

# Apoio à implementação dos métodos de procura informados

Procura sôfrega / A\* / Procura em Feixe / IDA\*

# Considerações sobre os métodos de procura informados:

$h$  – valor da heurística

$g$  – custo do caminho

- A fronteira é uma lista de nós ordenada de forma crescente pelo valor de  $f$ , tal como no método de procura uniforme
- Os métodos de procura informados requerem a implementação de uma heurística que avalie cada estado. No puzzle de 8, na classe *EightPuzzleState*, já estão dois métodos definidos:
  - *computeTileDistances(EightPuzzleState finalState)* – este método é invocado dentro do método *compute* da heurística *HeuristicTileDistance*
  - *ComputeTilesOutOfPlace(EightPuzzleState finalState)* – este método é invocado dentro do método *compute* da heurística *HeuristicTilesOutOfPlace*

# Heurísticas no *puzzle8*

- Considere que o objetivo é chegar ao estado seguinte:

	1	2
3	4	5
6	7	8

- Considere os seguintes estados:

1	2	3
	4	5
6	7	8

8	2	3
4	5	
6	1	7

<i>HeuristicTileDistance</i> (distância total à posição final)	1 2 3 4 5 6 7 8 h= 1+1+3+0+0+0+0+0=5	1 2 3 4 5 6 7 8 h= 2+1+3+1+1+0+1+4=13
<i>HeuristicTilesOutOfPlace</i>	h=3 (3 peças fora do lugar)	h=7 (7 peças fora do lugar)

# IMPLEMENTAR:

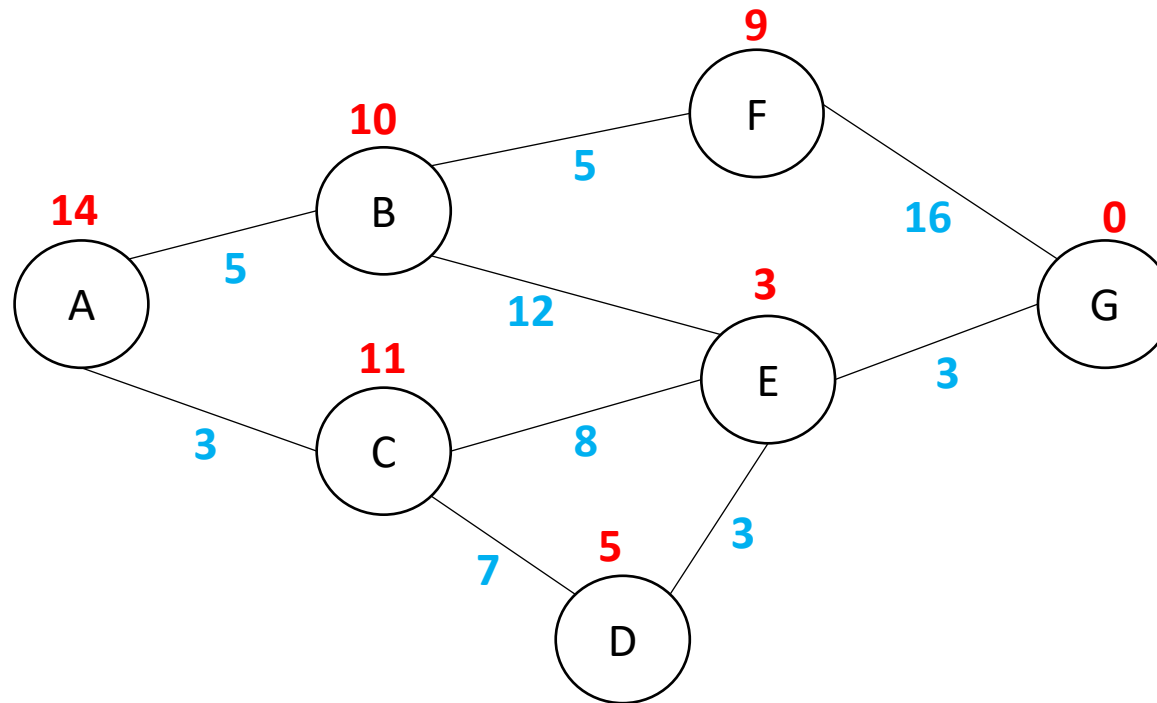
Métodos *compute(...)* das classes

*HeuristicTileDistance* e  
*HeuristicTilesOutOfPlace*

# Considere o seguinte exemplo:

vermelho – valor da heurística

azul – custo do caminho



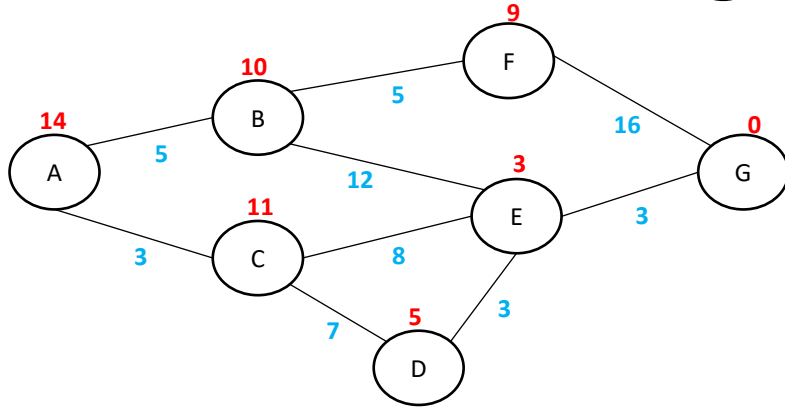
# Procura Sôfrega (*GreedyBestFirstSearch*)

$$f = h(n)$$

- Não é ótimo
- Tal como na procura uniforme utiliza uma lista de nós explorados
- Um nó só é adicionado à fronteira se o estado correspondente:
  - Não esteja já na fronteira e não tenha sido explorado
  - Já está na fronteira mas o custo do caminho até chegar a esse nó sucessor a partir do nó inicial (ou seja, o valor de  $g$ ) for inferior ao do estado que já está na fronteira. Neste caso, o nó que está na fronteira deve ser eliminado e o novo nó é adicionado (ver método `addSuccessorsToFrontier` da procura uniforme)
- A única diferença na implementação do `addSuccessorsToFrontier()` deste método para o procura uniforme, é que o valor de  $f$  do novo nó é igual ao valor da heurística (*`new Node (s, parent, g, heuristic.compute(s))`*)

