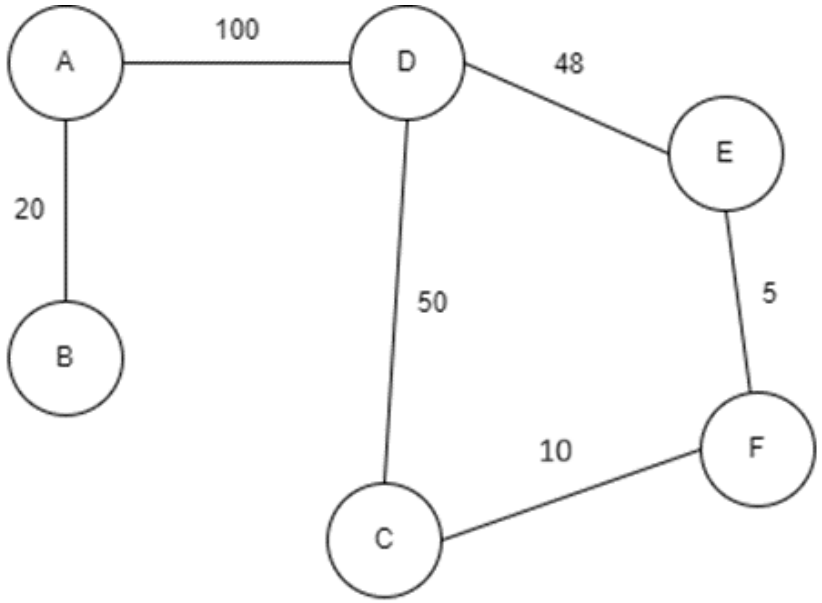
Exercícios

Daniel Nogueira

dnogueira@ipca.pt

1. Nos grafos a seguir, identifique todos os vértices e conexões possíveis (representação em lista) e os pesos das conexões (representação em matriz).

a)

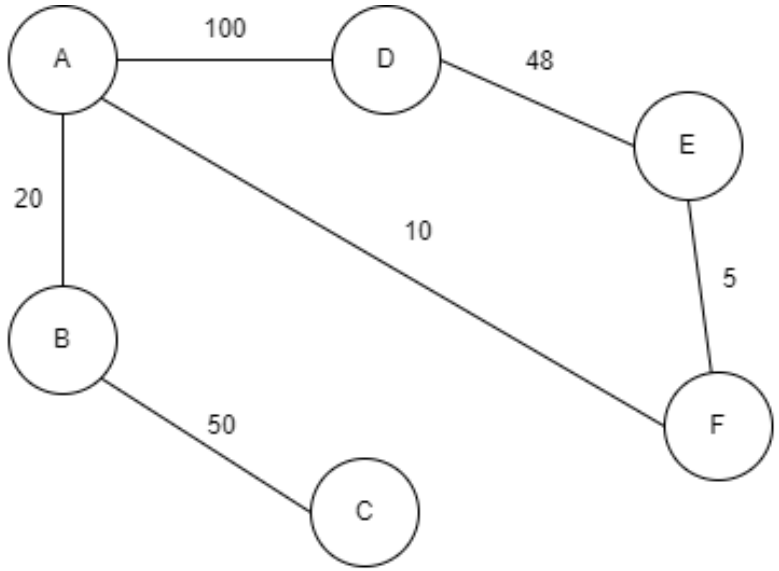


NÓ	CONEXÃO
A	B, D
B	A
C	D, F
D	A, C, E
E	D, F
F	C, E

	A	B	C	D	E	F
A	0	20	0	100	0	0
B	20	0	0	0	0	0
C	0	0	0	50	48	10
D	100	0	50	0	48	0
E	0	0	0	48	0	5
F	0	0	10	0	5	0

1. Nos grafos a seguir, identifique todos os vértices e conexões possíveis (representação em lista) e os pesos das conexões (representação em matriz).

b)

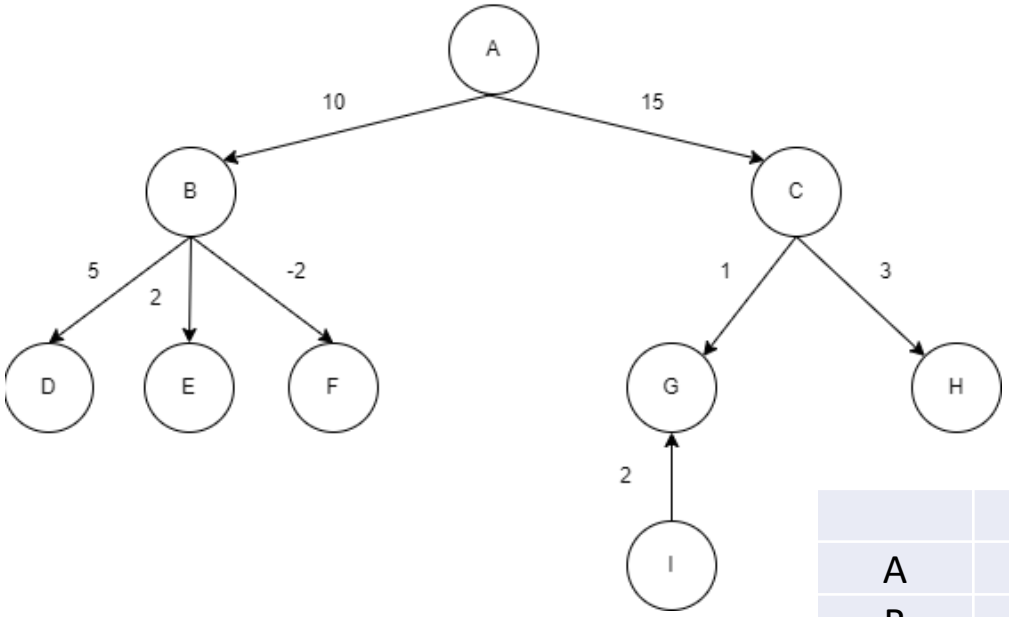


NÓ	CONEXÃO
A	B, D, F
B	A, C
C	B
D	A, E
E	D, F
F	A, E

	A	B	C	D	E	F
A	0	20	0	100	0	10
B	20	0	50	0	0	0
C	0	50	0	0	0	0
D	100	0	0	0	48	0
E	0	0	0	48	0	5
F	10	0	0	0	5	0

1. Nos grafos a seguir, identifique todos os vértices e conexões possíveis (representação em lista) e os pesos das conexões (representação em matriz).

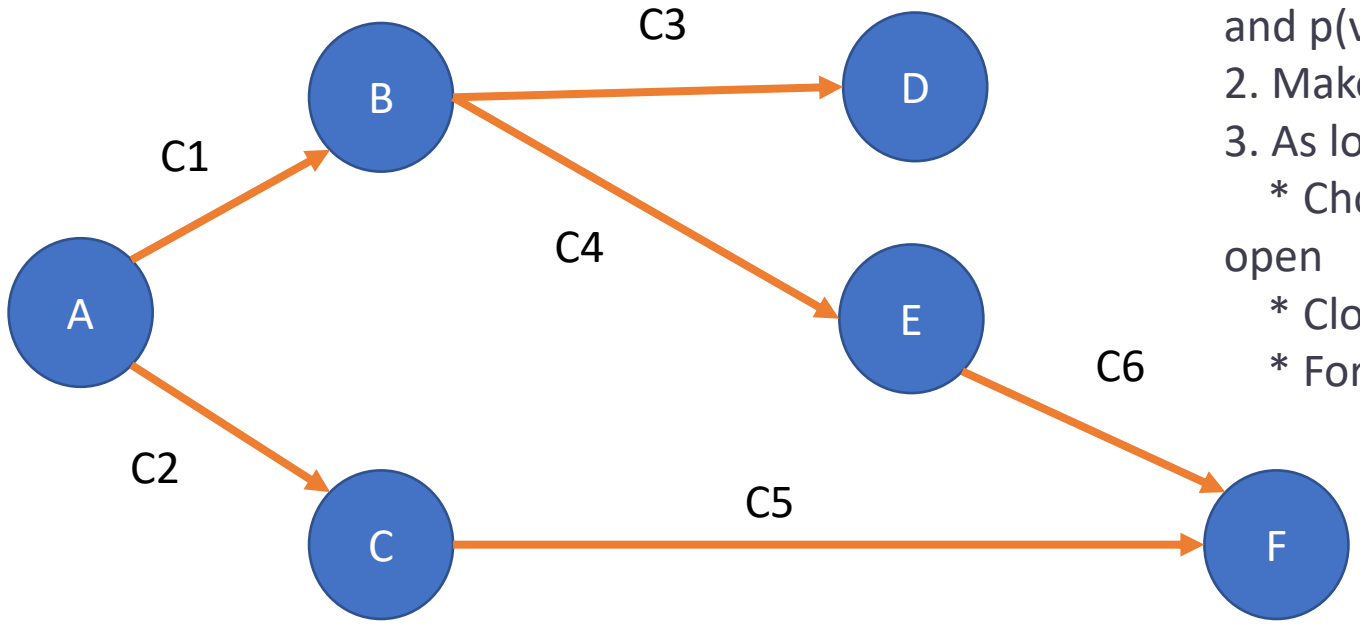
c)



NÓ	CONEXÃO
A	B, C
B	D, E, F
C	G, H
D	----
E	----
F	----
G	----
H	----
I	G

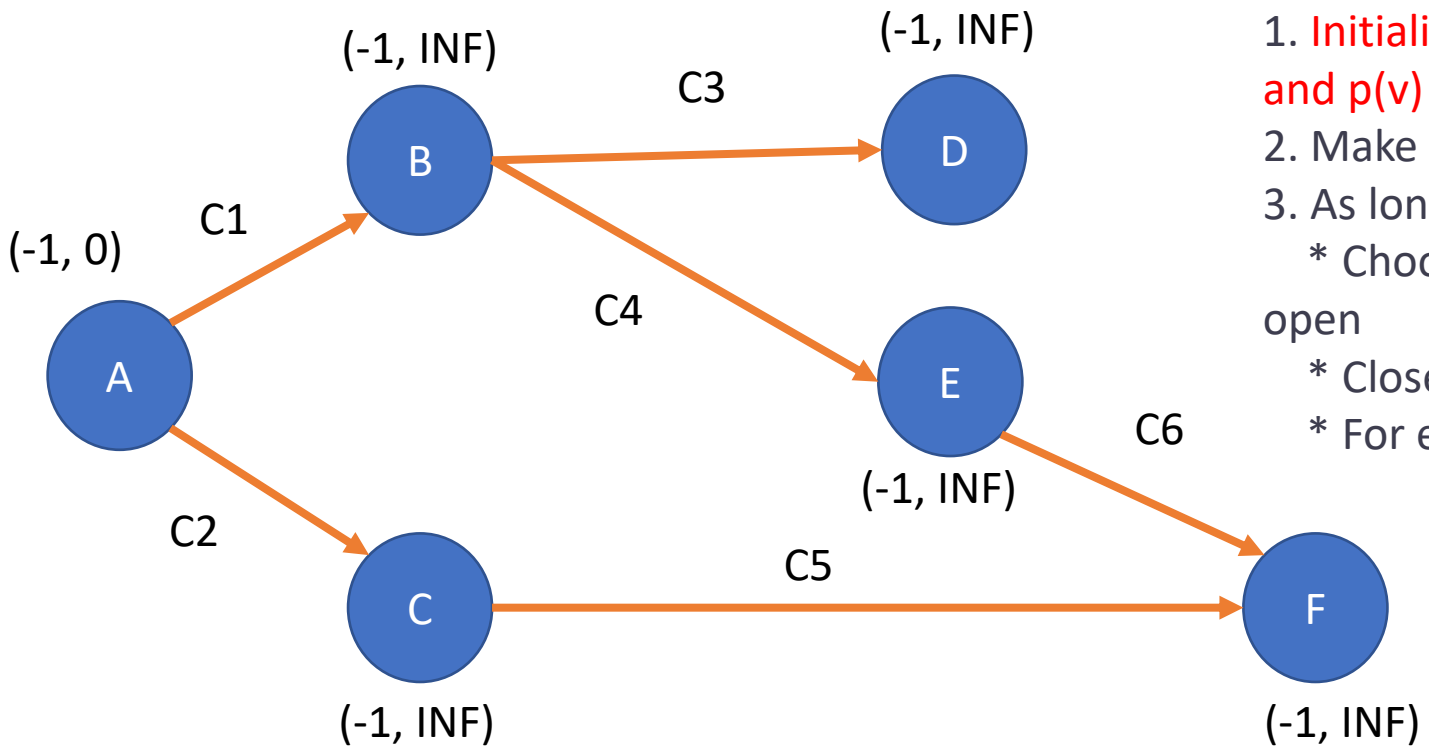
	A	B	C	D	E	F	G	H	I
A	INF	10	15	INF	INF	INF	INF	INF	INF
B	INF	INF	INF	5	2	-2	INF	INF	INF
C	INF	INF	INF	INF	INF	INF	1	3	INF
D	INF	INF	INF	INF	INF	INF	INF	INF	INF
E	INF	INF	INF	INF	INF	INF	INF	INF	INF
F	INF	INF	INF	INF	INF	INF	INF	INF	INF
G	INF	INF	INF	INF	INF	INF	INF	INF	INF
H	INF	INF	INF	INF	INF	INF	INF	INF	INF
I	INF	INF	INF	INF	INF	INF	2	INF	INF

2. Execute o Algoritmo Dijkstra no grafo a seguir (de A a F). Apresente todos os passos da execução do algoritmo e o caminho final escolhido



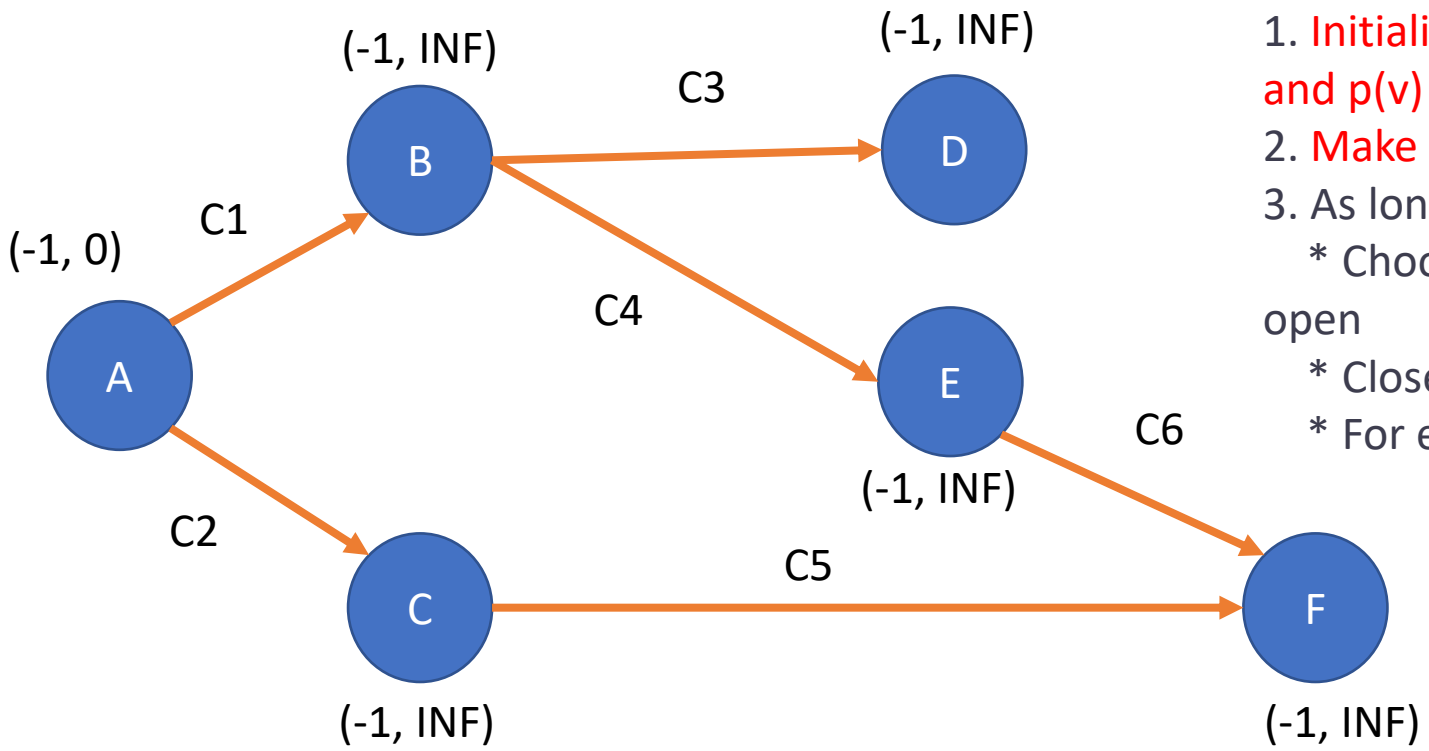
1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

2. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido



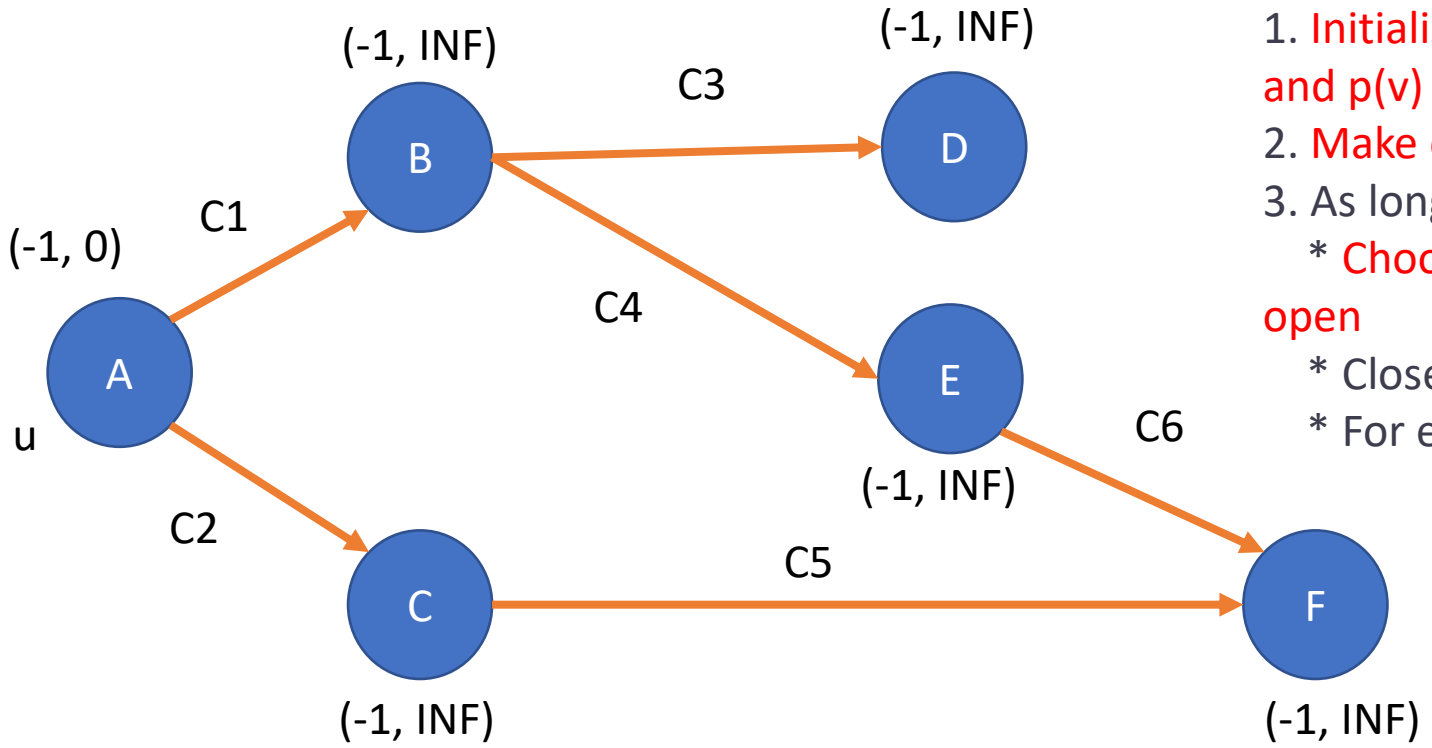
1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

2. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido



1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

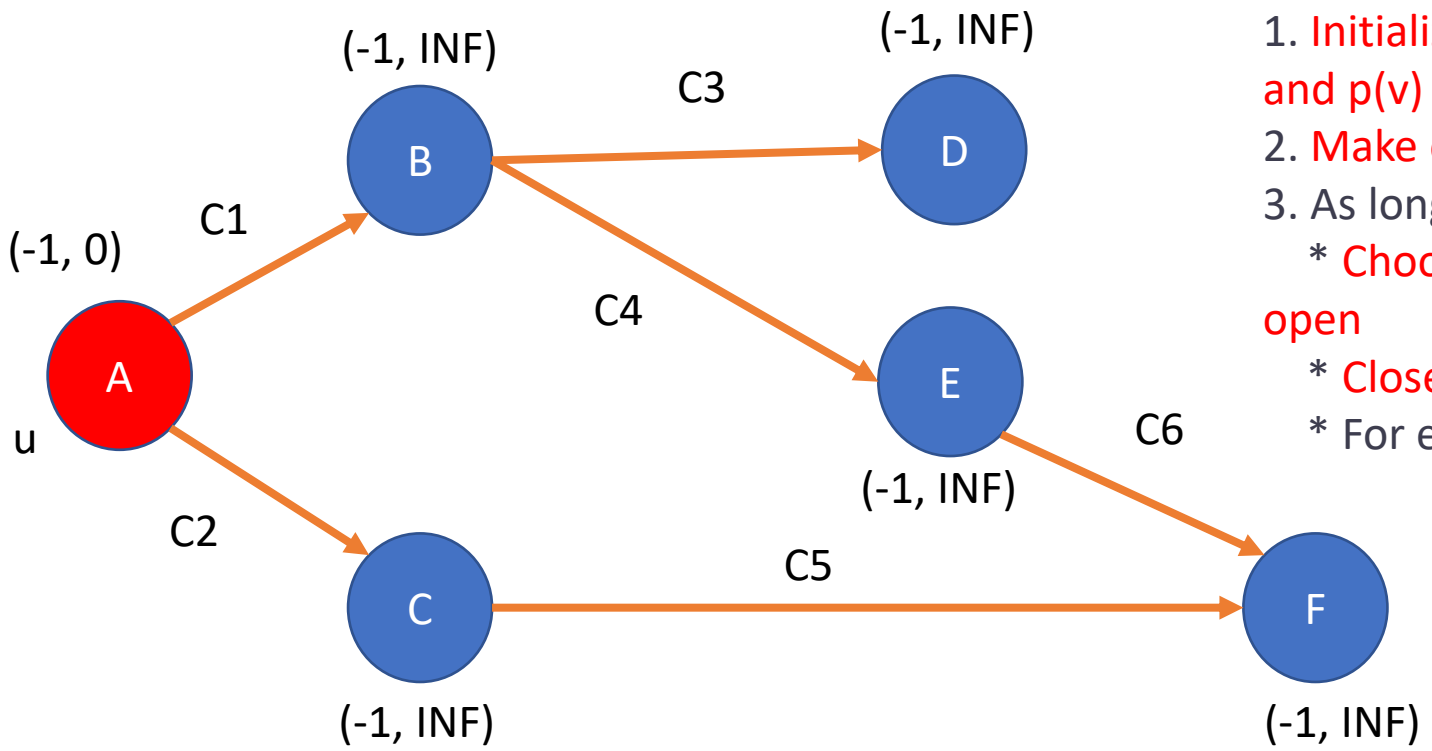
2. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido



1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

Suponha que:
C1 > C2 > C3 > C4 > C5 > C6

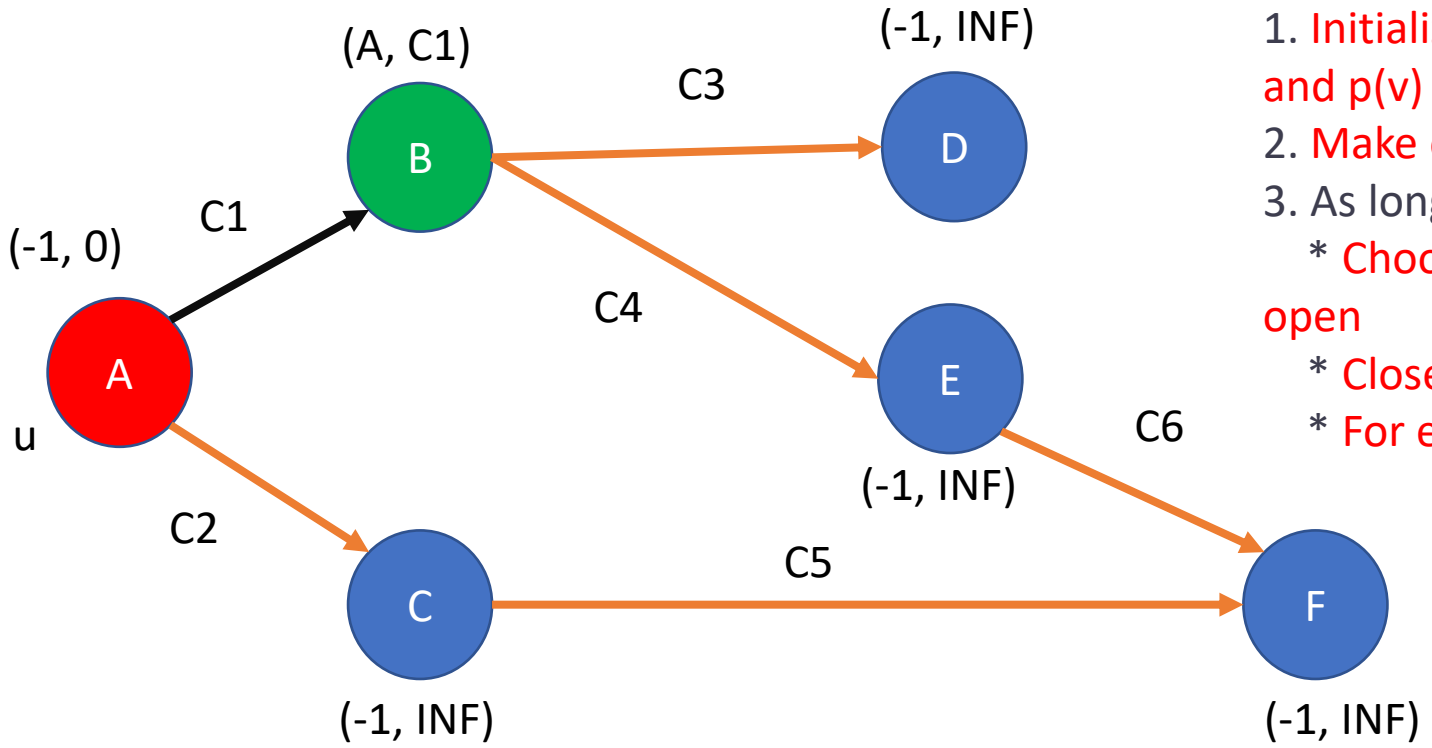
2. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido



1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

Suponha que:
C1 > C2 > C3 > C4 > C5 > C6

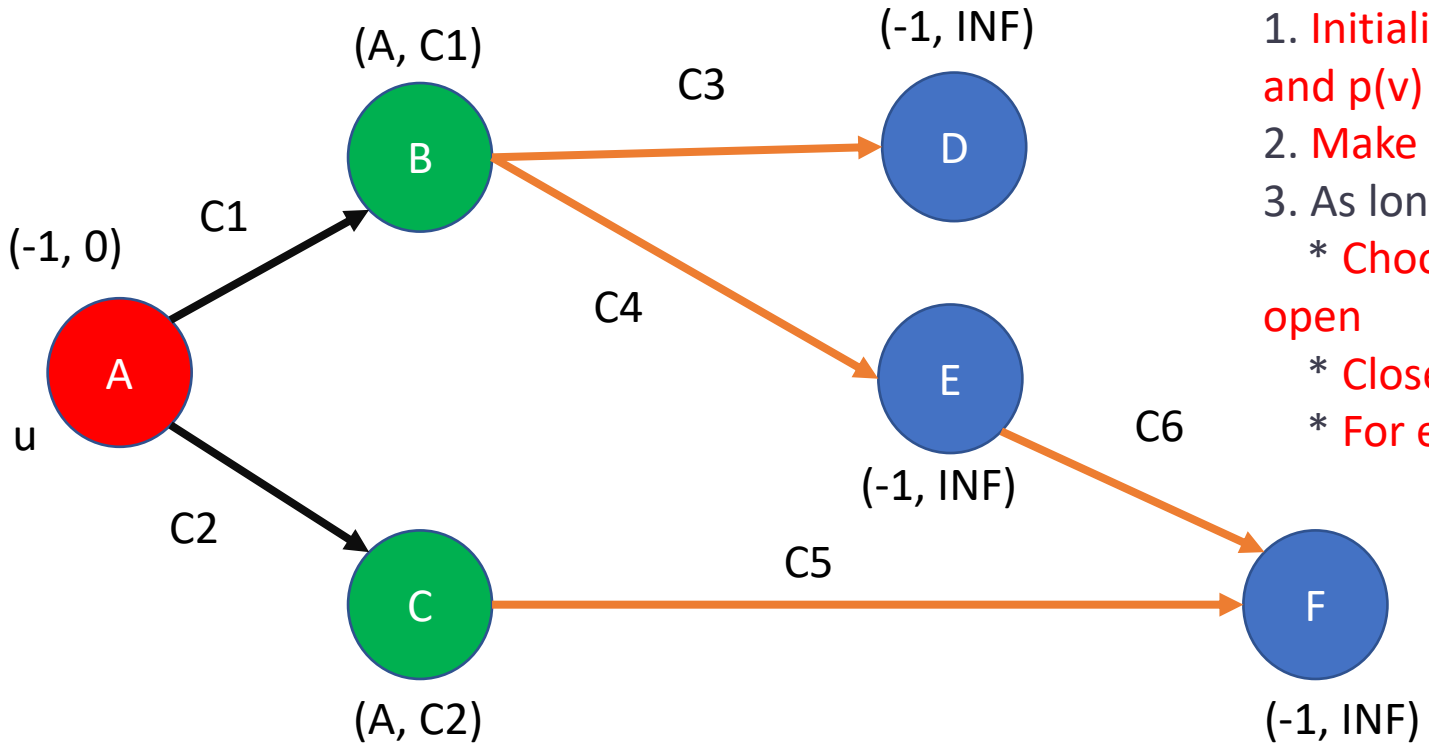
2. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido



