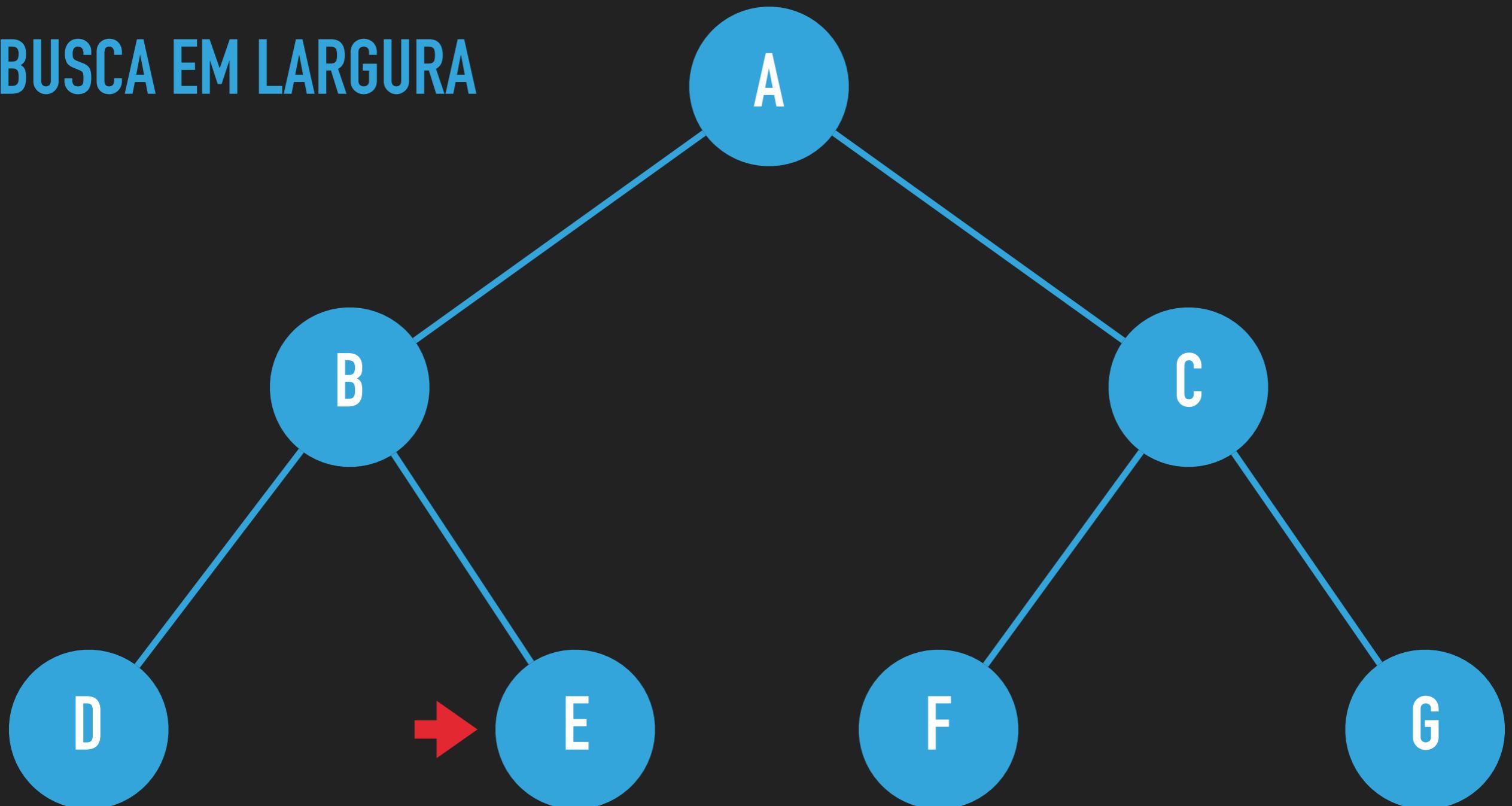
FRINGE: [C, D, E]

BUSCA EM LARGURA



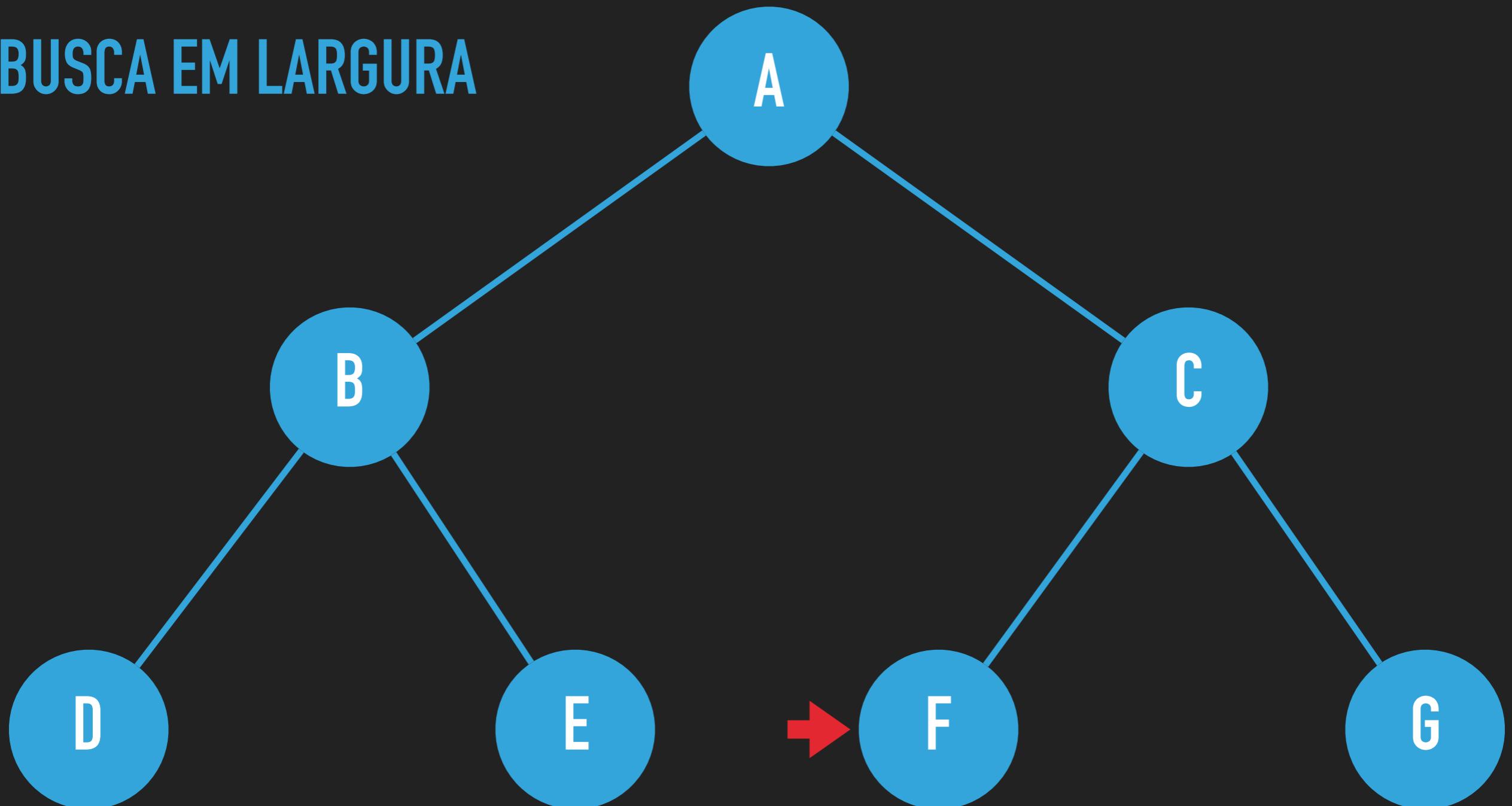
FRINGE: [D, E, F, G]

BUSCA EM LARGURA



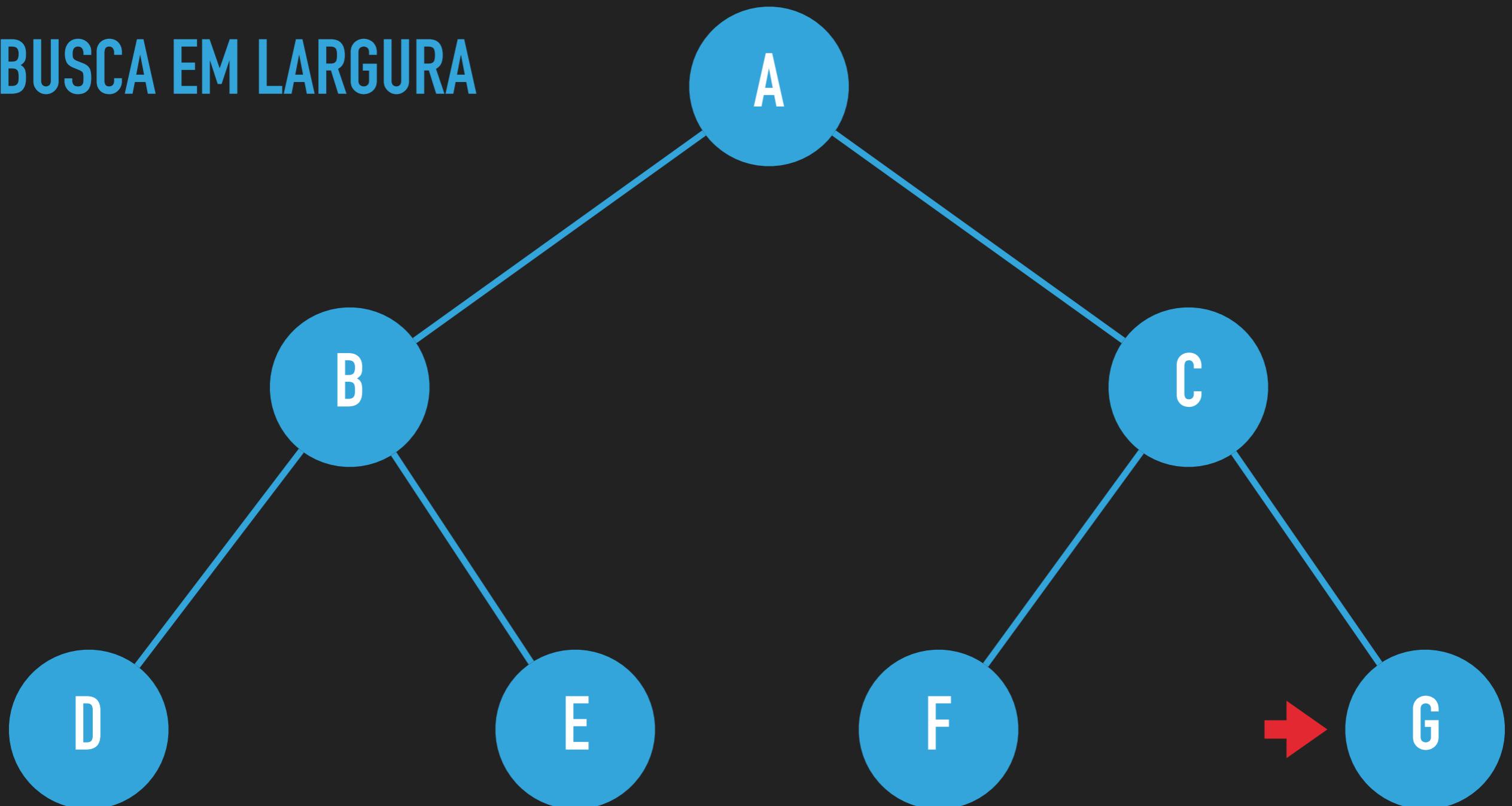
FRINGE: [E, F, G]

BUSCA EM LARGURA



FRINGE: [F, G]

BUSCA EM LARGURA



FRINGE: [G]

PROPRIEDADES DA BUSCA EM LARGURA

PROPRIEDADES DA BUSCA EM LARGURA

- ▶ Completo?

PROPRIEDADES DA BUSCA EM LARGURA

- ▶ Completo?
- ▶ Sim, se b for finito.

PROPRIEDADES DA BUSCA EM LARGURA

- ▶ Completo?
- ▶ Sim, se b for finito.
- ▶ Tempo?

PROPRIEDADES DA BUSCA EM LARGURA

- ▶ Completo?
 - ▶ Sim, se b for finito.
- ▶ Tempo?
 - ▶ $O(b^{d+1})$

PROPRIEDADES DA BUSCA EM LARGURA

- ▶ Completo?
 - ▶ Sim, se b for finito.
- ▶ Tempo?
 - ▶ $O(b^{d+1})$
- ▶ Espaço?