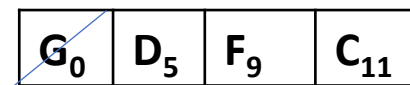
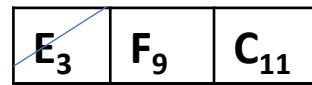
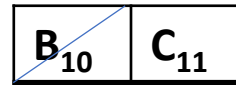
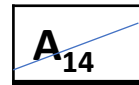
# Procura Sôfrega (*GreedyBestFirstSearch*)

$$f = h(n)$$

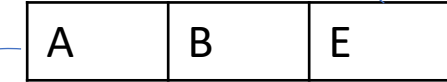


Sempre que um nó explorado não é objetivo, é adicionado à lista de explorados

• Fronteira:



• Nós explorados:



A

B

E

G

Não é adicionado à fronteira porque já foi explorado

Não é adicionado à fronteira porque já foi explorado

Não é adicionado à fronteira porque o **custo (g)** deste nó (25) é superior ao do que já lá está (3)

IMPLEMENTAR:

*Método addSuccessorsToFrontier(...)*  
da classe *GreedyBestFirstSearch*

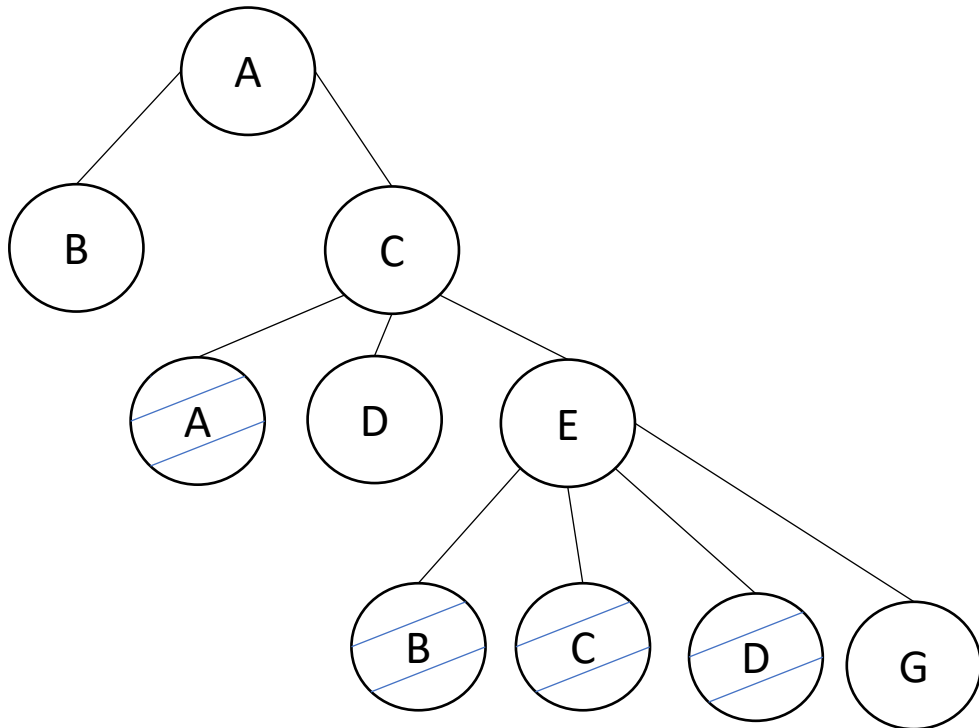
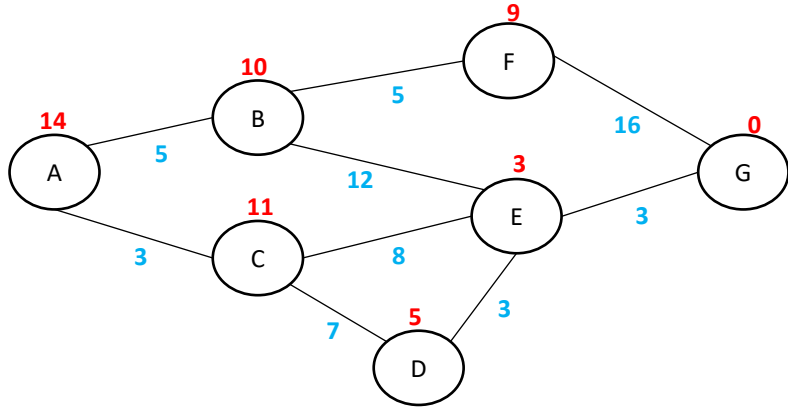


# $A^*$ (AStarSearch)

$$f = g(n) + h(n)$$

- É ótimo se a heurística for admissível
- Tal como na procura sôfrega, utiliza uma lista de nós explorados
- Assumindo que a heurística é consistente, um nó só é adicionado à fronteira se o estado correspondente não esteja já na fronteira e não tenha sido explorado
- Caso a heurística não seja consistente, tal como nos métodos de procura sôfrega e uniforme, se um estado sucessor for igual a um estado que já está na fronteira, este pode substituir o estado que está na fronteira se o custo do caminho até chegar a esse estado sucessor a partir do nó inicial (ou seja, o valor de  $g$ ) for inferior