1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

Suponha que:
C1 > C2 > C3 > C4 > C5 > C6

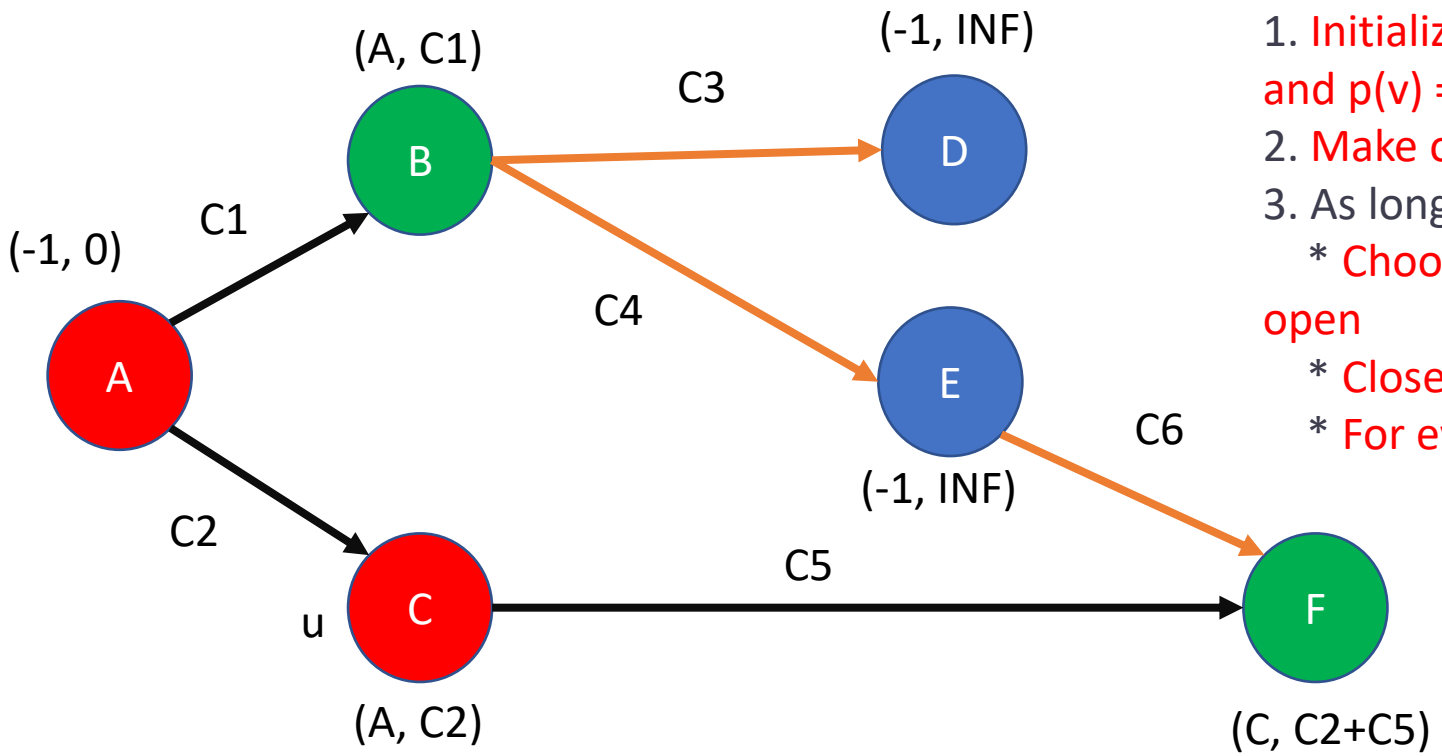
2. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido



1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

Suponha que:
C1 > C2 > C3 > C4 > C5 > C6

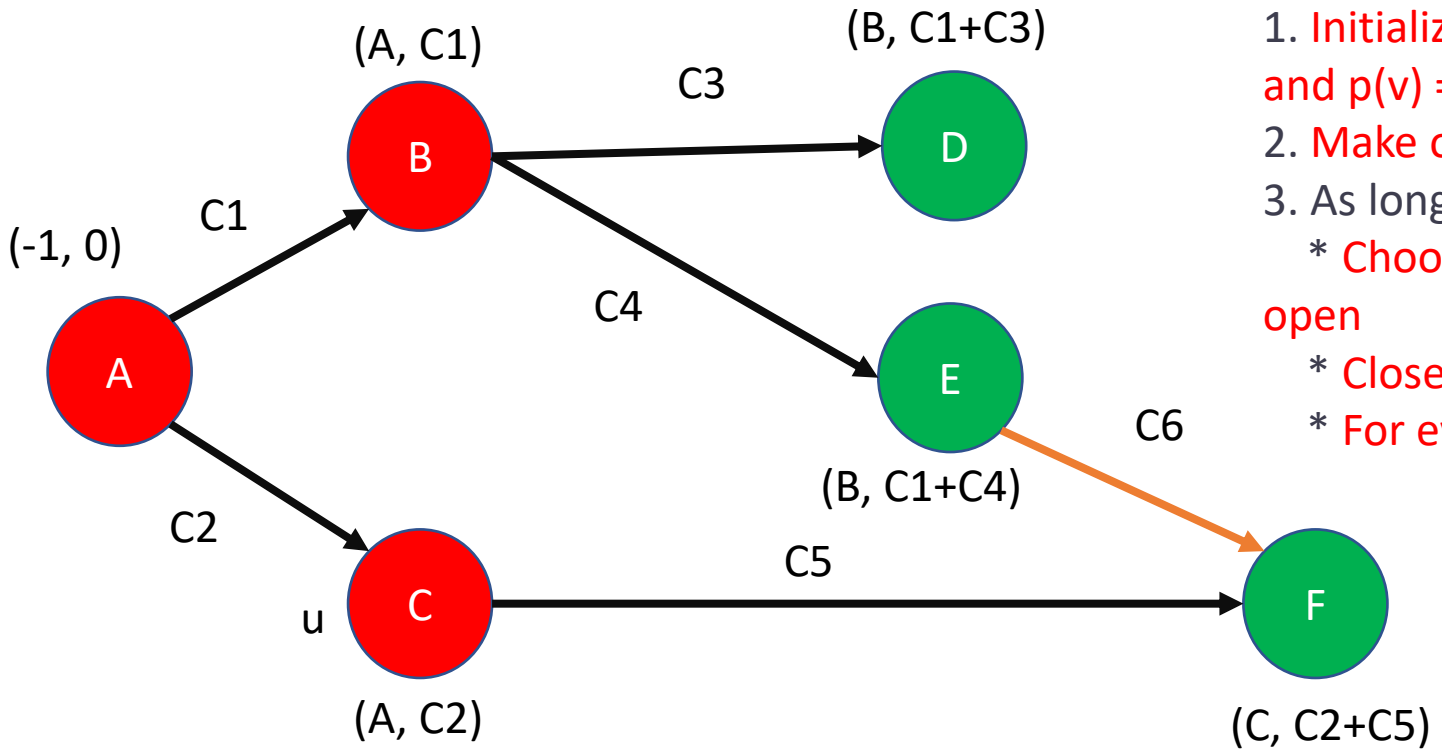
2. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido



1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all v
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose u whose estimate is the smallest among the open
   * Close u
   * For every open node v adjacent to u: relax edge (u,v)

Suponha que:
C1 > C2 > C3 > C4 > C5 > C6

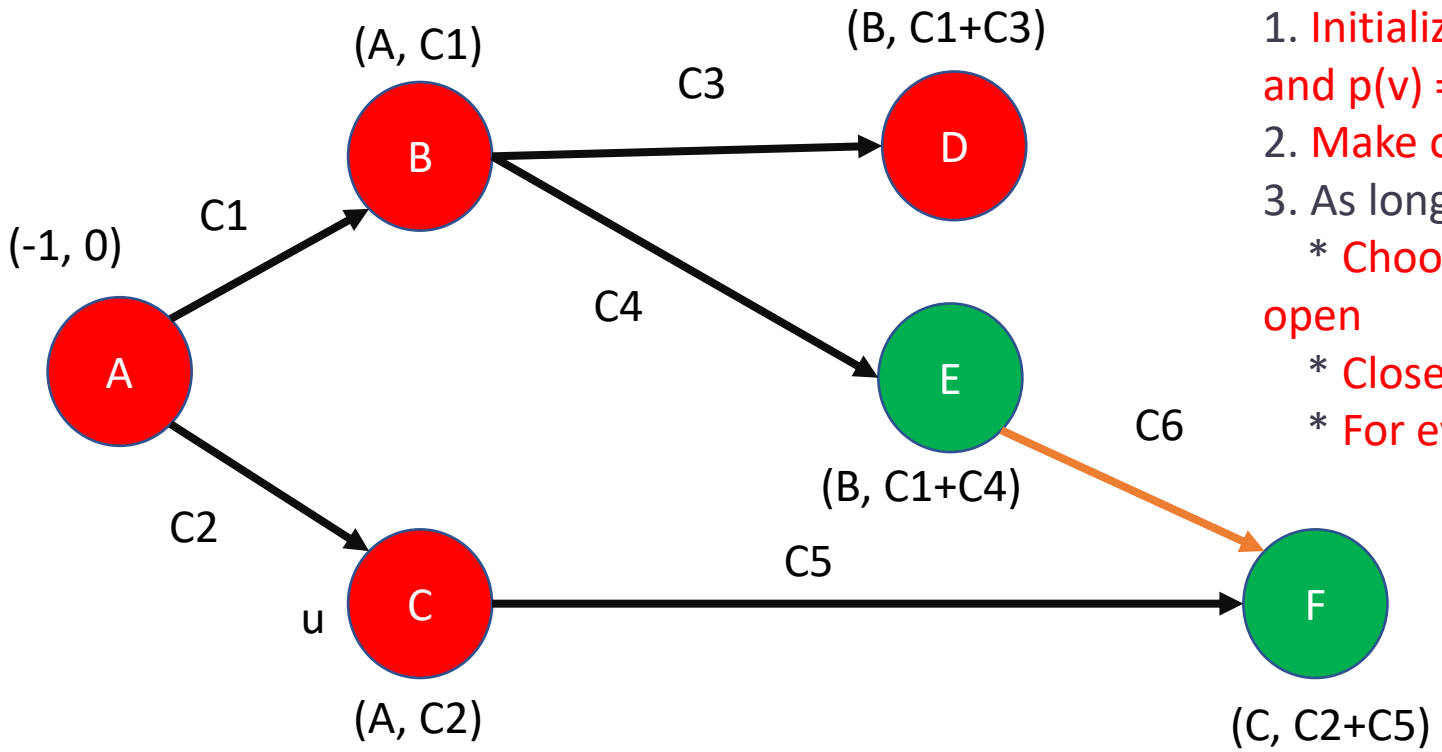
2. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido



1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

Suponha que:
C1 > C2 > C3 > C4 > C5 > C6

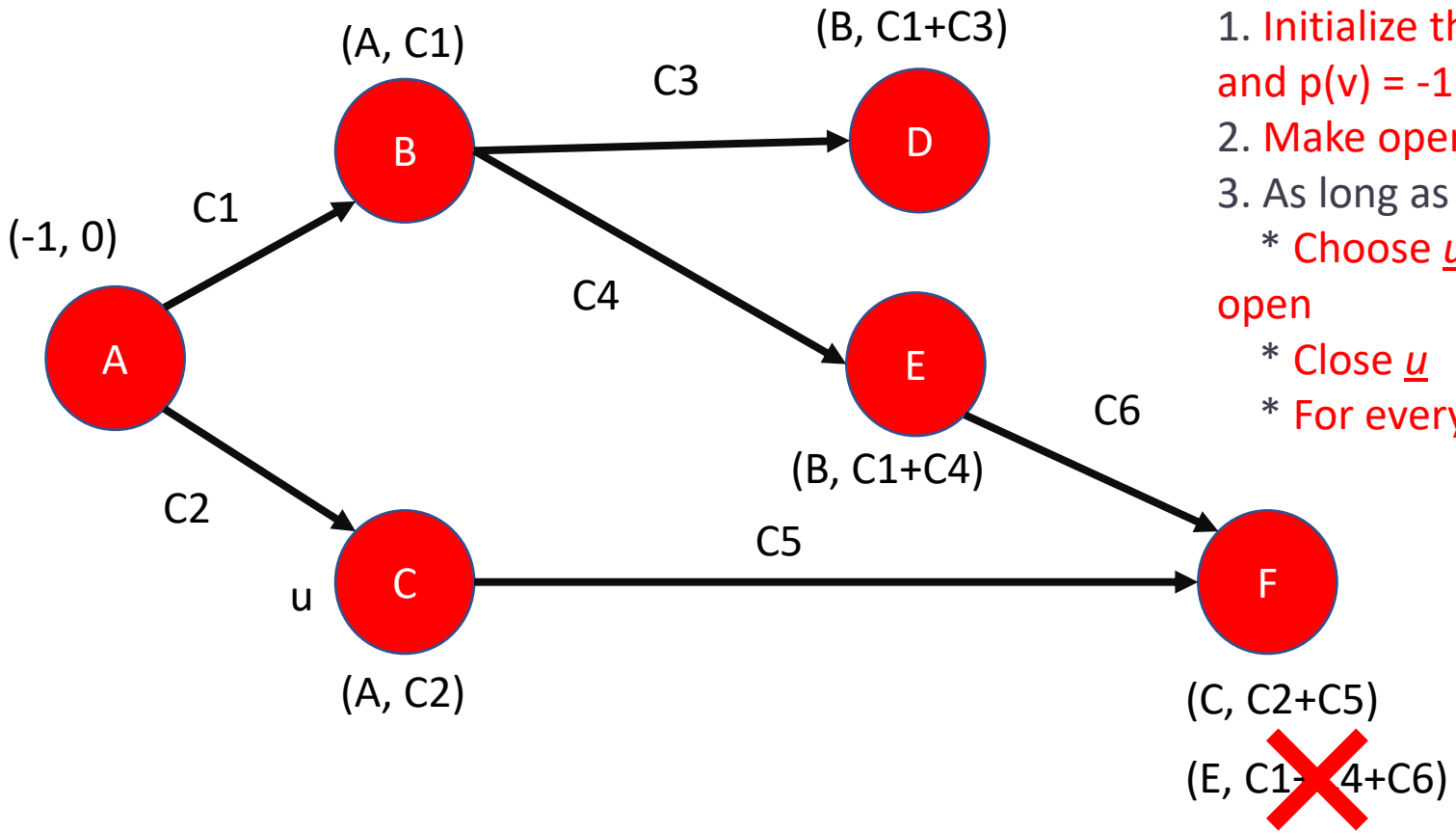
2. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido



1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

Suponha que:
C1 > C2 > C3 > C4 > C5 > C6

2. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido

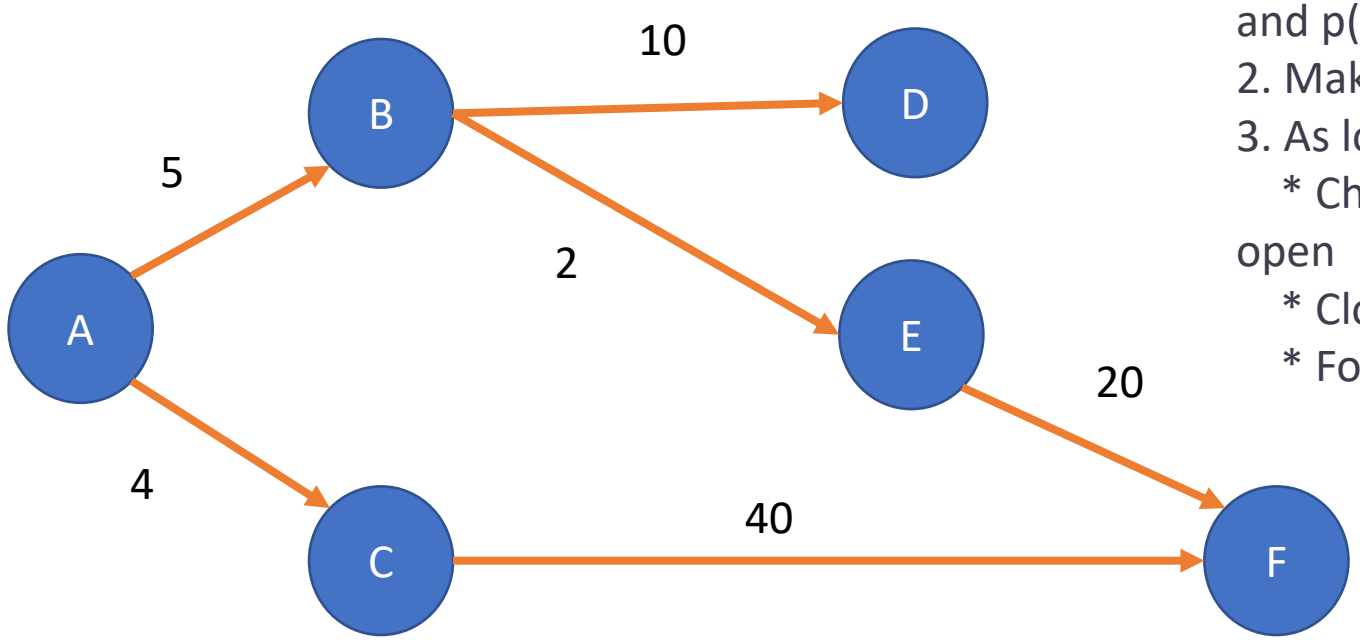


1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all v
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose u whose estimate is the smallest among the open
   * Close u
   * For every open node v adjacent to u: relax edge (u,v)

Suponha que:
C1 > C2 > C3 > C4 > C5 > C6

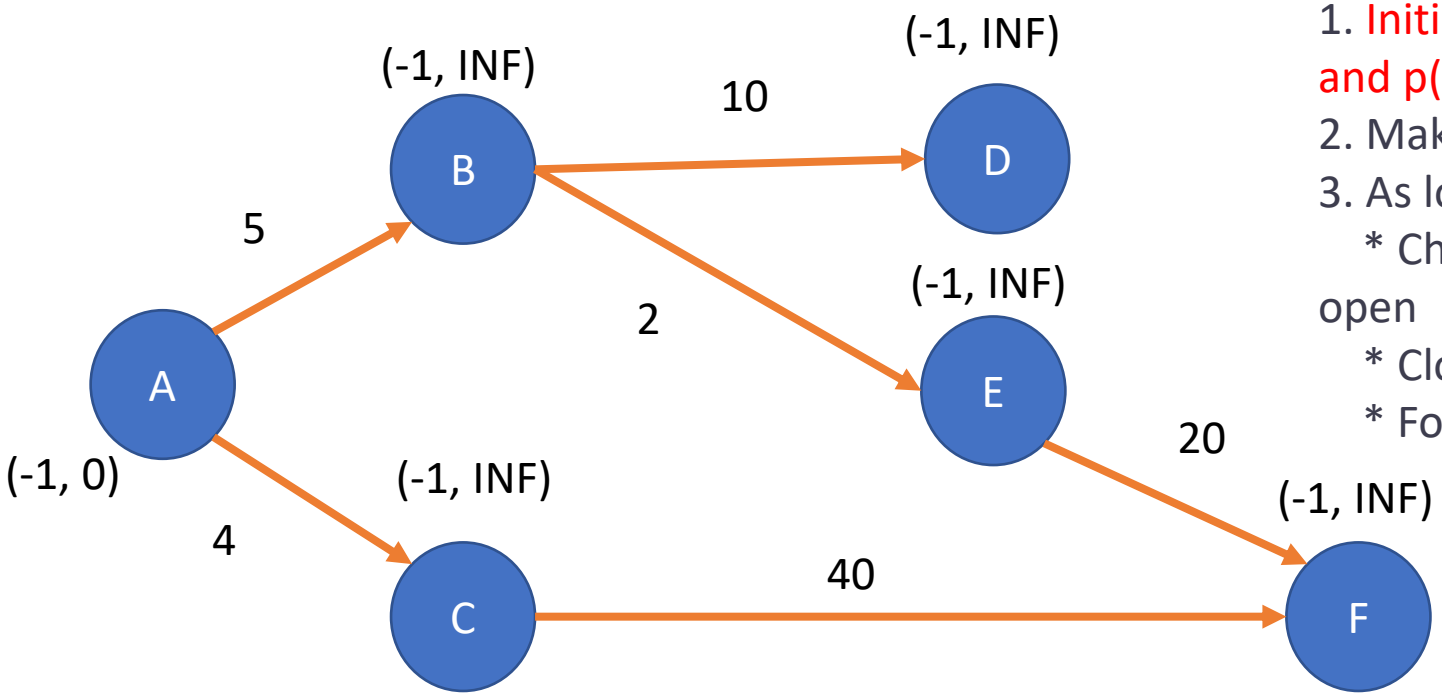
De A a F => A – C – F (Custo total = C2+C5)

3. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido.



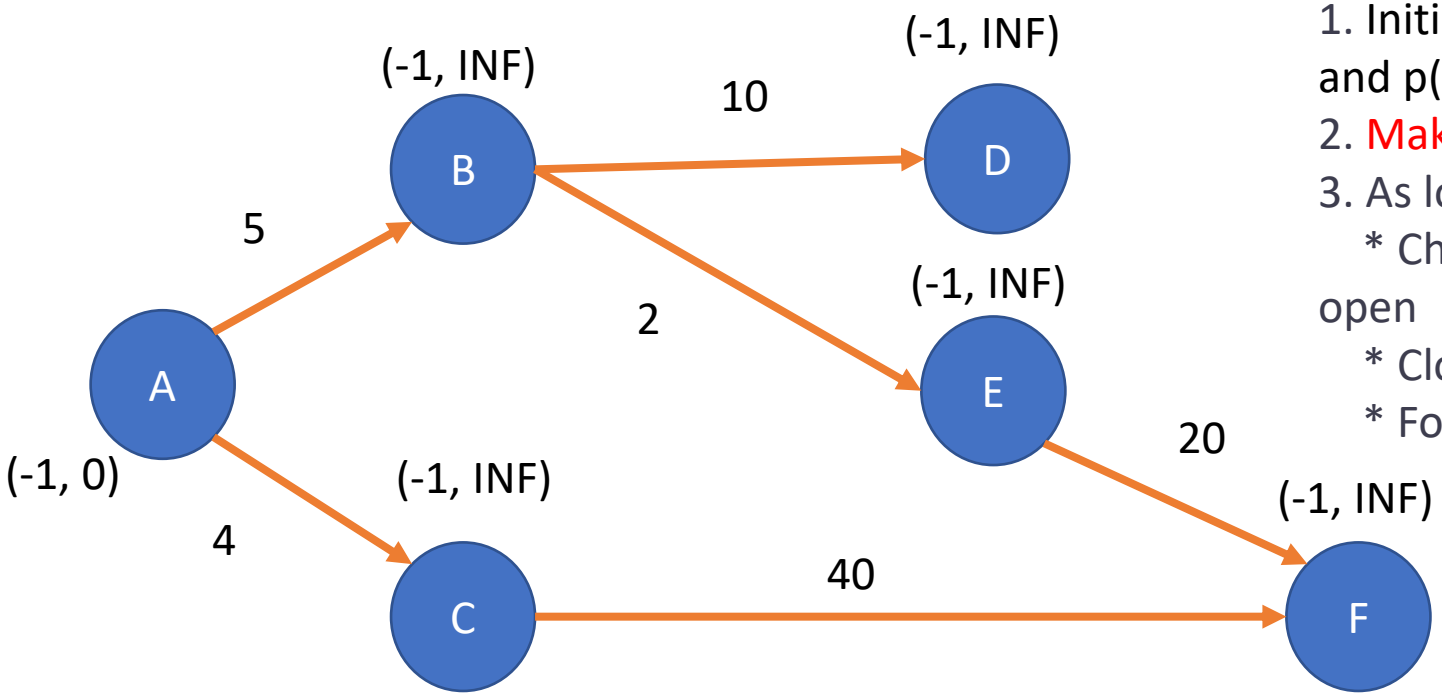
1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

3. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido.



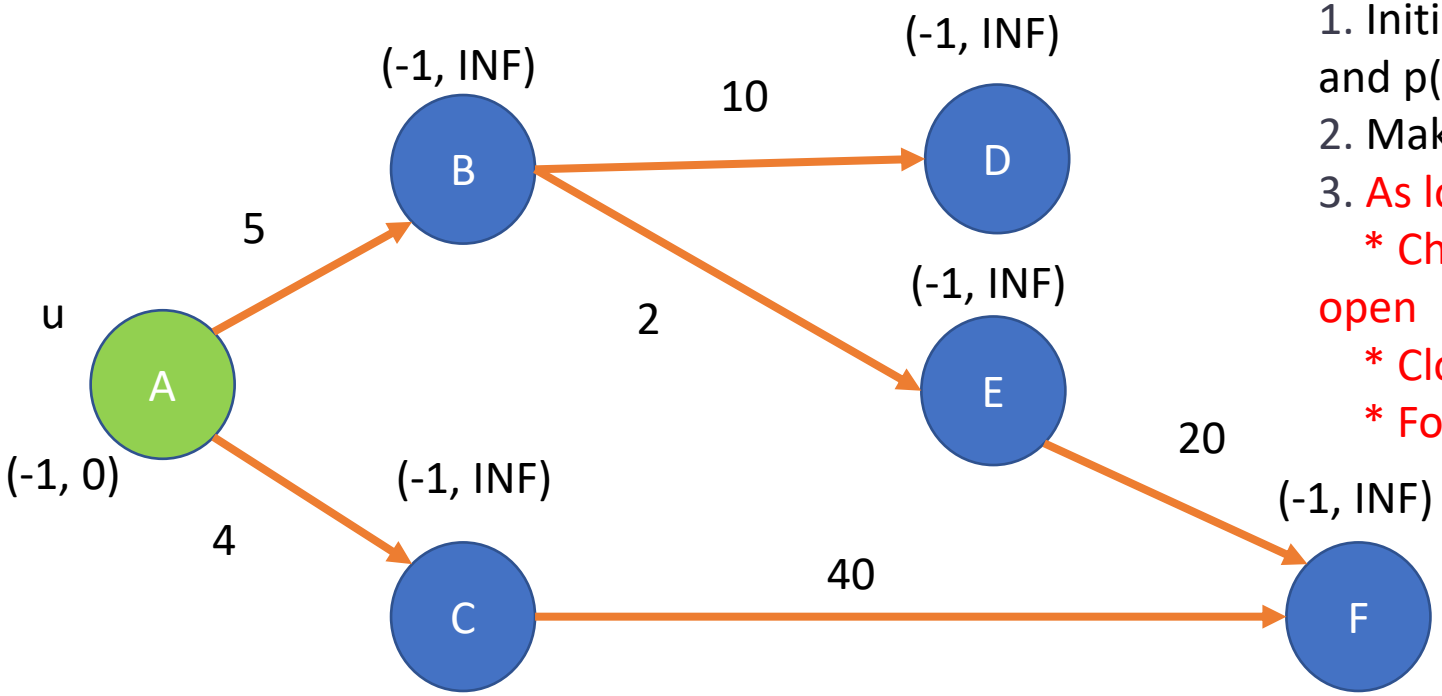
1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

3. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido.