PROPRIEDADES DA BUSCA EM LARGURA

- ▶ Completo?
 - ▶ Sim, se b for finito.
- ▶ Tempo?
 - ▶ $O(b^{d+1})$
- ▶ Espaço?
 - ▶ $O(b^{d+1})$

PROPRIEDADES DA BUSCA EM LARGURA

- ▶ Completo?
 - ▶ Sim, se b for finito.
- ▶ Tempo?
 - ▶ $O(b^{d+1})$
- ▶ Espaço?
 - ▶ $O(b^{d+1})$
- ▶ Ótimo?

PROPRIEDADES DA BUSCA EM LARGURA

- ▶ Completo?
 - ▶ Sim, se b for finito.
- ▶ Tempo?
 - ▶ $O(b^{d+1})$
- ▶ Espaço?
 - ▶ $O(b^{d+1})$
- ▶ Ótimo?
 - ▶ Não

PROPRIEDADES DA BUSCA EM LARGURA

- ▶ Completo?
 - ▶ Sim, se b for finito.
 - ▶ Tempo?
 - ▶ $O(b^{d+1})$
 - ▶ Espaço?
 - ▶ $O(b^{d+1})$
 - ▶ Ótimo?
 - ▶ Não

ESPAÇO



BUSCA EM PROFOUNDIDADE

BUSCA EM PROFOUNDIDADE

Expande o nós raíz, e expande seu sucessores, e cada um de seus sucessores subsequentes.

BUSCA EM PROFUNDIDADE

- ▶ Expande o nó mais profundo não expandido
- ▶ Fringe nada mais é do que uma

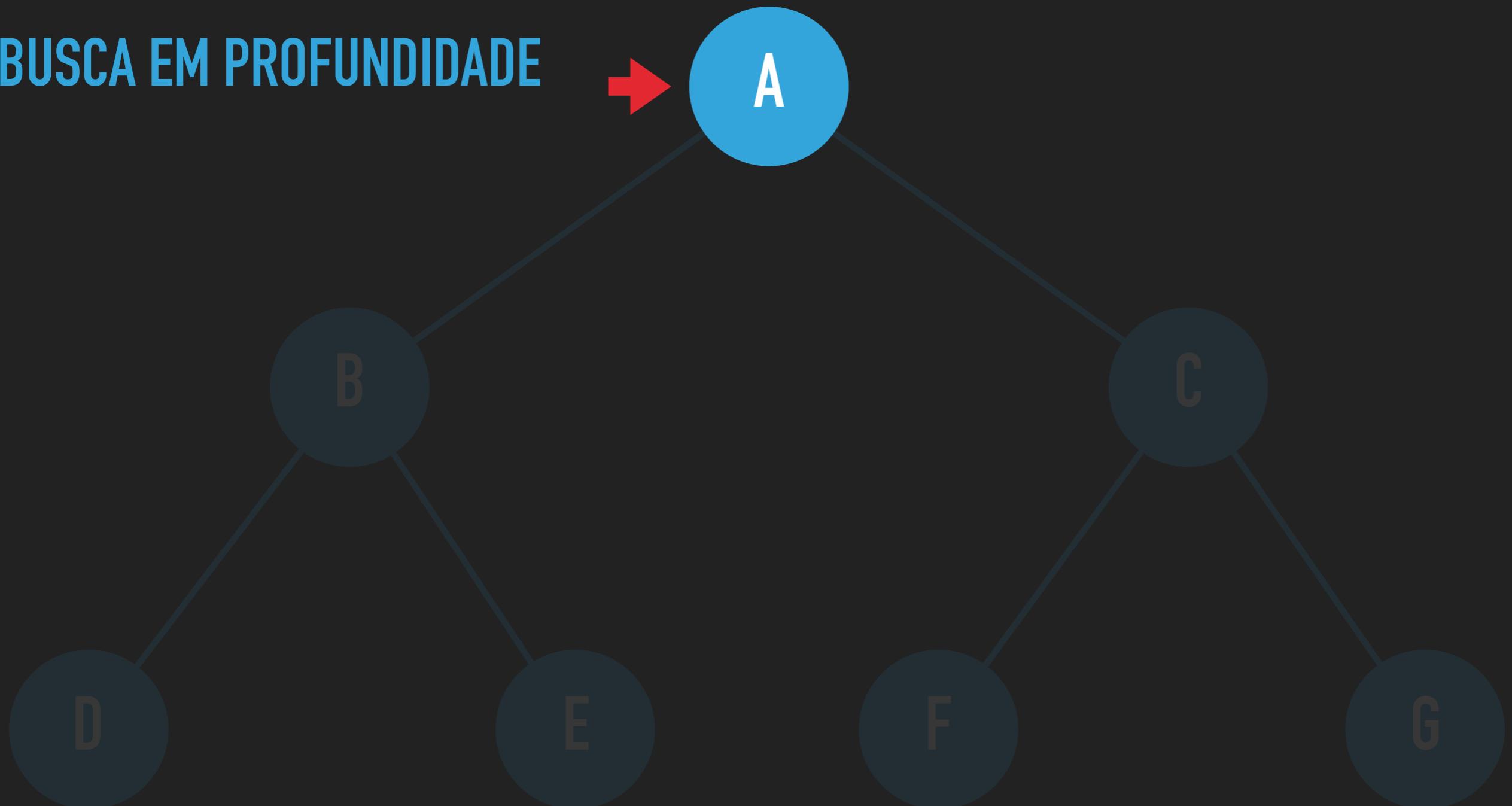
BUSCA EM PROFUNDIDADE

- ▶ Expande o nó mais profundo não expandido
- ▶ Fringe nada mais é do que uma



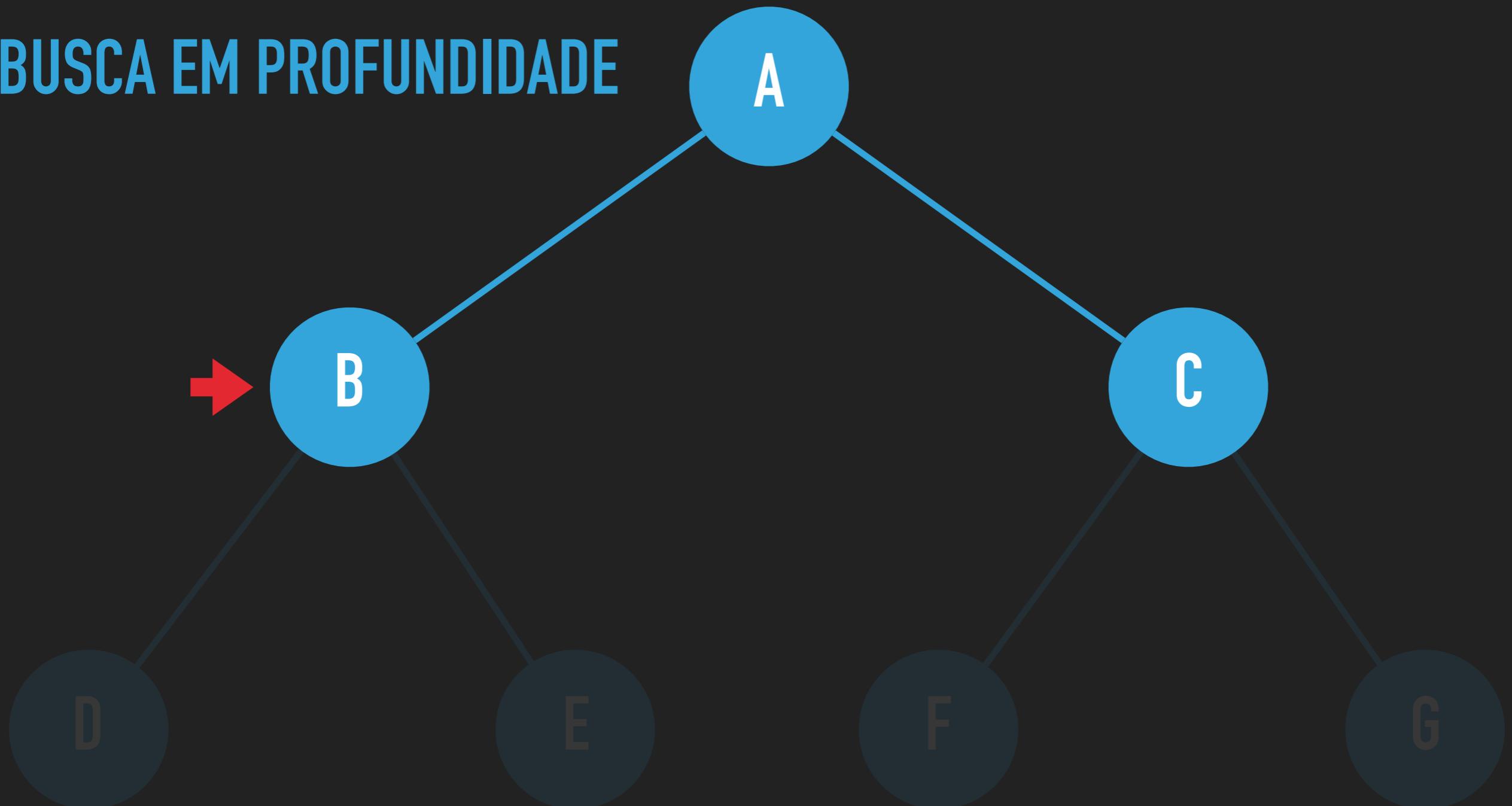
LIFO

BUSCA EM PROFUNDIDADE



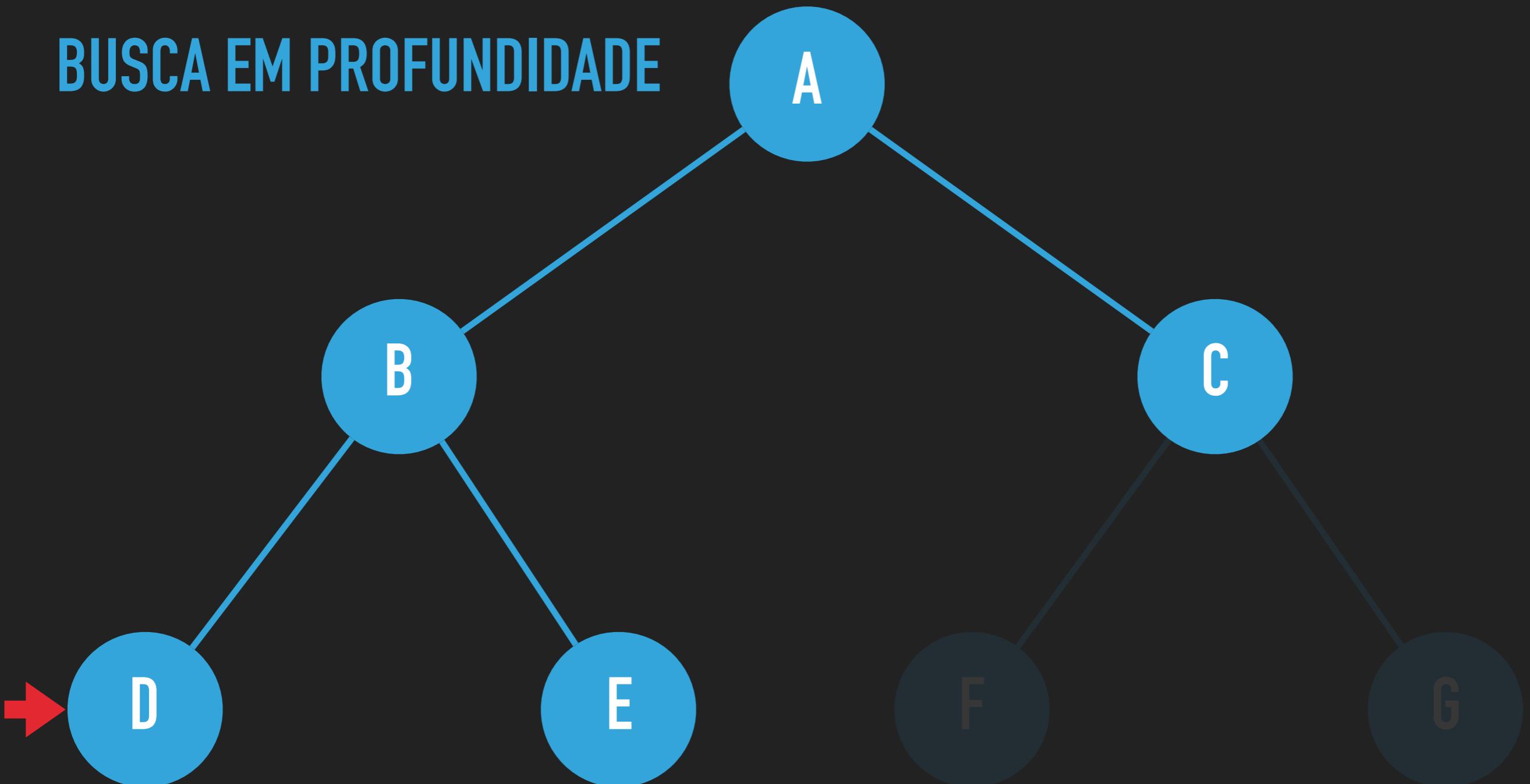
FRINGE: [A]

