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 implementados:
  - *computeTileDistances(EightPuzzleState finalState)* – este método deve ser invocado dentro do método *compute* da heurística *HeuristicTileDistance*
  - ComputeTilesOutOfPlace(EightPuzzleState finalState) – este método deve ser 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

HeuristicTileDistance (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
HeuristicTilesOutOfPlace	h=3 (3 peças fora do lugar)	h=7 (7 peças fora do lugar)

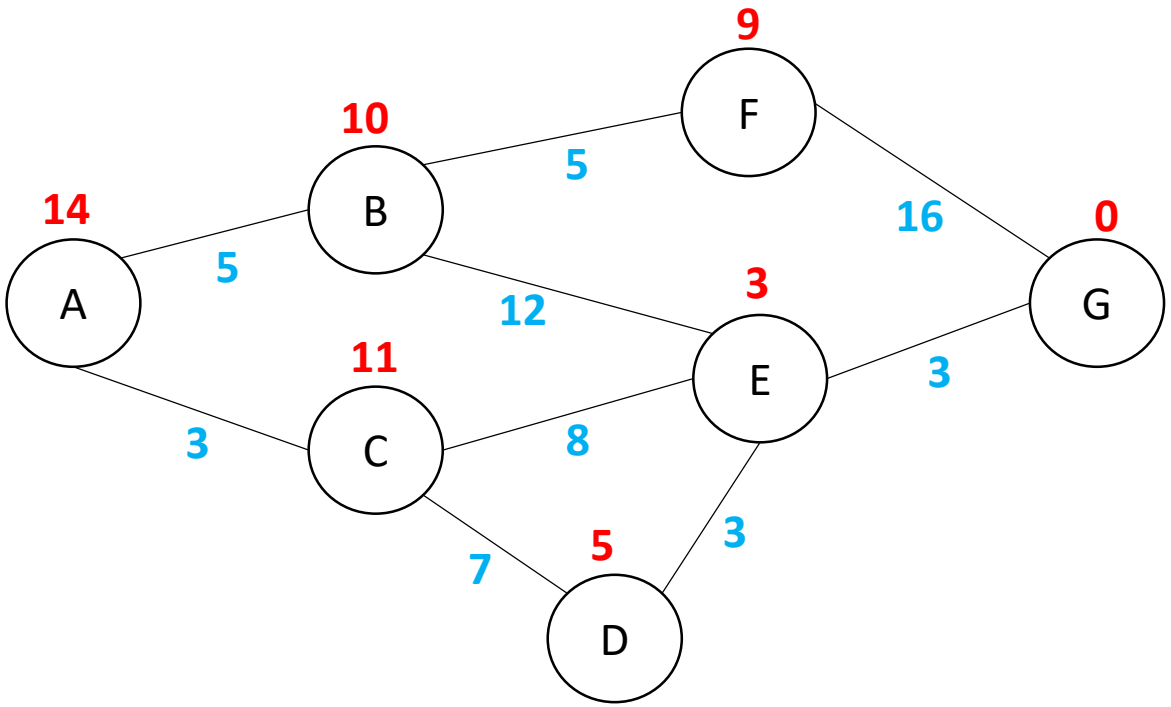
IMPLEMENTAR:

Métodos *compute(…)* das classes *HeuristicTileDistance* e *HeuristicTilesOutOfPlace*

Considere o seguinte exemplo:

vermelho – valor da heurística

azul – custo do caminho



Procura Sôfrega (*GreedyBestFirstSearch*)