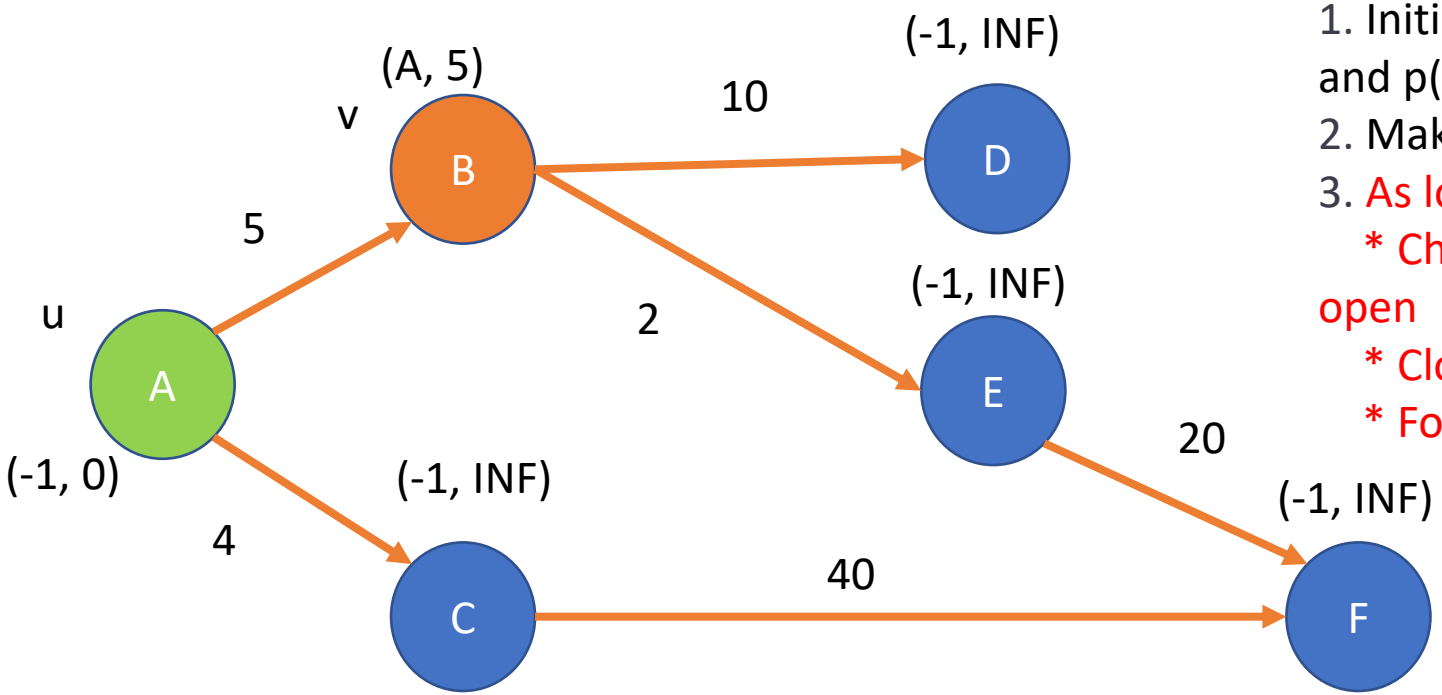
1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

3. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido.



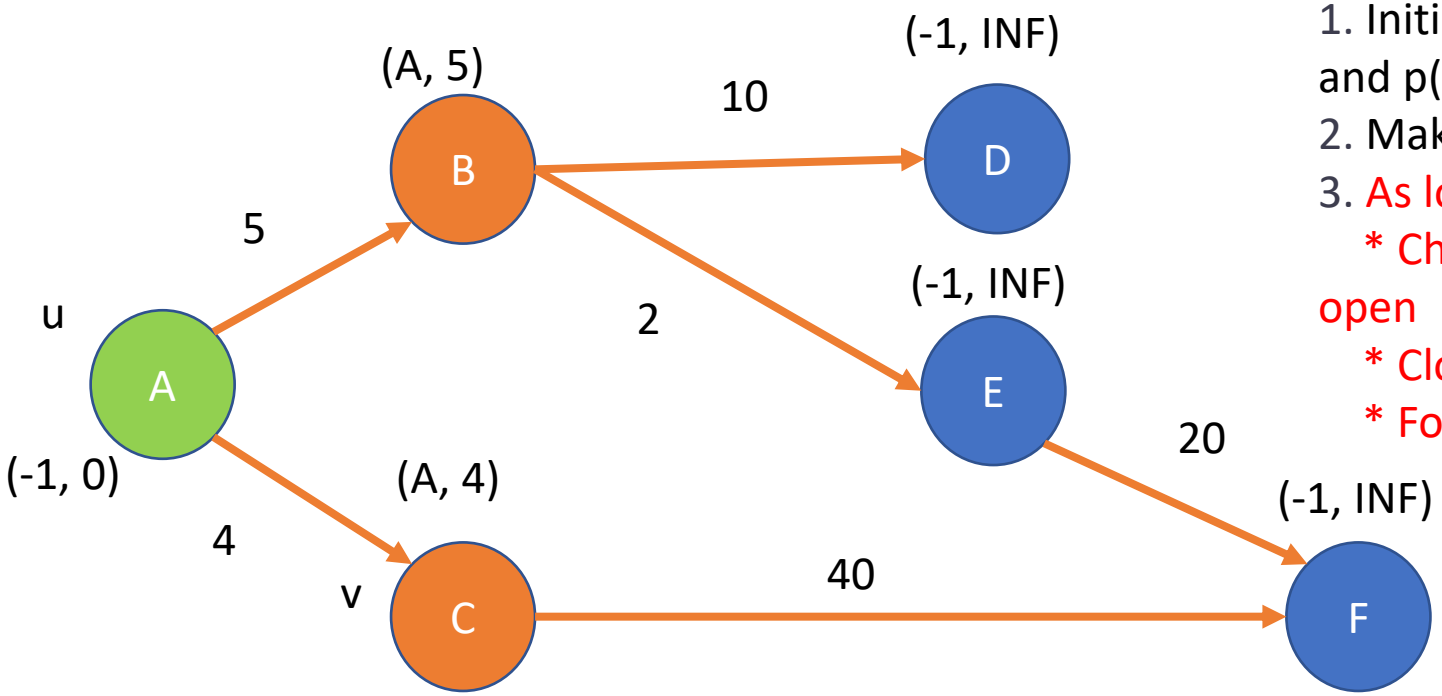
1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

3. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido.



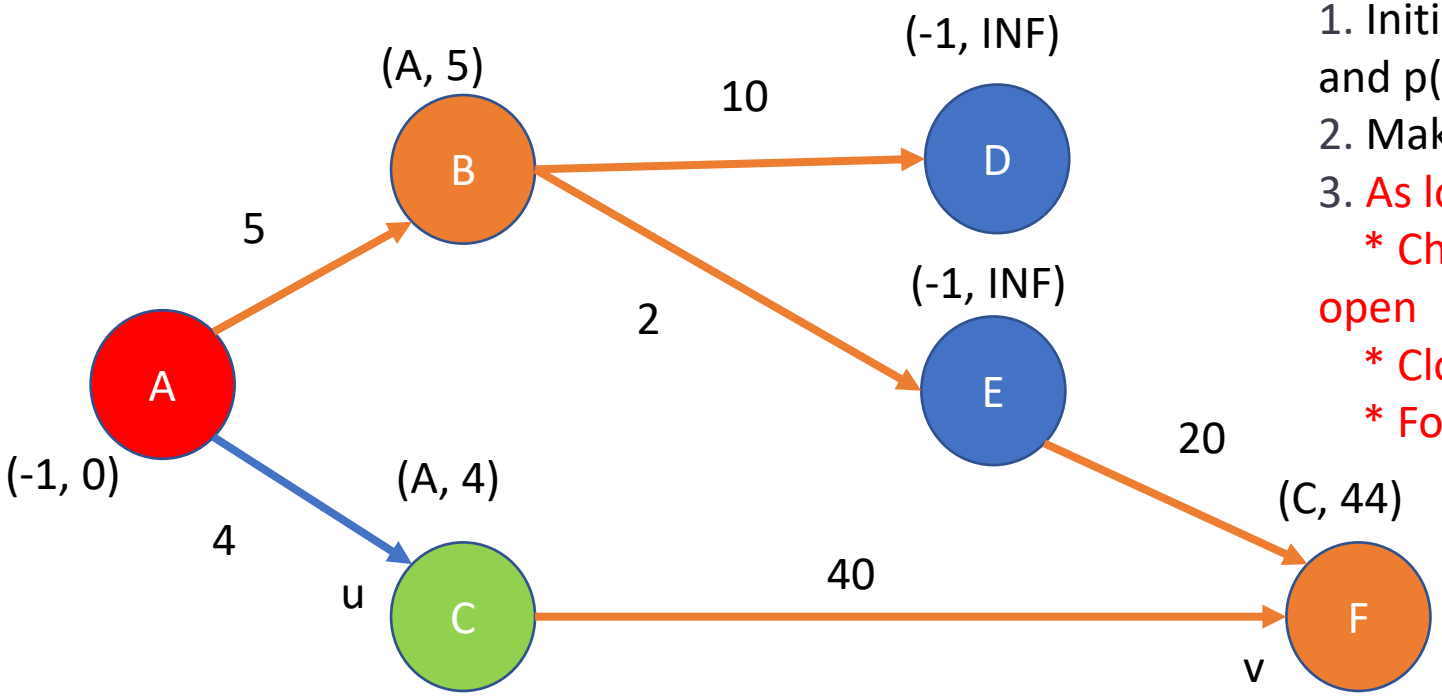
1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

3. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido.



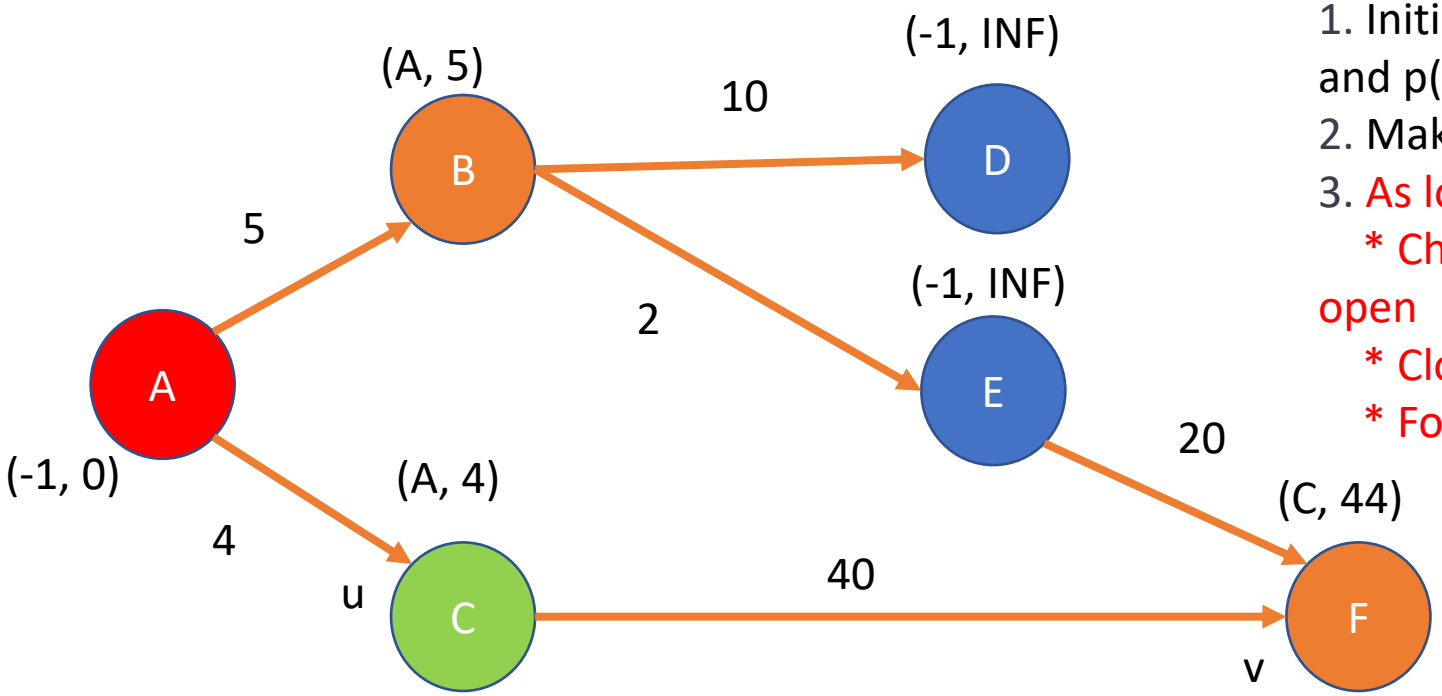
1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

3. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido.



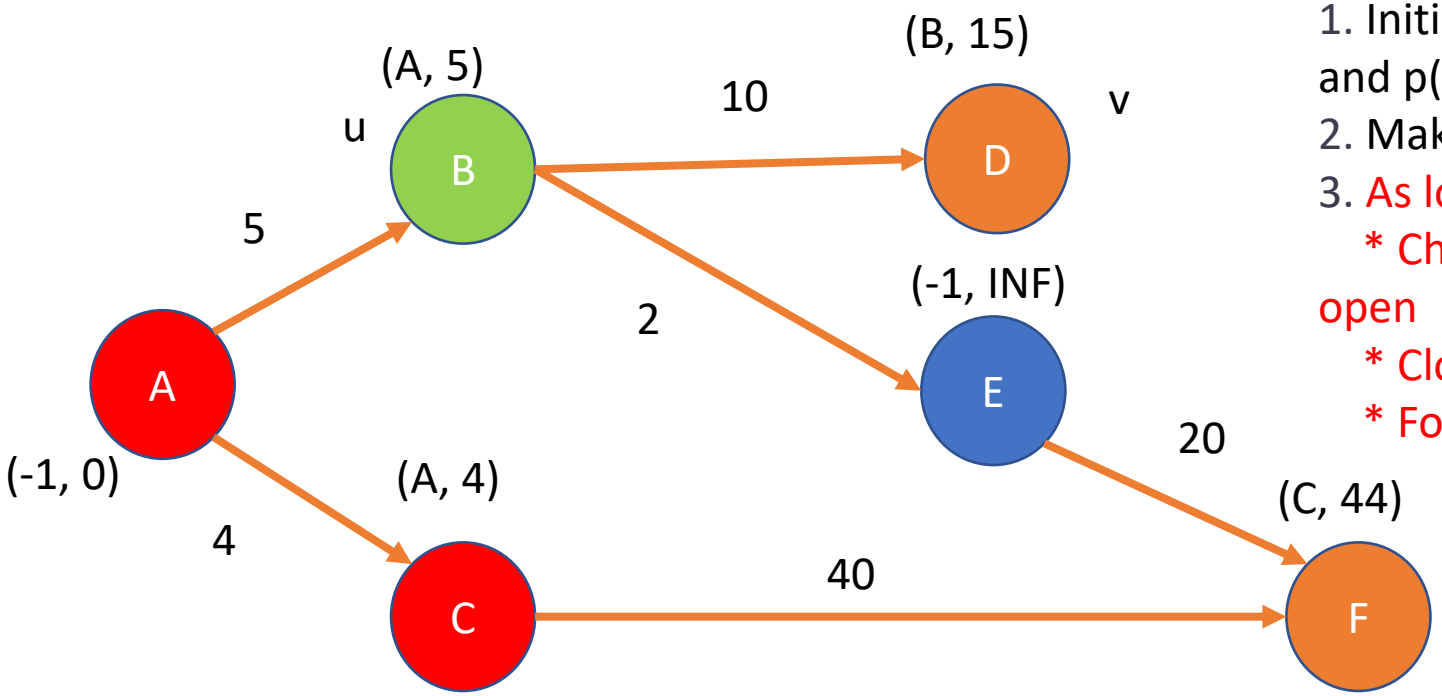
1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

3. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido.



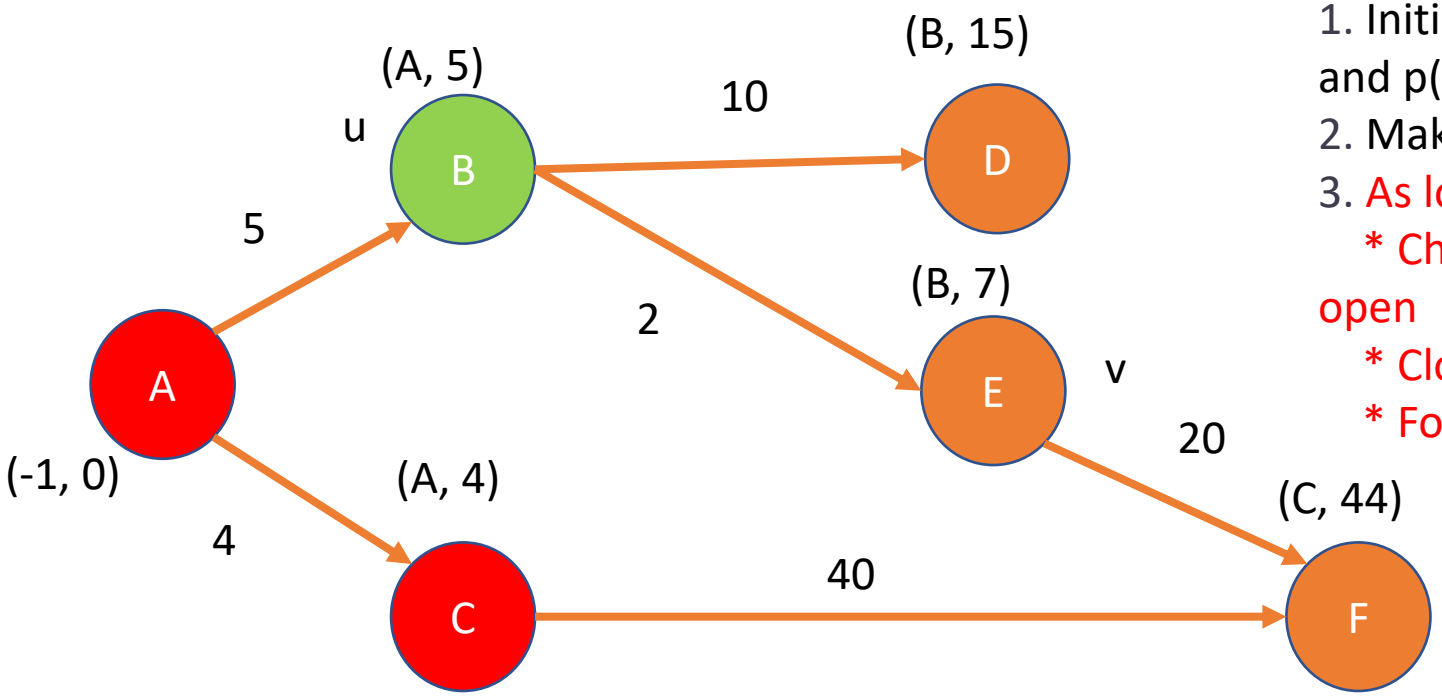
1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

3. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido.



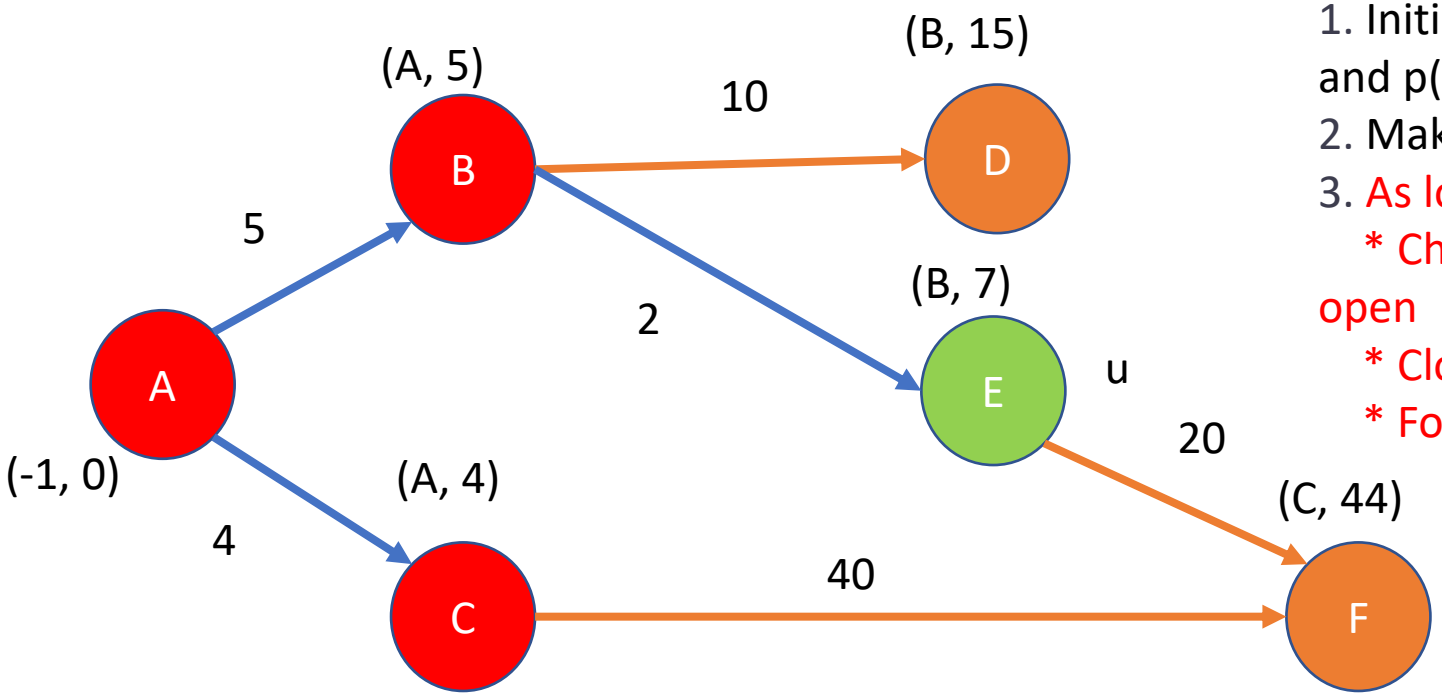
1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

3. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido.



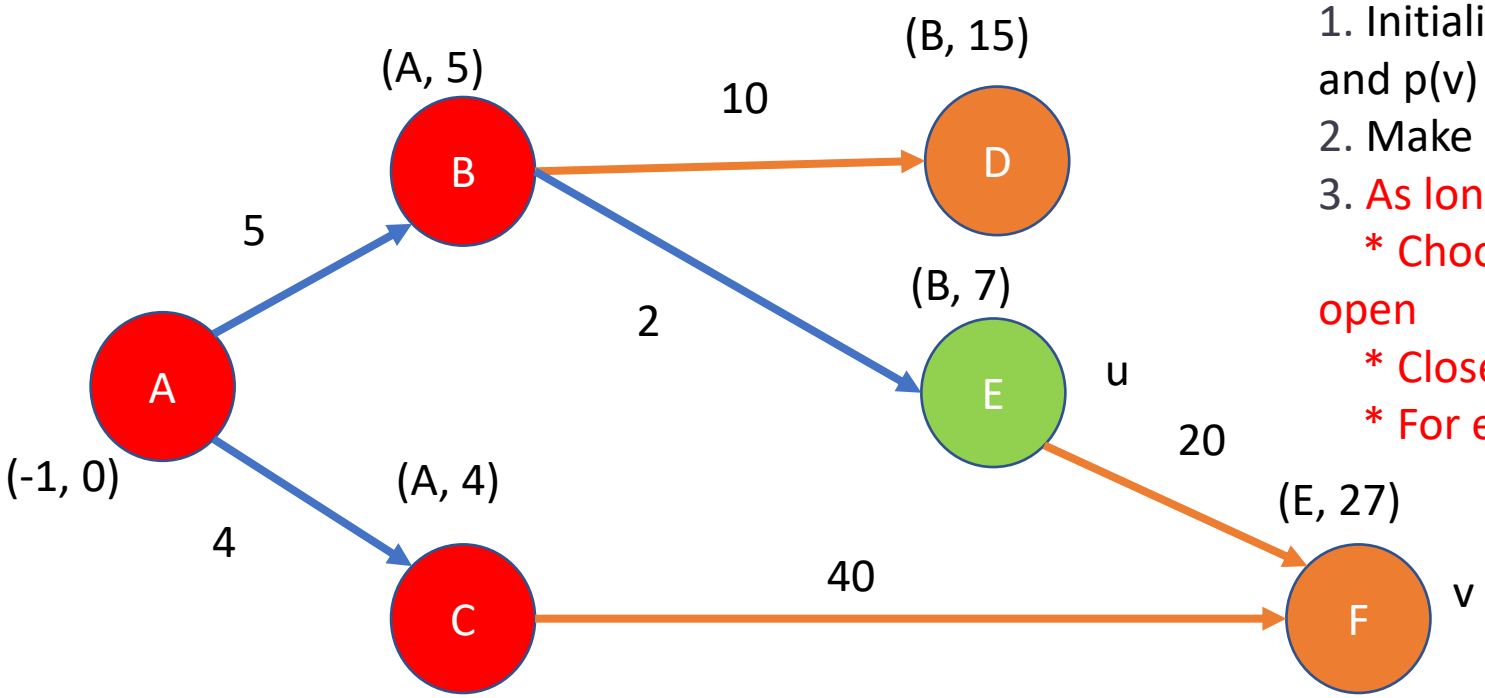
1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

3. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido.



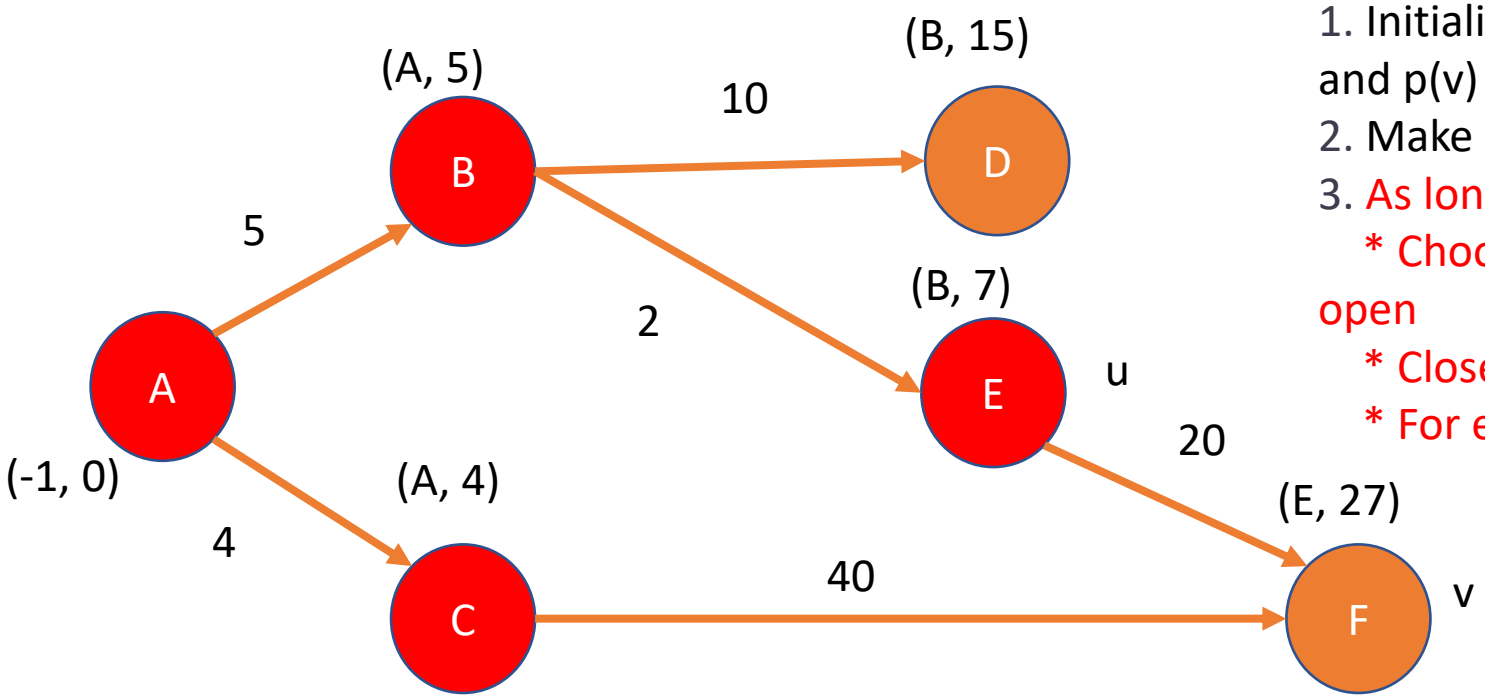
1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

3. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido.