BUSCA EM PROFUNDIDADE



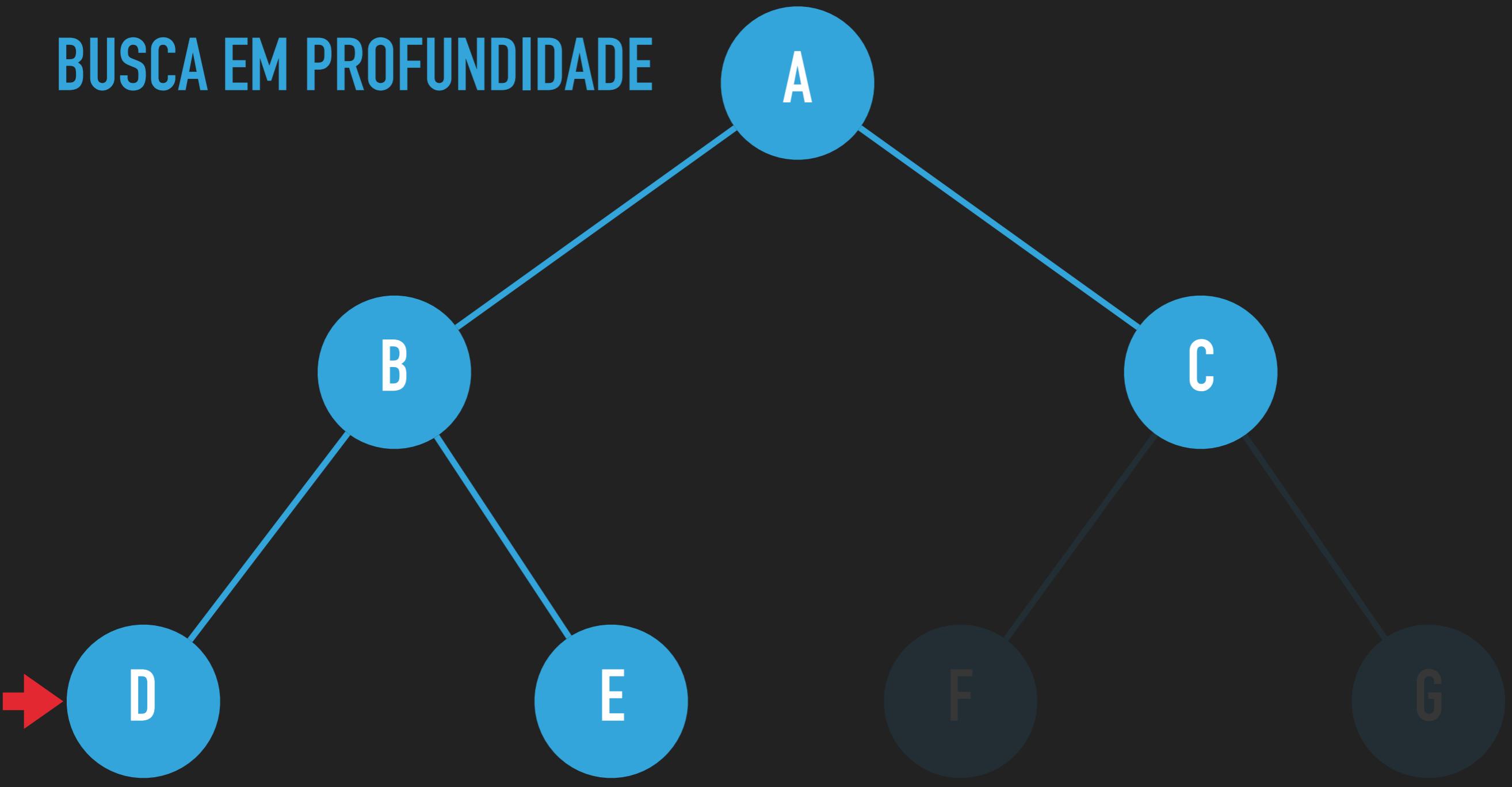
FRINGE: [B, C]

BUSCA EM PROFUNDIDADE



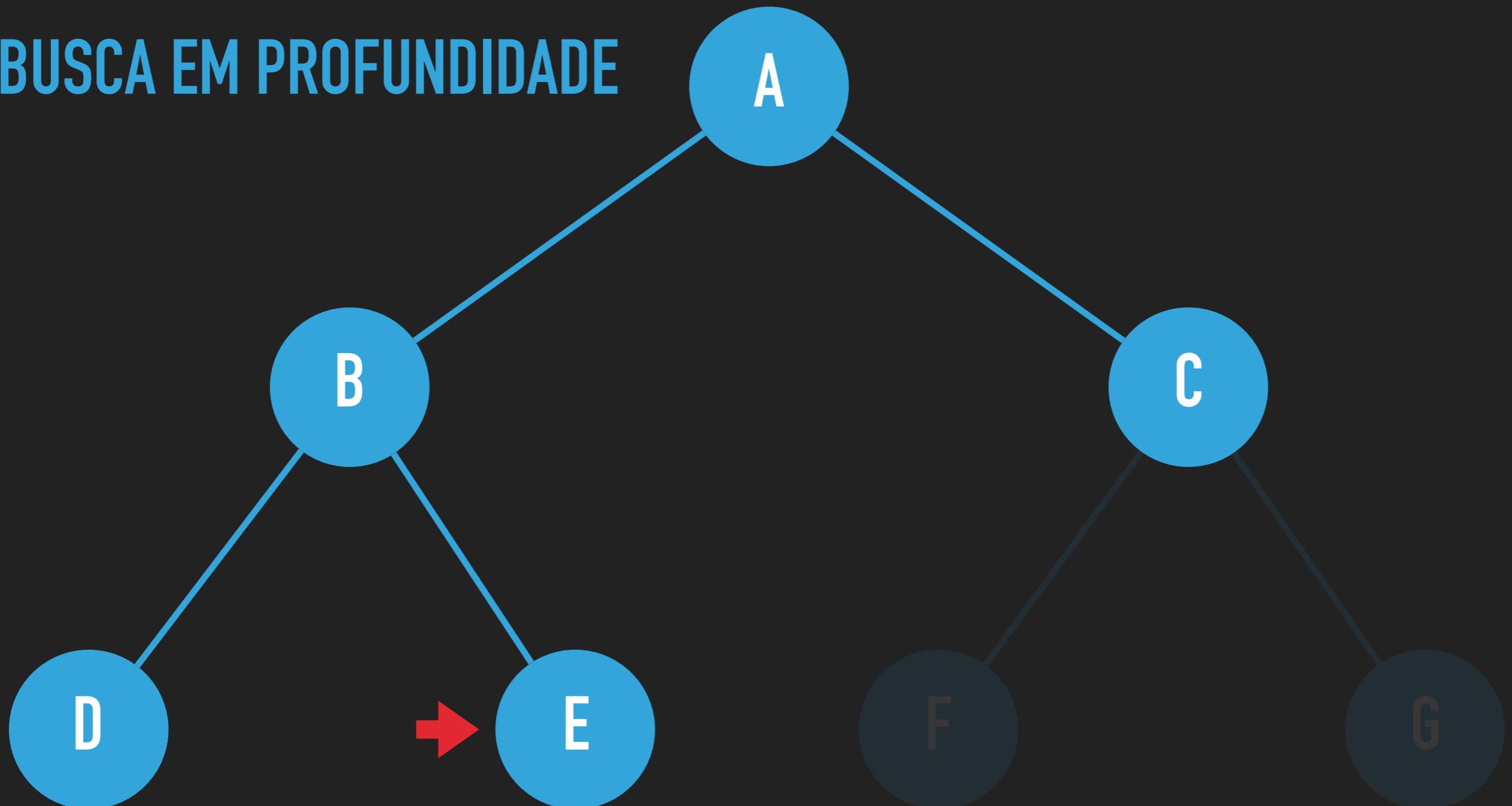
FRINGE: [D, E, C]

BUSCA EM PROFUNDIDADE



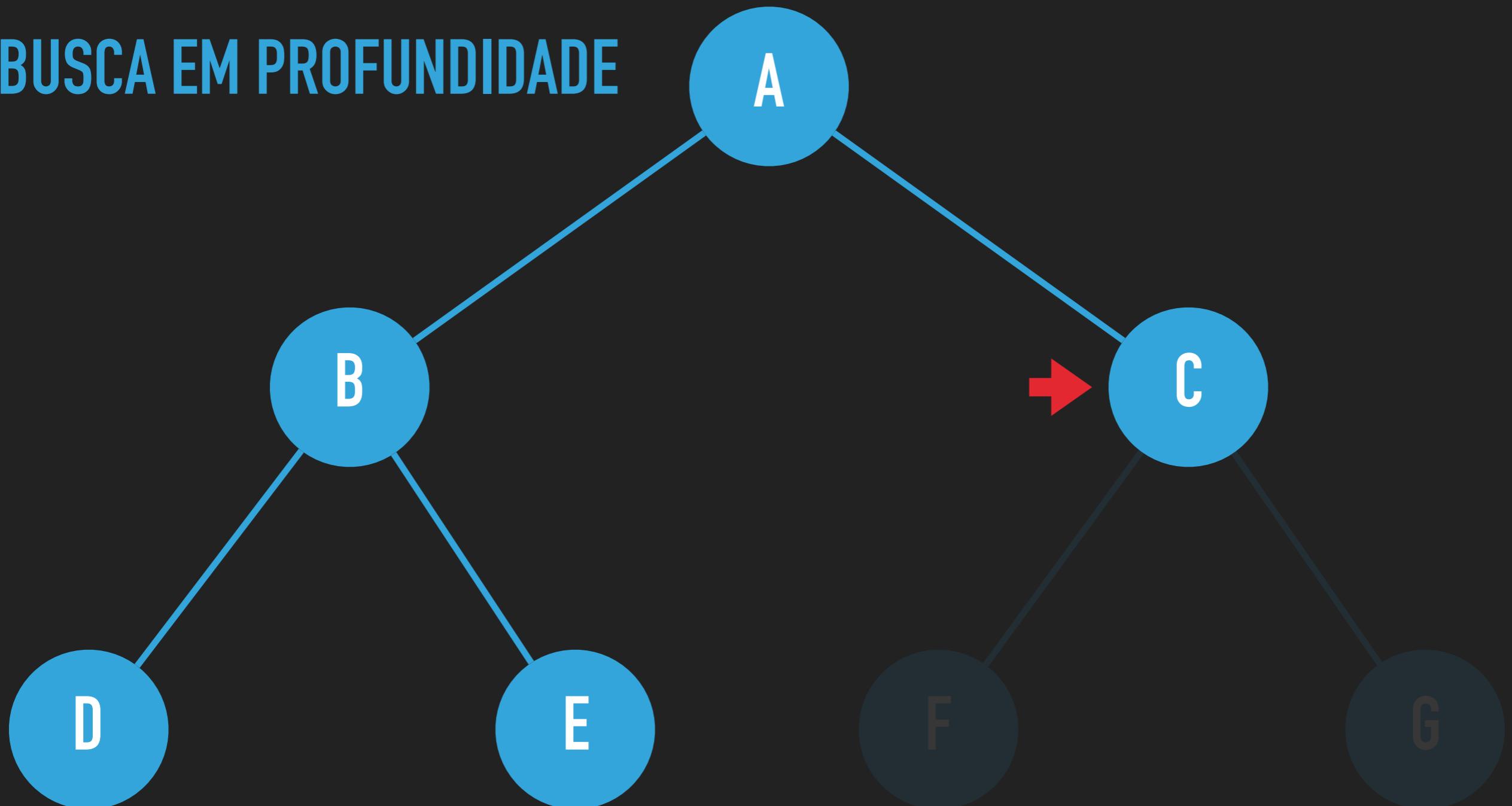
FRINGE: [D, E, C]