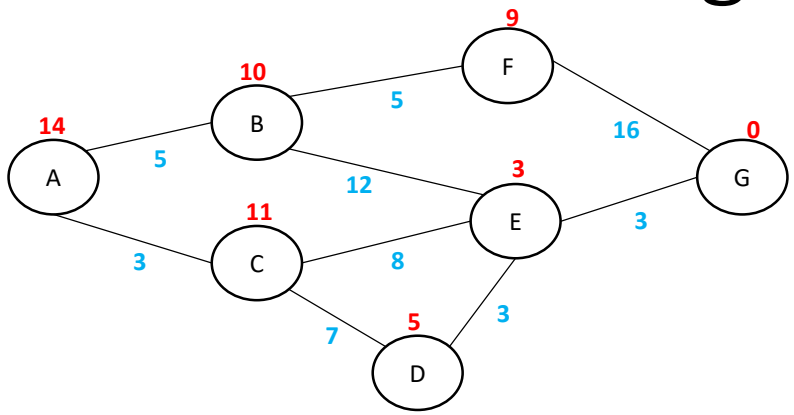
$$f = h(n)$$

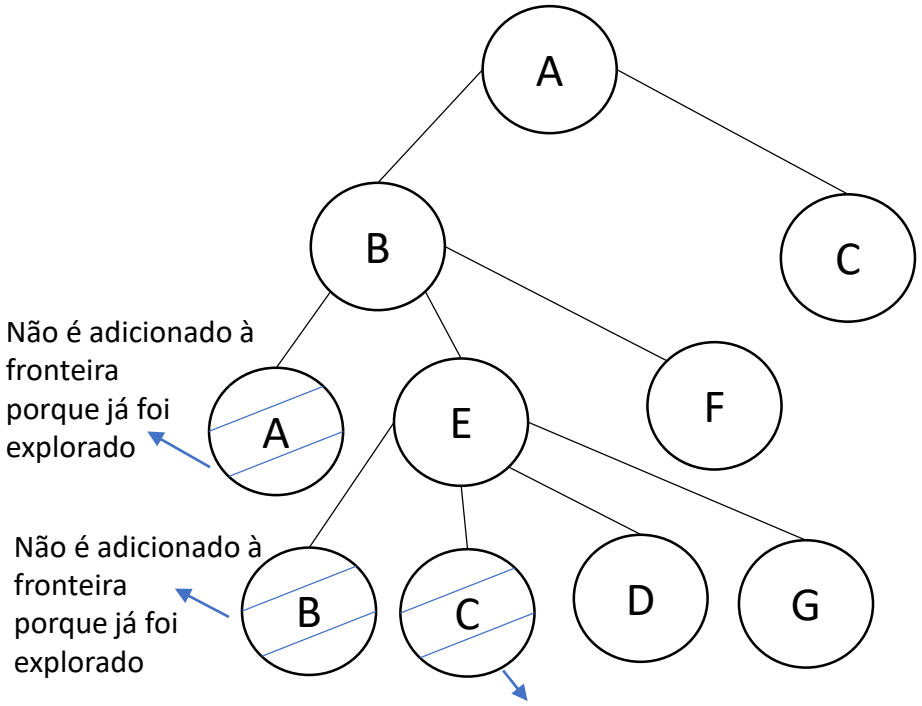
- Não é ótimo
- Tal como na procura uniforme utiliza uma lista de nós explorados
- Um nó só é adicionado à fronteira se o estado correspondente:
  - Não esteja já na fronteira e não tenha sido explorado
  - Já está na fronteira mas o custo do caminho até chegar a esse nó a partir do nó inicial (ou seja, 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)*)

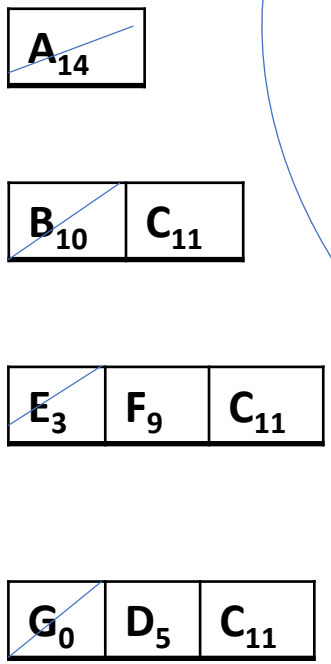
Procura Sôfrega (*GreedyBestFirstSearch*)

$f = h(n)$

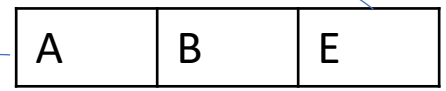




- Fronteira:



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



- Nós explorados:

A

B

E

G

IMPLEMENTAR:

Método addSuccessorsToFrontier(…)

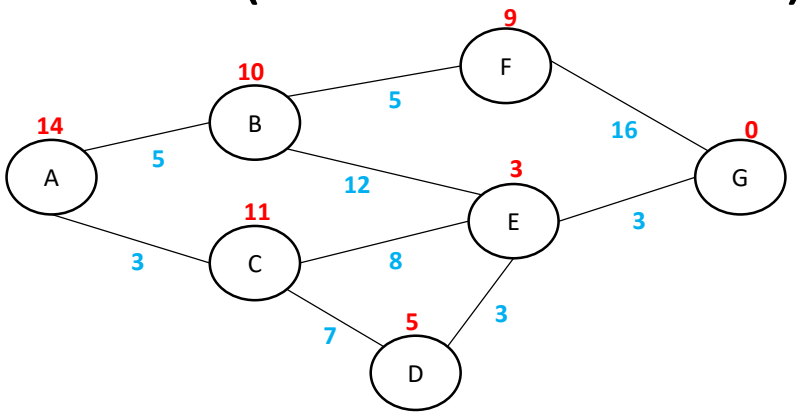
da classe *GreedyBestFirstSearch*

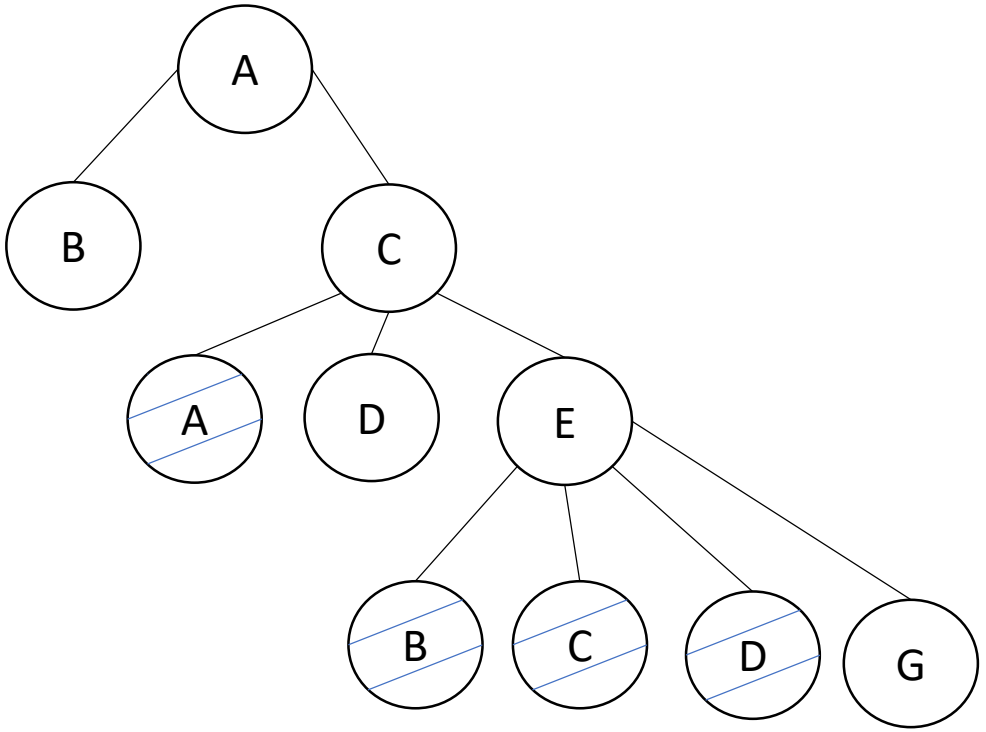
A*(*AStarSearch*)