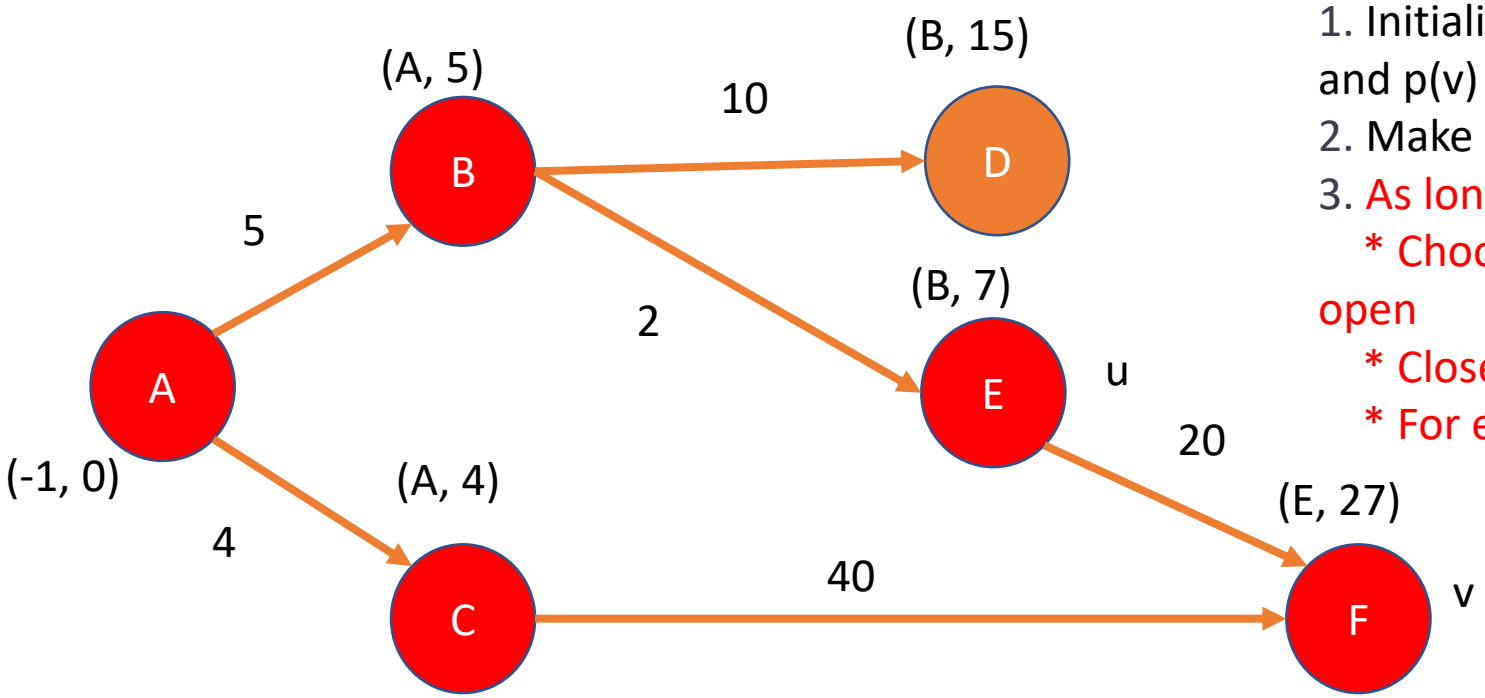
1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

3. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido.



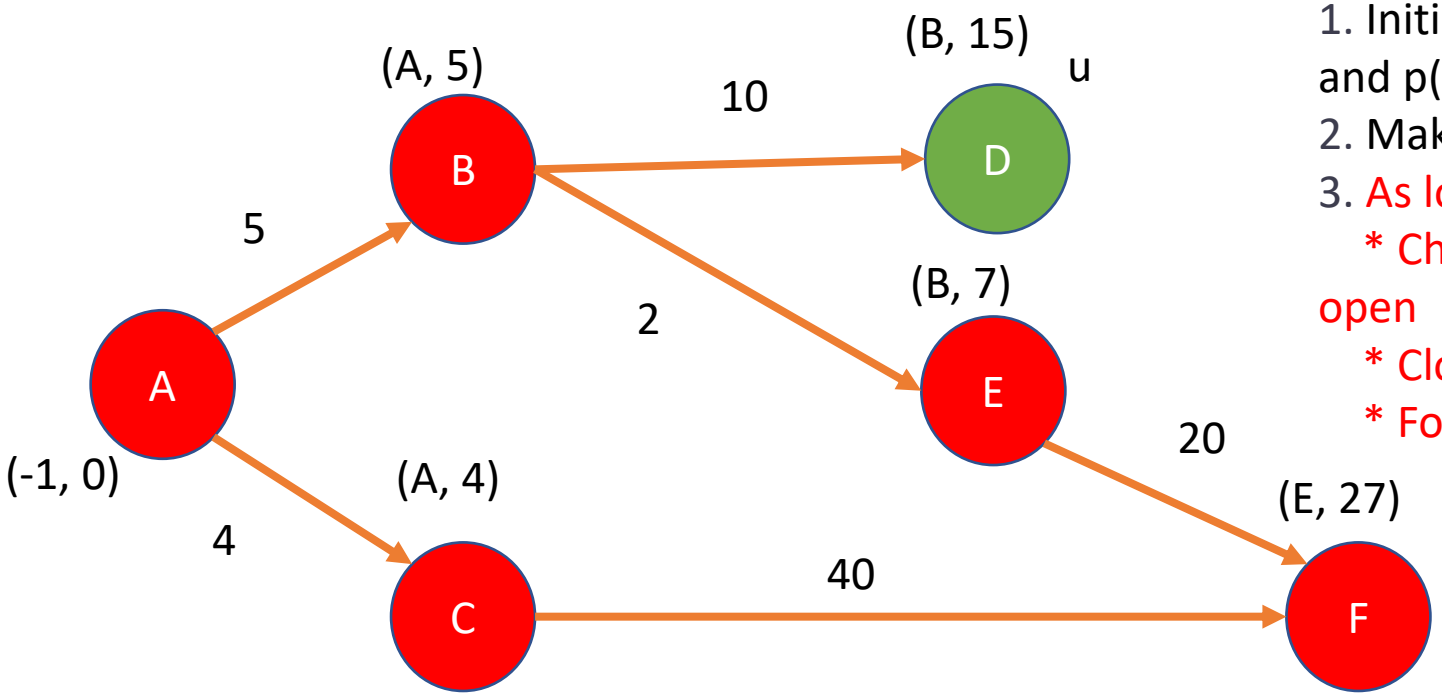
1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

3. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido.



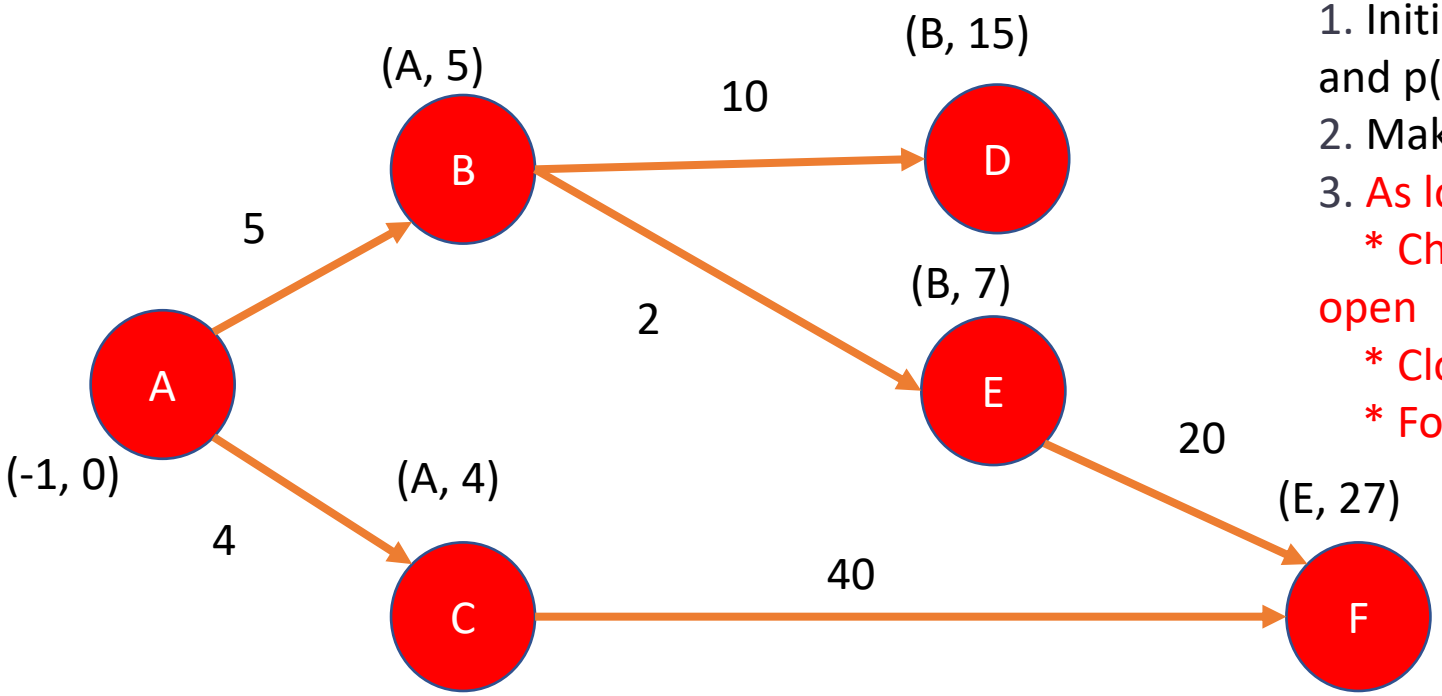
1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

3. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido.



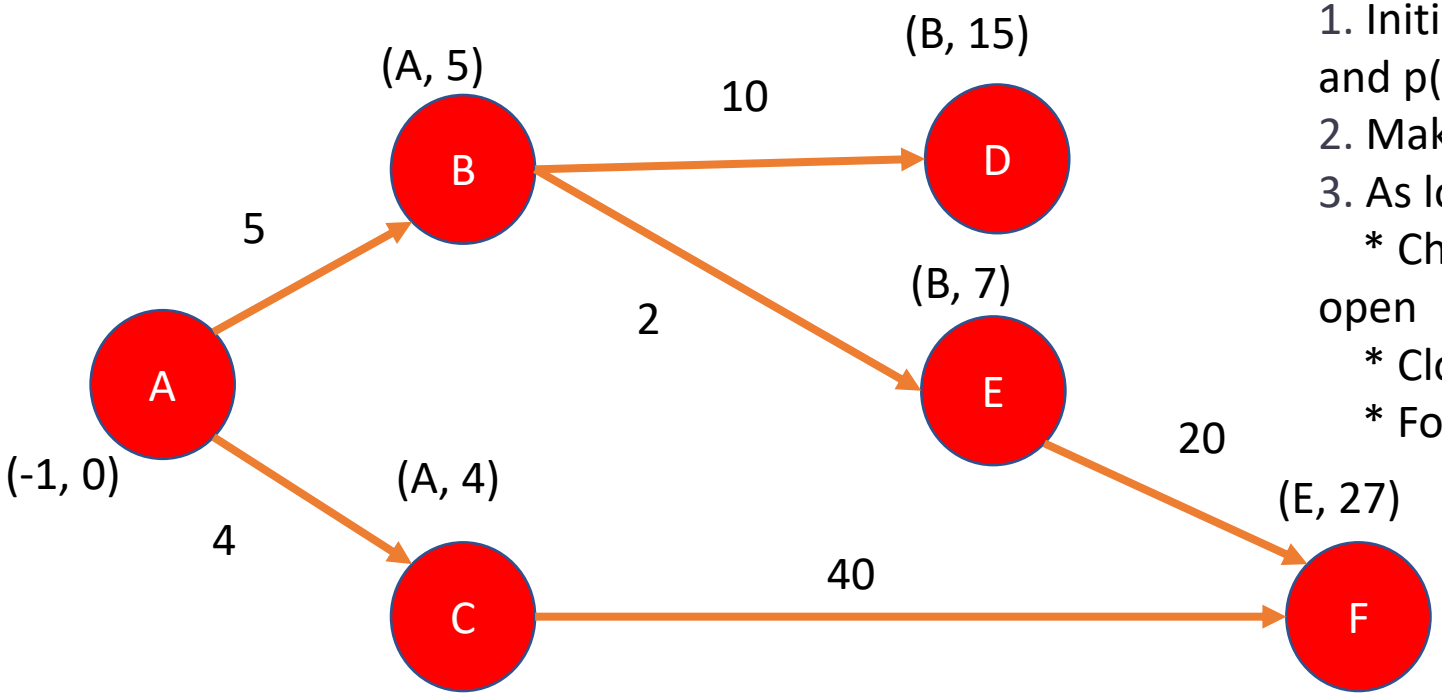
1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

3. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido.



1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

3. Execute o Algoritmo Dijkstra no grafo a seguir. Apresente todos os passos da execução do algoritmo e o caminho final escolhido.

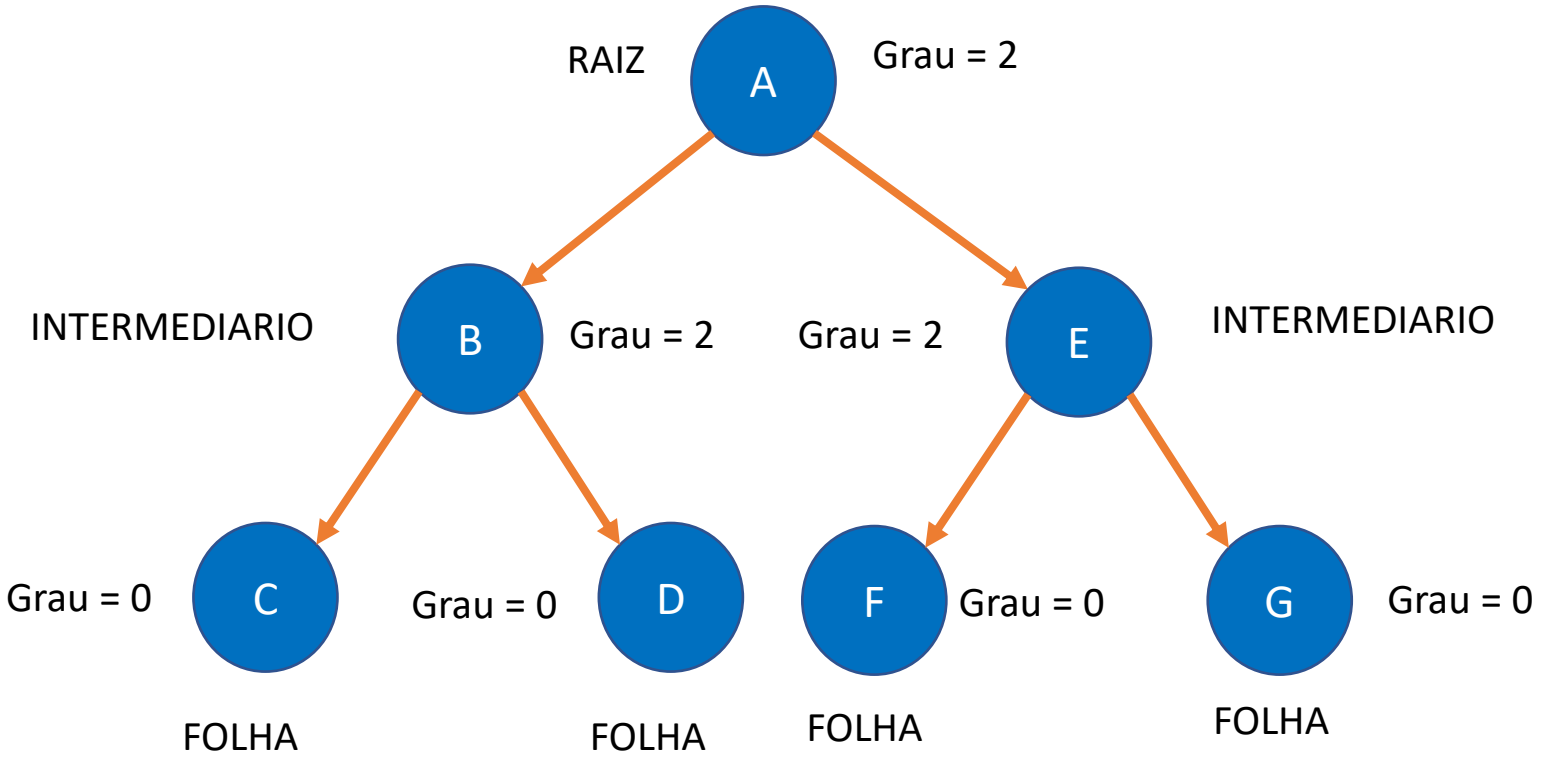


1. Initialize the graph with d(s) = 0, d(v) = INF, for all v ≠ s, and p(v) = -1 for all *v*
2. Make open(v) = True for every v in the graph
3. As long as there is an open vertex:
   * Choose *u* whose estimate is the smallest among the open
   * Close *u*
   * For every open node *v* adjacent to *u*: relax edge (*u*,*v*)

A => F F – E – B - A Custo: 27

3. Com relação a Árvore de Decisão a seguir, responda as seguintes questões

a) Identifique os nós em Raiz, Folha e Intermediários
b) Apresente o valor do Grau para cada nó
c) Defina *Entropia* e *Ganho de Informação* em algoritmos de Árvore de Decisão



ENTROPIA:

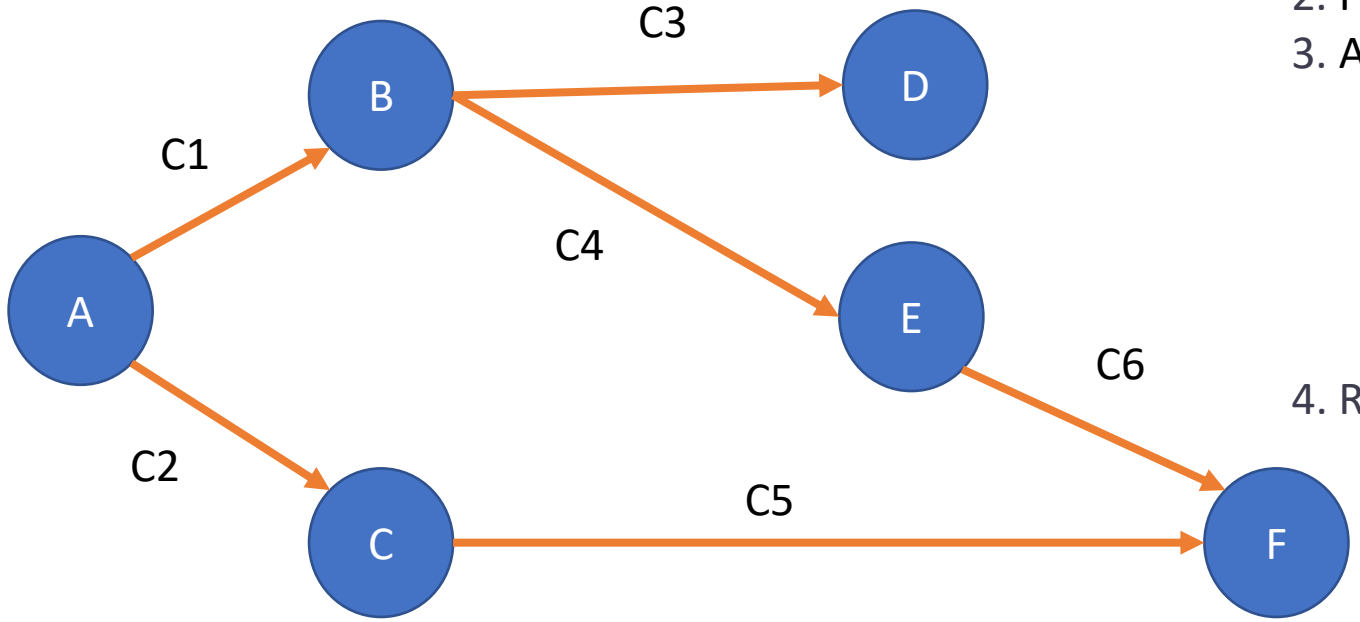
- É uma métrica da teoria da informação que mede a impureza ou incerteza em um grupo de observações.
- Ajuda a decidir o melhor atributo para a divisão dos nós em uma árvore de decisões.
- Ajuda a identificar o atributo com maior ganho de informação.
- É a presença de impureza (grau de aleatoriedade).

GANHO DE INFORMAÇÂO:

- Mede quanta informação um recurso fornece sobre uma classe.
- É a redução da entropia depois que um conjunto de dados é dividido com base em um atributo, de modo que ajuda a decidir qual atributo deve ser selecionado como nó de decisão.
- Ajuda a determinar a ordem dos atributos nos nós de uma árvore de decisão.
- Construir uma árvore de decisão envolve encontrar o atributo que retorna o maior ganho de informação.

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

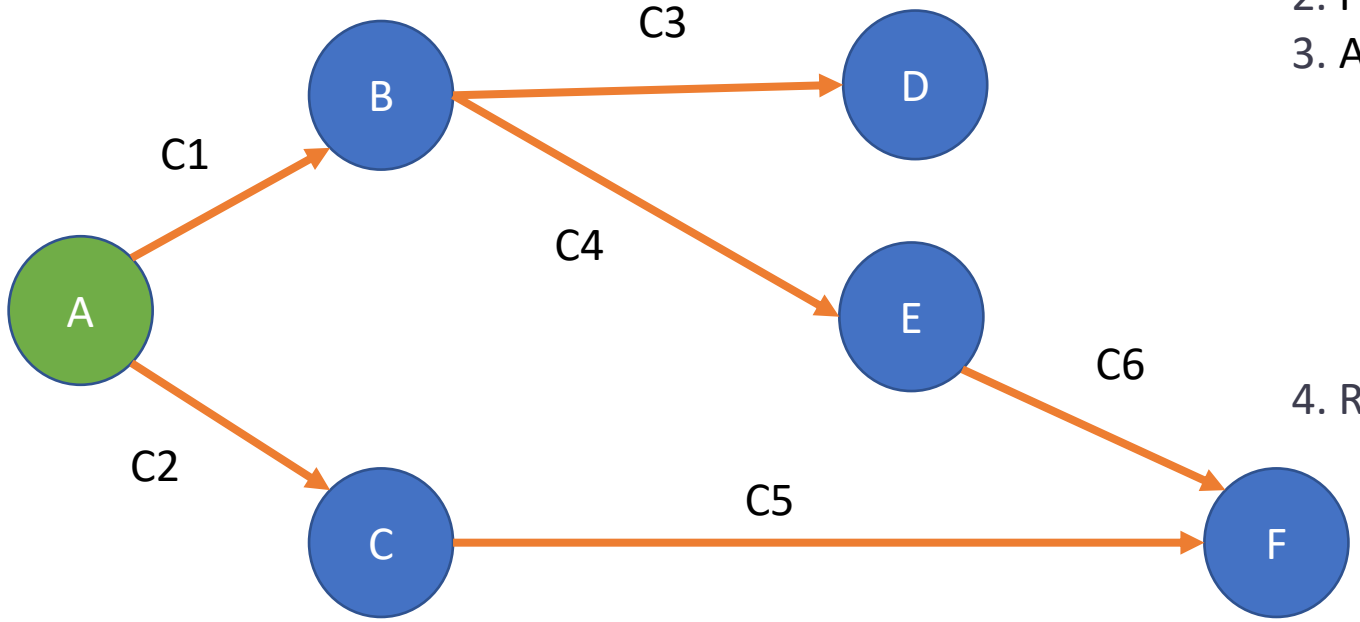
Breadth-First Search (BFS)



1. Define an initial node, marking it as explored
2. Put it on the list
3. As long as the queue is not empty:
   - Remove the 1st node from the list, u
   - For each neighbour v of u:
     * If v is not explored:
       ** Mark v as explored
       ** Put v at the end of the list
4. Repeat from another starting node, if there is one

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

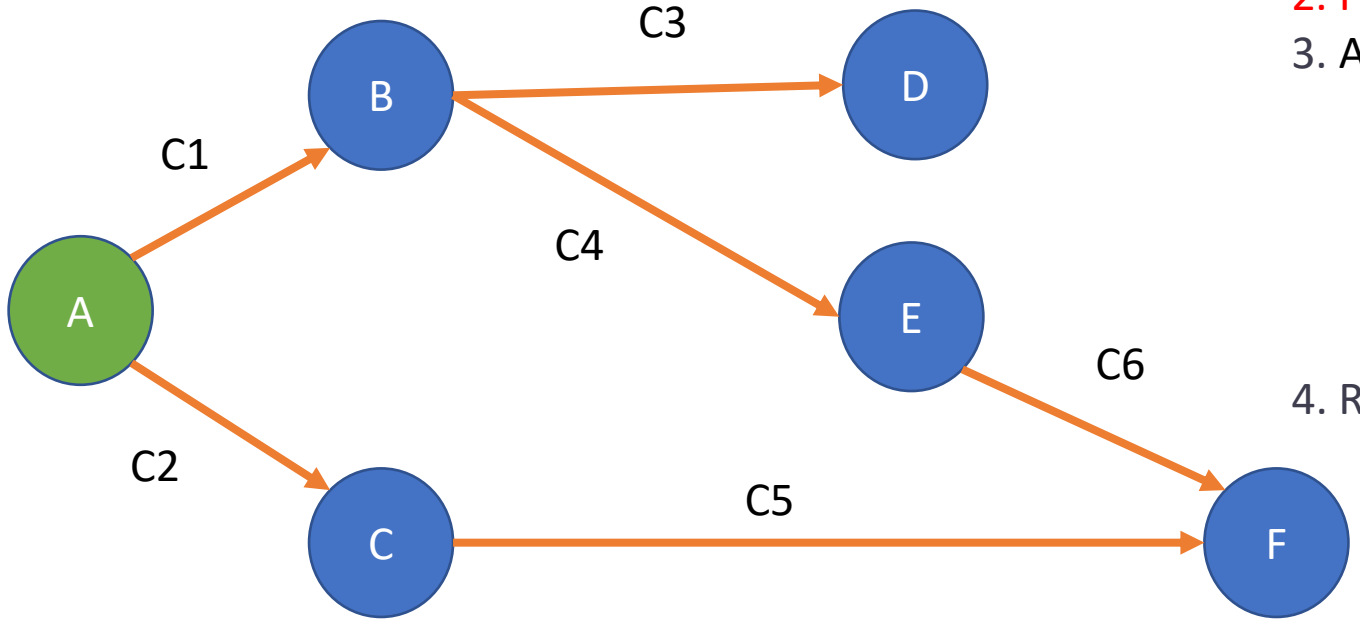
Breadth-First Search (BFS)



1. Define an initial node, marking it as explored
2. Put it on the list
3. As long as the queue is not empty:
   - Remove the 1st node from the list, u
   - For each neighbour v of u:
     * If v is not explored:
       ** Mark v as explored
       ** Put v at the end of the list
4. Repeat from another starting node, if there is one

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

Breadth-First Search (BFS)

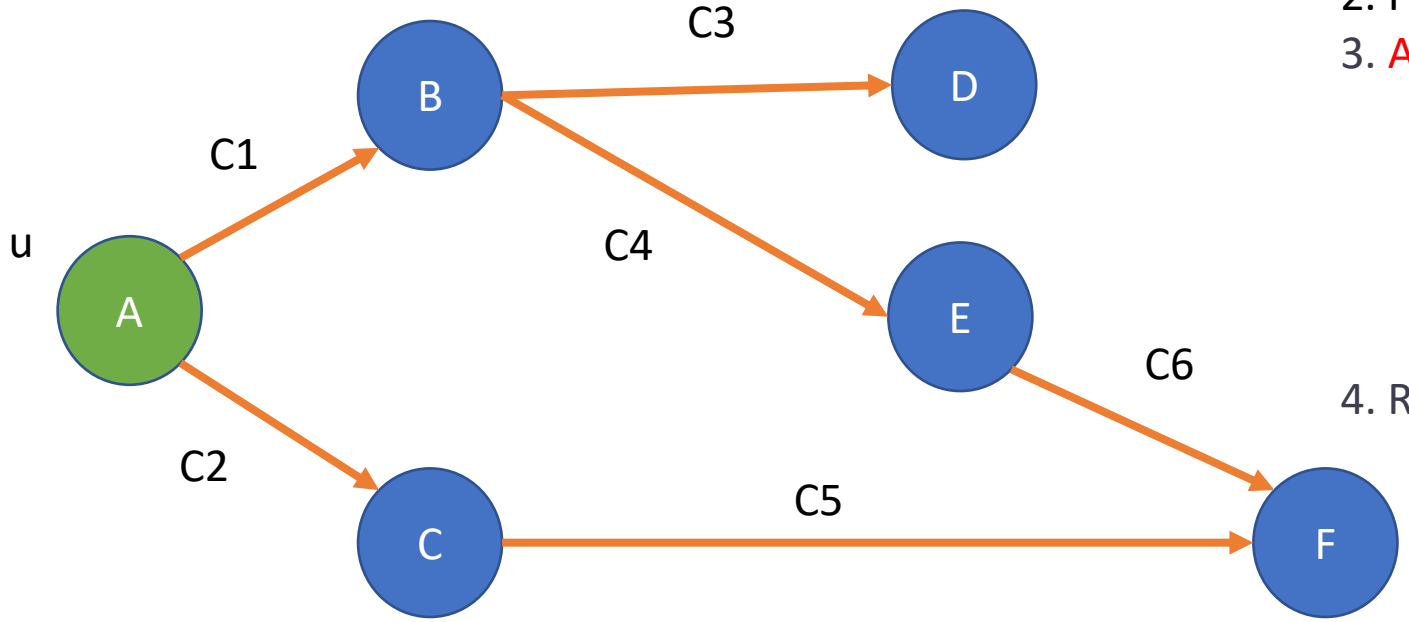


1. Define an initial node, marking it as explored
2. Put it on the list
3. As long as the queue is not empty:
   - Remove the 1st node from the list, u
   - For each neighbour v of u:
     * If v is not explored:
       ** Mark v as explored
       ** Put v at the end of the list
4. Repeat from another starting node, if there is one