BUSCA EM PROFUNDIDADE



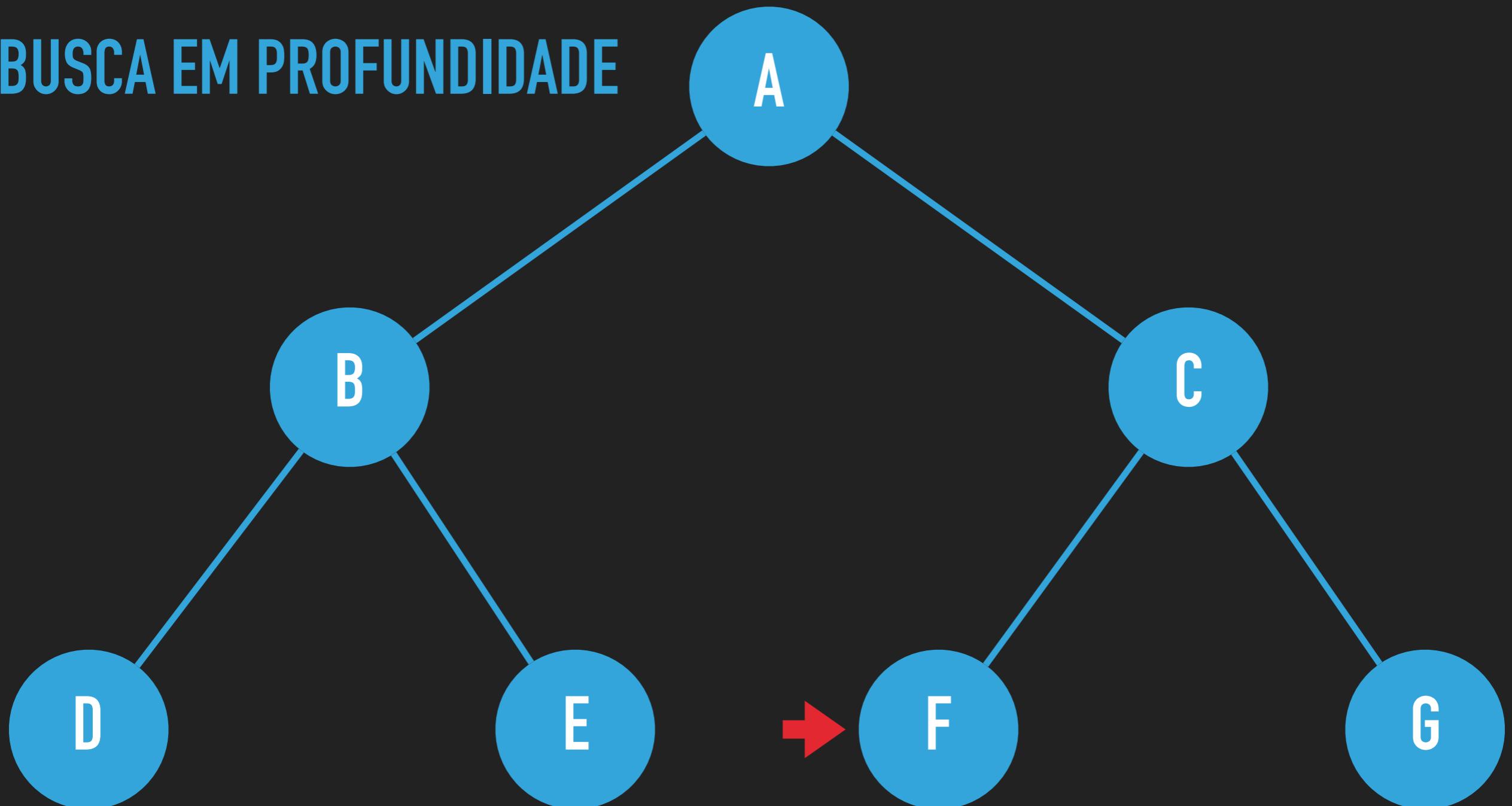
FRINGE: [E, C]

BUSCA EM PROFUNDIDADE



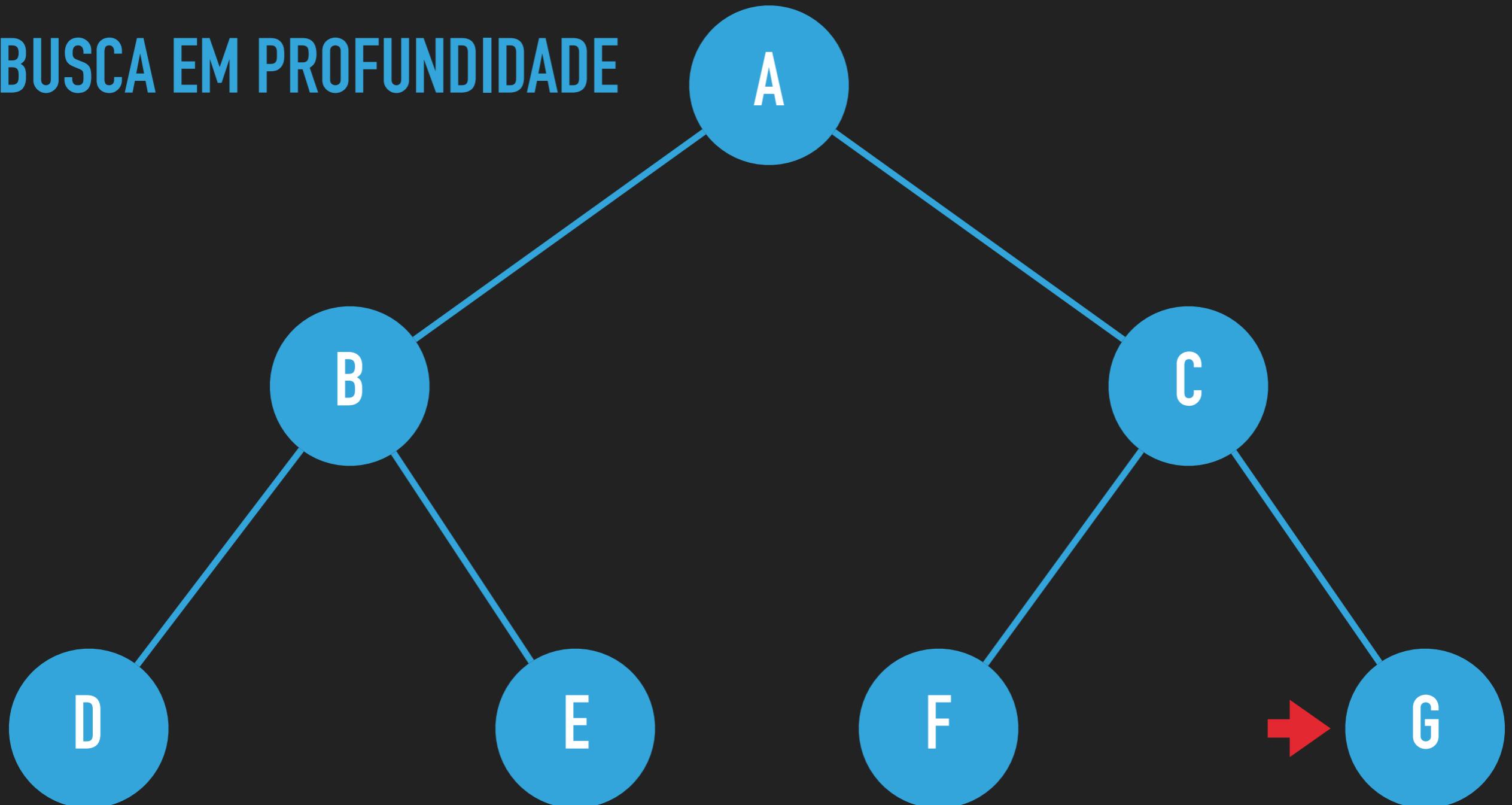
FRINGE: [C]

BUSCA EM PROFUNDIDADE



FRINGE: [F, G]

BUSCA EM PROFUNDIDADE



FRINGE: [G]

PROPRIEDADES DA BUSCA EM PROFUNDIDADE

PROPRIEDADES DA BUSCA EM PROFUNDIDADE

- ▶ Completo?

PROPRIEDADES DA BUSCA EM PROFUNDIDADE

- ▶ Completo?
- ▶ Não, falha em espaços infinitos e com loops

PROPRIEDADES DA BUSCA EM PROFUNDIDADE

- ▶ Completo?
- ▶ Não, falha em espaços infinitos e com loops
- ▶ Tempo?

PROPRIEDADES DA BUSCA EM PROFUNDIDADE

- ▶ Completo?
 - ▶ Não, falha em espaços infinitos e com loops
- ▶ Tempo?
 - ▶ $O(b^m)$, muito pior se $m \gg b$

PROPRIEDADES DA BUSCA EM PROFUNDIDADE

- ▶ Completo?
 - ▶ Não, falha em espaços infinitos e com loops
- ▶ Tempo?
 - ▶ $O(b^m)$, muito pior se $m \gg b$
- ▶ Espaço?

PROPRIEDADES DA BUSCA EM PROFUNDIDADE

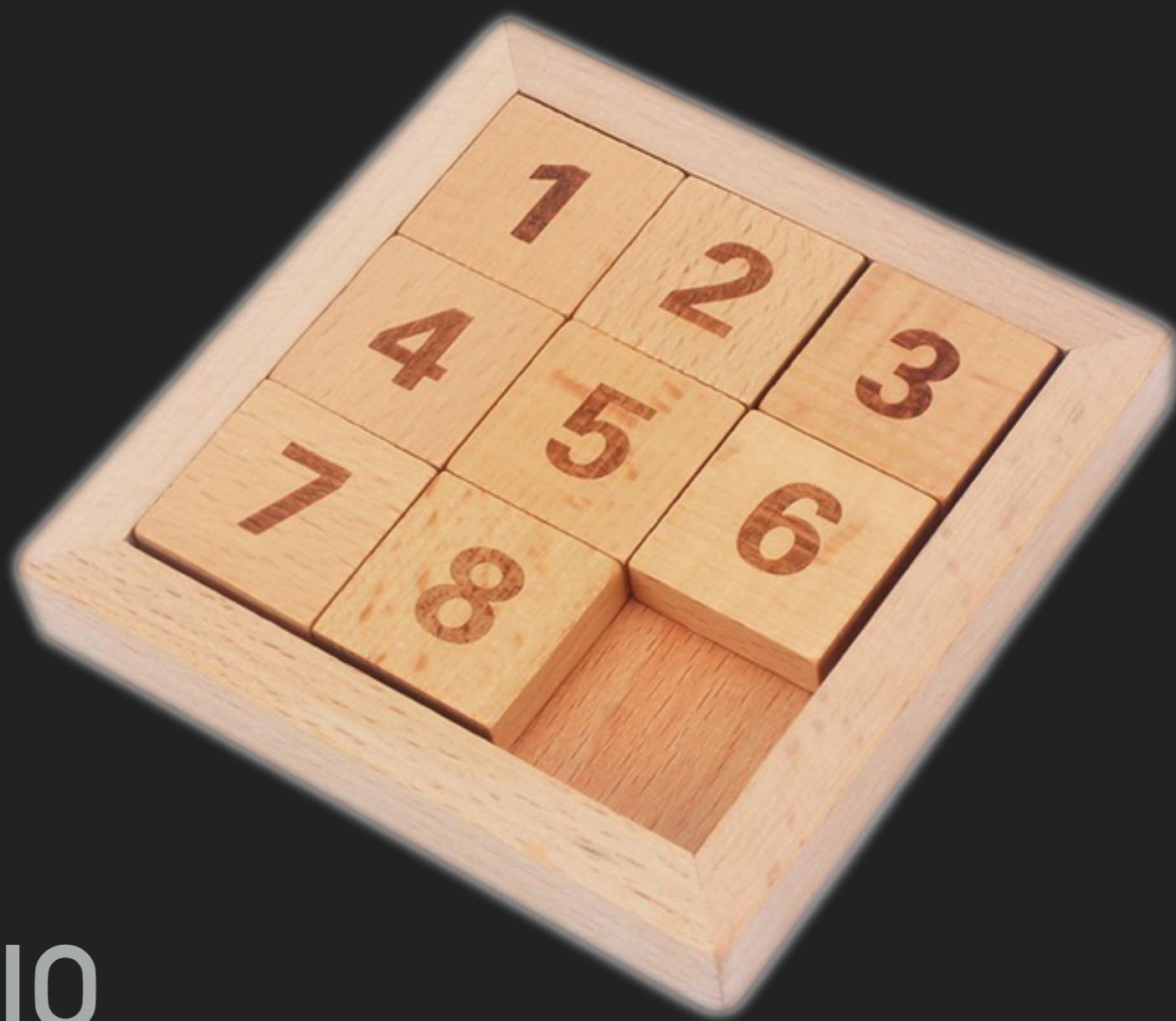
- ▶ Completo?
 - ▶ Não, falha em espaços infinitos e com loops
- ▶ Tempo?
 - ▶ $O(b^m)$, muito pior se $m \gg b$
- ▶ Espaço?
 - ▶ $O(bm)$, linear

PROPRIEDADES DA BUSCA EM PROFUNDIDADE

- ▶ Completo?
 - ▶ Não, falha em espaços infinitos e com loops
- ▶ Tempo?
 - ▶ $O(b^m)$, muito pior se $m \gg b$
- ▶ Espaço?
 - ▶ $O(bm)$, linear
- ▶ Ótimo?