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

- 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

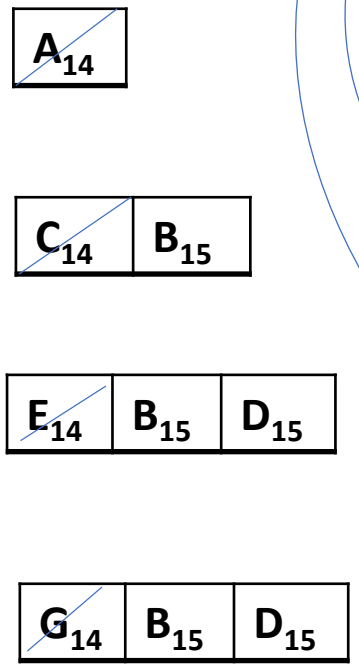
A*(*AStarSearch*)



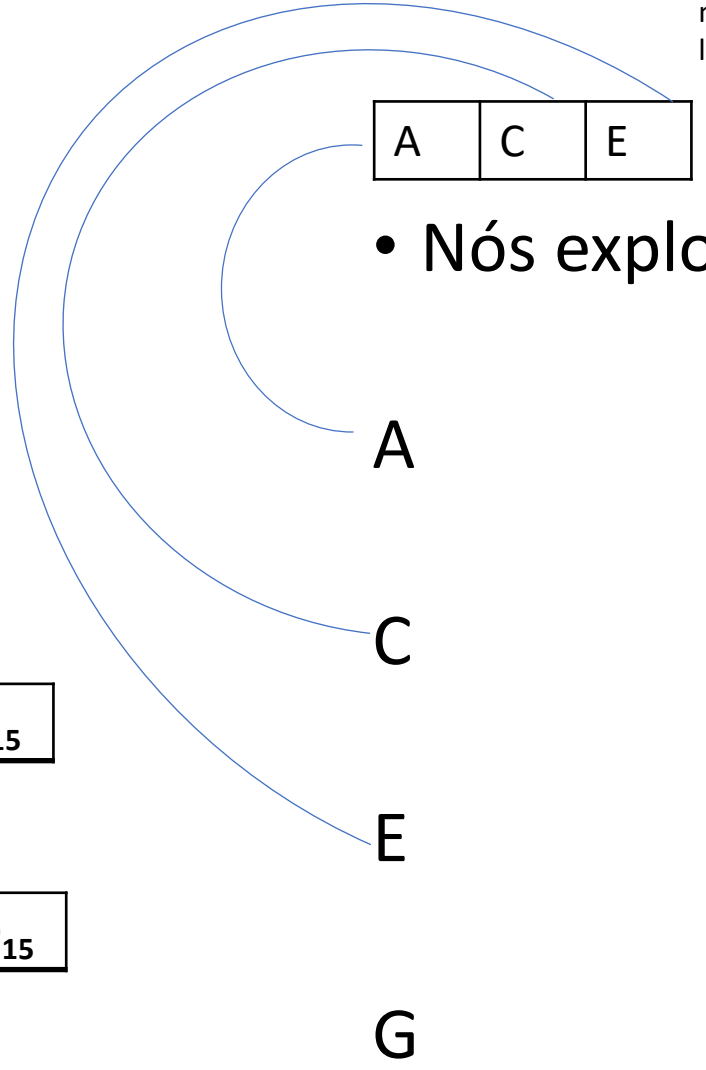


$$f = g(n) + h(n)$$

- Fronteira:



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



- Nós explorados:

Nós sucessores não são adicionados à fronteira, ou porque estão na lista de nós explorados ou porque já estão na fronteira

IMPLEMENTAR:

Método addSuccessorsToFrontier(…)

da classe *A**

Procura em Feixe (*BeamSearch*)