A

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

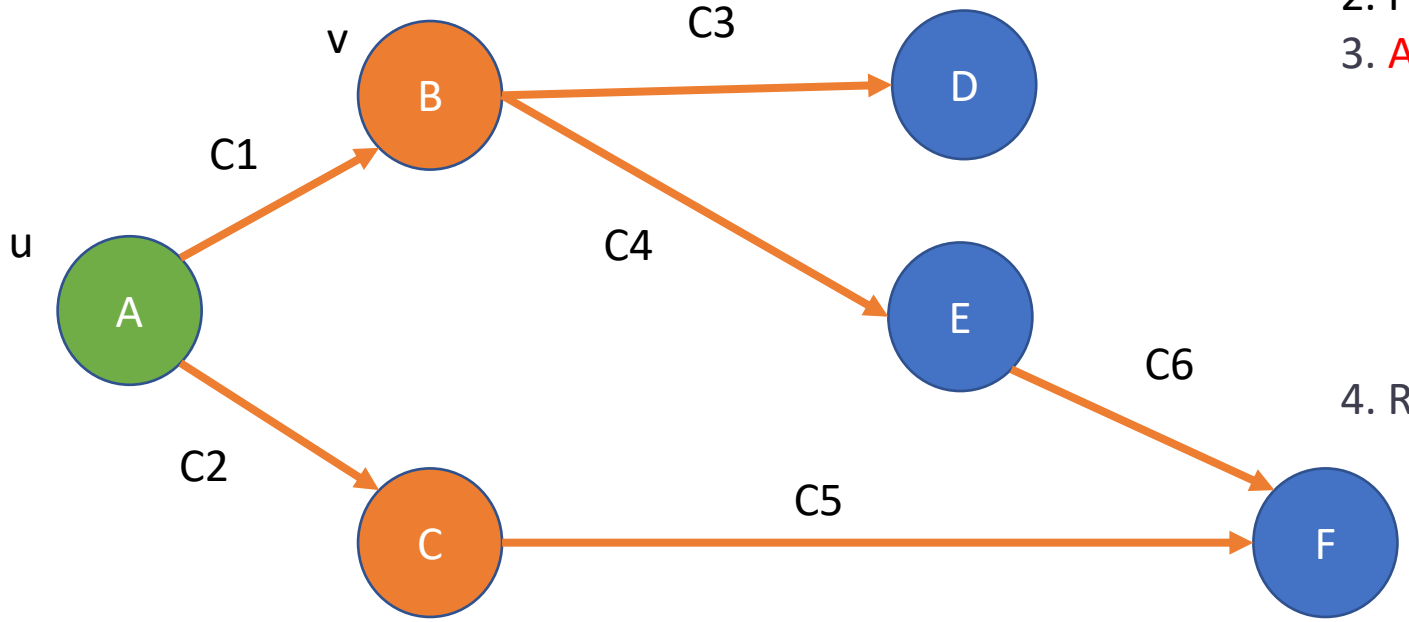
Breadth-First Search (BFS)



1. Define an initial node, marking it as explored
2. Put it on the list
3. As long as the queue is not empty:
   - Remove the 1st node from the list, u
   - For each neighbour v of u:
     * If v is not explored:
       ** Mark v as explored
       ** Put v at the end of the list
4. Repeat from another starting node, if there is one

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

Breadth-First Search (BFS)

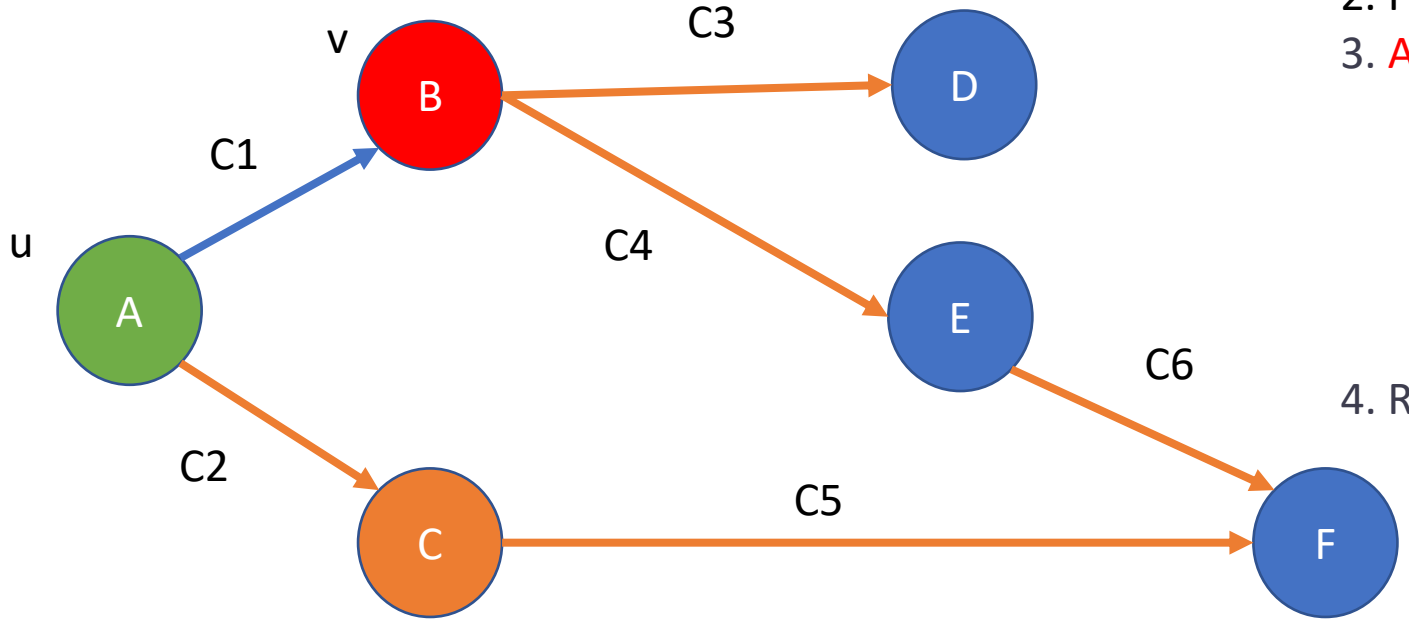


1. Define an initial node, marking it as explored
2. Put it on the list
3. As long as the queue is not empty:
   - Remove the 1st node from the list, u
   - For each neighbour v of u:
     * If v is not explored:
       ** Mark v as explored
       ** Put v at the end of the list
4. Repeat from another starting node, if there is one

B

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

Breadth-First Search (BFS)

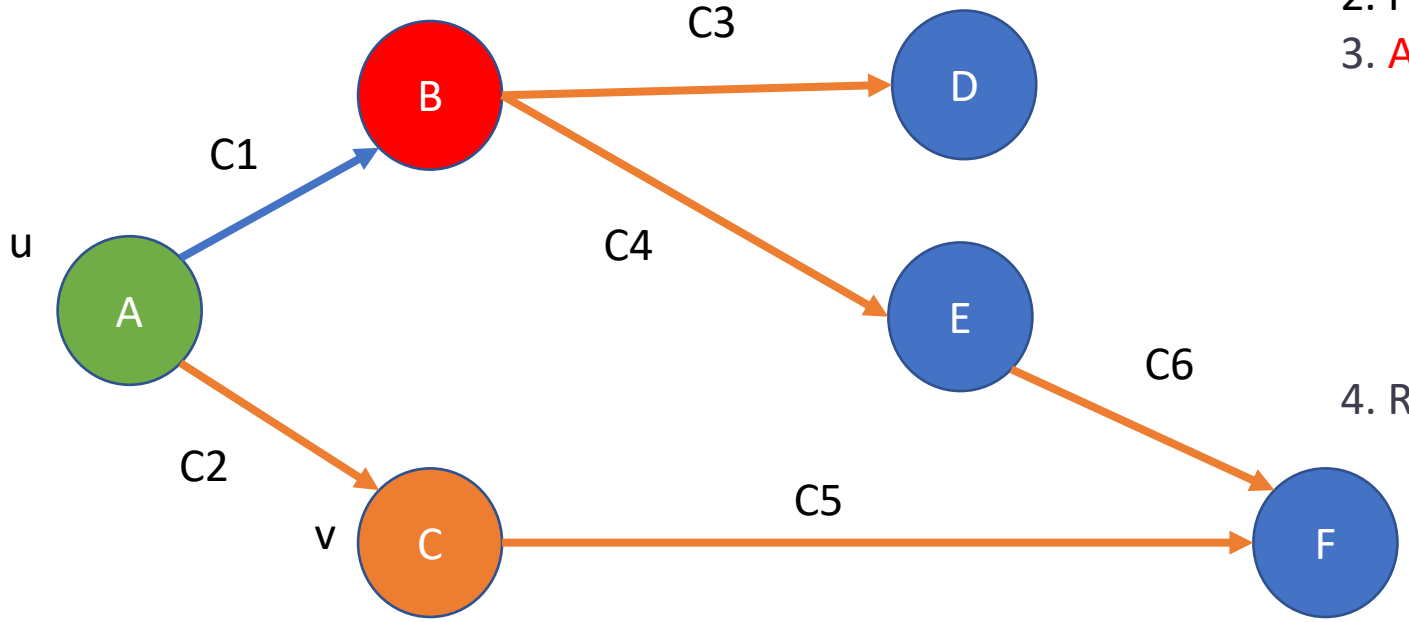


1. Define an initial node, marking it as explored
2. Put it on the list
3. As long as the queue is not empty:
   - Remove the 1st node from the list, u
   - For each neighbour v of u:
     * If v is not explored:
       ** Mark v as explored
       ** Put v at the end of the list
4. Repeat from another starting node, if there is one

	C

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

Breadth-First Search (BFS)

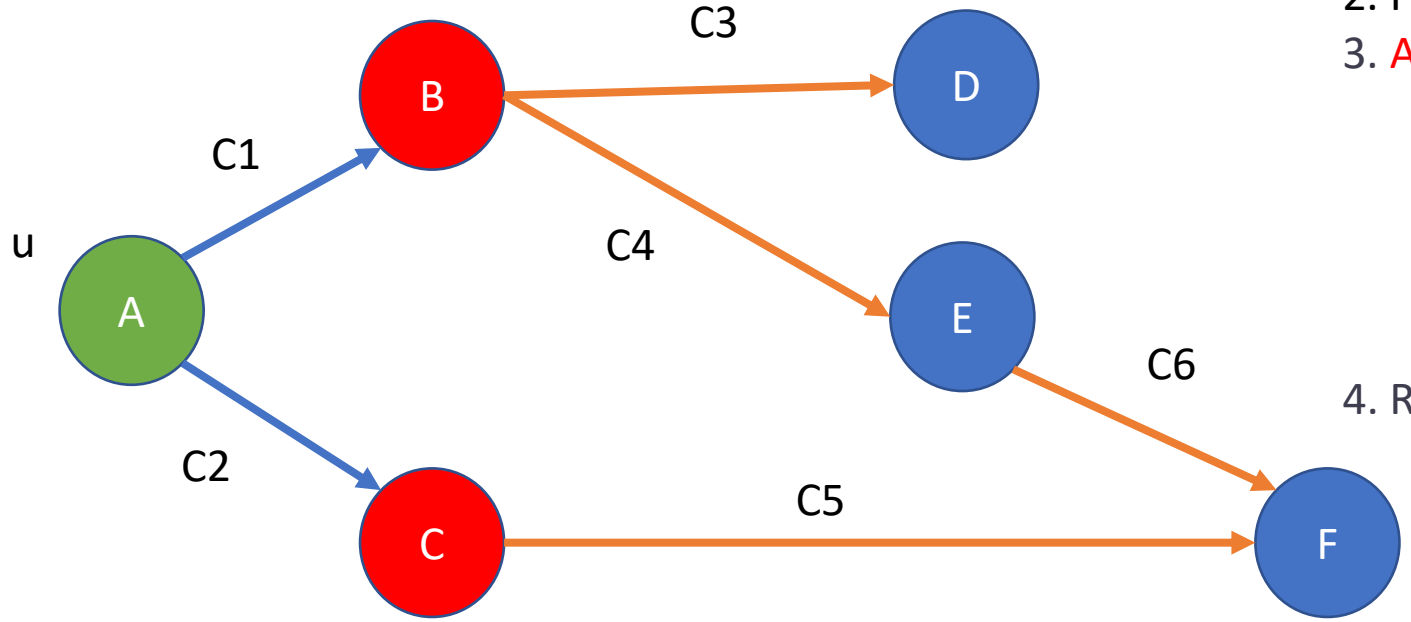


1. Define an initial node, marking it as explored
2. Put it on the list
3. As long as the queue is not empty:
   - Remove the 1st node from the list, u
   - For each neighbour v of u:
     * If v is not explored:
       ** Mark v as explored
       ** Put v at the end of the list
4. Repeat from another starting node, if there is one

	C

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

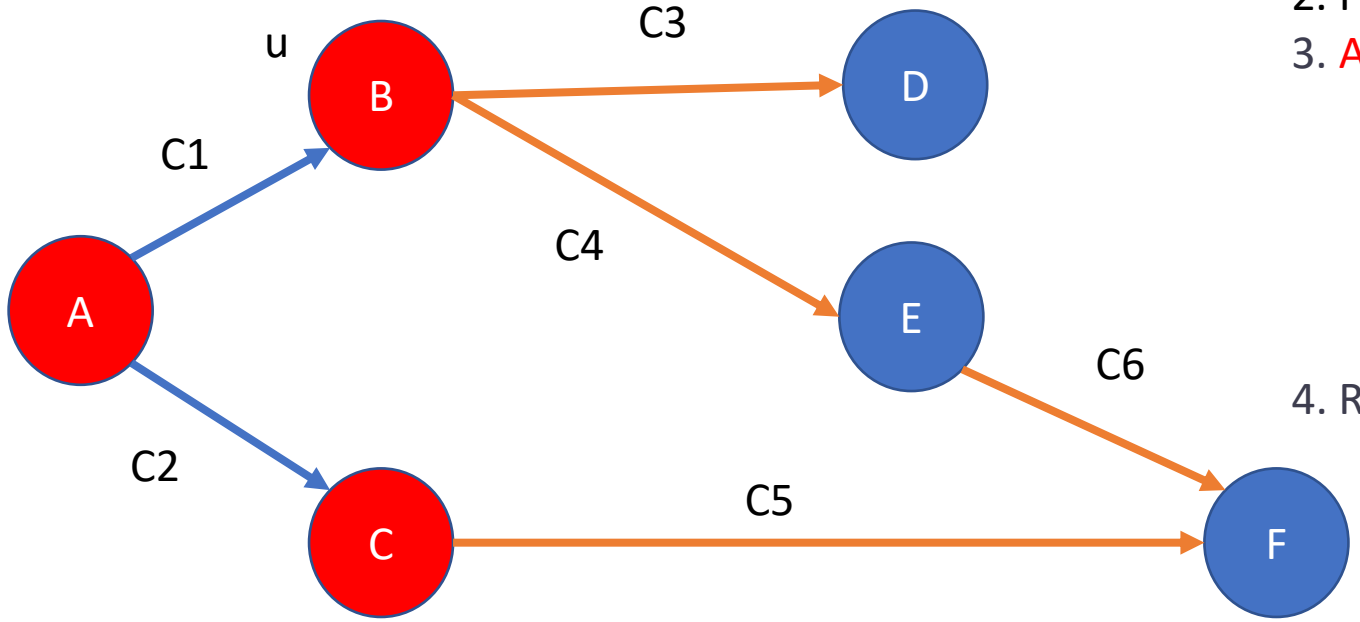
Breadth-First Search (BFS)



1. Define an initial node, marking it as explored
2. Put it on the list
3. As long as the queue is not empty:
   - Remove the 1st node from the list, u
   - For each neighbour v of u:
     * If v is not explored:
       ** Mark v as explored
       ** Put v at the end of the list
4. Repeat from another starting node, if there is one

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

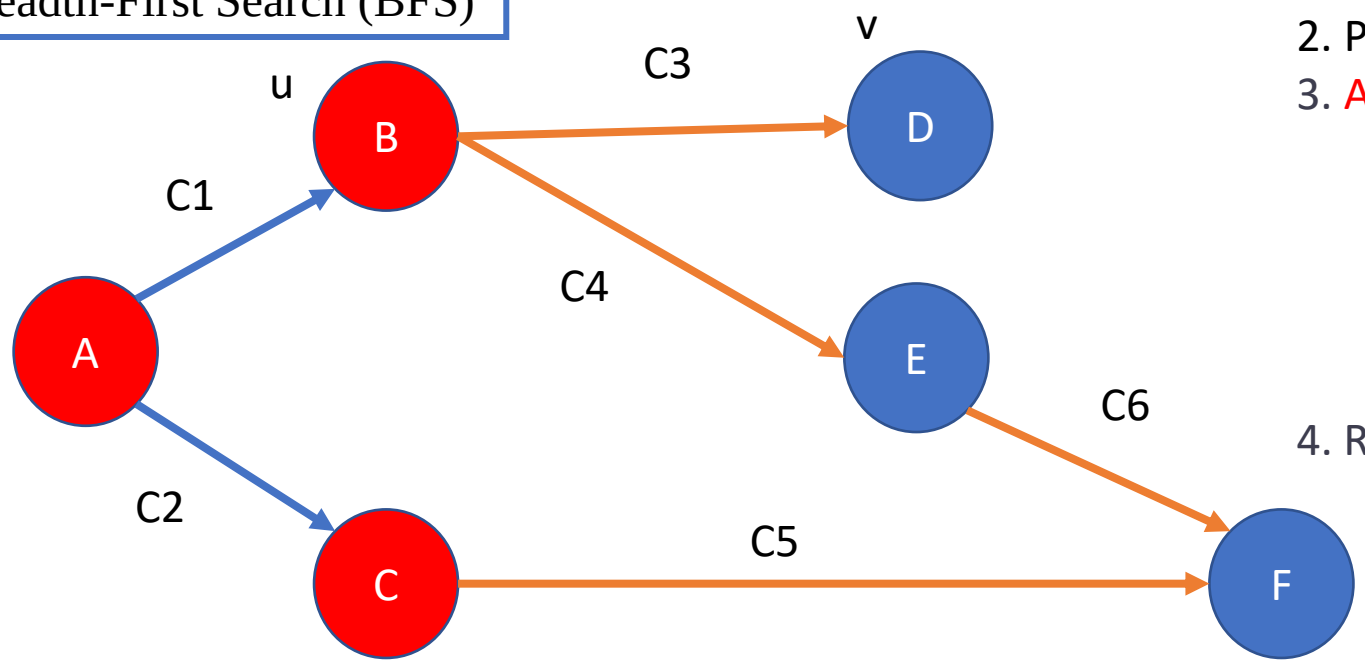
Breadth-First Search (BFS)



1. Define an initial node, marking it as explored
2. Put it on the list
3. As long as the queue is not empty:
   - Remove the 1st node from the list, u
   - For each neighbour v of u:
     * If v is not explored:
       ** Mark v as explored
       ** Put v at the end of the list
4. Repeat from another starting node, if there is one

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

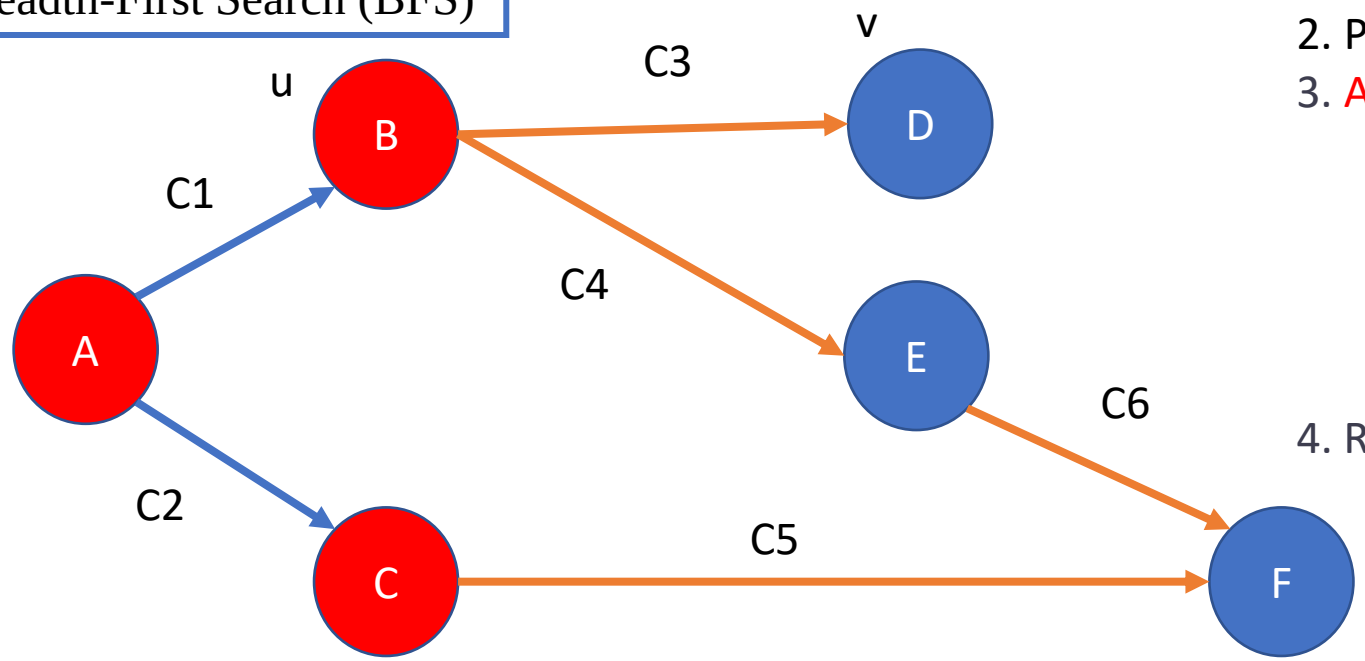
Breadth-First Search (BFS)



1. Define an initial node, marking it as explored
2. Put it on the list
3. As long as the queue is not empty:
   - Remove the 1st node from the list, u
   - For each neighbour v of u:
     * If v is not explored:
       ** Mark v as explored
       ** Put v at the end of the list
4. Repeat from another starting node, if there is one

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

Breadth-First Search (BFS)

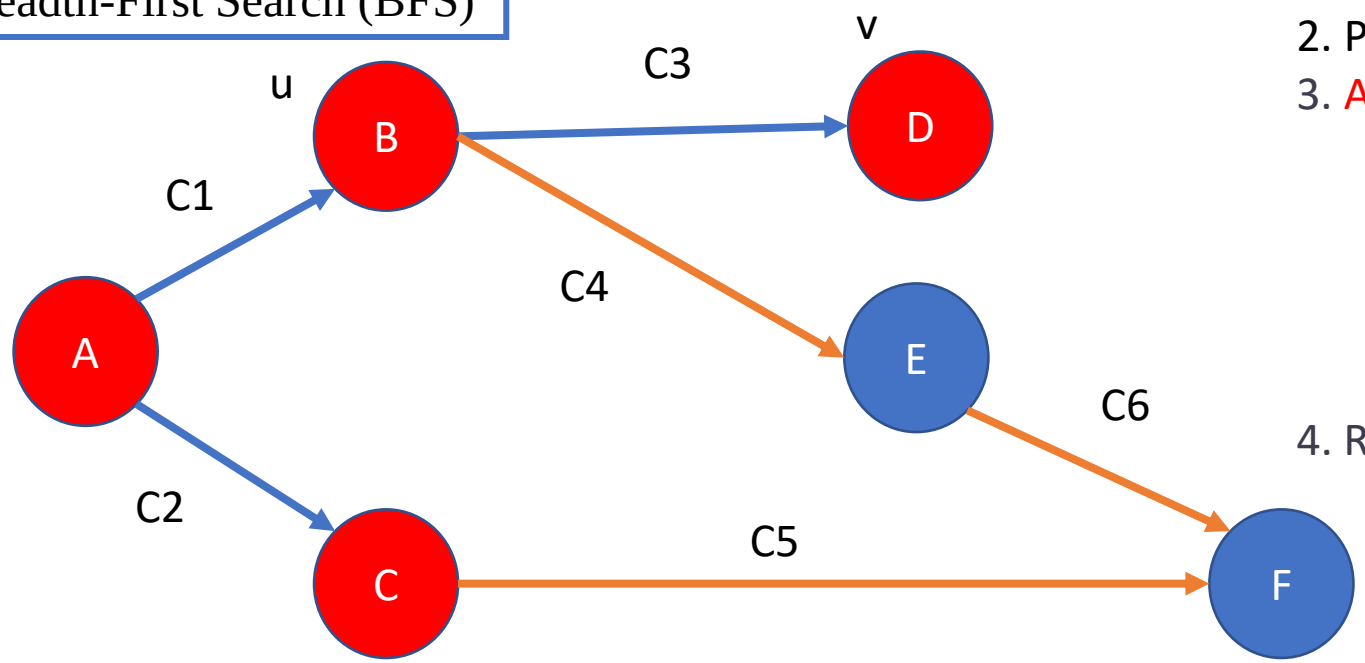


1. Define an initial node, marking it as explored
2. Put it on the list
3. As long as the queue is not empty:
   - Remove the 1st node from the list, u
   - For each neighbour v of u:
     * If v is not explored:
       ** Mark v as explored
       ** Put v at the end of the list
4. Repeat from another starting node, if there is one

D	E

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

Breadth-First Search (BFS)

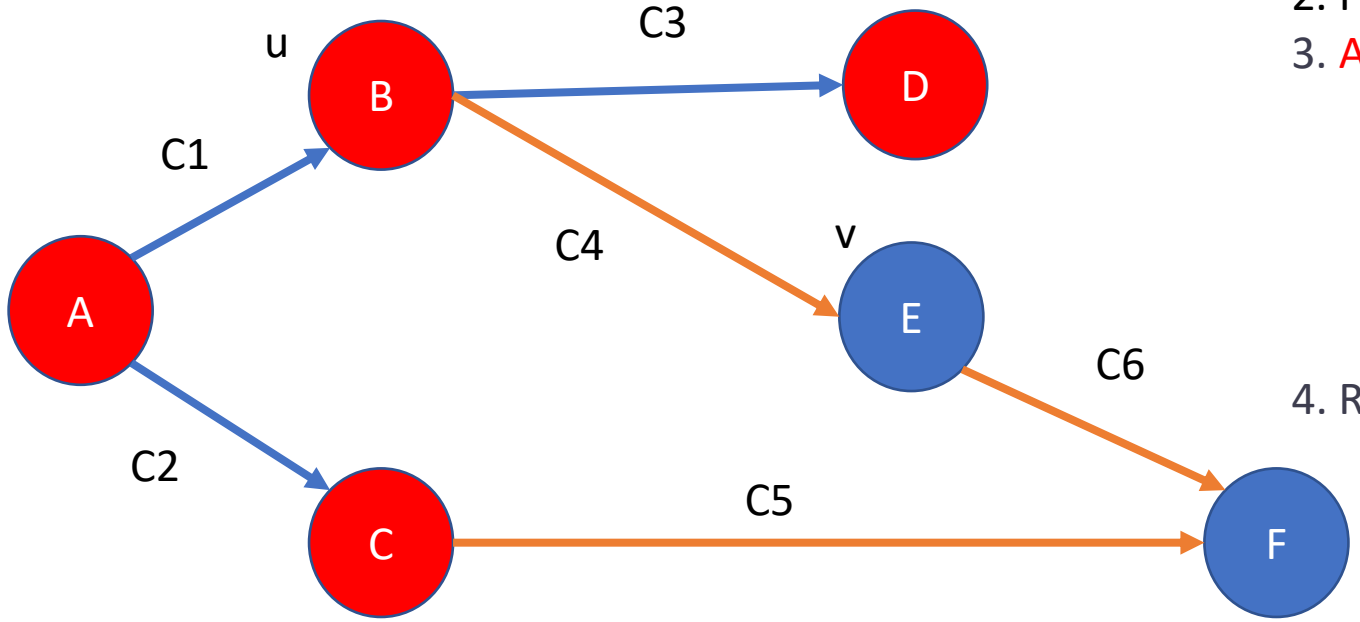


1. Define an initial node, marking it as explored
2. Put it on the list
3. As long as the queue is not empty:
   - Remove the 1st node from the list, u
   - For each neighbour v of u:
     * If v is not explored:
       ** Mark v as explored
       ** Put v at the end of the list
4. Repeat from another starting node, if there is one

	E

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

Breadth-First Search (BFS)