PROPRIEDADES DA BUSCA EM PROFUNDIDADE

- ▶ Completo?
 - ▶ Não, falha em espaços infinitos e com loops
- ▶ Tempo?
 - ▶ $O(b^m)$, muito pior se $m \gg b$
- ▶ Espaço?
 - ▶ $O(bm)$, linear
- ▶ Ótimo?
 - ▶ Não



EXERCÍCIO

IMPLEMENTAR O 8-PUZZLE USANDO BUSCA SEM INFORMAÇÃO



IA - BUSCA EM
ESPAÇO DE ESTADOS

BUSCA COM
INFORMAÇÃO

BUSCA COM INFORMAÇÃO

- ▶ Usa um conhecimento sobre o nó atual.
- ▶ Para cada nó é possível estimar o quão perto ele está da solução.
- ▶ Heurística.



REVISÃO: BUSCA EM ÁRVORE

```
function Tree-Search(problem, fringe) returns a solution, or failure
    fringe ← Insert(Make-Node(Initial-State[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← Remove-Front(fringe)
        if Goal-Test(problem, State(node)) then return node
        fringe ← InsertAll(Expand(node, problem), fringe)
```

A estratégia envolve a política de *Remove-Front* e *InsertAll* na lista *fringe*

HEURÍSTICA

Lucas Baggio Figueira [*@lucasfigueira*]

HEURÍSTICA

Quantificação de proximidade a um determinado objetivo.

Uma boa heurística indica que o objeto de avaliação está muito próximo do objetivo;
Uma má heurística indica que o objeto avaliado estiver muito longe do objetivo.

É uma medida, mas uma medida advinda de subjetividade.

PERDIDO NA MONTANHA COM SEDE, O QUE FAZER?

PERDIDO NA MONTANHA COM SEDE, O QUE FAZER?

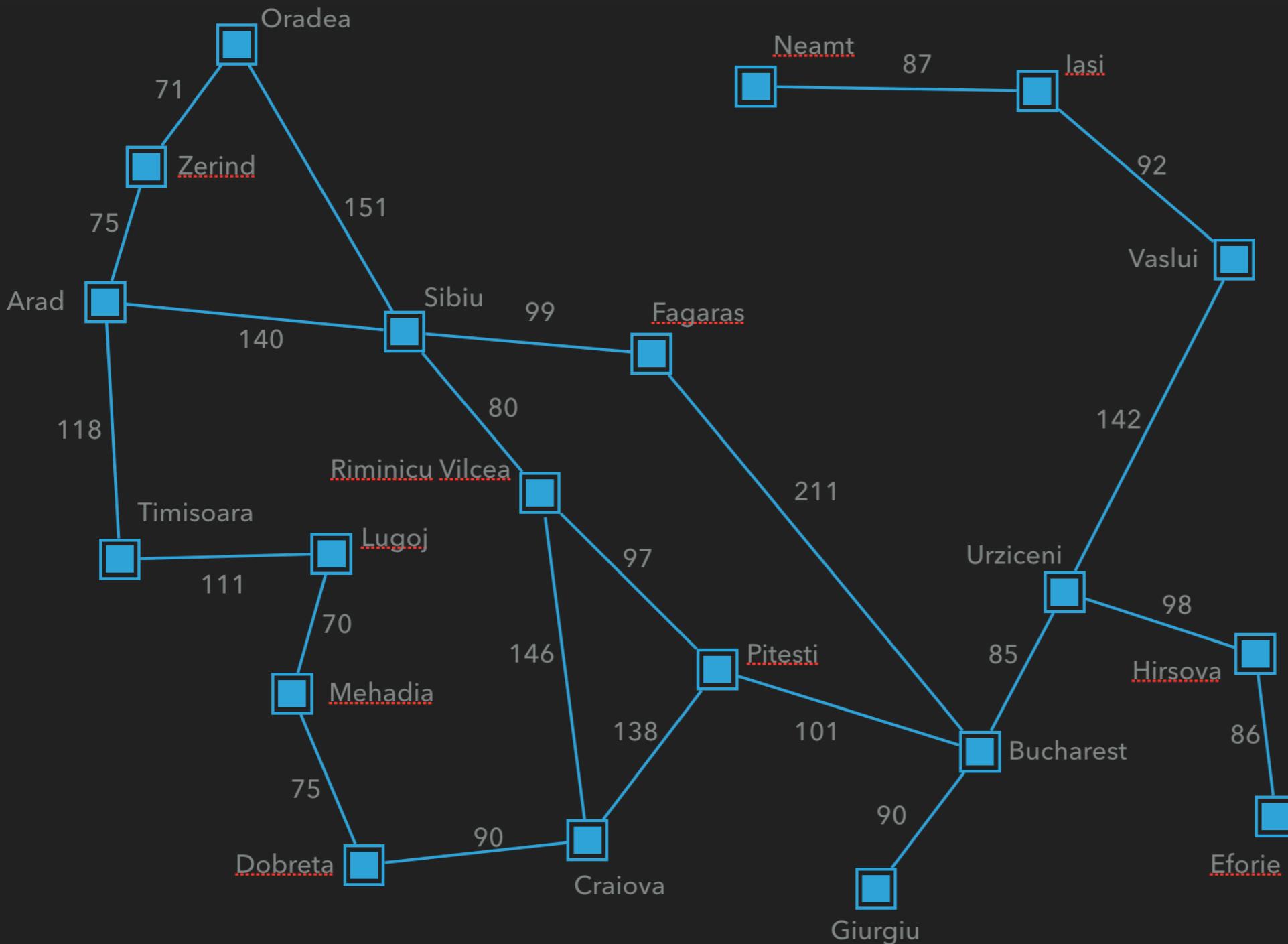


ESTRATÉGIAS BASEADAS EM HEURÍSTICAS

- ▶ Best First: organiza o “horizonte de busca” por heurística.
- ▶ Greedy Search (Busca Gulosa)
 - ▶ Usa conhecimento heurístico pontual
- ▶ A* (A-star)
 - ▶ Usa conhecimento concreto (passado) e heurístico.

Os nós de fringe são ordenados por uma medida que contempla a heurística.

COMO DESCOBRIR UM CAMINHO ENTRE ARAD E BUCARESTE?



**HEURÍSTICA: DISTÂNCIA
EM LINHA RETA ATÉ
BUCARESTE**

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
Urzecini	80
Vaslui	199
Zerind	374

BUSCA GULOSA

- ▶ Expande o nó que parece estar mais próximo do objetivo.
- ▶ Cada nó é submetido à uma função de avaliação $f(n)$.
- ▶ $f(n) = h(n)$, heurística do problema.
- ▶ Exemplo: no problema da Romênia usa a heurística dada - menor distância em linha reta de Bucareste

BUSCA GULOSA NO PROBLEMA DA ROMÊNIA

HEURÍSTICA:
DISTÂNCIA EM
LINHA RETA

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
Rimin.	193
Sibiu	253
Timisoara	329
Urzecini	80
Vaslui	199
Zerind	374

BUSCA GULOSA NO PROBLEMA DA ROMÊNIA

ARAD

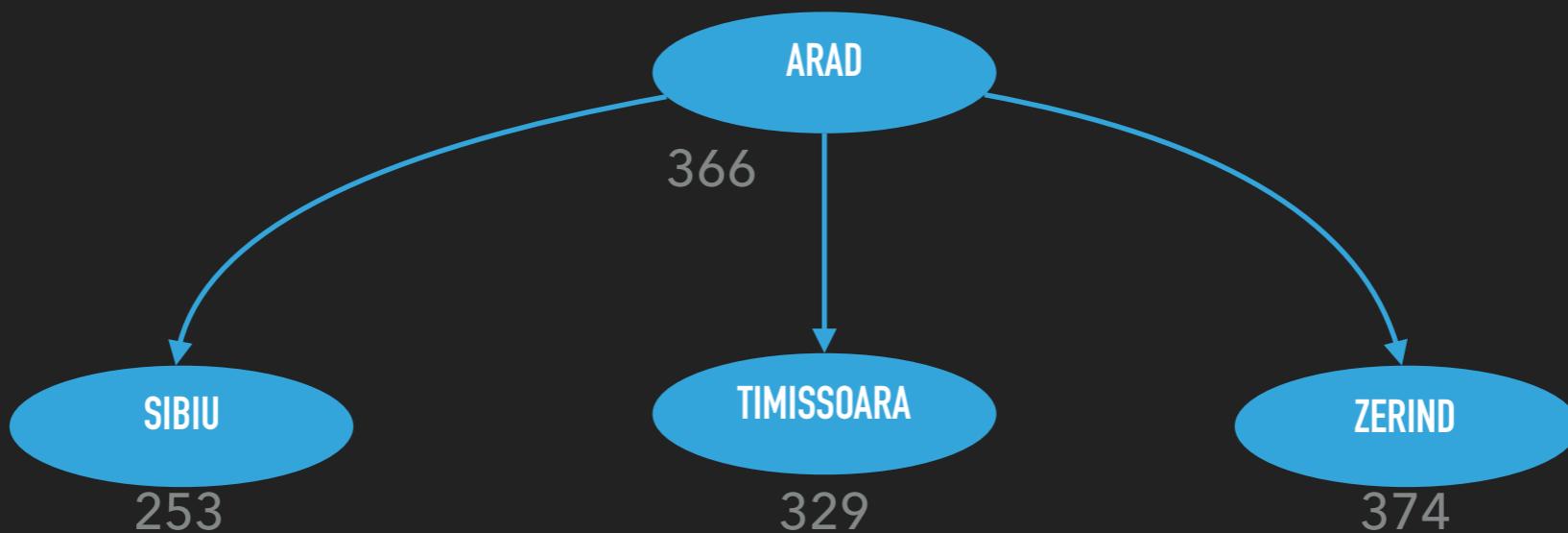
366

HEURÍSTICA:
DISTÂNCIA EM
LINHA RETA

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
Rimin.	193
Sibiu	253
Timisoara	329
Urzecini	80
Vaslui	199
Zerind	374

BUSCA GULOSA NO PROBLEMA DA ROMÊNIA

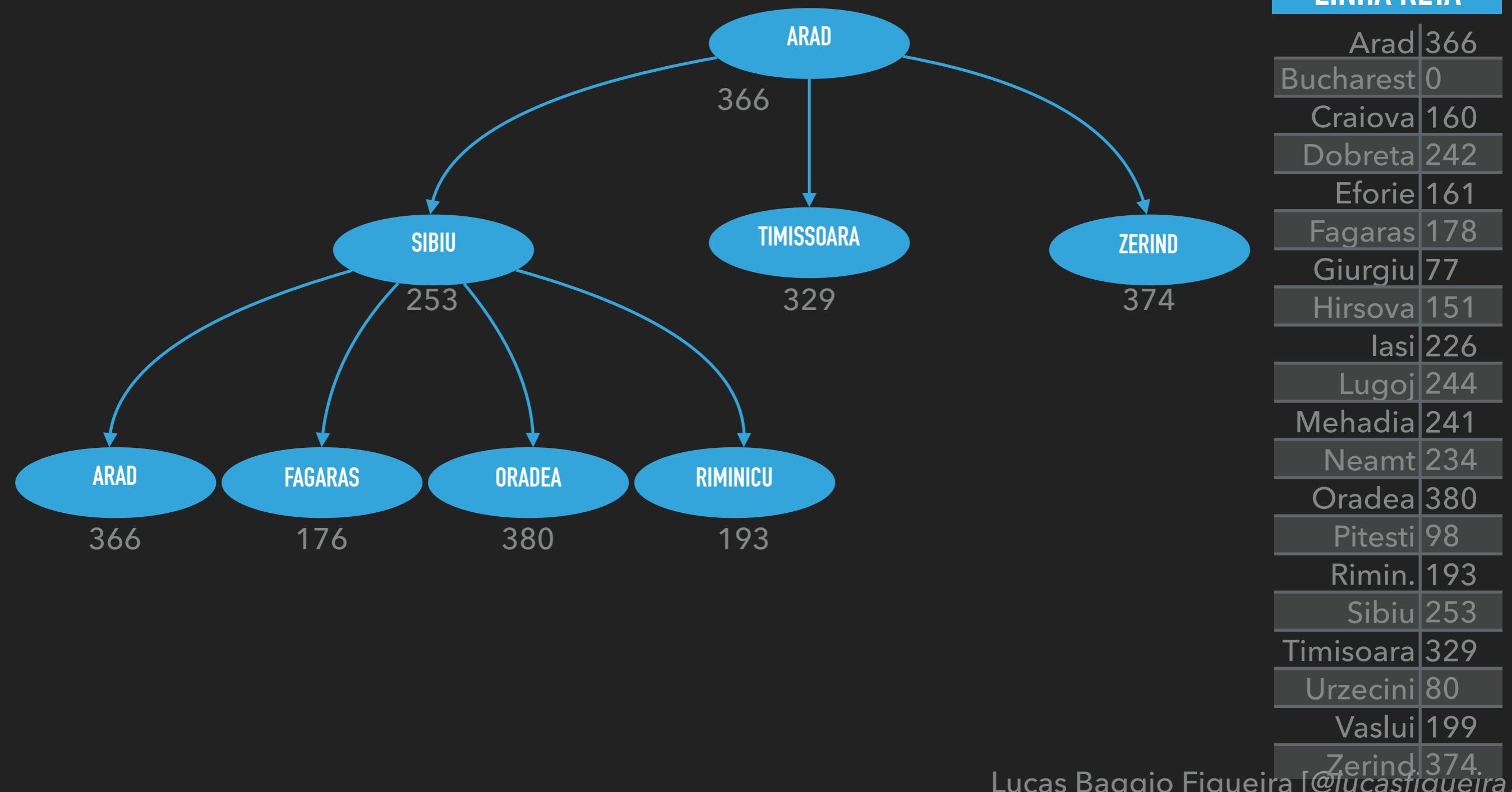
HEURÍSTICA:
DISTÂNCIA EM
LINHA RETA



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
Rimin.	193
Sibiu	253
Timisoara	329
Urzecini	80
Vaslui	199
Zerind	374