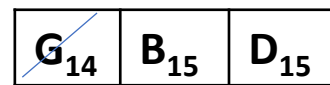
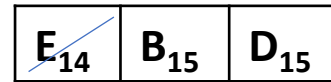
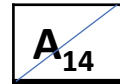
# A\*(AStarSearch)



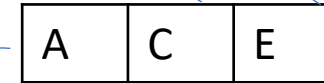
$$f = g(n) + h(n)$$

Sempre que um nó explorado não é objetivo, é adicionado à lista de explorados

• Fronteira:



• Nós explorados:



A

C

E

G

Nós sucessores não são adicionados à fronteira, ou porque estão na lista de nós explorados ou porque já estão na fronteira e têm custo do caminho superior

IMPLEMENTAR:

*Método addSuccessorsToFrontier(...)*

da classe  $A^*$

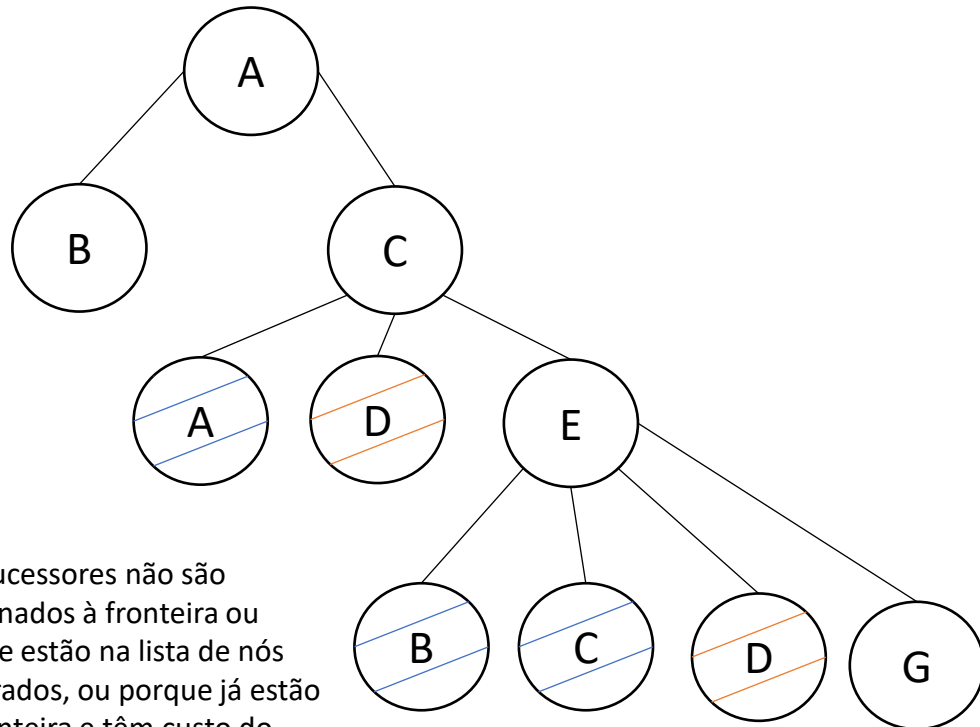
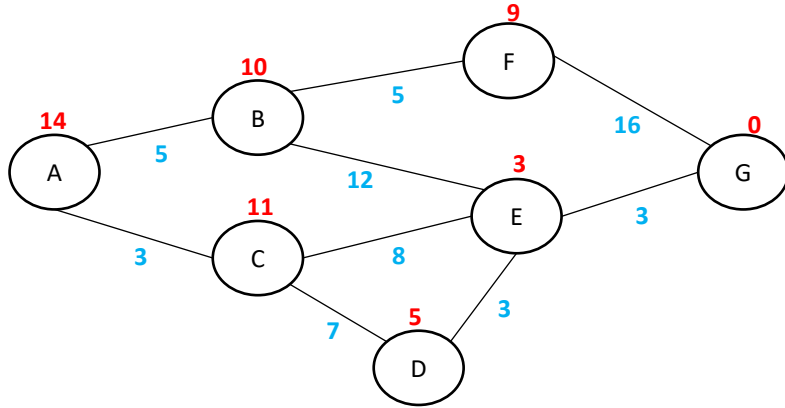
# Procura em Feixe (*BeamSearch*)

$$f = g(n) + h(n)$$

- Não é completo e não é ótimo
- Idêntico ao A\*, mas com imposição de um limite máximo para o tamanho da fronteira (*beamSize*)
- Após adicionarem os sucessores à fronteira devem verificar se o seu tamanho excede o *beamSize* e caso isso aconteça:
  - 1 – criar uma lista auxiliar (do tipo *NodePriorityQueue*)
  - 2 – adicionar à lista auxiliar os elementos da fronteira até ao *beamSize*
  - 3 – substituir a fronteira pela lista auxiliar
- Nota: Seria mais simples ir removendo simplesmente os últimos nós da fronteira até que esta ficasse com o tamanho máximo do *beamSize*. No entanto, a classe *PriorityQueue*, da qual estende a classe *NodePriorityQueue*, não tem o método *removeLast()*. Se quiséssemos usar esta solução, teríamos que implementar esse método na *NodePriorityQueue* ou implementar de raiz uma *PriorityQueue* com o método *removeLast()*

# Procura em Feixe (*BeamSearch*)

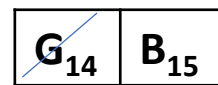
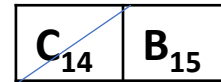
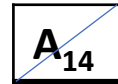
$$f = g(n) + h(n) \quad \text{beamSize}=2$$



Nós sucessores não são adicionados à fronteira ou porque estão na lista de nós explorados, ou porque já estão na fronteira e têm custo do caminho superior

Nós sucessores são removidos da fronteira (a partir do limite), caso esta atinga o limite definido (*beamSize*)

- Fronteira:



- Nós explorados:

A

C

E

G

IMPLEMENTAR:

*Método addSuccessorsToFrontier(...)*

da classe *BeamSearch*

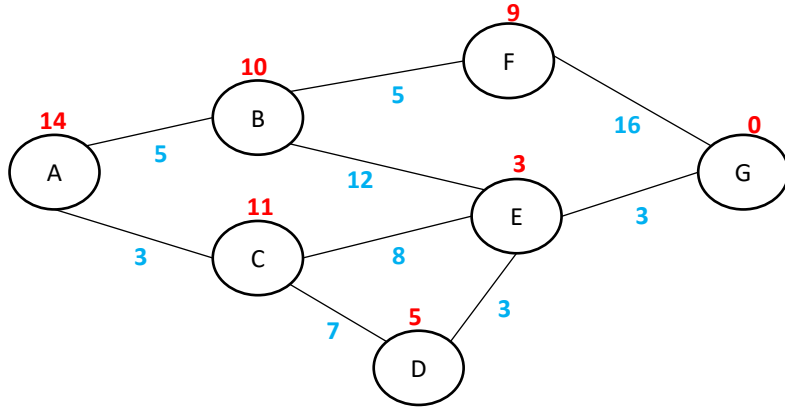
# IDA\* (*IDAStarSearch*)

$$f = g(n) + h(n)$$

- É ótimo
- Tal como o método de procura em profundidade não tem lista de nós explorados
- O método de procura é aplicado sucessivas vezes impondo um limite (*limiteInicial* =  $f$  do nó inicial, próximos limites = menor valor de  $f$  entre os sucessores não adicionados à fronteira)
- Um nó só é adicionado à fronteira se o estado correspondente:
  - Não esteja já na fronteira, o valor de  $f(n) \leq \text{limite}$  e nenhum dos pais desse nó até ao topo da árvore seja igual a esse nó
  - Já está na fronteira mas o custo do caminho até chegar a esse nó a partir do nó inicial (ou seja, o valor de  $g$ ) for inferior ao do estado que já está na fronteira. Neste caso, o nó que está na fronteira deve ser eliminado e o novo nó é adicionado

# IDA\* (IDASearch) – Exemplo 1

$$f = g(n) + h(n) \text{ limiteInicial} = f(A) = 14$$



• Fronteira:

~~A<sub>14</sub>~~

~~C<sub>14</sub>~~

~~E<sub>14</sub>~~

~~G<sub>14</sub>~~

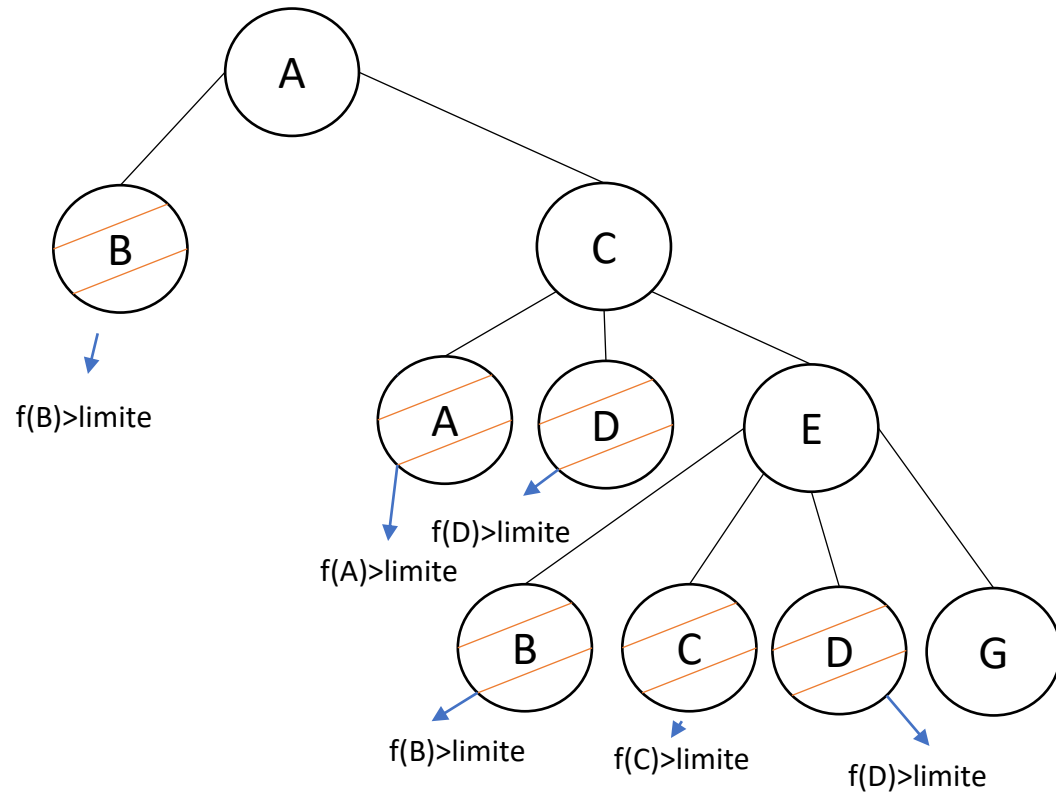
• Nós explorados:

A

C

E

G



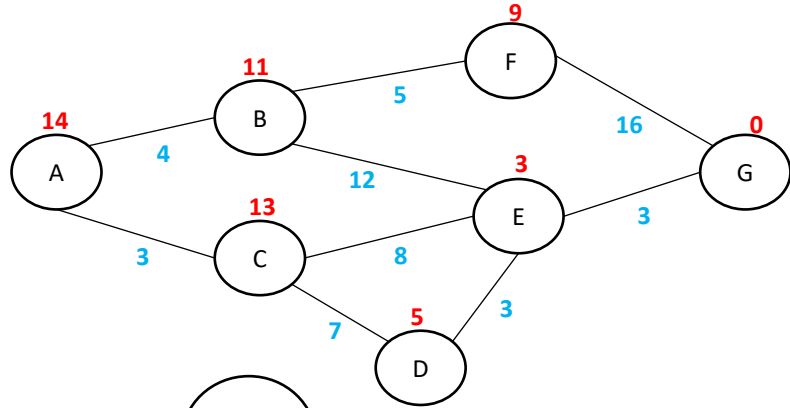
Menor  $f$  entre nós sucessores não adicionados à fronteira