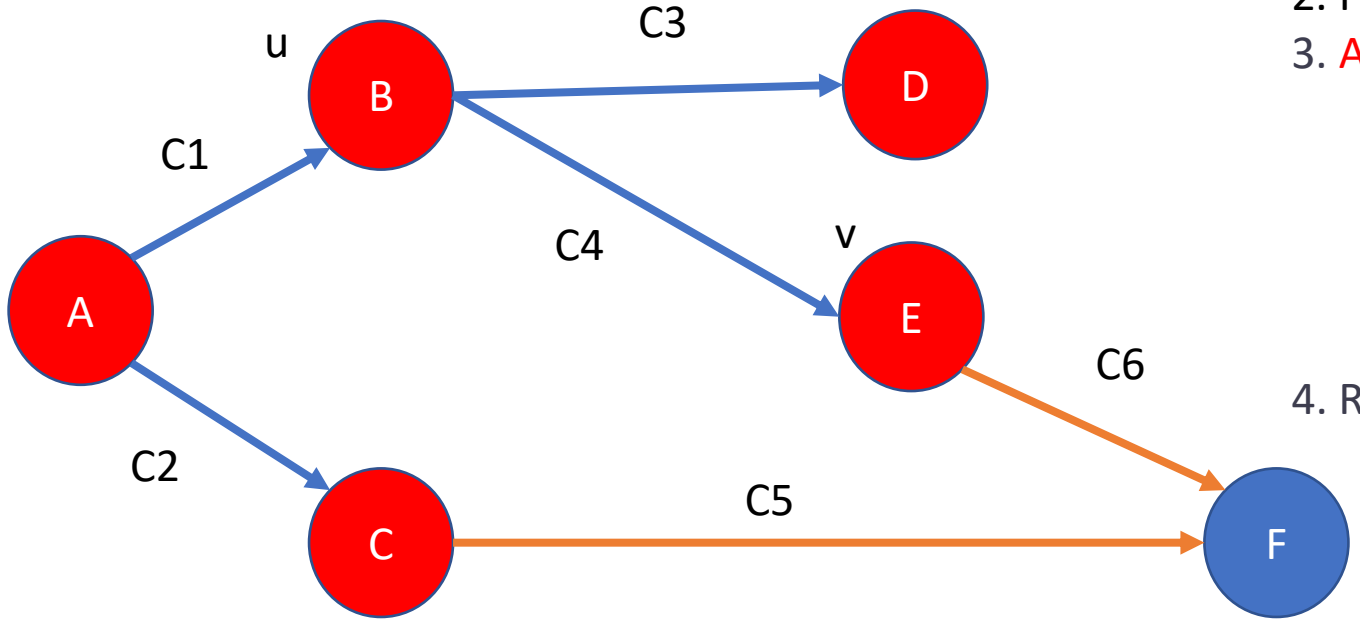


1. Define an initial node, marking it as explored
2. Put it on the list
3. As long as the queue is not empty:
   - Remove the 1st node from the list, u
   - For each neighbour v of u:
     * If v is not explored:
       ** Mark v as explored
       ** Put v at the end of the list
4. Repeat from another starting node, if there is one

	E

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

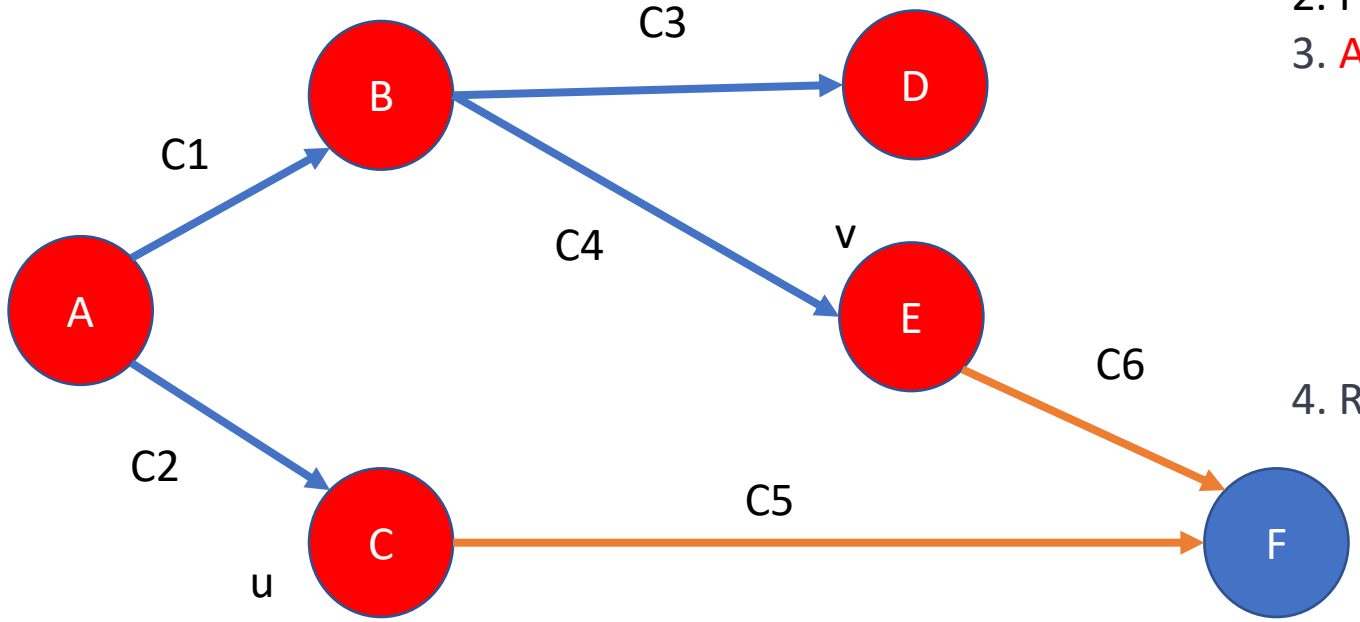
Breadth-First Search (BFS)



1. Define an initial node, marking it as explored
2. Put it on the list
3. As long as the queue is not empty:
   - Remove the 1st node from the list, u
   - For each neighbour v of u:
     * If v is not explored:
       ** Mark v as explored
       ** Put v at the end of the list
4. Repeat from another starting node, if there is one

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

Breadth-First Search (BFS)

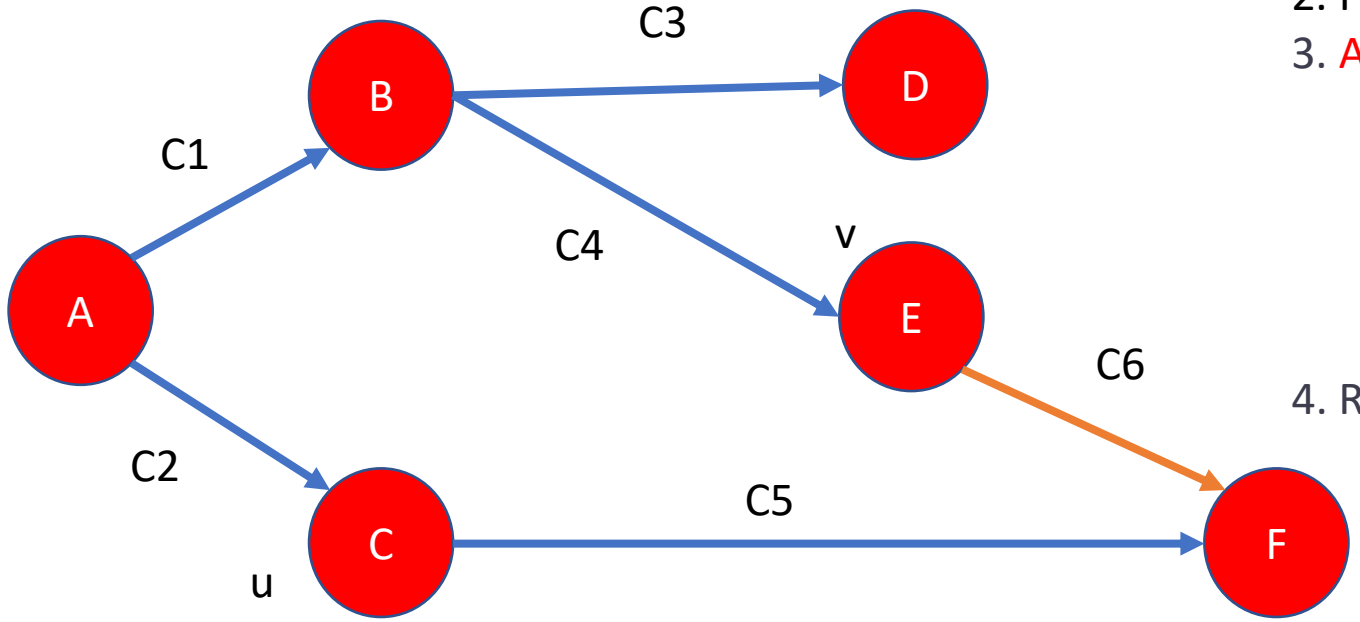


1. Define an initial node, marking it as explored
2. Put it on the list
3. As long as the queue is not empty:
   - Remove the 1st node from the list, u
   - For each neighbour v of u:
     * If v is not explored:
       ** Mark v as explored
       ** Put v at the end of the list
4. Repeat from another starting node, if there is one

F	

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

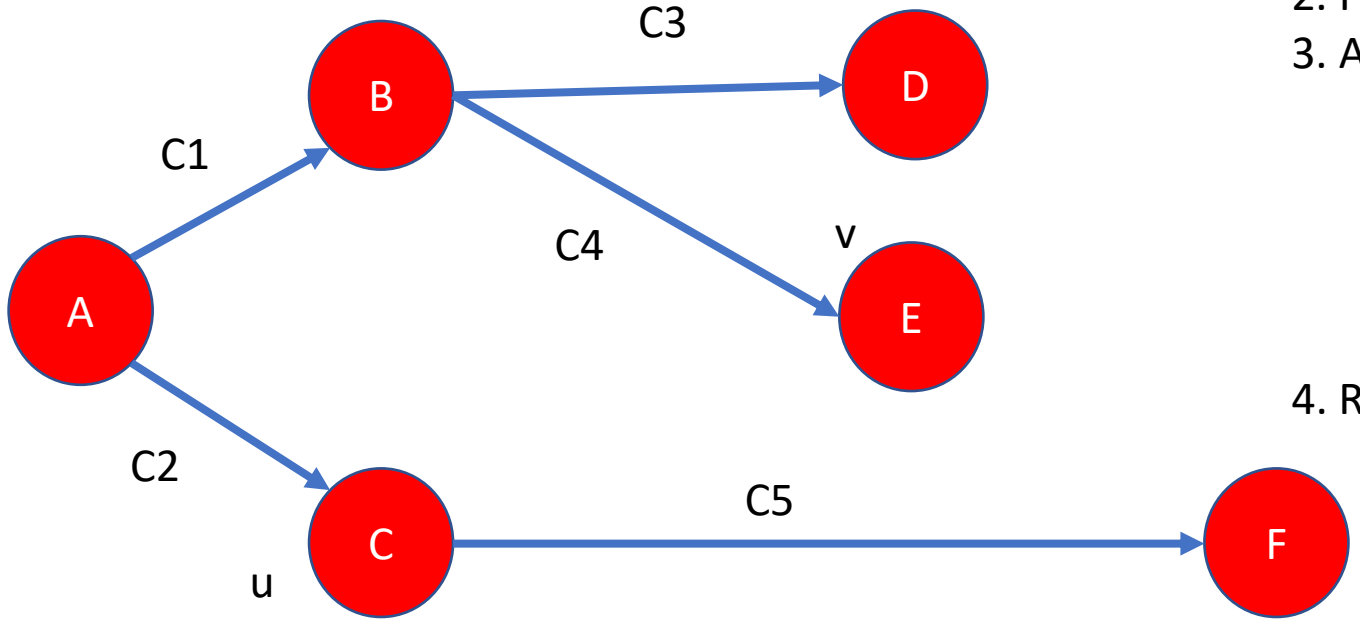
Breadth-First Search (BFS)



1. Define an initial node, marking it as explored
2. Put it on the list
3. As long as the queue is not empty:
   - Remove the 1st node from the list, u
   - For each neighbour v of u:
     * If v is not explored:
       ** Mark v as explored
       ** Put v at the end of the list
4. Repeat from another starting node, if there is one

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

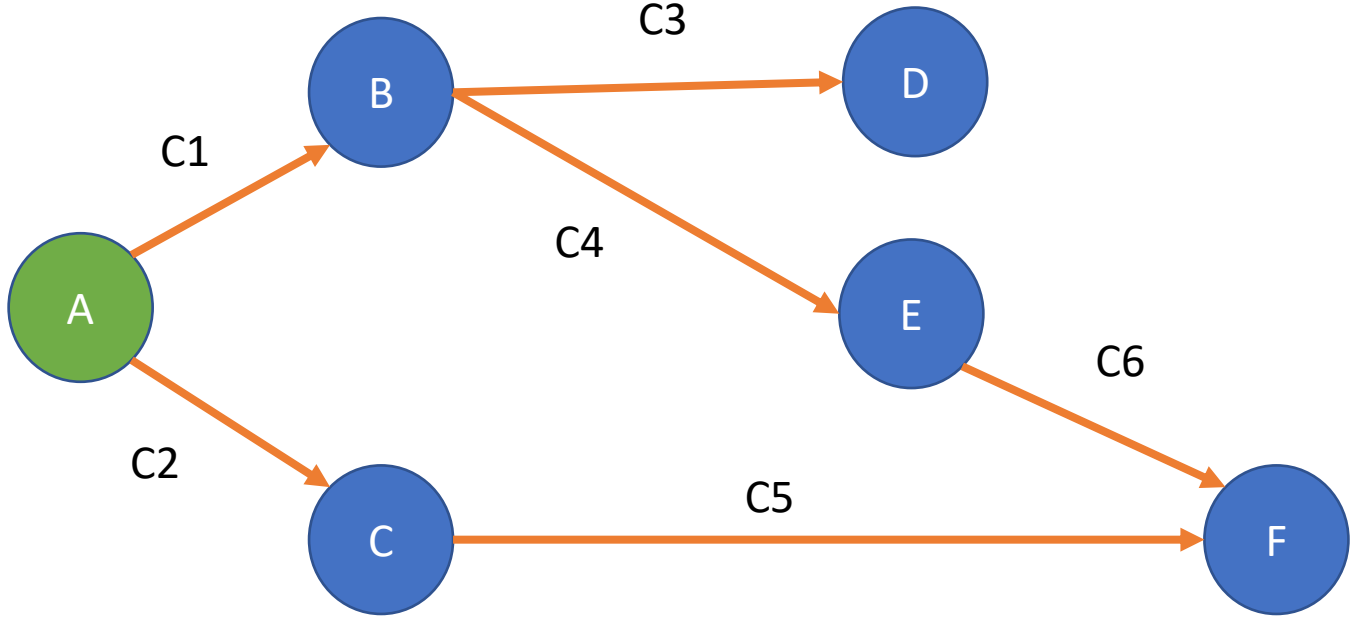
Breadth-First Search (BFS)



1. Define an initial node, marking it as explored
2. Put it on the list
3. As long as the queue is not empty:
   - Remove the 1st node from the list, u
   - For each neighbour v of u:
     * If v is not explored:
       ** Mark v as explored
       ** Put v at the end of the list
4. Repeat from another starting node, if there is one

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

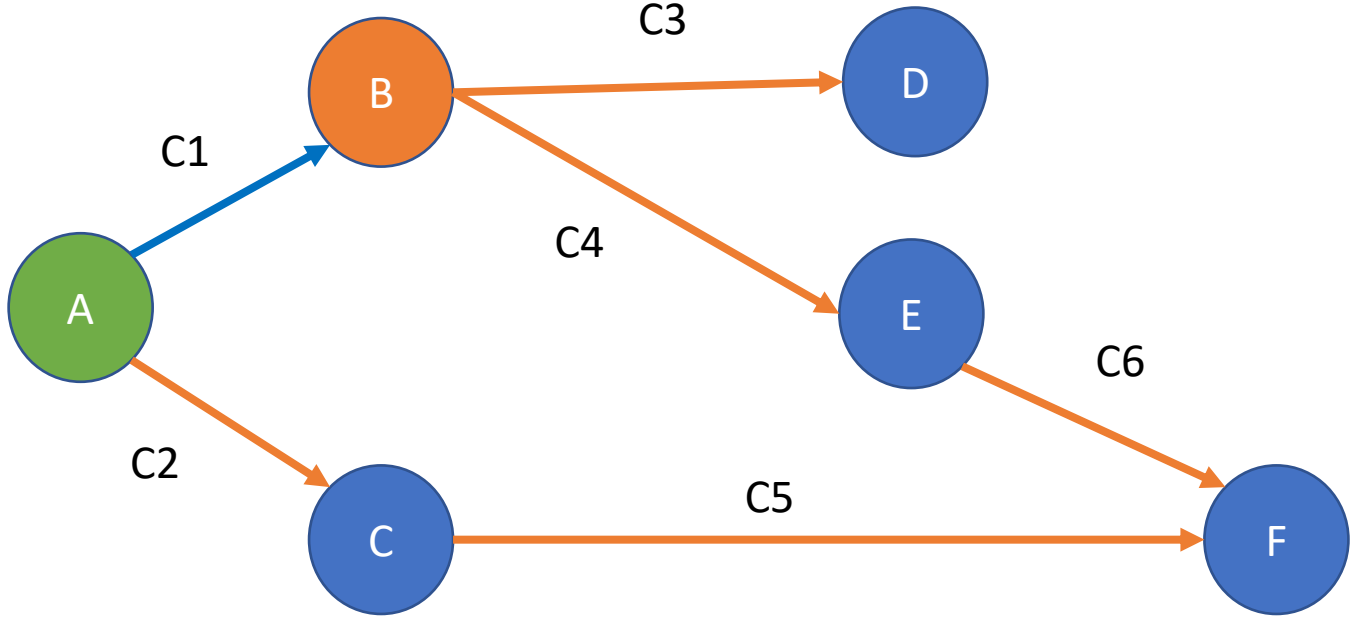
Depth-First Search (DFS)



1. Set a start node
2. While this is not an objective or final node (node whose adjacency has already been visited):
   - Choose an adjacent node not yet visited
   - Visit it
3. If it is a non-objective end node:
   - Return to this father
   - If there is a father, repeat. If there is no parent, choose another start node

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

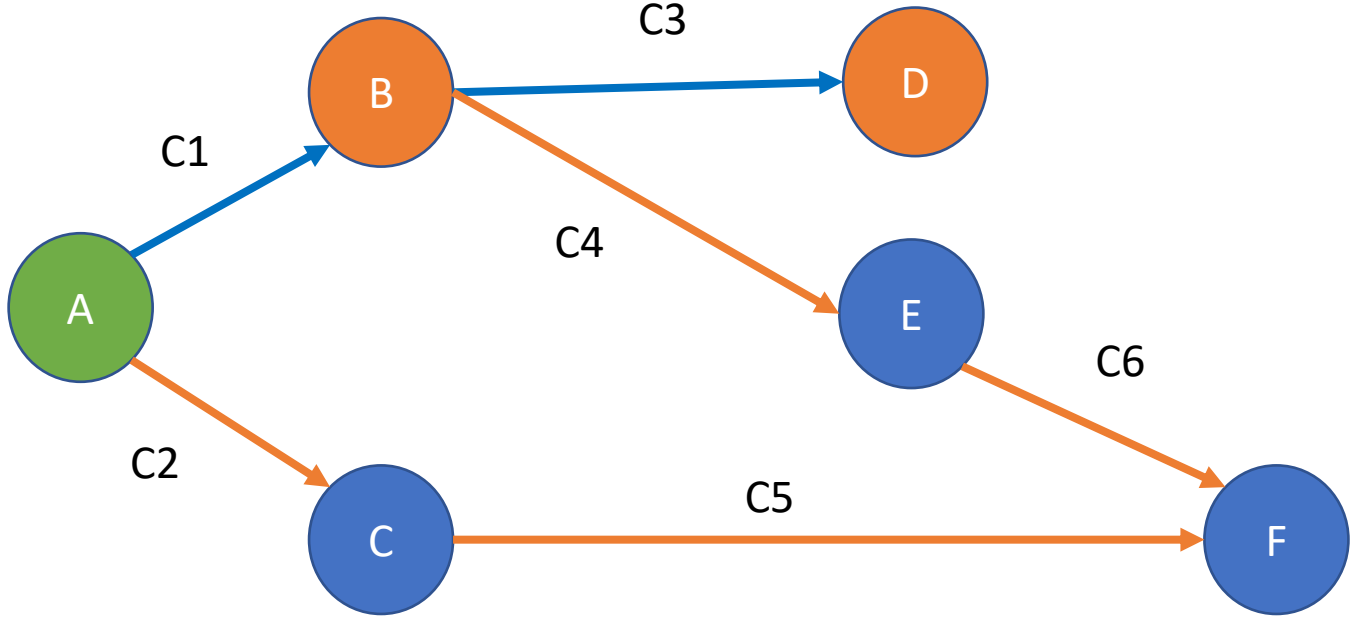
Depth-First Search (DFS)



1. Set a start node
2. While this is not an objective or final node (node whose adjacency has already been visited):
   - Choose an adjacent node not yet visited
   - Visit it
3. If it is a non-objective end node:
   - Return to this father
   - If there is a father, repeat. If there is no parent, choose another start node

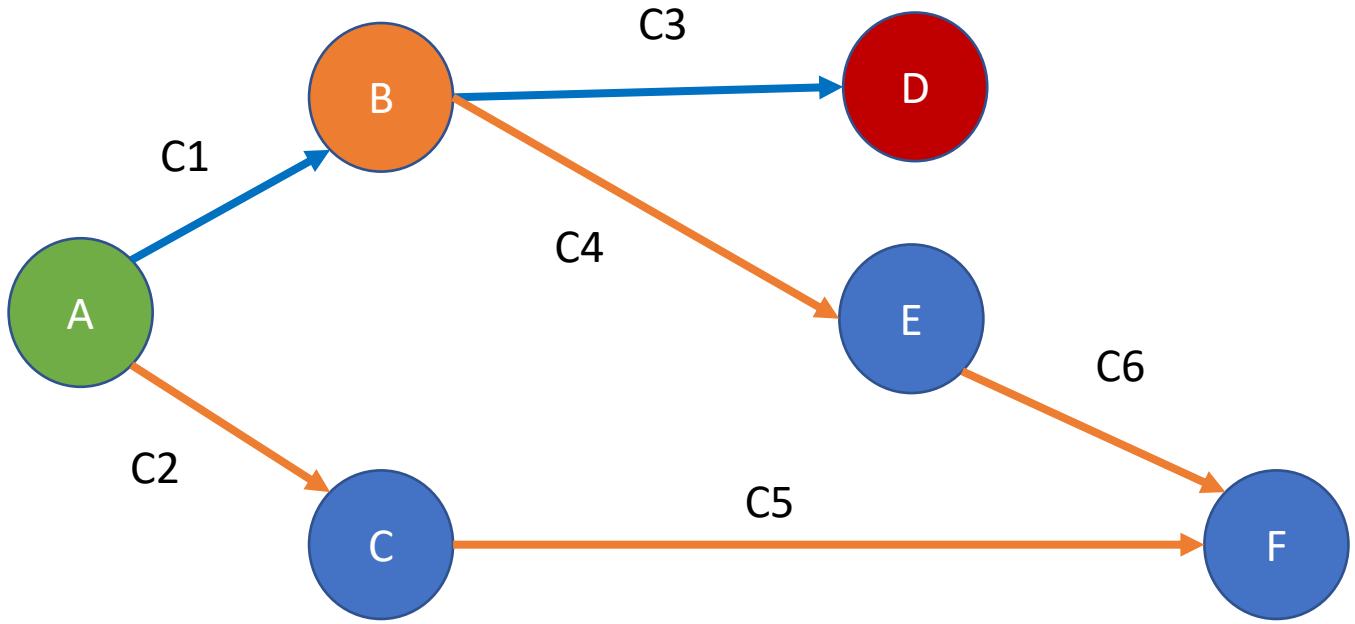
4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

Depth-First Search (DFS)



1. Set a start node
2. While this is not an objective or final node (node whose adjacency has already been visited):
   - Choose an adjacent node not yet visited
   - Visit it
3. If it is a non-objective end node:
   - Return to this father
   - If there is a father, repeat. If there is no parent, choose another start node

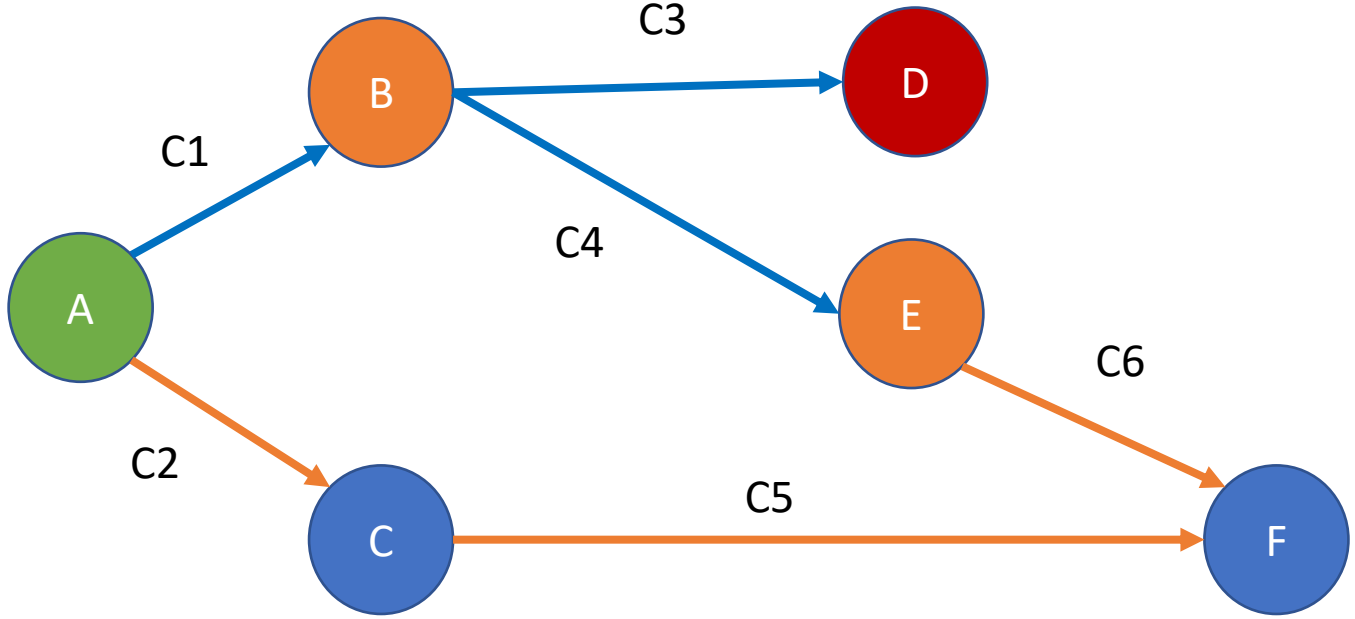
4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.