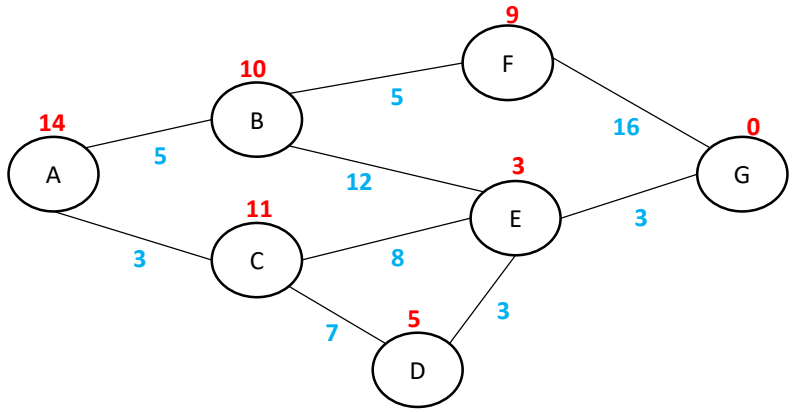
$$f = g(n) + h(n)$$

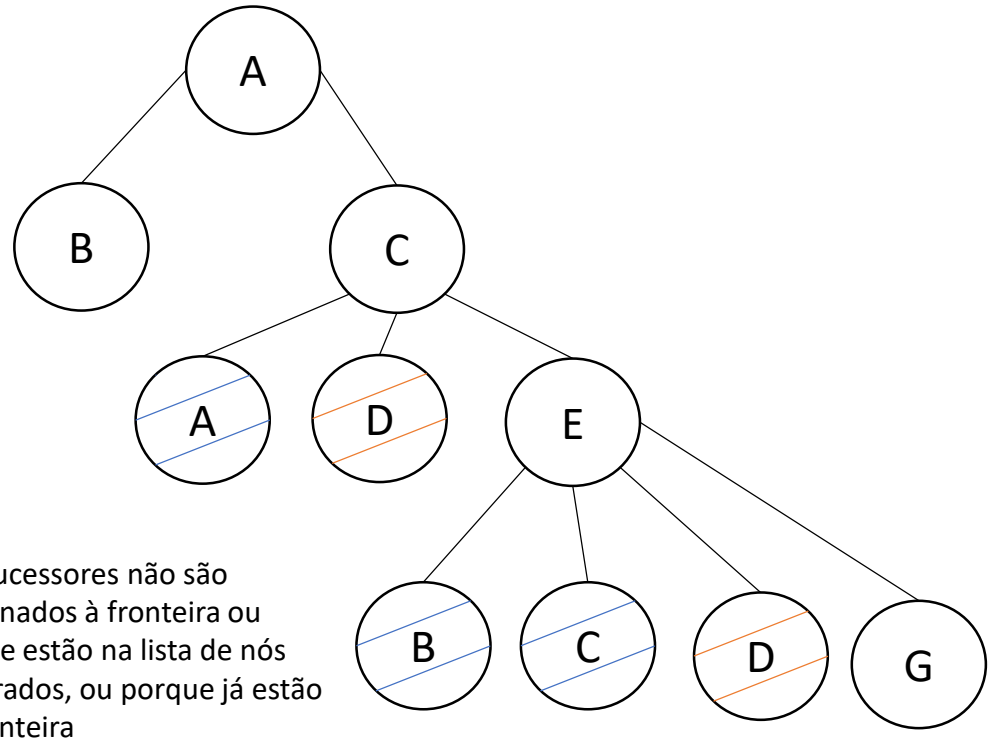
- Não é completo e não é ótimo
- Idêntico ao A*, mas com imposição de um limite máximo para o tamanho da fronteira (*beamSize*)
- Após adicionarem os sucessores à fronteira devem verificar se o seu tamanho excede o *beamSize* e caso isso aconteça:
  - 1 – criar uma lista auxiliar (do tipo *NodePriorityQueue*)
  - 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 raíz uma *PriorityQueue* com o método *removeLast()*

Procura em Feixe (*BeamSearch*)