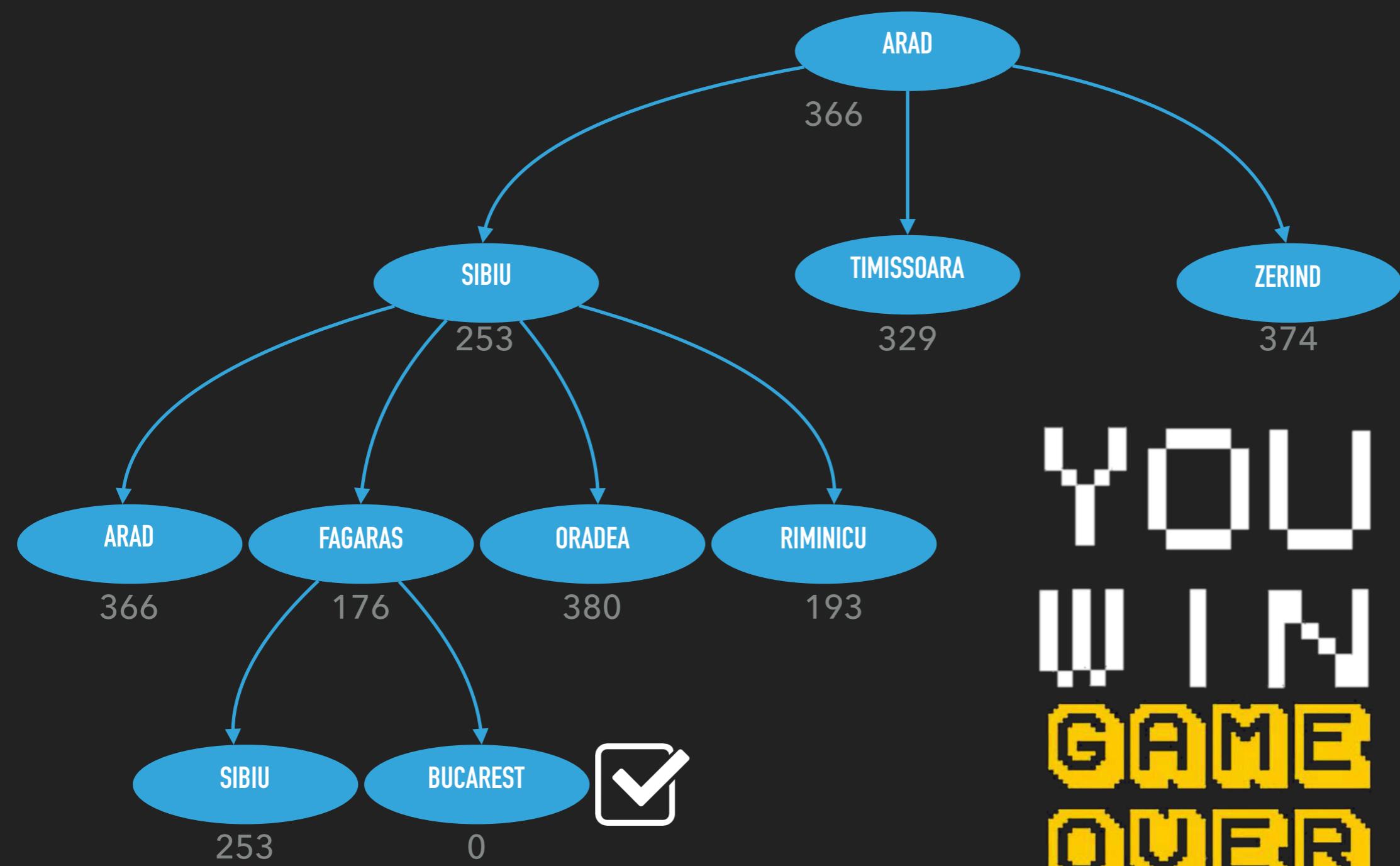
BUSCA GULOSA NO PROBLEMA DA ROMÊNIA

**HEURÍSTICA:
DISTÂNCIA EM
LINHA RETA**



BUSCA GULOSA NO PROBLEMA DA ROMÊNIA

**HEURÍSTICA:
DISTÂNCIA EM
LINHA RETA**



YOU
WIN
GAME
OVER

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
Rimin.	193
Sibiu	253
Timisoara	329
Urzecini	80
Vaslui	199
Zerind	374

PROPRIEDADES DA BUSCA GULOSA

PROPRIEDADES DA BUSCA GULOSA

- ▶ Completo?

PROPRIEDADES DA BUSCA GULOSA

- ▶ Completo?
- ▶ Não - pode ficar preso em loops - adotar checagem de nó visitado.

PROPRIEDADES DA BUSCA GULOSA

- ▶ Completo?
- ▶ Não - pode ficar preso em loops - adotar checagem de nó visitado.
- ▶ Tempo?

PROPRIEDADES DA BUSCA GULOSA

- ▶ Completo?
 - ▶ Não - pode ficar preso em loops - adotar checagem de nó visitado.
- ▶ Tempo?
 - ▶ $O(b^m)$ - drástica redução usando uma boa heurística

PROPRIEDADES DA BUSCA GULOSA

- ▶ Completo?
 - ▶ Não - pode ficar preso em loops - adotar checagem de nó visitado.
- ▶ Tempo?
 - ▶ $O(b^m)$ - drástica redução usando uma boa heurística
- ▶ Espaço?

PROPRIEDADES DA BUSCA GULOSA

- ▶ Completo?
 - ▶ Não - pode ficar preso em loops - adotar checagem de nó visitado.
- ▶ Tempo?
 - ▶ $O(b^m)$ - drástica redução usando uma boa heurística
- ▶ Espaço?
 - ▶ $O(b^m)$ - mantém todos os nós na memória

PROPRIEDADES DA BUSCA GULOSA

- ▶ Completo?
 - ▶ Não - pode ficar preso em loops - adotar checagem de nó visitado.
- ▶ Tempo?
 - ▶ $O(b^m)$ - drástica redução usando uma boa heurística
- ▶ Espaço?
 - ▶ $O(b^m)$ - mantém todos os nós na memória
- ▶ Ótimo?

PROPRIEDADES DA BUSCA GULOSA

- ▶ Completo?
 - ▶ Não - pode ficar preso em loops - adotar checagem de nó visitado.
- ▶ Tempo?
 - ▶ $O(b^m)$ - drástica redução usando uma boa heurística
- ▶ Espaço?
 - ▶ $O(b^m)$ - mantém todos os nós na memória
- ▶ Ótimo?
 - ▶ Não

IA - BUSCA EM ESPAÇO DE ESTADOS

A* (A-STAR)

A* (A-STAR)

A* (A-STAR)

- ▶ Idéia básica: evita expandir os nós que já estão custosos.
- ▶ Função de avaliação: $f(n) = g(n) + h(n)$
 - ▶ $h(n)$ - heurística, "custo" estimado de n até o objetivo.
 - ▶ $g(n)$ - custo para se alcançar n , custo real.
- ▶ A* usa uma heurística admissível, ou seja, $h(n) \leq h^*(n)$ - verdadeira medida.
- ▶ Teorema: A* é ótimo quando $h(n)$ é admissível.