1. Set a start node
2. While this is not an objective or final node (node whose adjacency has already been visited):
   - Choose an adjacent node not yet visited
   - Visit it
3. If it is a non-objective end node:
   - Return to this father
   - If there is a father, repeat. If there is no parent, choose another start node

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

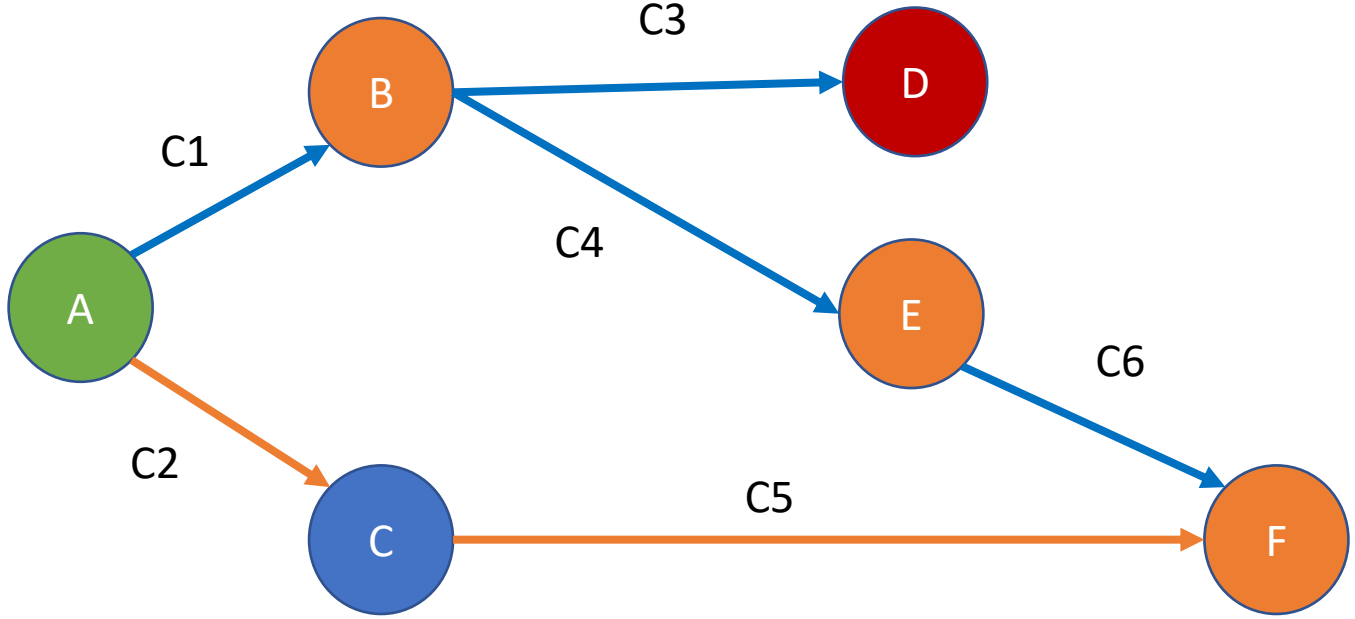
Depth-First Search (DFS)



1. Set a start node
2. While this is not an objective or final node (node whose adjacency has already been visited):
   - Choose an adjacent node not yet visited
   - Visit it
3. If it is a non-objective end node:
   - Return to this father
   - If there is a father, repeat. If there is no parent, choose another start node

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

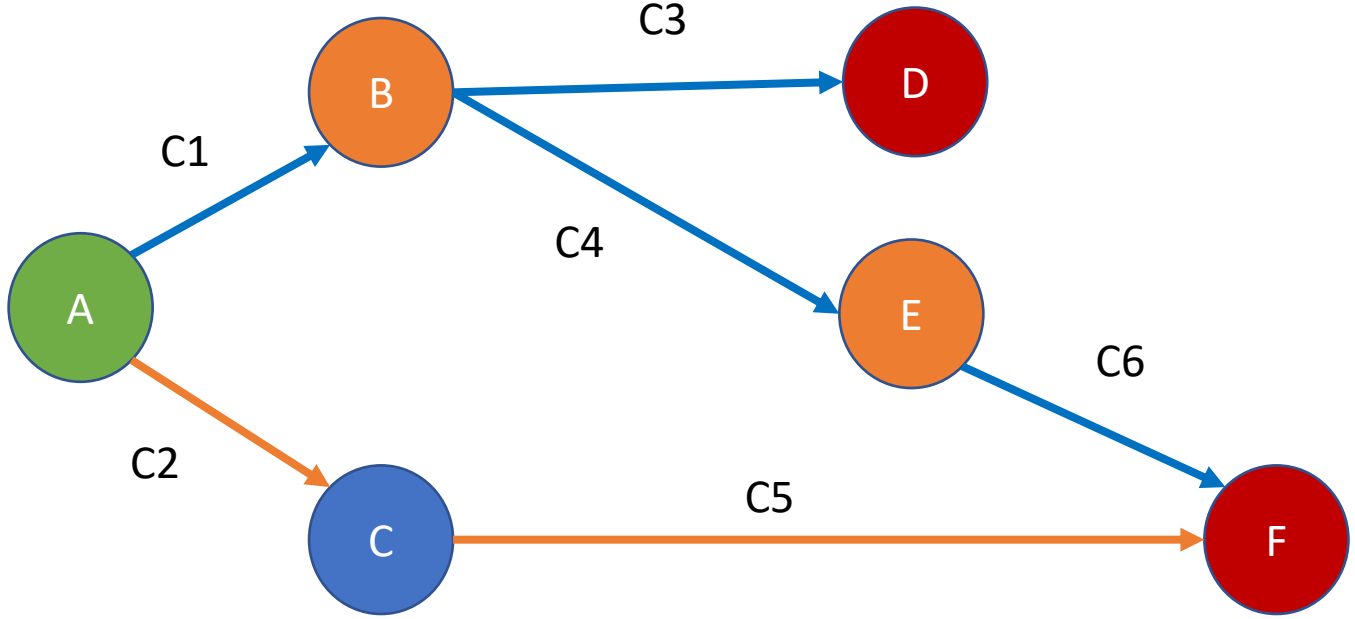
Depth-First Search (DFS)



1. Set a start node
2. While this is not an objective or final node (node whose adjacency has already been visited):
   - Choose an adjacent node not yet visited
   - Visit it
3. If it is a non-objective end node:
   - Return to this father
   - If there is a father, repeat. If there is no parent, choose another start node

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

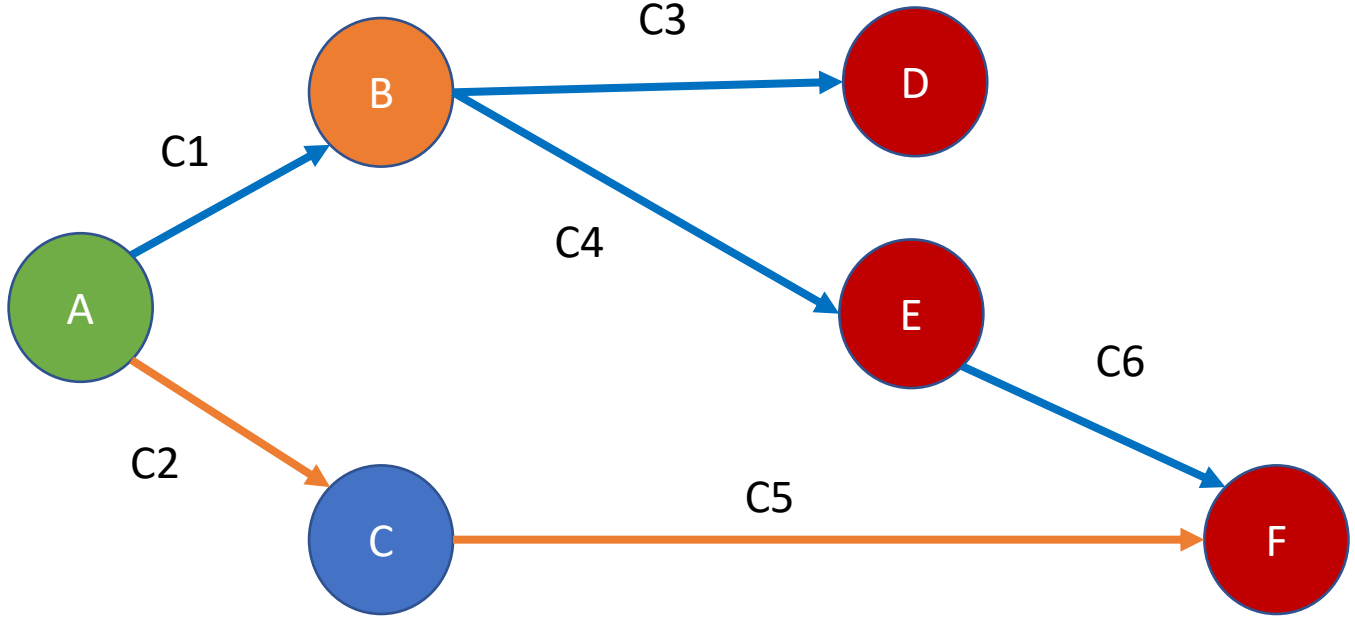
Depth-First Search (DFS)



1. Set a start node
2. While this is not an objective or final node (node whose adjacency has already been visited):
   - Choose an adjacent node not yet visited
   - Visit it
3. If it is a non-objective end node:
   - Return to this father
   - If there is a father, repeat. If there is no parent, choose another start node

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

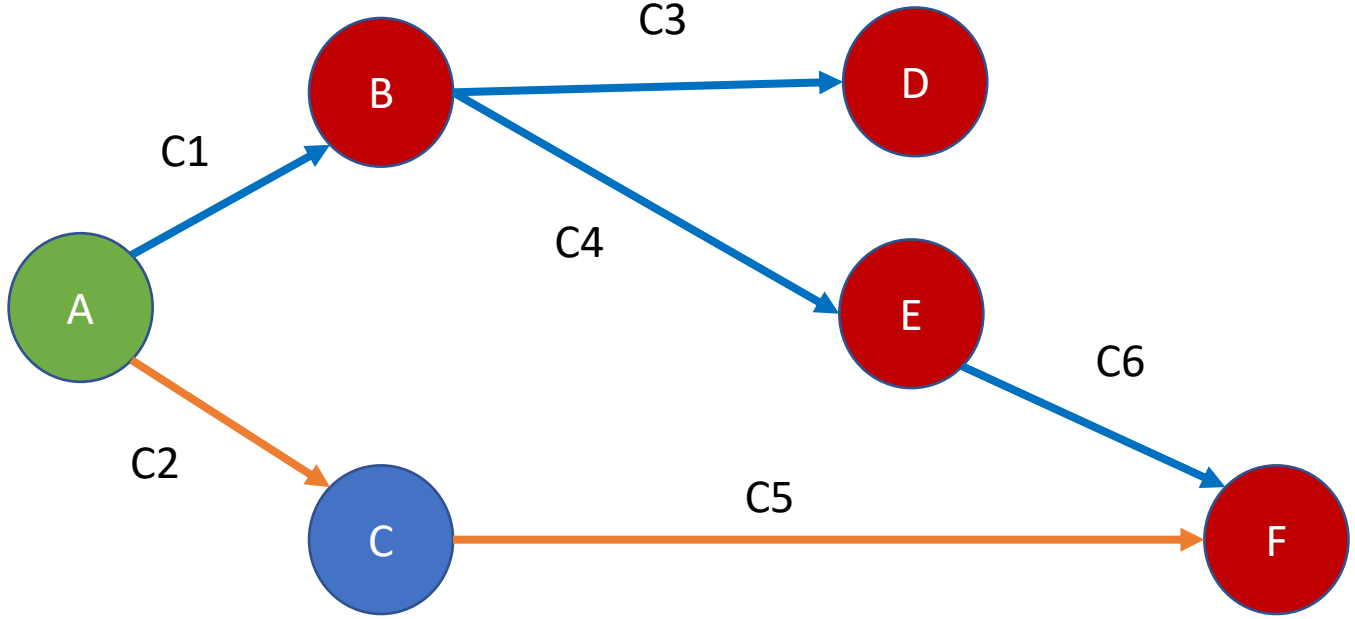
Depth-First Search (DFS)



1. Set a start node
2. While this is not an objective or final node (node whose adjacency has already been visited):
   - Choose an adjacent node not yet visited
   - Visit it
3. If it is a non-objective end node:
   - Return to this father
   - If there is a father, repeat. If there is no parent, choose another start node

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

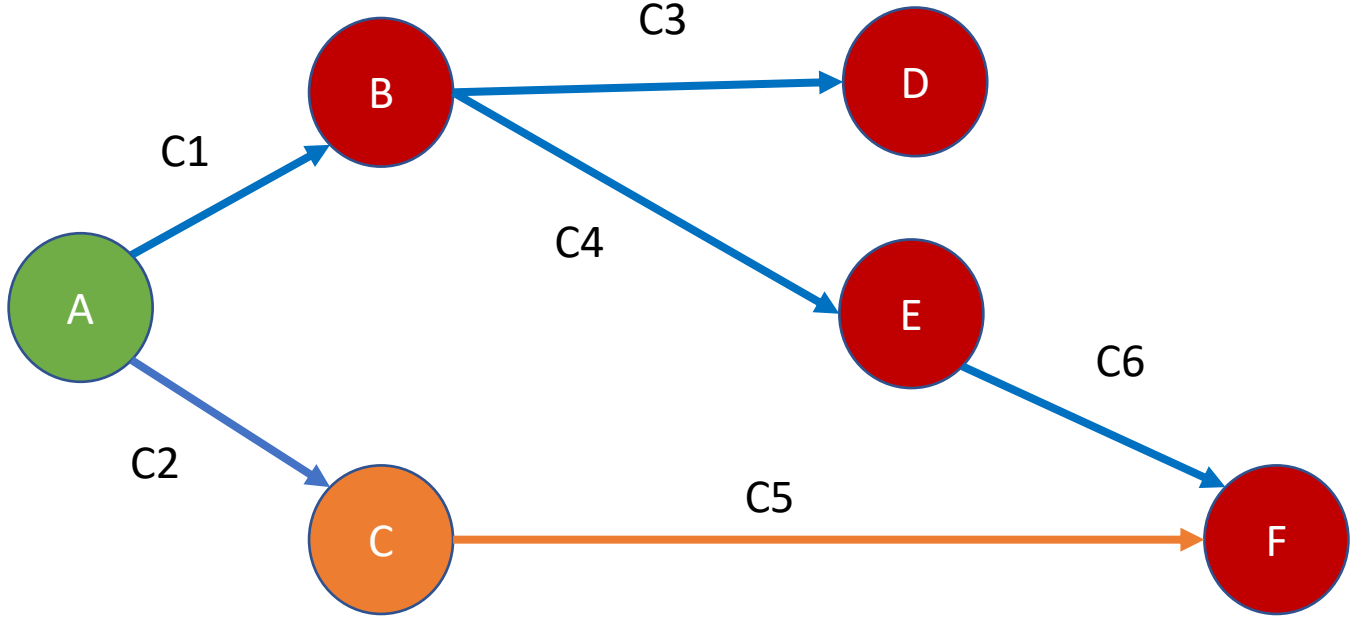
Depth-First Search (DFS)



1. Set a start node
2. While this is not an objective or final node (node whose adjacency has already been visited):
   - Choose an adjacent node not yet visited
   - Visit it
3. If it is a non-objective end node:
   - Return to this father
   - If there is a father, repeat. If there is no parent, choose another start node

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

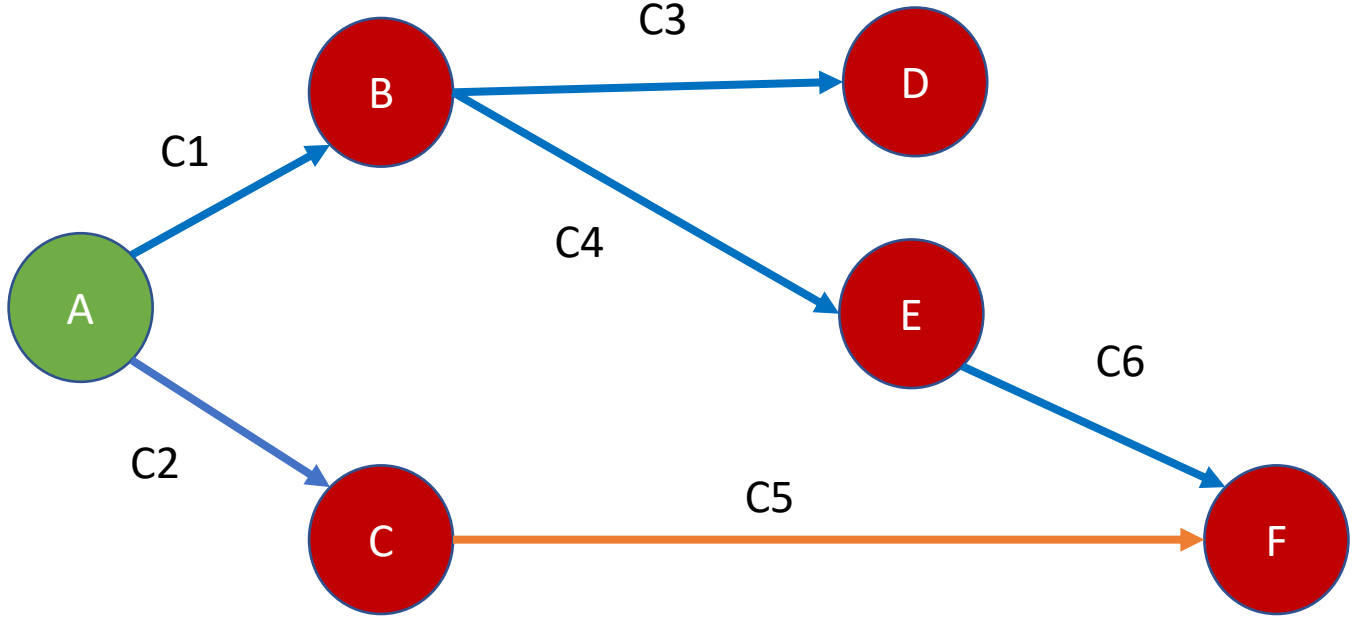
Depth-First Search (DFS)



1. Set a start node
2. While this is not an objective or final node (node whose adjacency has already been visited):
   - Choose an adjacent node not yet visited
   - Visit it
3. If it is a non-objective end node:
   - Return to this father
   - If there is a father, repeat. If there is no parent, choose another start node

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

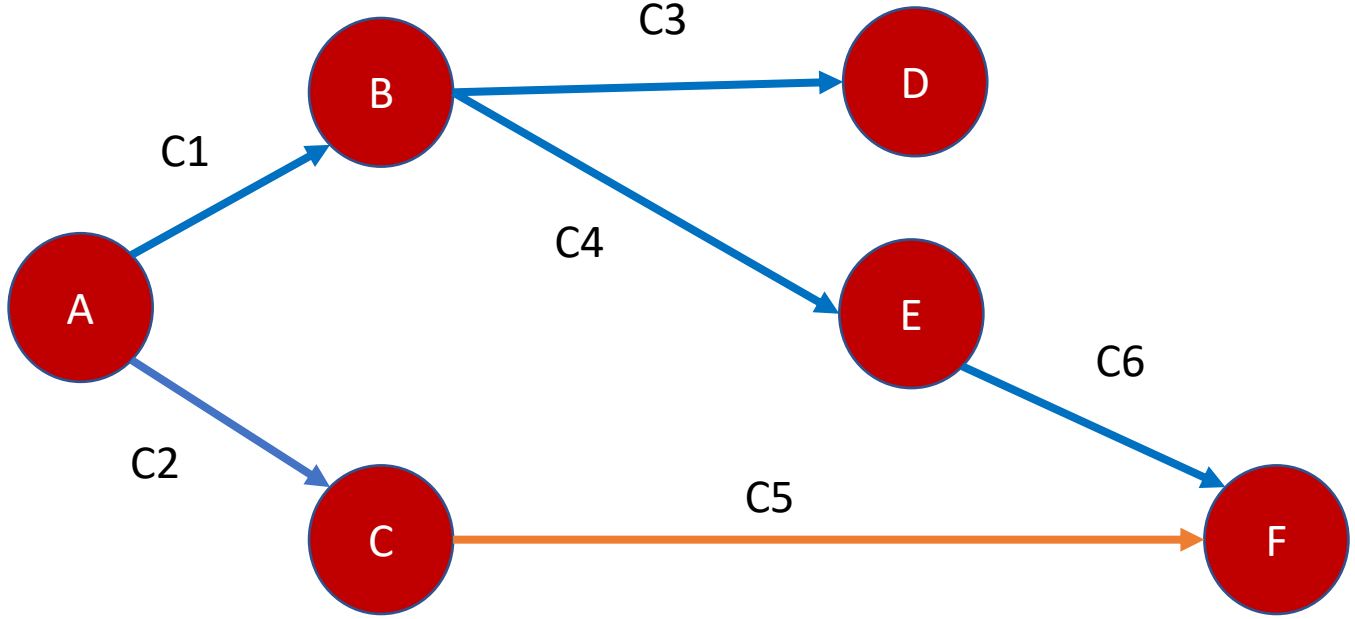
Depth-First Search (DFS)



1. Set a start node
2. While this is not an objective or final node (node whose adjacency has already been visited):
   - Choose an adjacent node not yet visited
   - Visit it
3. If it is a non-objective end node:
   - Return to this father
   - If there is a father, repeat. If there is no parent, choose another start node

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

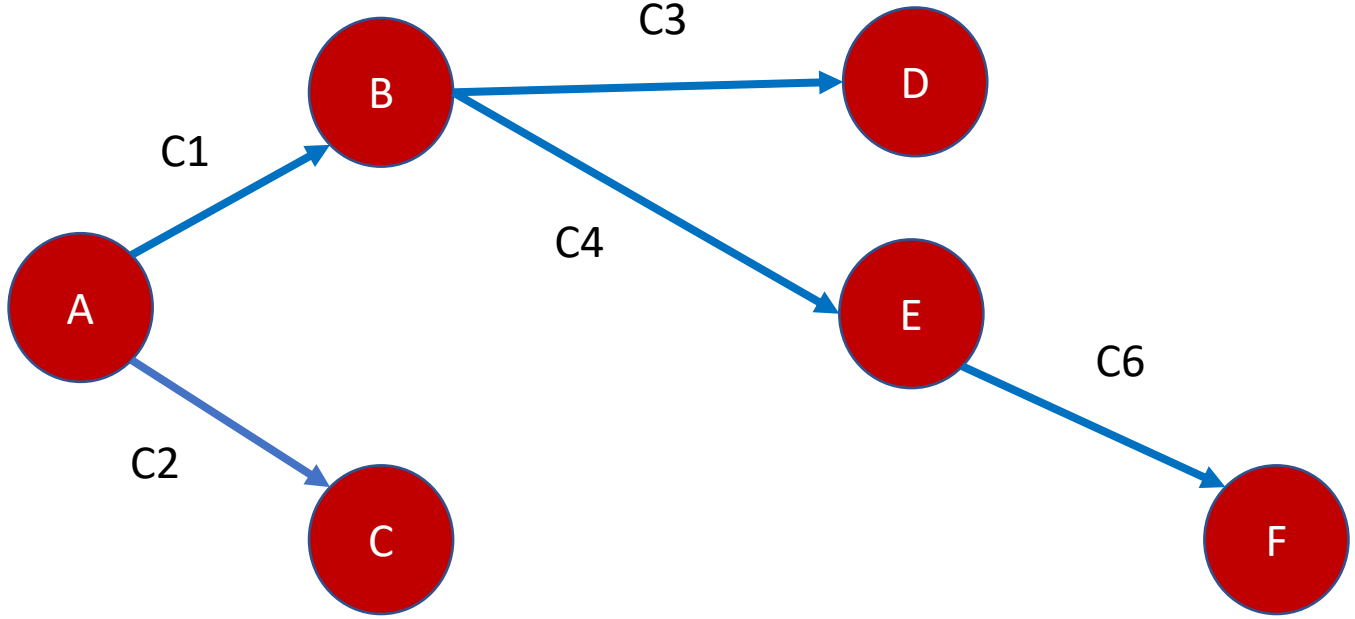
Depth-First Search (DFS)



1. Set a start node
2. While this is not an objective or final node (node whose adjacency has already been visited):
   - Choose an adjacent node not yet visited
   - Visit it
3. If it is a non-objective end node:
   - Return to this father
   - If there is a father, repeat. If there is no parent, choose another start node

4. Utilize o grafo a seguir para explicar o funcionamento do Algoritmo de Busca em Profundidade (DFS) e do Algoritmo de Busca em Largura (BFS). Destaque todos os passos da execução dos algoritmos e apresente o caminho final escolhido para ambos.

Depth-First Search (DFS)

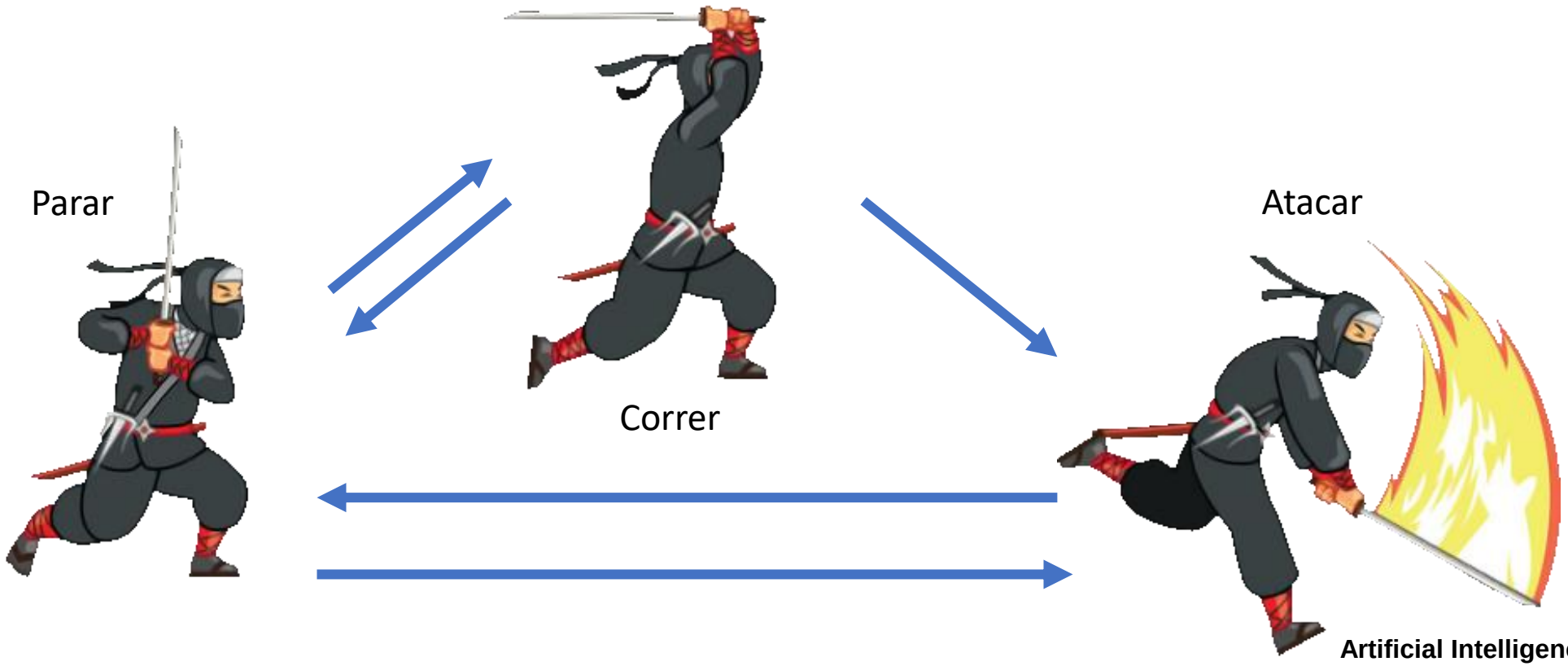


1. Set a start node
2. While this is not an objective or final node (node whose adjacency has already been visited):
   - Choose an adjacent node not yet visited
   - Visit it
3. If it is a non-objective end node:
   - Return to this father
   - If there is a father, repeat. If there is no parent, choose another start node

5. Defina Máquina de Estados e Árvore de Comportamentos aplicados no desenvolvimento de jogos digitais. Apresente um exemplo de aplicação para cada uma das definições.

Máquina de Estados:

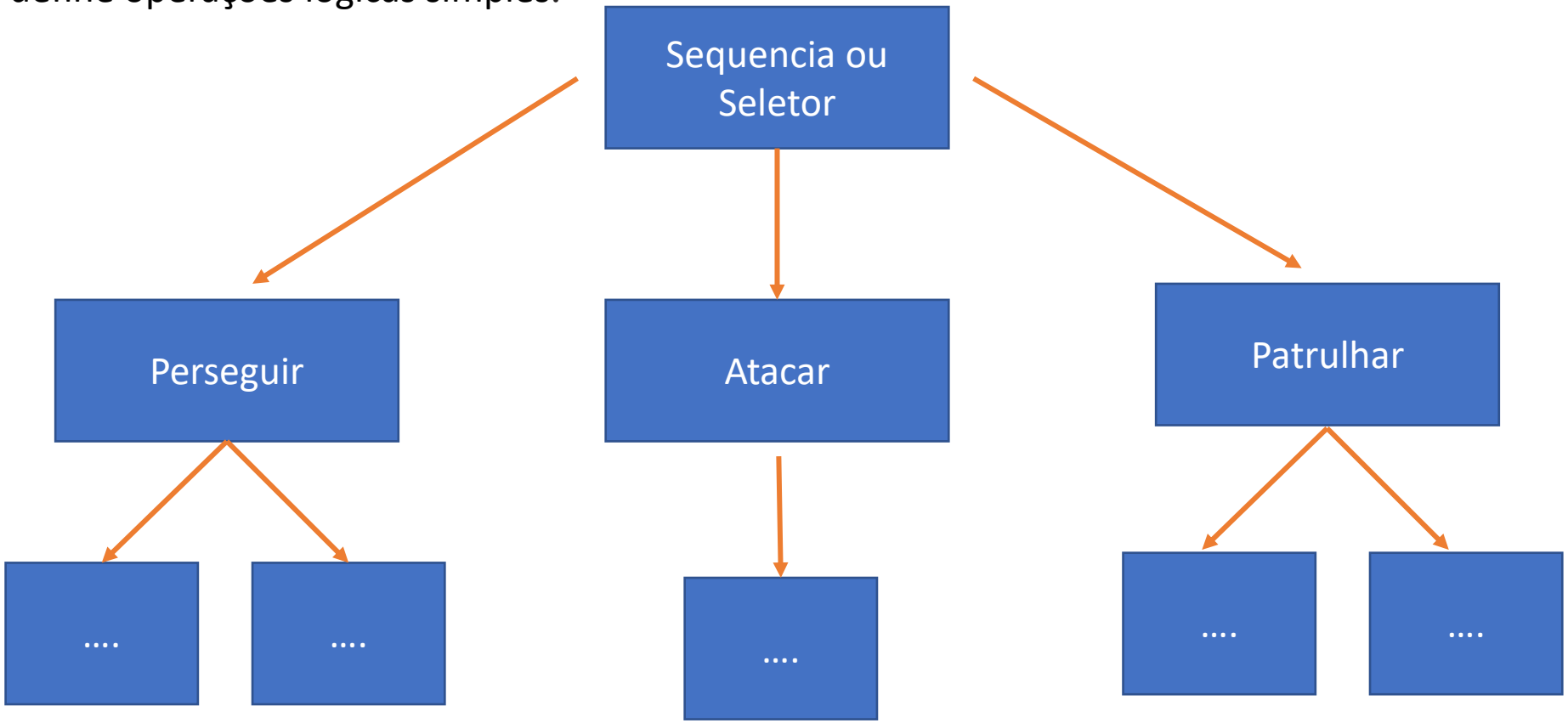
- *É um modelo matemático utilizado para representar os diversos comportamentos e as respetivas transições entre estes em um programa.*
- *É **composta por estados e transições**.*
- *É um conjunto de estados finitos que funcionam como intermediários entre uma relação de entrada e saídas. Desta forma, a saída dependerá do estado das entradas naquele momento*



5. Defina Máquina de Estados e Árvore de Comportamentos aplicados no desenvolvimento de jogos digitais. Apresente um exemplo de aplicação para cada uma das definições.