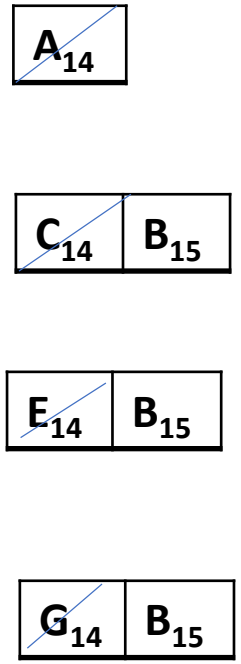
$f = g(n) + h(n)$ *beamSize=2*

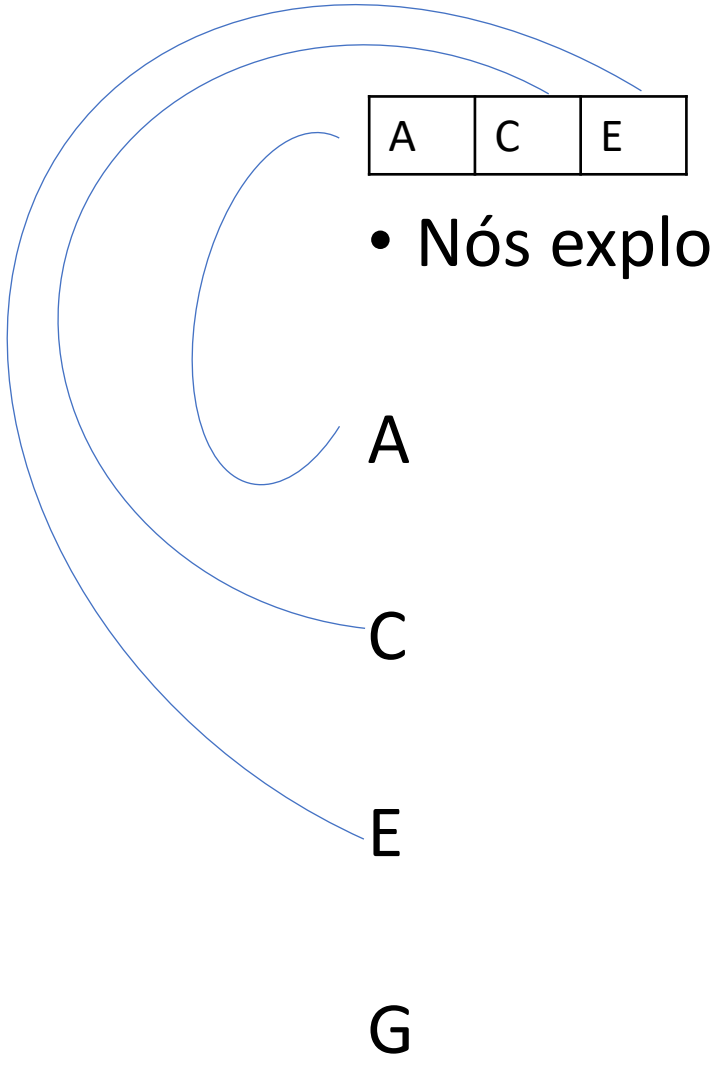




Nós sucessores não são adicionados à fronteira ou porque estão na lista de nós explorados, ou porque já estão na fronteira

- Fronteira:





- Nós explorados:

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

IMPLEMENTAR:

Método addSuccessorsToFrontier(…)
da classe *BeamSearch*

IDA* (*IDAStarSearch*)

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

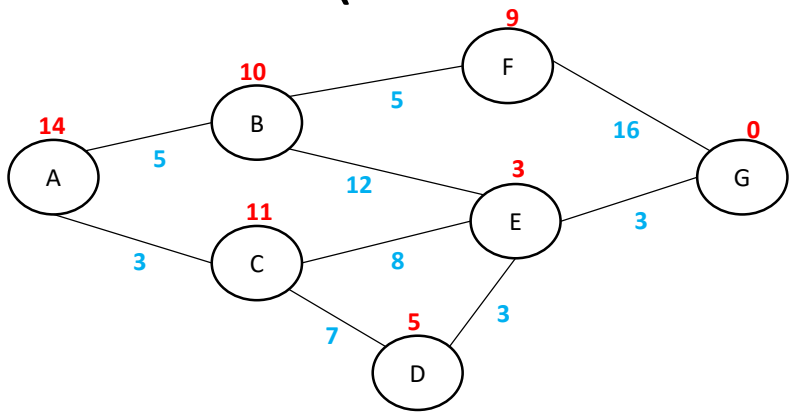
- É ótimo
- Tal como o método de procura em profundidade não tem lista de nós explorados
- O método de procura é aplicado sucessivas vezes impondo um limite

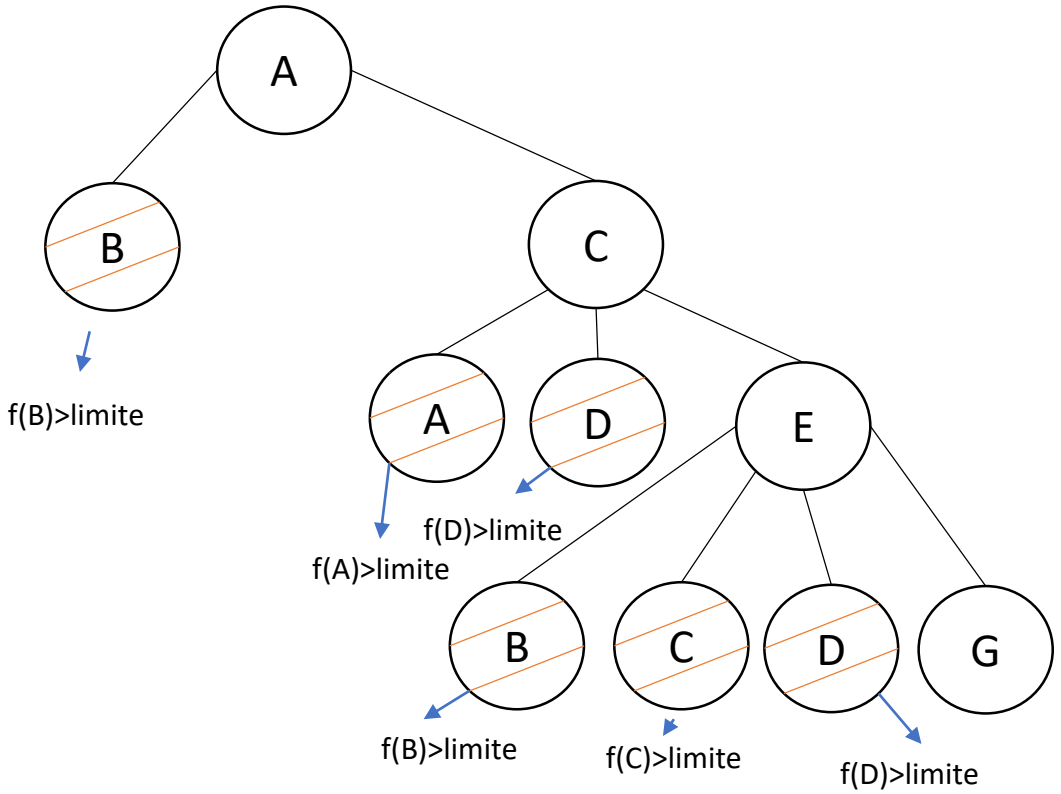
(*limiteInicial* = *f* do nó inicial, próximos limites = menor valor de *f* entre os sucessores não adicionados à fronteira)

- Um nó só é adicionado à fronteira se o estado correspondente:
  - Não esteja já na fronteira, o valor de *f(n)<=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*(*IDAStarSearch*) – Exemplo 1

$f = g(n) + h(n)$ *limiteInicial=f(A) =14*





- Fronteira:

A_{14}

C_{14}

E_{14}

G_{14}

Menor *f* entre nós sucessores não adicionados à fronteira

15

- Nós explorados:

A

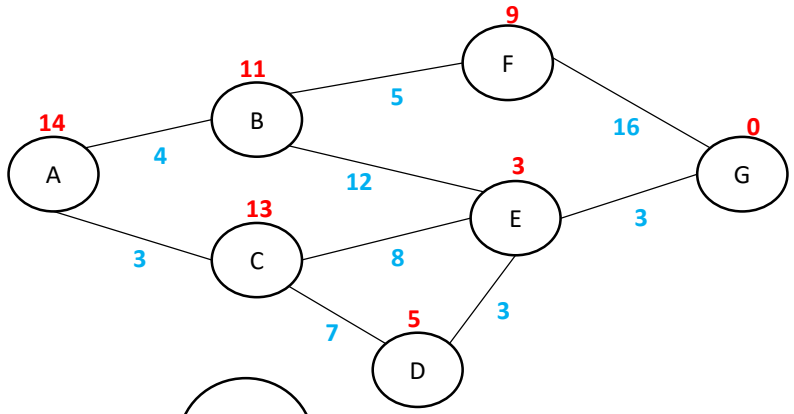
C

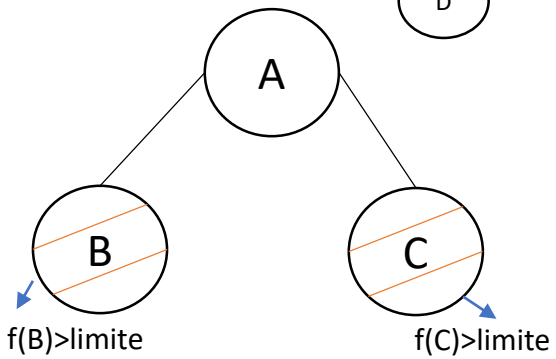
E

G

IDA*(*IDAStarSearch*) – Exemplo 2

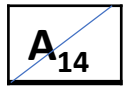
$f = g(n) + h(n)$ *limiteInicial=f(A) =14*





limite =14

- Fronteira:



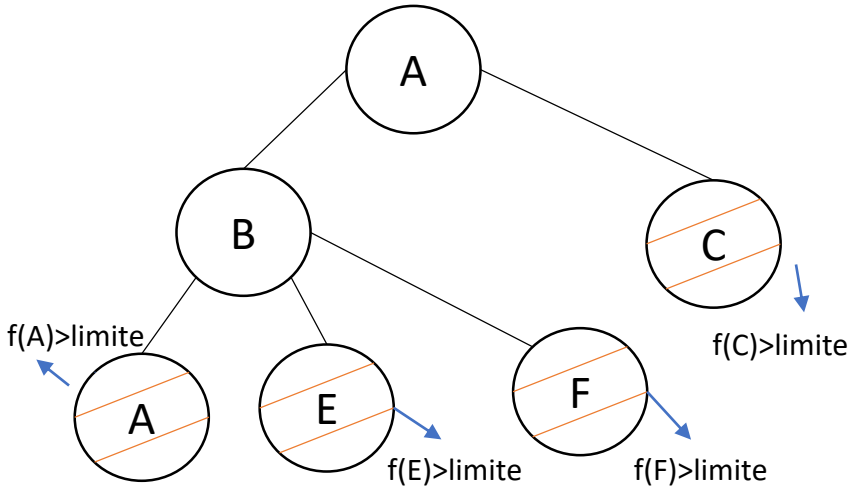
Menor *f* entre nós sucessores não adicionados à fronteira

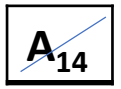
15

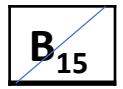
- Nós explorados:

A

limite=15







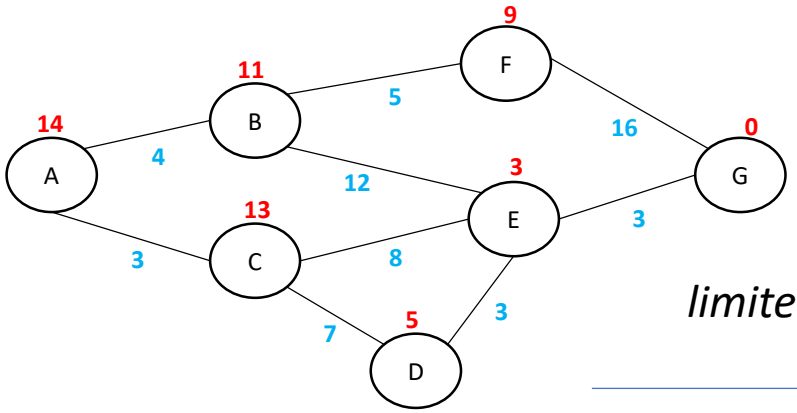
Menor *f* entre nós sucessores não adicionados à fronteira

16

A

B

IDA*(*IDAStarSearch*) – Exemplo 2 (continuação)



limite =14

- Fronteira: …
- Nós explorados:

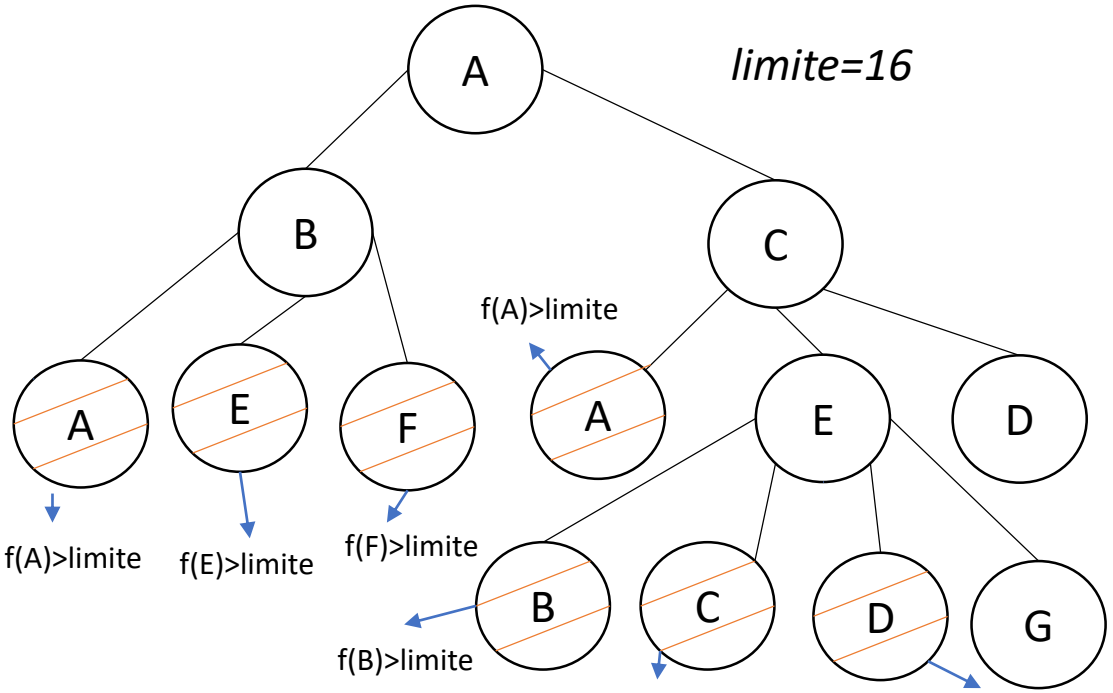
A

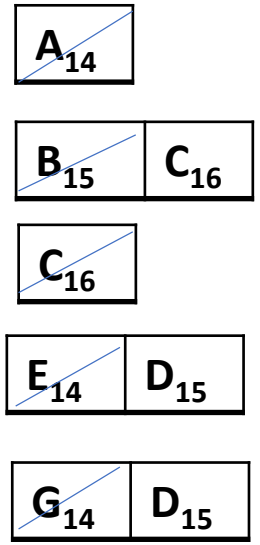
limite=15

- Fronteira: …
- Nós explorados:

A
B

limite=16





Menor *f* entre nós sucessores não adicionados à fronteira

18

- Nós explorados:

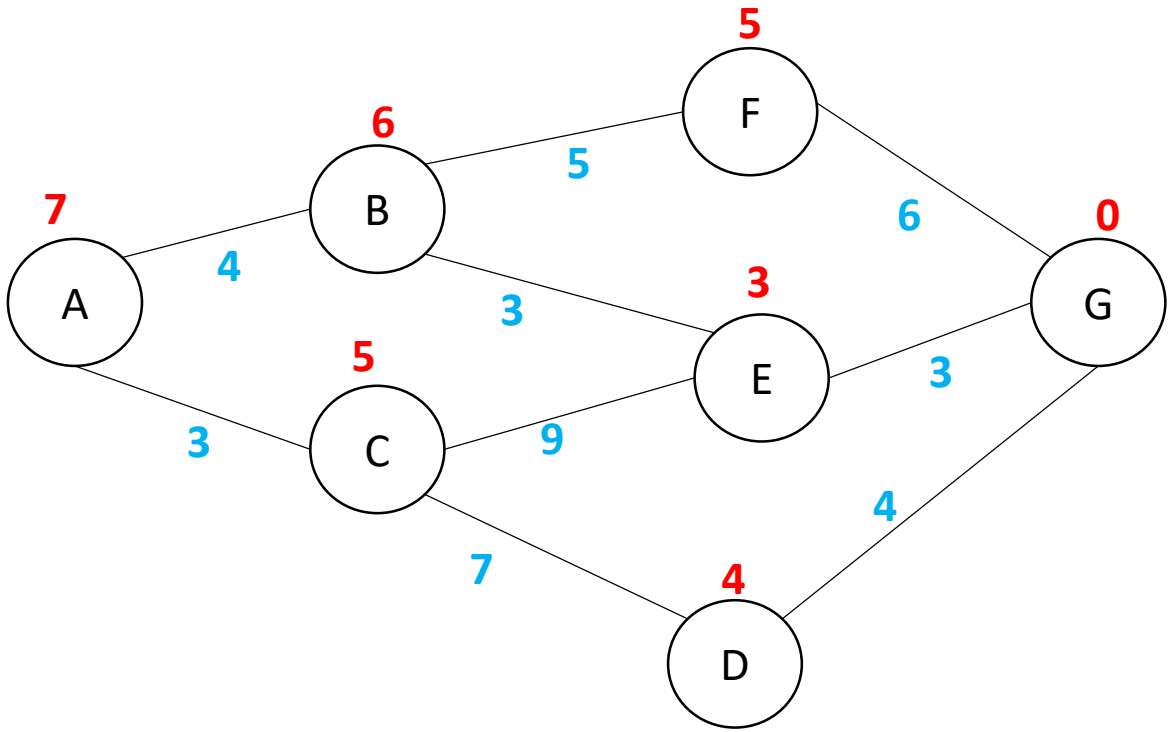
A
B
C
E
G

Já está na fronteira e custo (g) é superior (14) ao que já lá está (10)