BUSCA A*

NO PROBLEMA DA ROMÊNIA

BUSCA A*

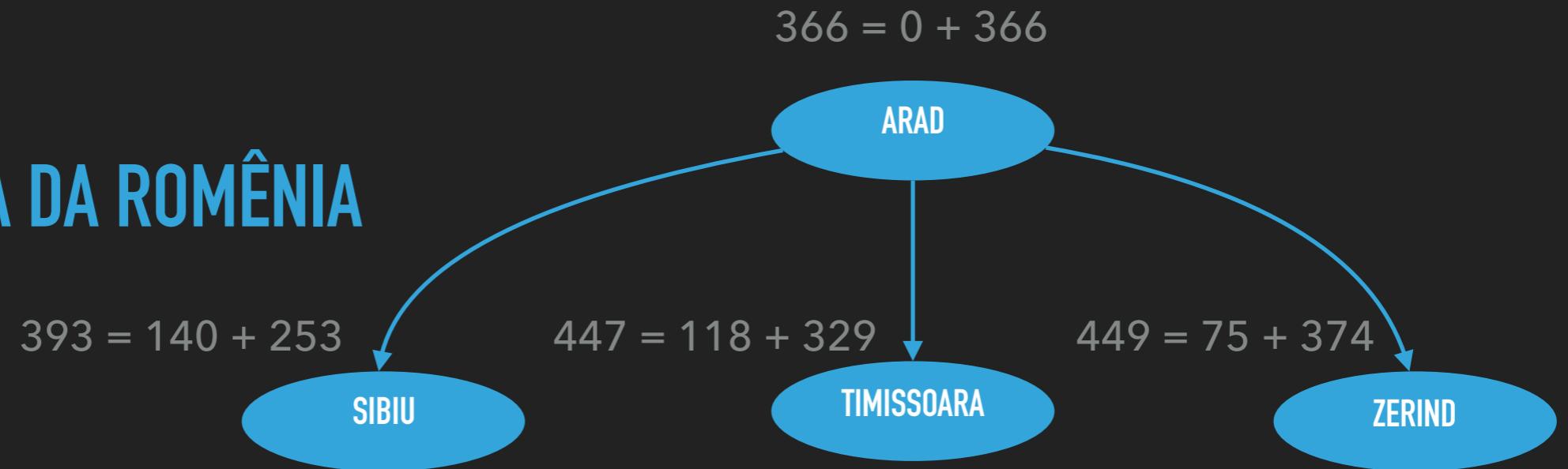
NO PROBLEMA DA ROMÊNIA

$$366 = 0 + 366$$

ARAD

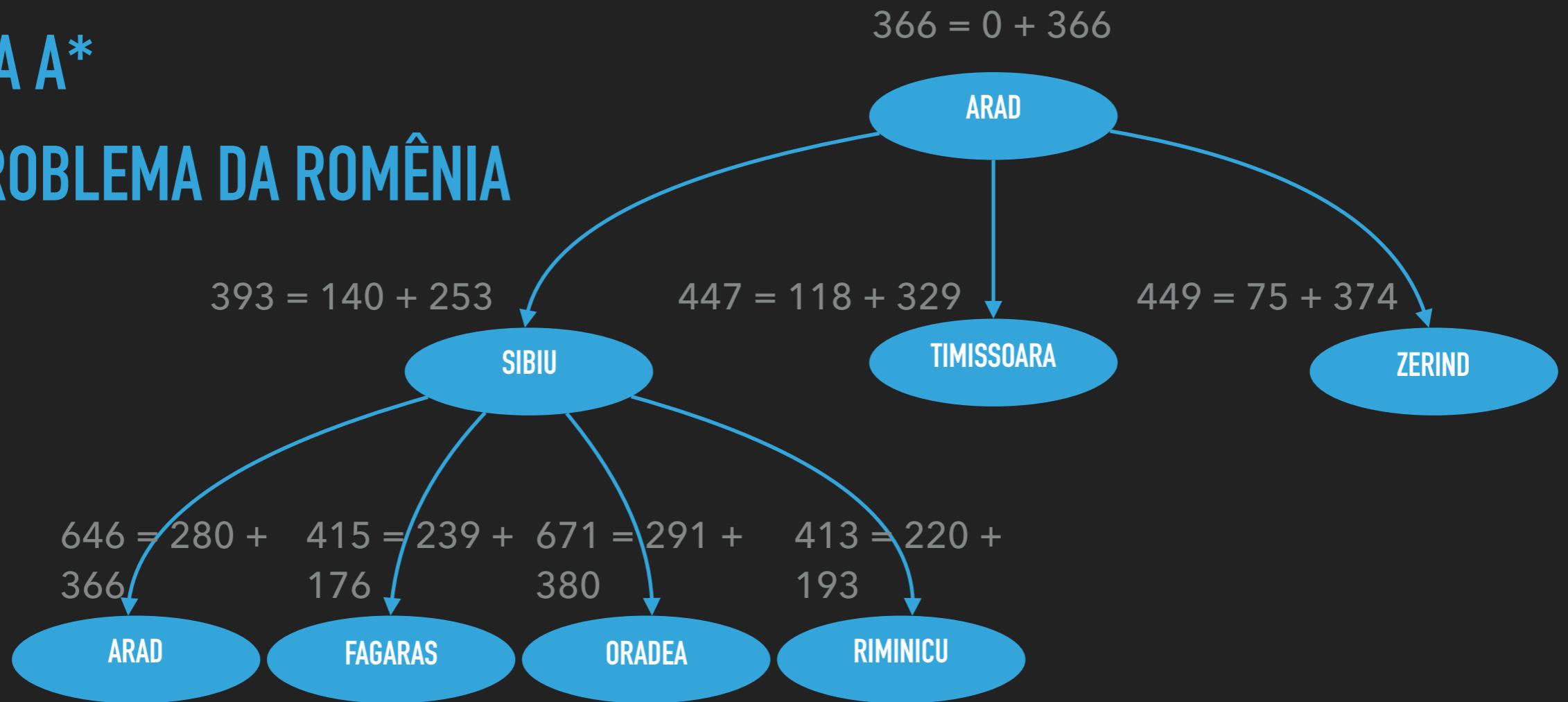
BUSCA A*

NO PROBLEMA DA ROMÊNIA



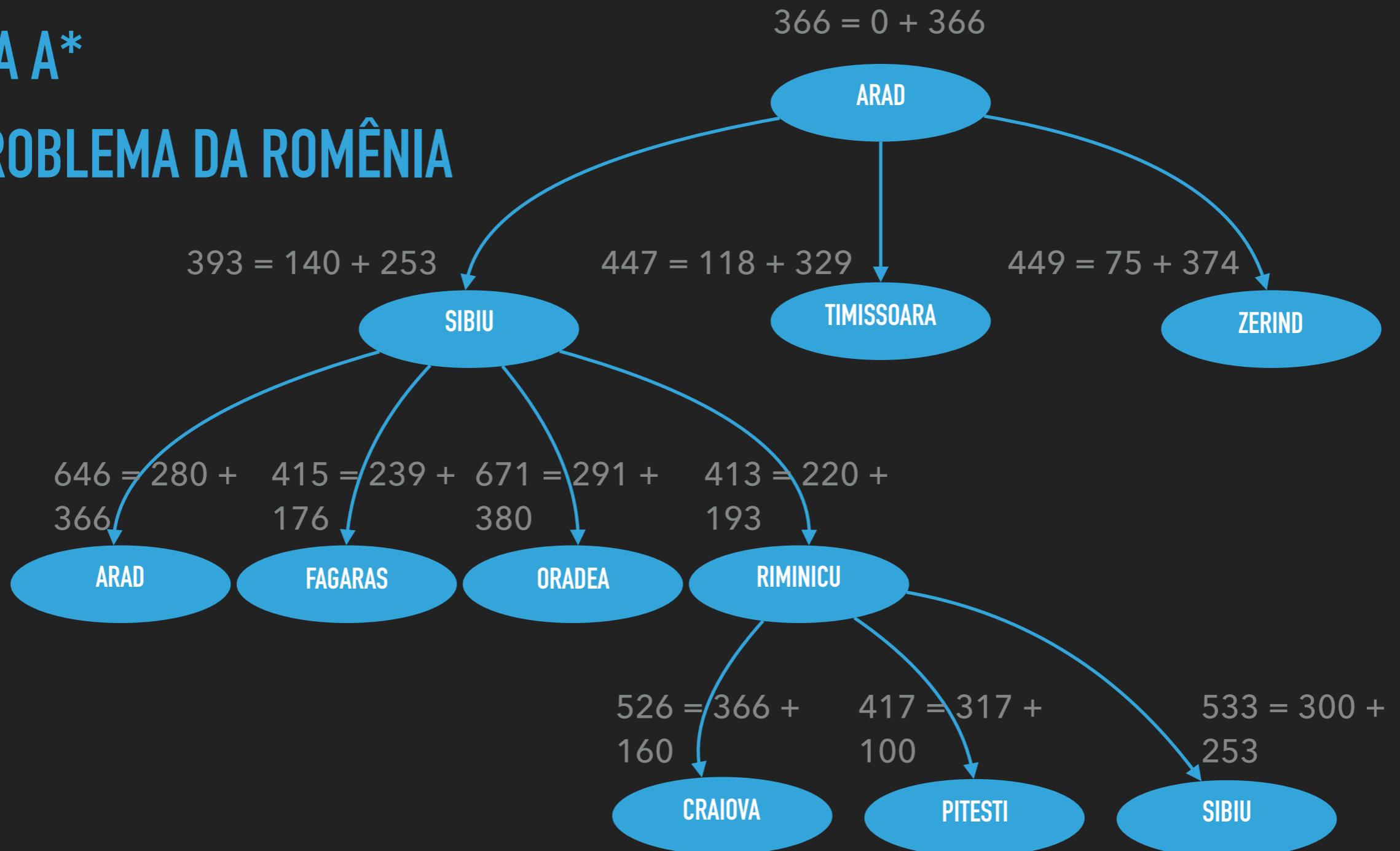
BUSCA A*

NO PROBLEMA DA ROMÊNIA



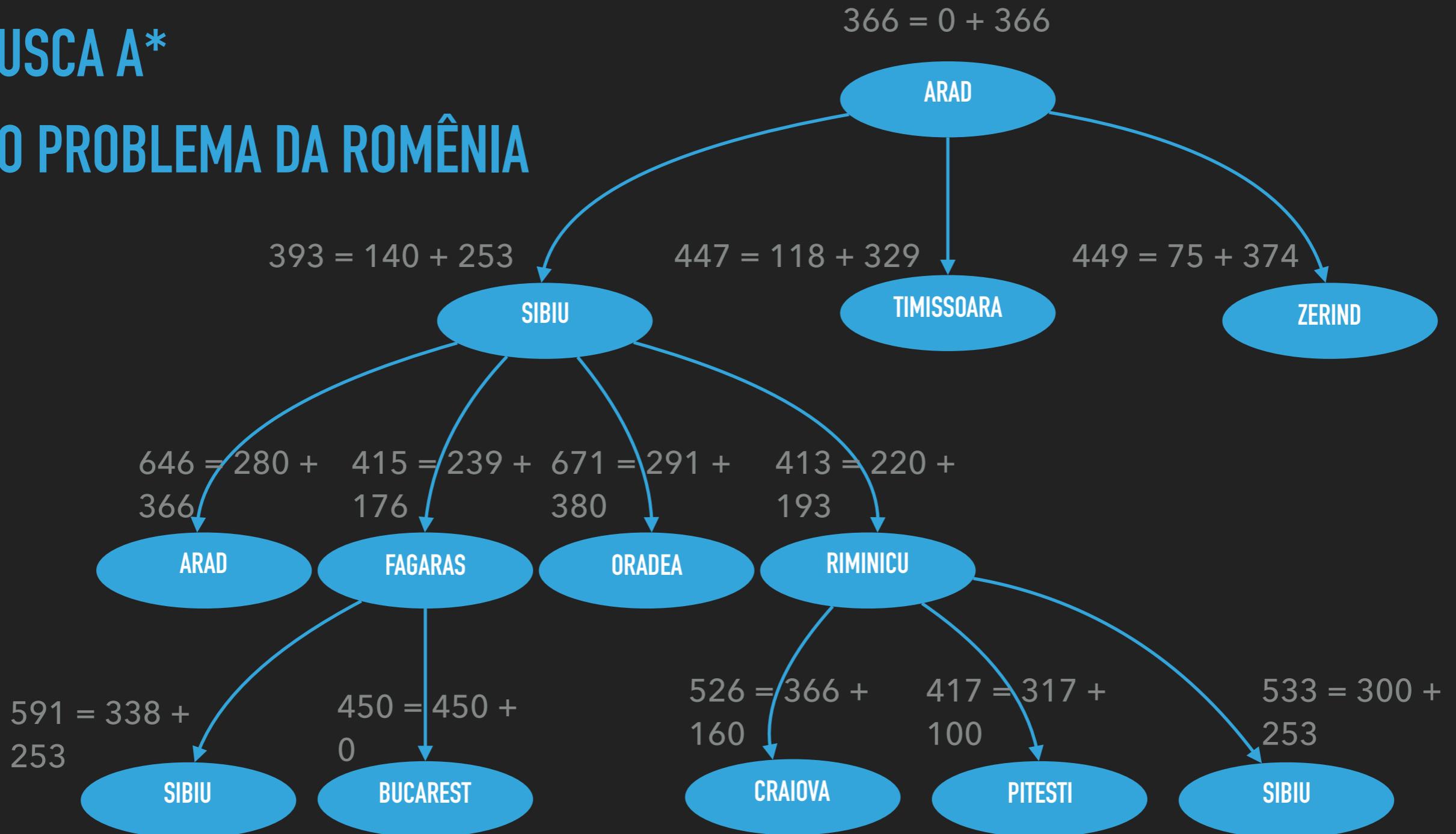
BUSCA A*

NO PROBLEMA DA ROMÊNIA



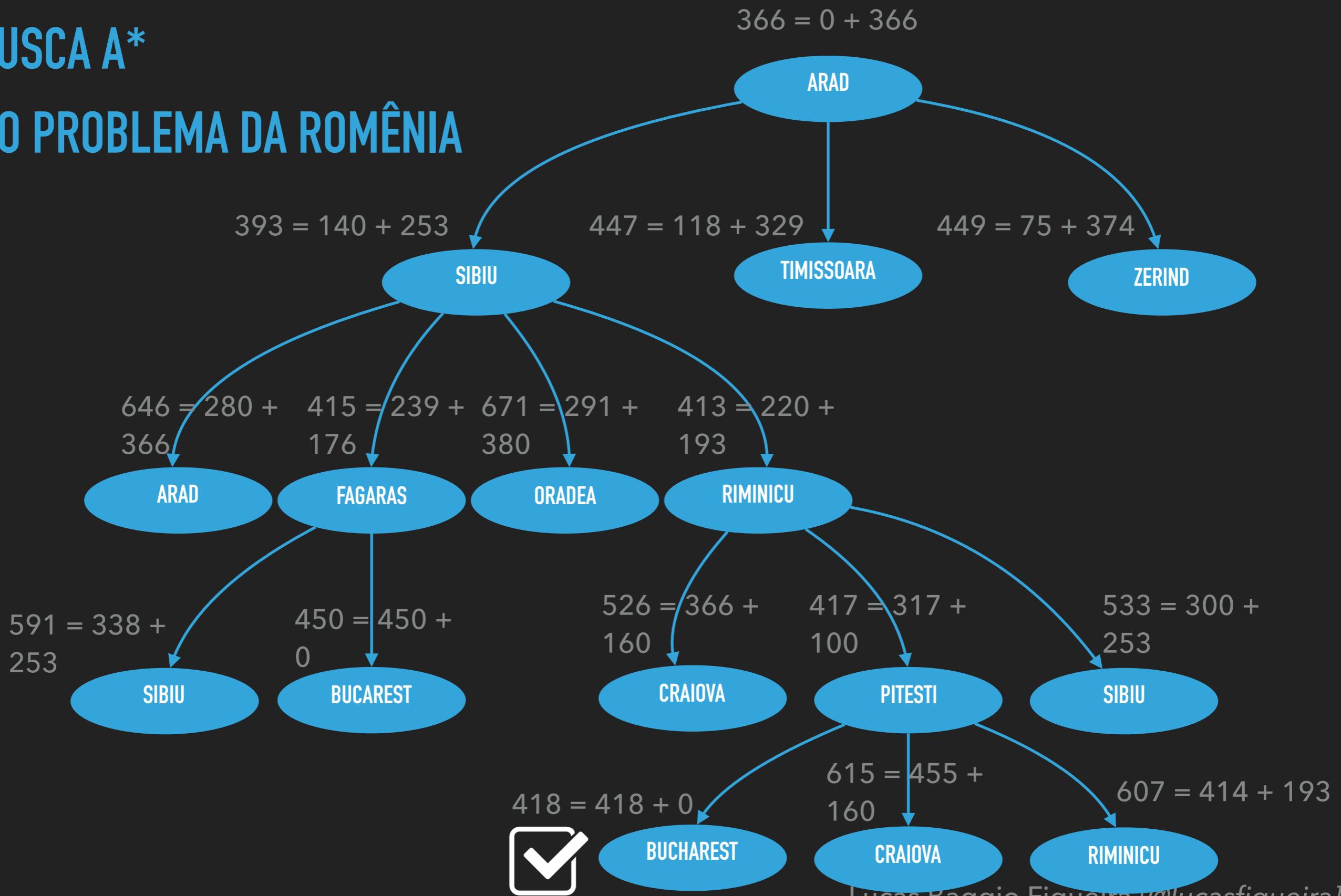
BUSCA A*

NO PROBLEMA DA ROMÊNIA



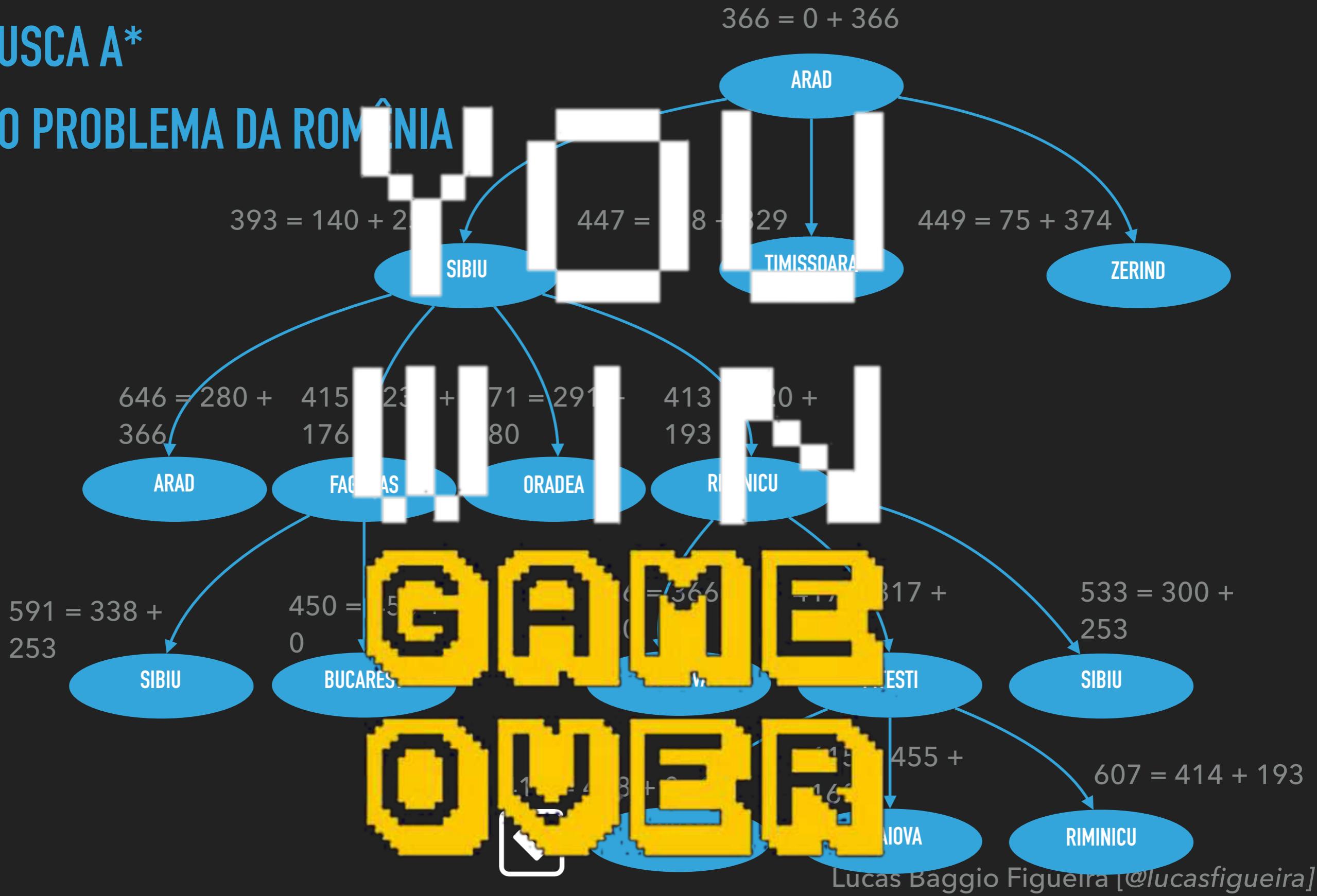
BUSCA A*

NO PROBLEMA DA ROMÊNIA



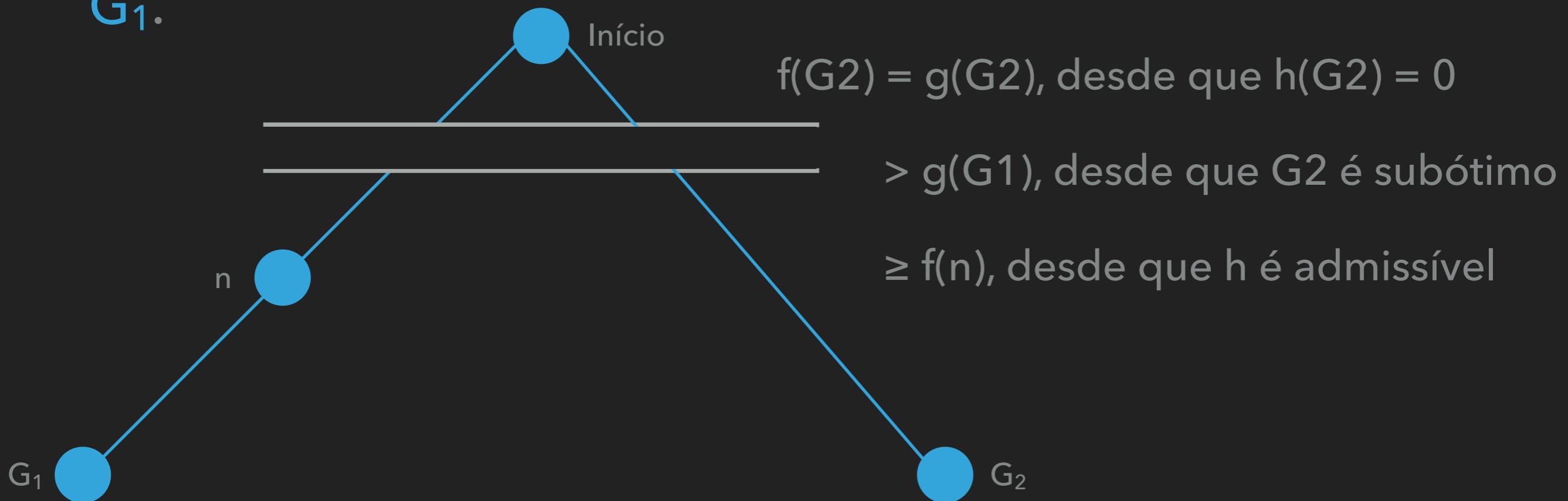
BUSCA A*

NO PROBLEMA DA ROMÉNIA



OTIMALIDADE DO A*

- ▶ Suponha um caminho ao objetivo subótimo G_2 que foi gerado e está na fila (*fringe*). Seja n um nó não expandido que está no caminho mais curto (ótimo) para o objetivo G_1 .



PROPRIEDADES DO A*

PROPRIEDADES DO A*

- ▶ Completo?

PROPRIEDADES DO A*

- ▶ Completo?
- ▶ Sim, a não ser que existam infinitos nós $f < f(G)$

PROPRIEDADES DO A*

- ▶ Completo?
 - ▶ Sim, a não ser que existam infinitos nós $f < f(G)$
- ▶ Tempo?

PROPRIEDADES DO A*

- ▶ Completo?
 - ▶ Sim, a não ser que existam infinitos nós $f < f(G)$
- ▶ Tempo?
 - ▶ Exponencial.