15

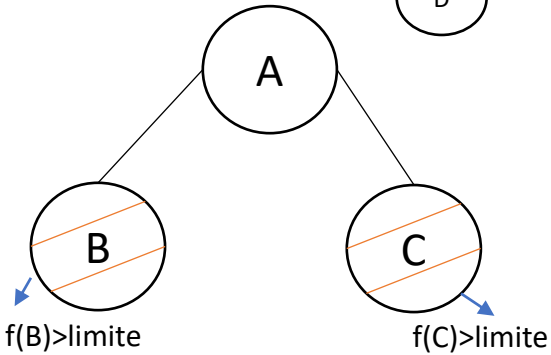


# IDA\* (IDASTarSearch) – Exemplo 2

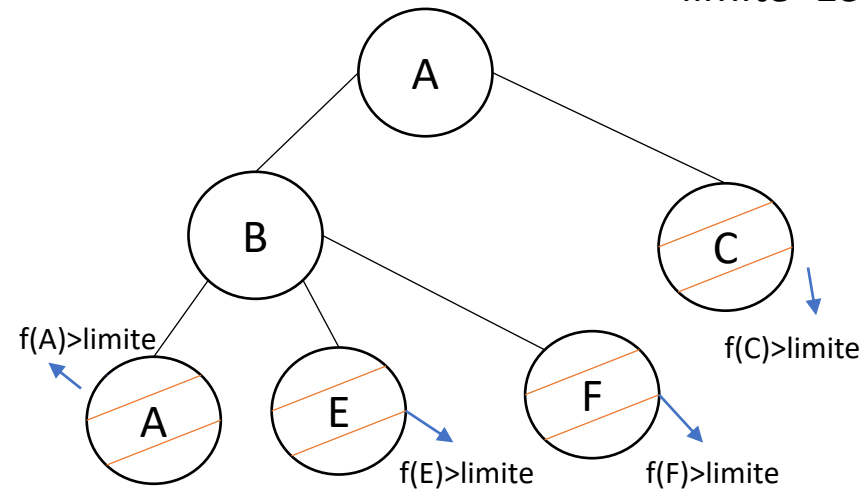
$$f = g(n) + h(n) \text{ limiteInicial} = f(A) = 14$$



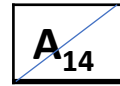
limite = 14



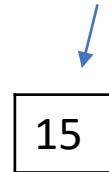
limite = 15



## • Fronteira:

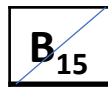
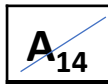


Menor  $f$  entre nós sucessores não adicionados à fronteira

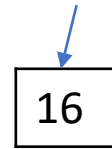


## • Nós explorados:

A



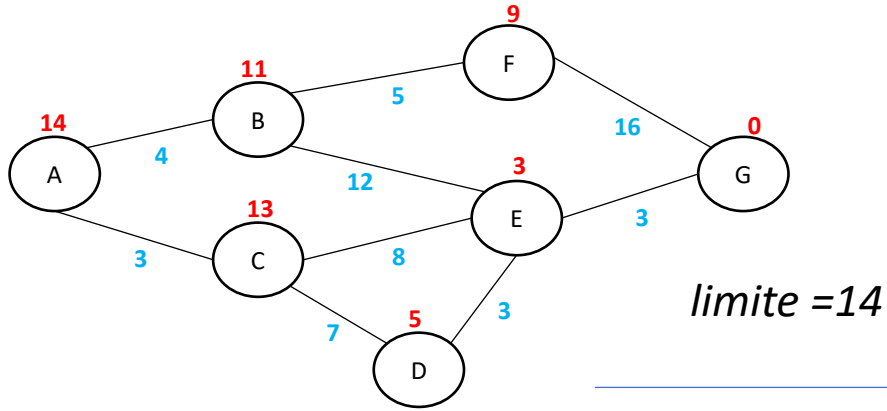
Menor  $f$  entre nós sucessores não adicionados à fronteira



A

B

# IDA\* (IDASearch) – Exemplo 2 (continuação)



• Fronteira:

...

• Nós explorados:

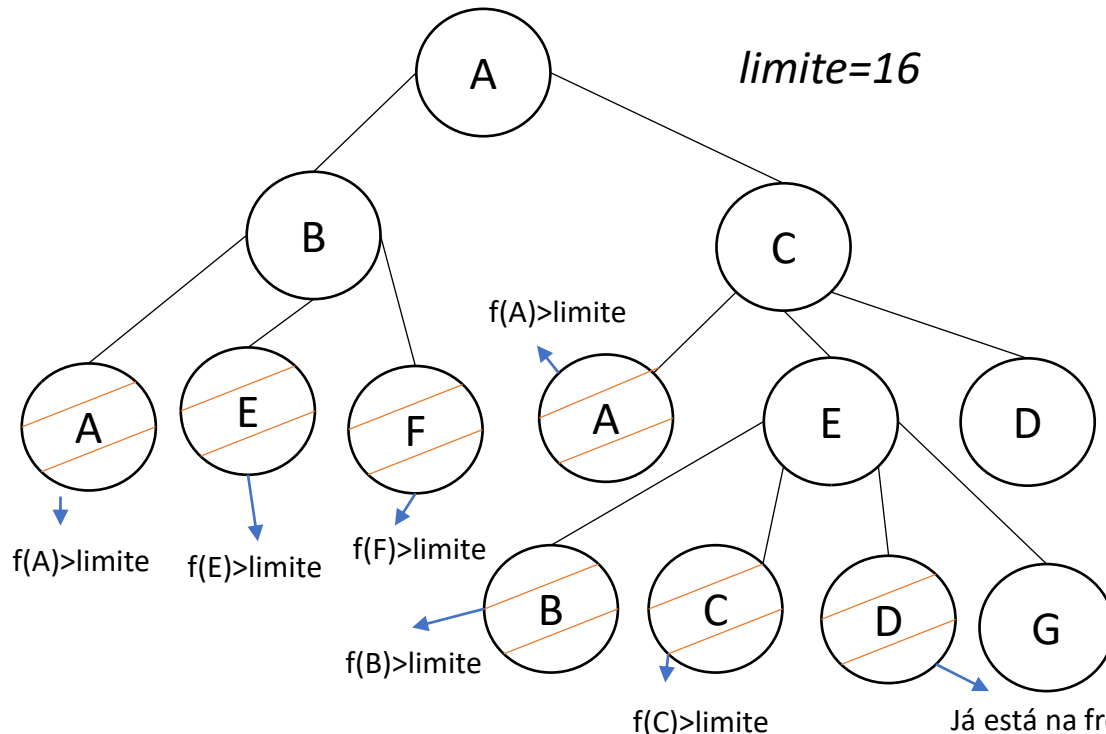
A

...

A

B

limite = 15



~~A<sub>14</sub>~~

~~B<sub>15</sub>~~ | C<sub>16</sub>

~~C<sub>16</sub>~~

~~E<sub>14</sub>~~ | D<sub>15</sub>

~~G<sub>14</sub>~~ | D<sub>15</sub>

Menor  $f$  entre nós sucessores não adicionados à fronteira

18

A

B

C

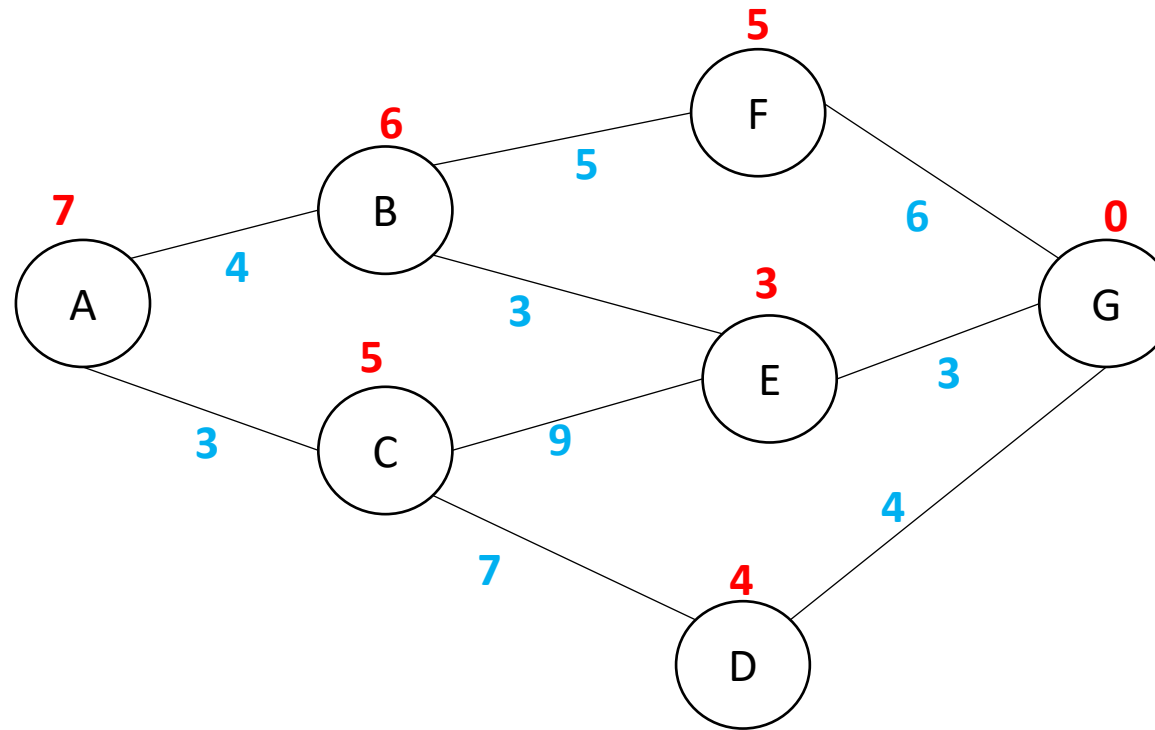
E

G

IMPLEMENTAR:

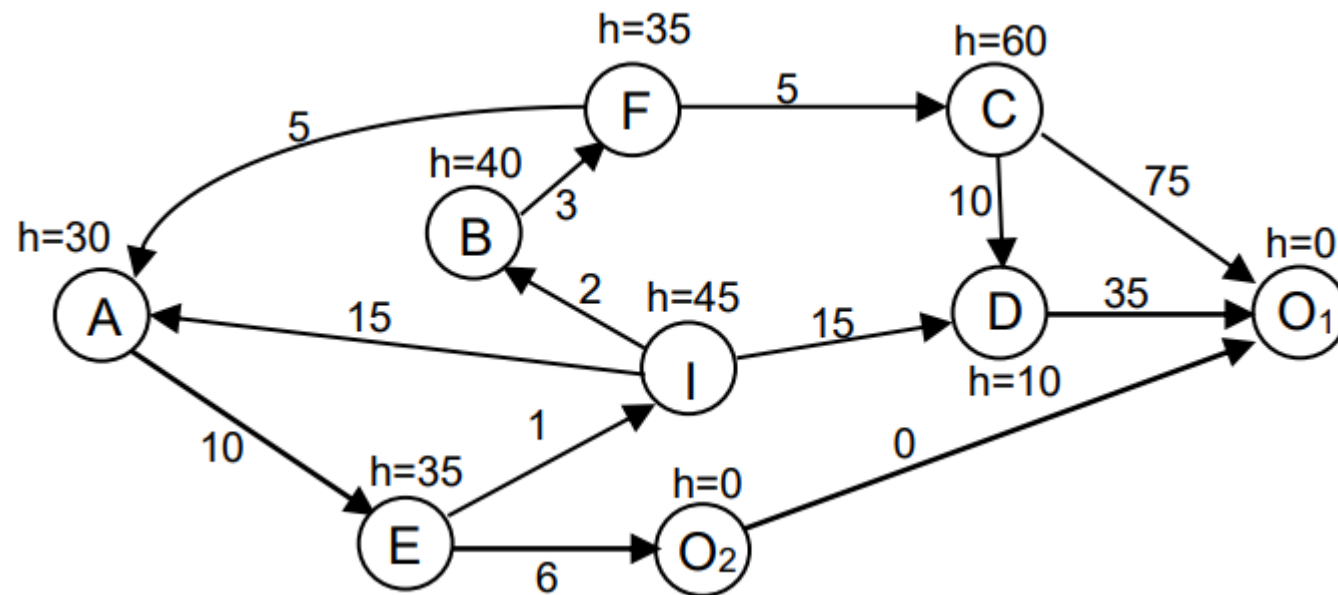
*Métodos da classe IDAStarSearch*  
(só se houver tempo)