IMPLEMENTAR:

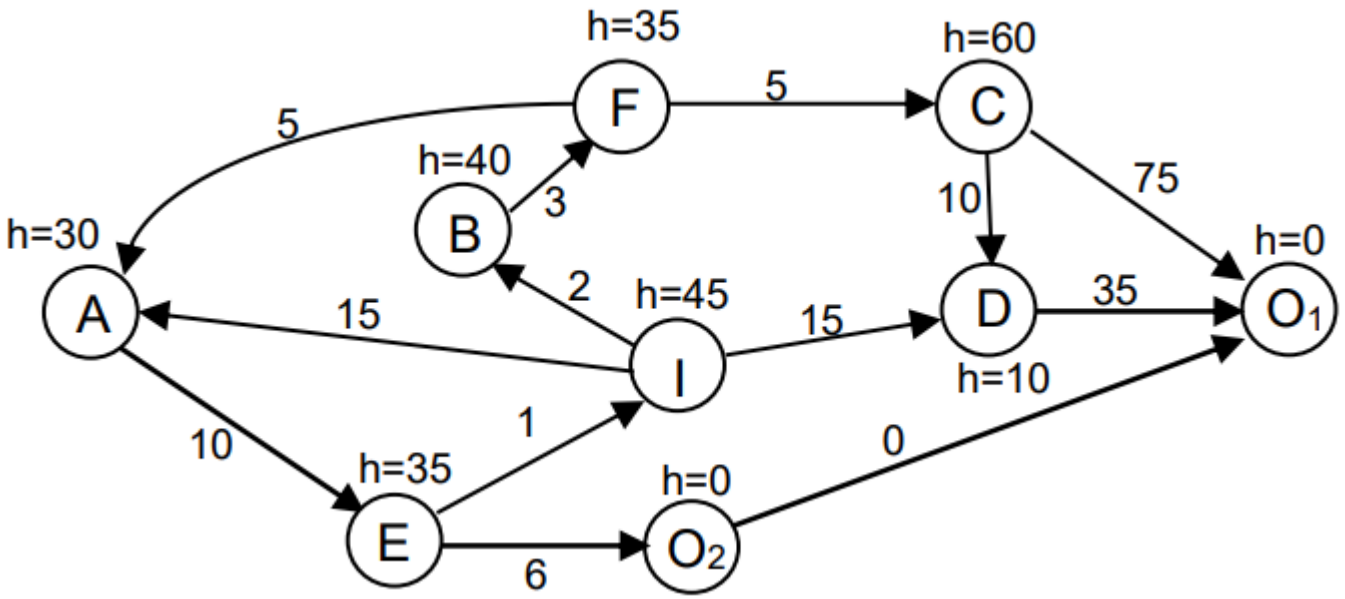
Métodos da classe *IDAStarSearch*
(só se houver tempo)

Exercício para estudo autónomo:



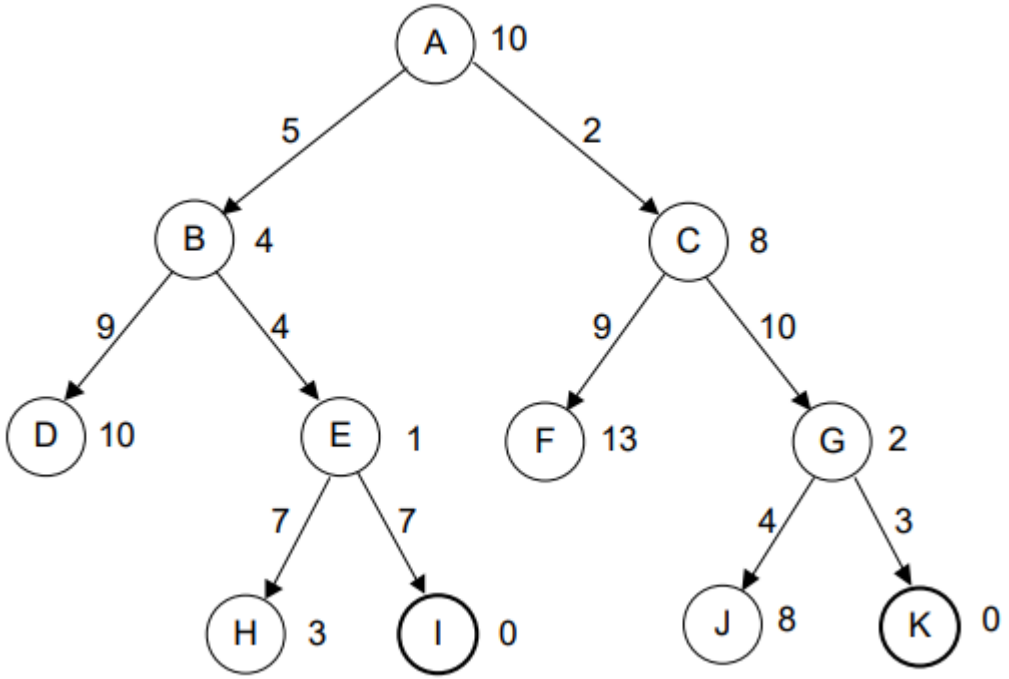
Nó inicial - A

Exercício para estudo autónomo:



Nó inicial – I
Nós objetivos – O1/O2

Exercício para estudo autónomo:



Nós objetivos – I/K

Bons estudos!