PROPRIEDADES DO A*

- ▶ Completo?
 - ▶ Sim, a não ser que existam infinitos nós $f < f(G)$
- ▶ Tempo?
 - ▶ Exponencial.
- ▶ Espaço?

PROPRIEDADES DO A*

- ▶ Completo?
 - ▶ Sim, a não ser que existam infinitos nós $f < f(G)$
- ▶ Tempo?
 - ▶ Exponencial.
- ▶ Espaço?
 - ▶ Mantém todos os nós na memória.

PROPRIEDADES DO A*

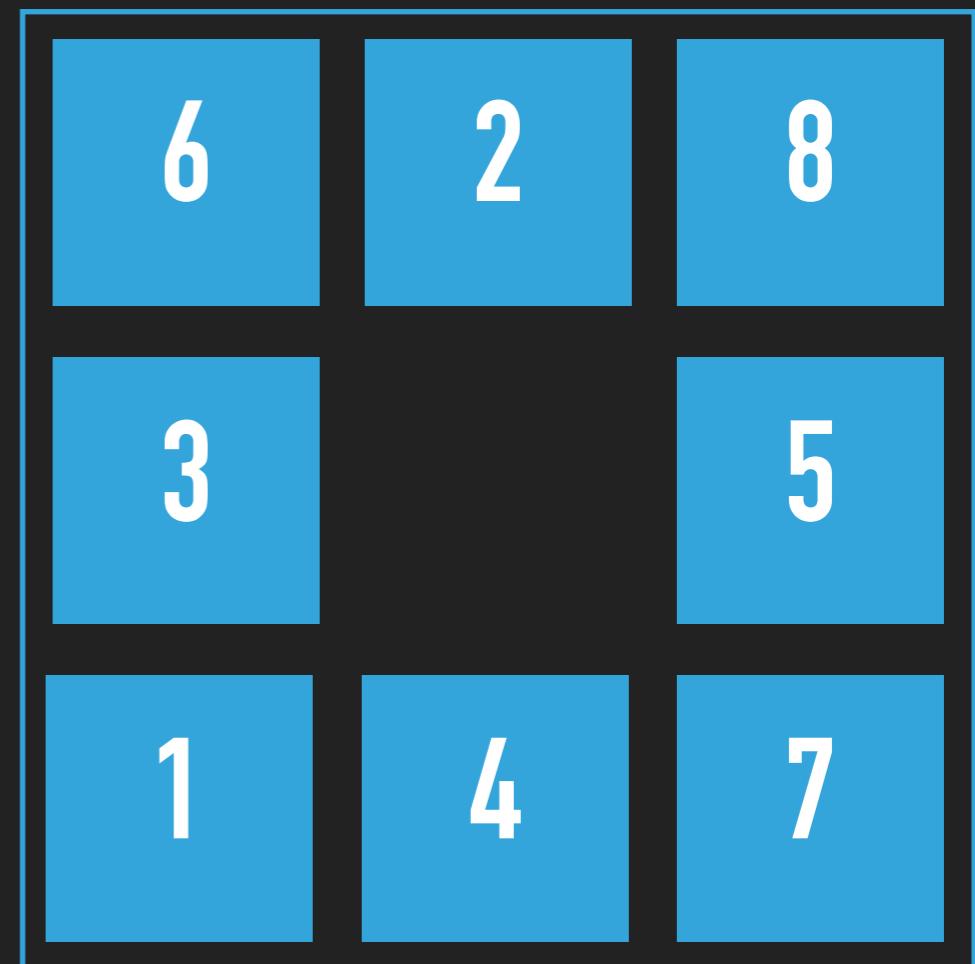
- ▶ Completo?
 - ▶ Sim, a não ser que existam infinitos nós $f < f(G)$
- ▶ Tempo?
 - ▶ Exponencial.
- ▶ Espaço?
 - ▶ Mantém todos os nós na memória.
- ▶ Optimalidade?

PROPRIEDADES DO A*

- ▶ Completo?
 - ▶ Sim, a não ser que existam infinitos nós $f < f(G)$
- ▶ Tempo?
 - ▶ Exponencial.
- ▶ Espaço?
 - ▶ Mantém todos os nós na memória.
- ▶ Otimalidade?
 - ▶ Sim

HEURÍSTICAS - 8-PUZZLE

- ▶ $h_1(n)$ - número de peças fora do lugar.
- ▶ $h_2(n)$ - distância da Manhattan.
- ▶ Distância da peça de seu lugar "certo"



HEURÍSTICA: DOMINÂNCIA

- ▶ $d = 14$, IDS = 3.473.941 nós
- ▶ $A^*(h_1) = 539$ nós.
- ▶ $A^*(h_2) = 113$ nós.
- ▶ $d = 24$, IDS $\approx 54.000.000.000$ nós
- ▶ $A^*(h_1) = 39.135$ nós.
- ▶ $A^*(h_2) = 1.641$ nós.



EXERCÍCIO

**IMPLEMENTAR O 8-PUZZLE
USANDO BUSCA COM INFORMAÇÃO**