## Exercício para estudo autónomo:

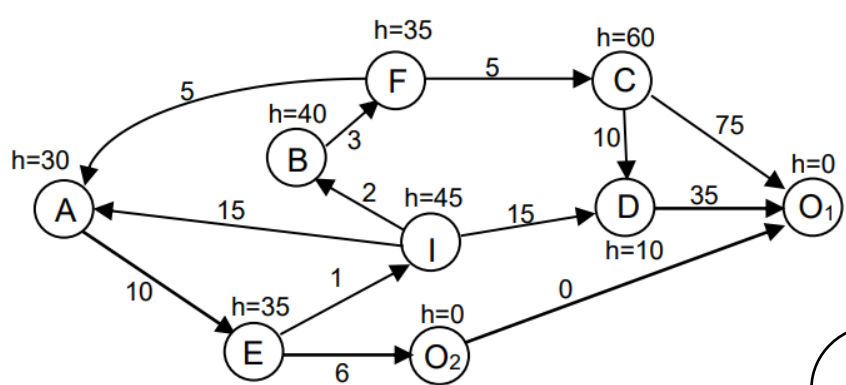


Nó inicial - A

# Exercício para estudo autónomo:



Nó inicial – I  
Nós objetivos – O<sub>1</sub>/O<sub>2</sub>



## Uniforme

Fronteira

Explorados

$\emptyset$

I

~~B~~<sub>2</sub> A<sub>15</sub> D<sub>15</sub>

B

~~F~~<sub>5</sub> A<sub>15</sub> D<sub>15</sub>

F

~~A~~<sub>10</sub> C<sub>10</sub> D<sub>15</sub>

A

~~E~~<sub>10</sub> D<sub>15</sub> E<sub>20</sub>

C

~~D~~<sub>15</sub> E<sub>20</sub> O1<sub>85</sub>

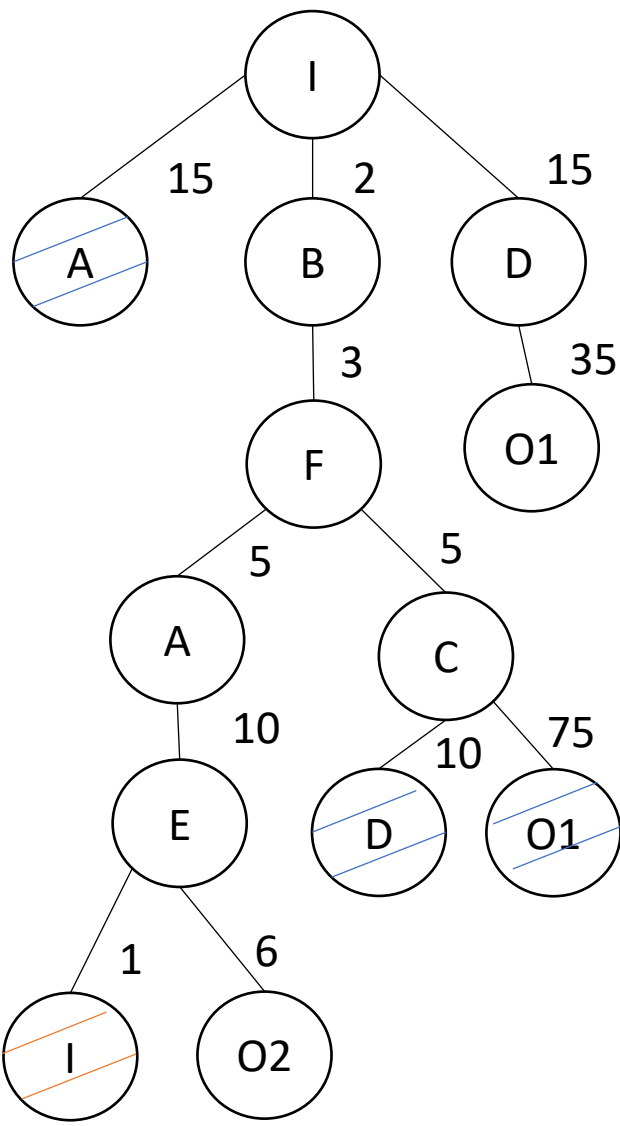
D

~~E~~<sub>20</sub> O1<sub>50</sub>

E

~~O~~<sub>26</sub> O1<sub>50</sub>

O2



## Procura Sôfrega

Fronteira

Explorados

$\emptyset$

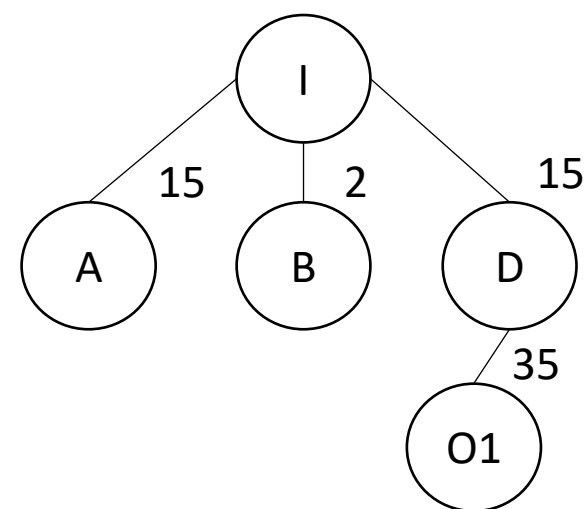
I

~~D~~<sub>10</sub> A<sub>30</sub> B<sub>40</sub>

D

~~O~~<sub>10</sub> A<sub>30</sub> B<sub>40</sub>

O



## Procura A\*

Fronteira

Explorados

$\emptyset$

I

~~D~~<sub>25</sub> B<sub>42</sub> A<sub>45</sub>

D

~~B~~<sub>42</sub> A<sub>45</sub> O1<sub>50</sub>

B

~~F~~<sub>40</sub> A<sub>45</sub> O1<sub>50</sub>

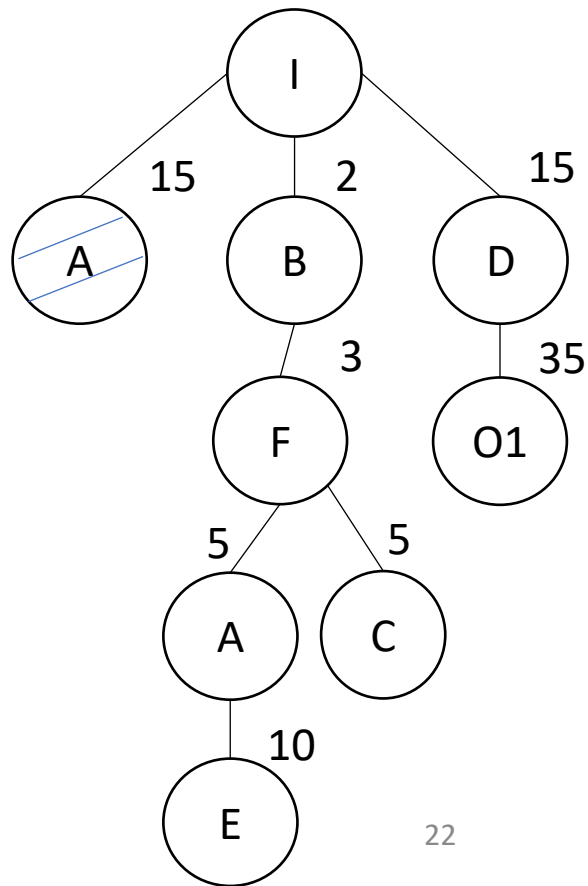
F

~~A~~<sub>40</sub> O1<sub>50</sub> C<sub>70</sub>

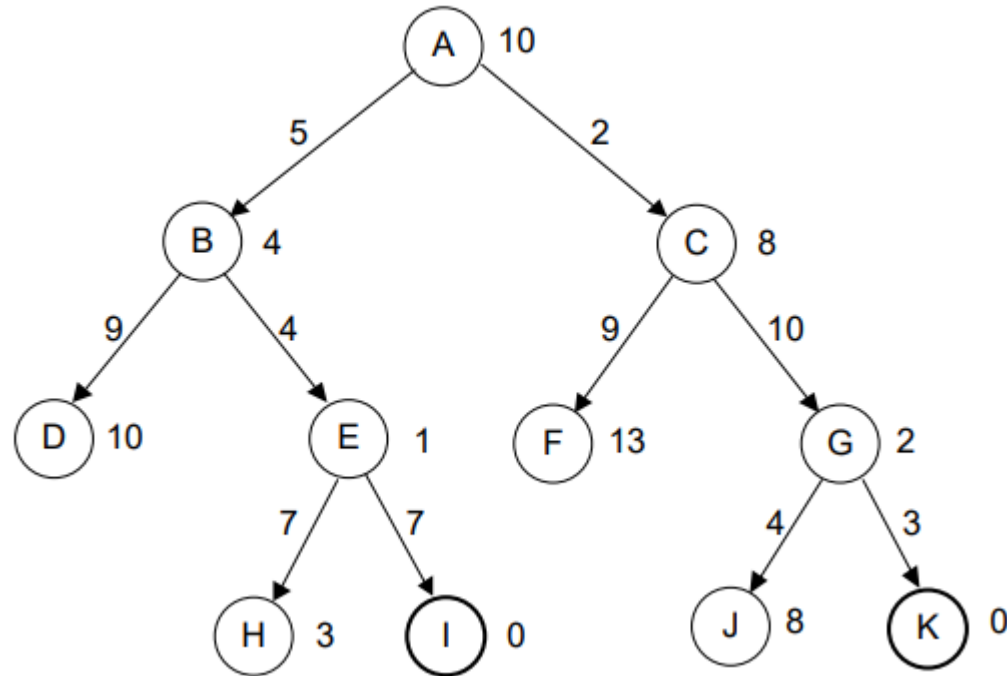
A

~~O~~<sub>150</sub> E<sub>55</sub> C<sub>70</sub>

O1



# Exercício para estudo autónomo:



Nós objetivos – I/K

# Bons estudos!