Árvore de Comportamentos:

- É uma árvore de nós hierárquicos que controlam o fluxo de tomada de decisão de uma entidade de Inteligência Artificial (IA)
- É uma arquitetura de IA que fornece aos **Non Player Characters (NPC)** do jogo a capacidade de selecionar comportamentos e executá-los, por meio de uma arquitetura semelhante a uma árvore que define operações lógicas simples.

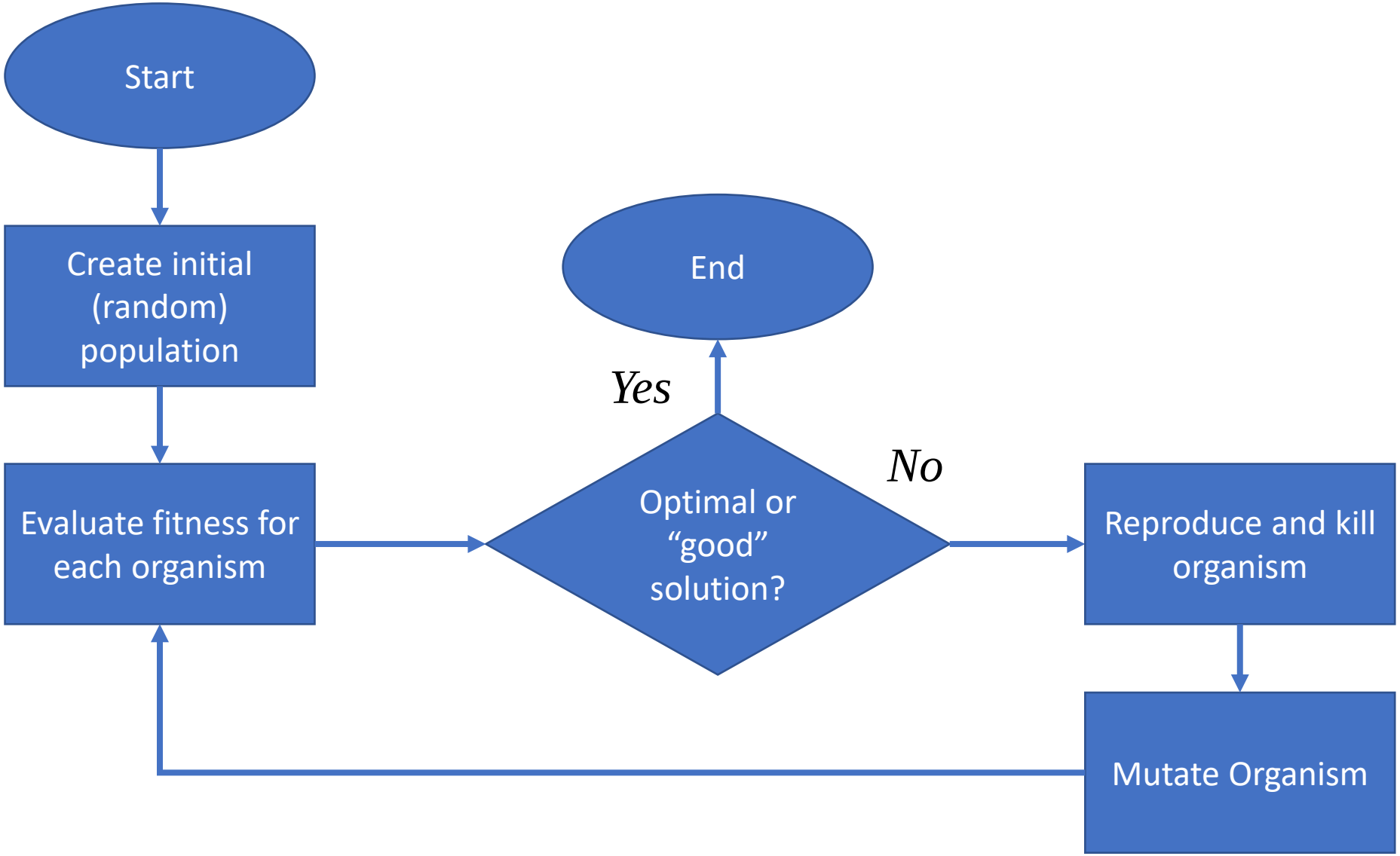


6. Explique o que é Algoritmo Genético, destaque suas características e em quais aplicações eles são mais bem utilizados. Adicionalmente, apresente e explique o diagrama de funcionamento de um Algoritmo Genético.

- Algoritmos Genéticos são algoritmos baseados em analogias biológicas e na evolução das espécies
- Tem procedimentos, com passos distintos e bem definidos.

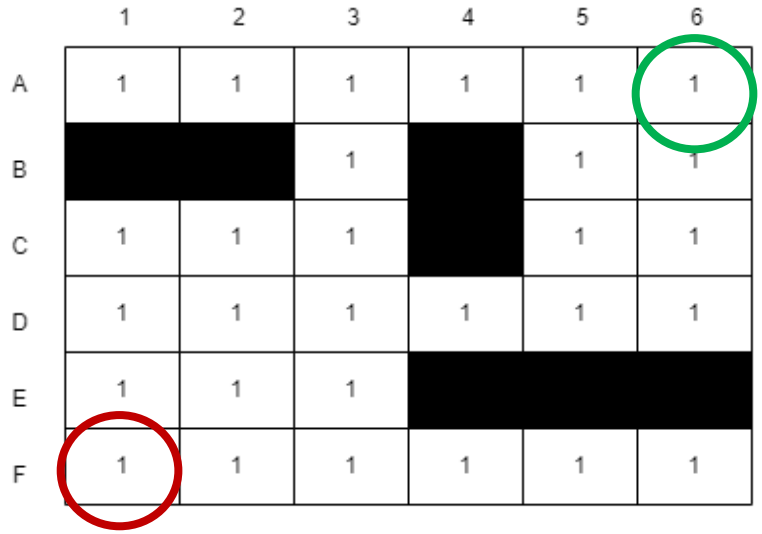
São melhores aplicados em Pesquisa e Otimização e são amplamente utilizado em problemas de difícil resolução com técnicas tradicionais

6. Explique o que é Algoritmo Genético, destaque suas características e em quais aplicações eles são mais bem utilizados. Adicionalmente, apresente e explique o diagrama de funcionamento de um Algoritmo Genético.



7. Supondo um labirinto como o apresentado na Figura 1, onde todos os passos têm peso 1 e as paredes (pontos escuros) tem valor infinitamente altos, qual o percurso o algoritmo de Path Finding A* escolherá para o personagem localizado no ponto F1 e com o objetivo localizado no ponto A6? Apresente todos os passos de execução do algoritmo A* e o caminho escolhido no final. Considere os valores de heurística apresentados na Figura 2.

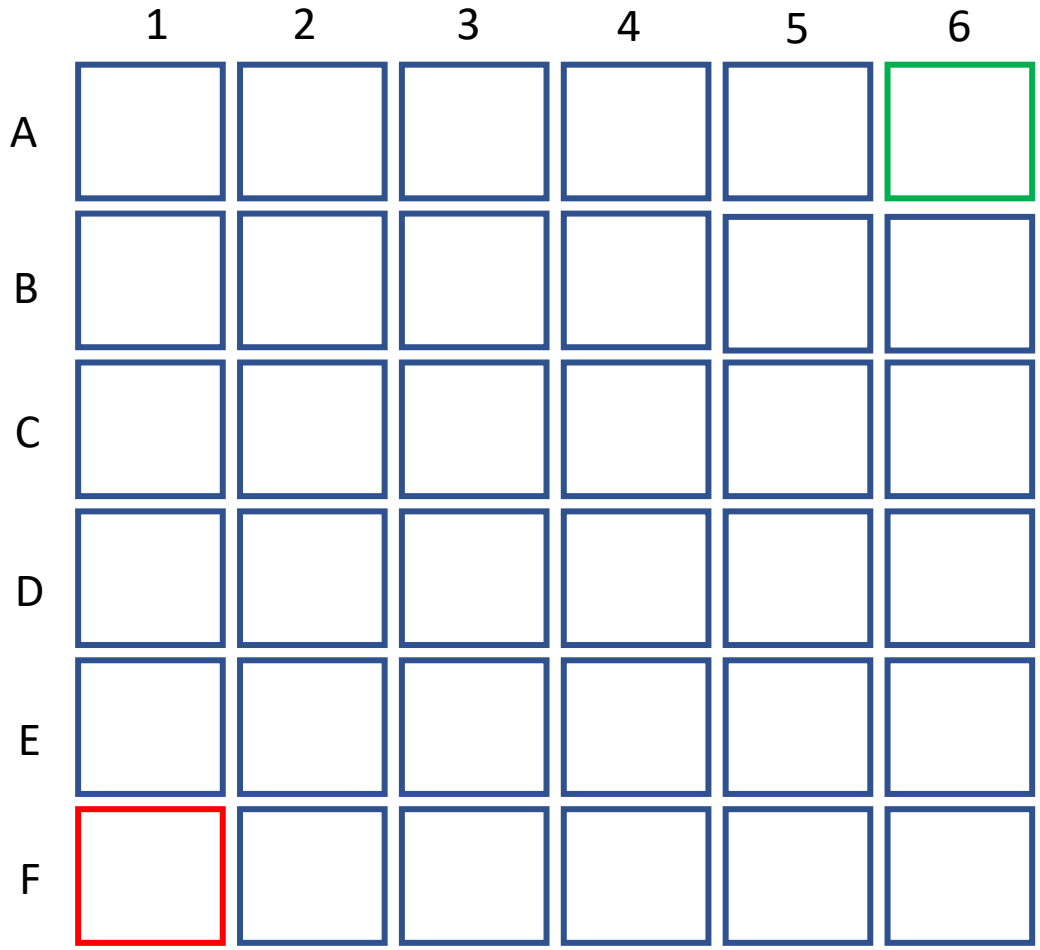
Figura 1



	1	2	3	4	5	6
A	1	1	1	1	1	1
B			1		1	1
C	1	1	1		1	1
D	1	1	1	1	1	1
E	1	1	1			
F	1	1	1	1	1	1

Figura 2

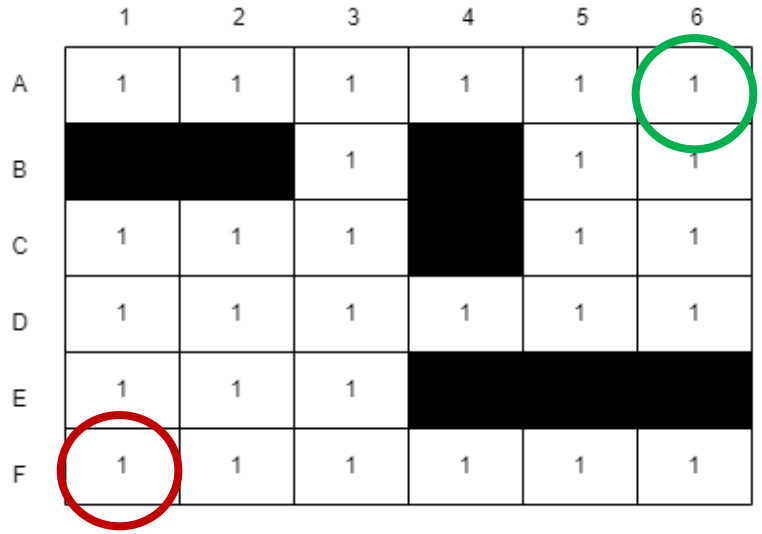
	1	2	3	4	5	6
A	5	4	3	2	1	0
B	6	5	4	3	2	1
C	7	6	5	4	3	2
D	8	7	6	5	4	3
E	9	8	7	6	5	4
F	10	9	8	7	6	5



manhattan((x1, y1), (x2, y2)) = |x1 — x2| + |y1 — y2|

7. Supondo um labirinto como o apresentado na Figura 1, onde todos os passos têm peso 1 e as paredes (pontos escuros) tem valor infinitamente altos, qual o percurso o algoritmo de Path Finding A* escolherá para o personagem localizado no ponto F1 e com o objetivo localizado no ponto A6? Apresente todos os passos de execução do algoritmo A* e o caminho escolhido no final. Considere os valores de heurística apresentados na Figura 2.

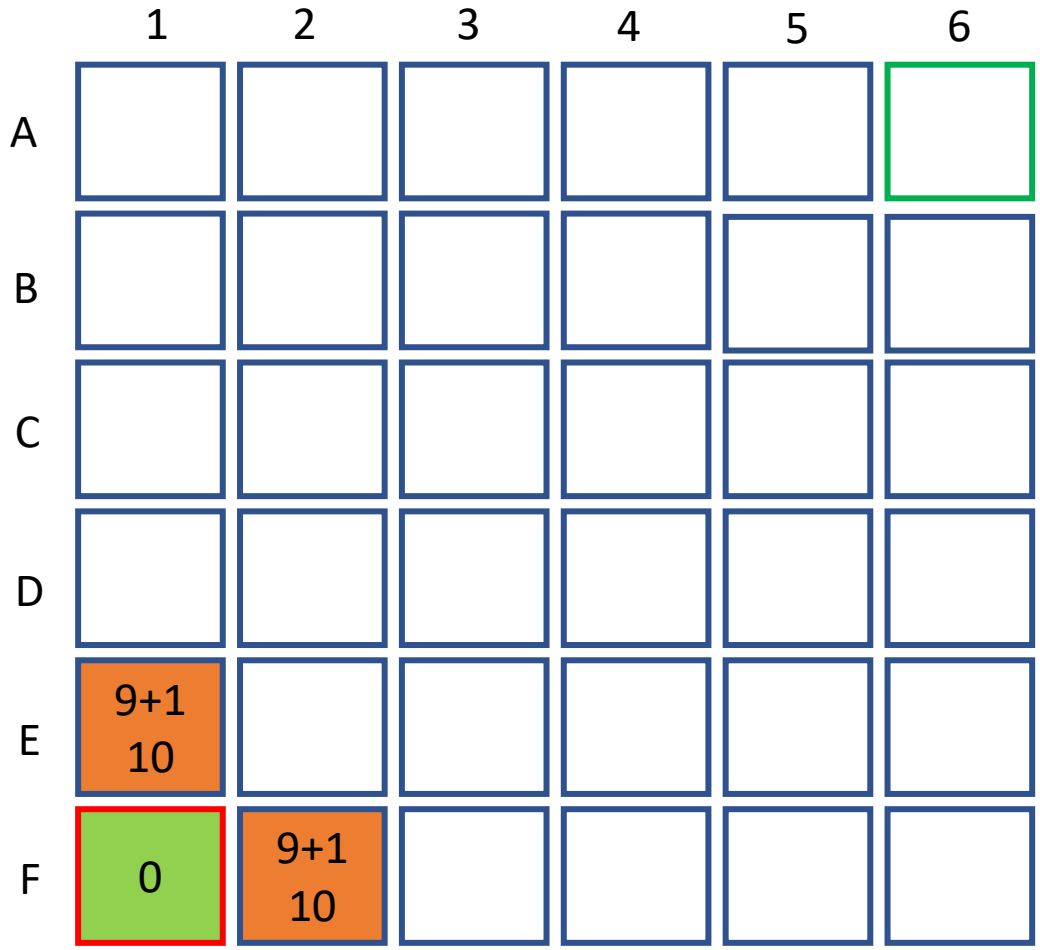
Figura 1



	1	2	3	4	5	6
A	1	1	1	1	1	1
B			1		1	1
C	1	1	1		1	1
D	1	1	1	1	1	1
E	1	1	1			
F	1	1	1	1	1	1

Figura 2

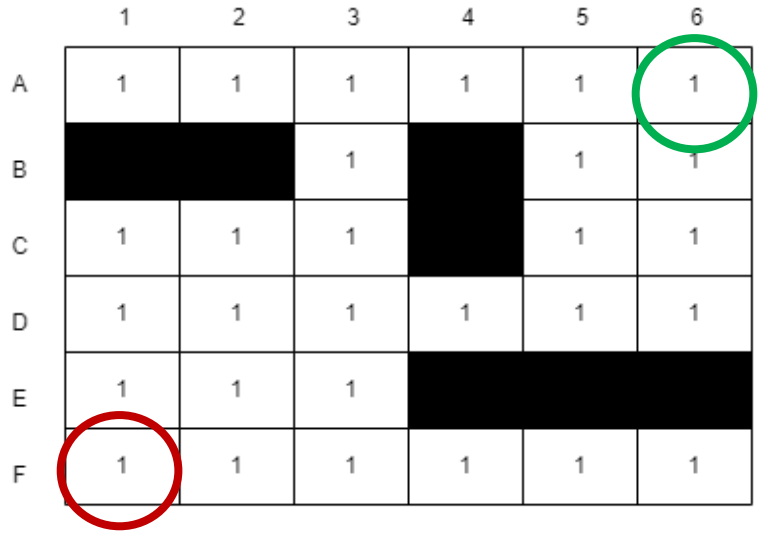
	1	2	3	4	5	6
A	5	4	3	2	1	0
B	6	5	4	3	2	1
C	7	6	5	4	3	2
D	8	7	6	5	4	3
E	9	8	7	6	5	4
F	10	9	8	7	6	5



manhattan((x1, y1), (x2, y2)) = |x1 — x2| + |y1 — y2|

7. Supondo um labirinto como o apresentado na Figura 1, onde todos os passos têm peso 1 e as paredes (pontos escuros) tem valor infinitamente altos, qual o percurso o algoritmo de Path Finding A* escolherá para o personagem localizado no ponto F1 e com o objetivo localizado no ponto A6? Apresente todos os passos de execução do algoritmo A* e o caminho escolhido no final. Considere os valores de heurística apresentados na Figura 2.

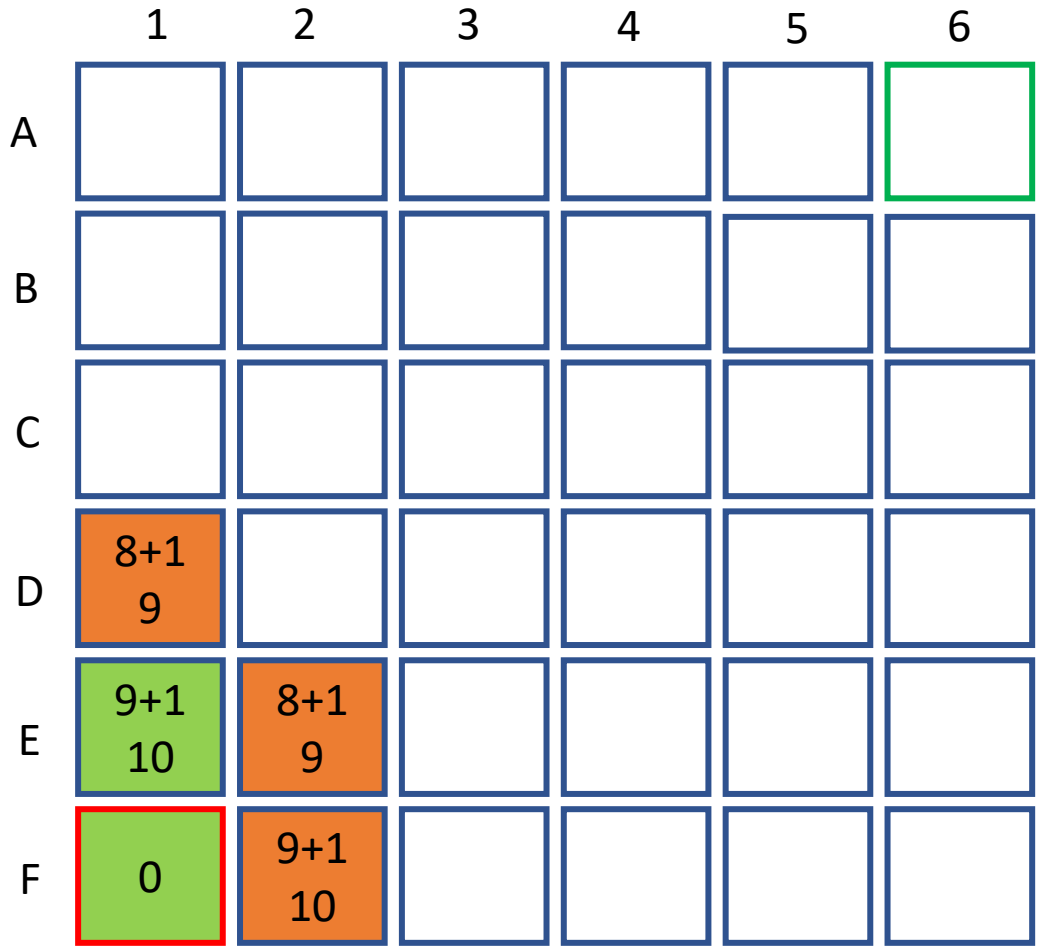
Figura 1



	1	2	3	4	5	6
A	1	1	1	1	1	1
B			1		1	1
C	1	1	1		1	1
D	1	1	1	1	1	1
E	1	1	1			
F	1	1	1	1	1	1

Figura 2

	1	2	3	4	5	6
A	5	4	3	2	1	0
B	6	5	4	3	2	1
C	7	6	5	4	3	2
D	8	7	6	5	4	3
E	9	8	7	6	5	4
F	10	9	8	7	6	5



manhattan((x1, y1), (x2, y2)) = |x1 — x2| + |y1 — y2|

7. Supondo um labirinto como o apresentado na Figura 1, onde todos os passos têm peso 1 e as paredes (pontos escuros) tem valor infinitamente altos, qual o percurso o algoritmo de Path Finding A* escolherá para o personagem localizado no ponto F1 e com o objetivo localizado no ponto A6? Apresente todos os passos de execução do algoritmo A* e o caminho escolhido no final. Considere os valores de heurística apresentados na Figura 2.

Figura 1

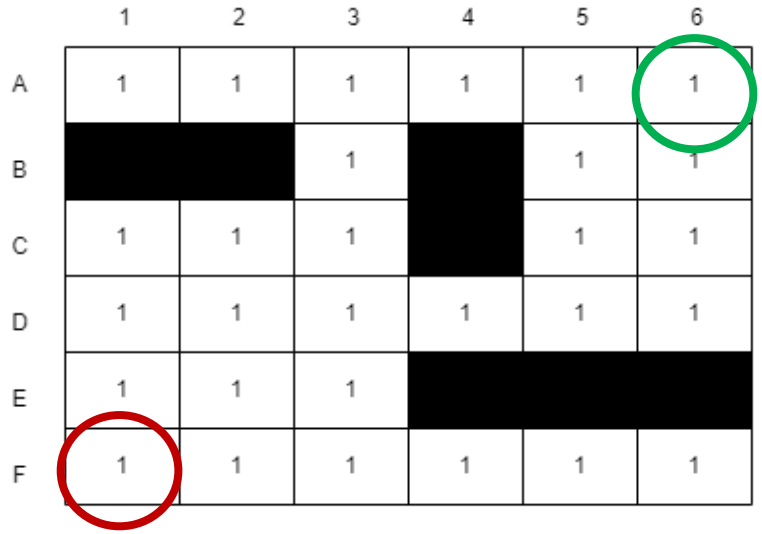
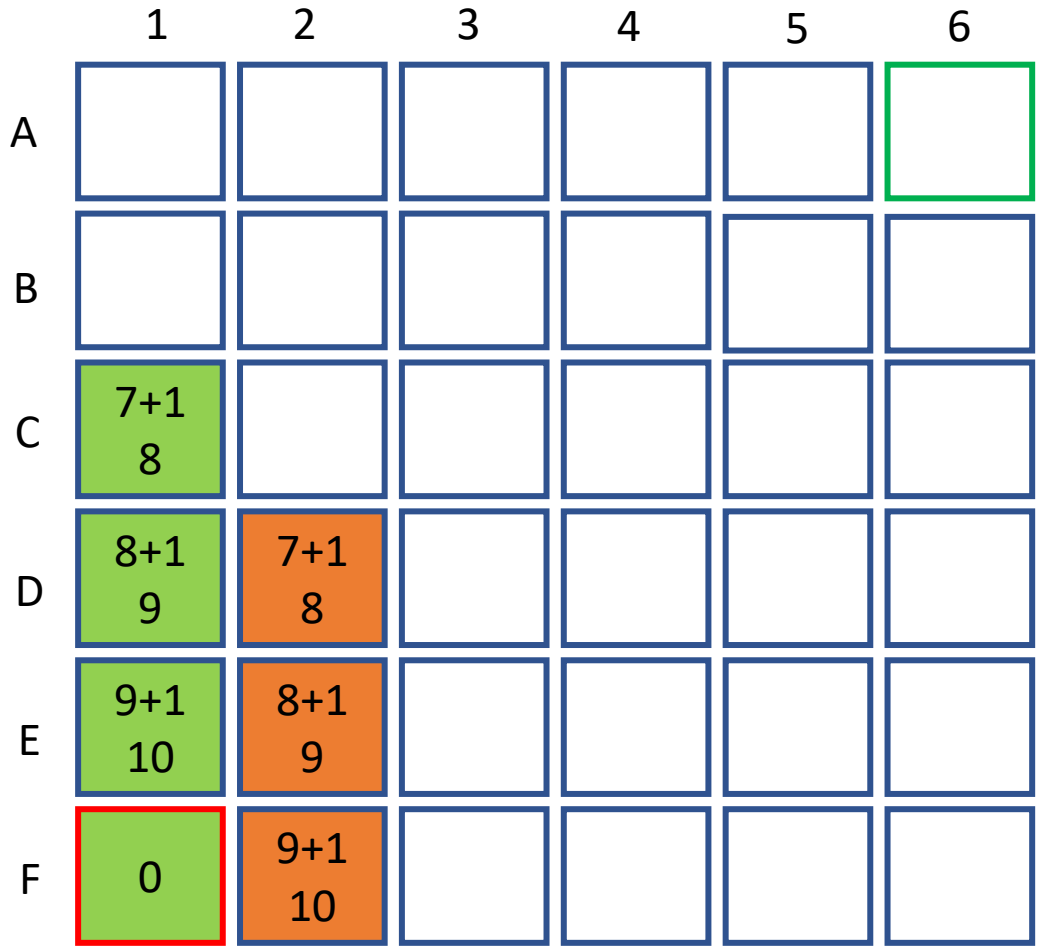


Figura 2

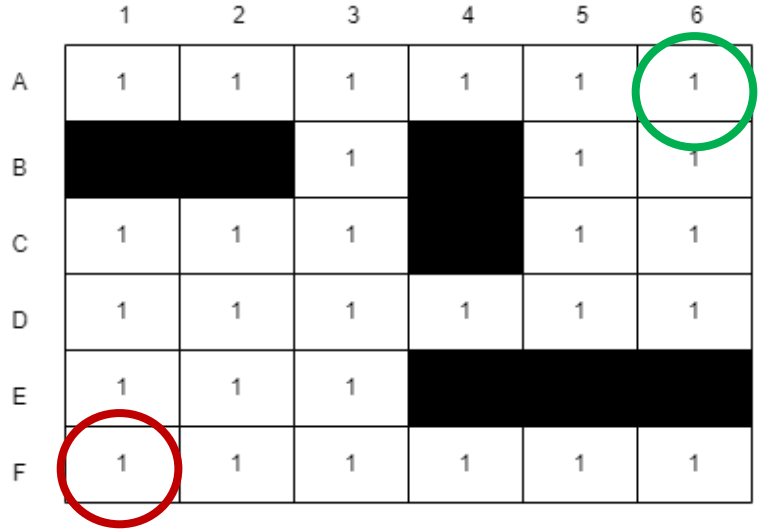
	1	2	3	4	5	6
A	5	4	3	2	1	0
B	6	5	4	3	2	1
C	7	6	5	4	3	2
D	8	7	6	5	4	3
E	9	8	7	6	5	4
F	10	9	8	7	6	5



manhattan((x1, y1), (x2, y2)) = |x1 — x2| + |y1 — y2|

7. Supondo um labirinto como o apresentado na Figura 1, onde todos os passos têm peso 1 e as paredes (pontos escuros) tem valor infinitamente altos, qual o percurso o algoritmo de Path Finding A* escolherá para o personagem localizado no ponto F1 e com o objetivo localizado no ponto A6? Apresente todos os passos de execução do algoritmo A* e o caminho escolhido no final. Considere os valores de heurística apresentados na Figura 2.

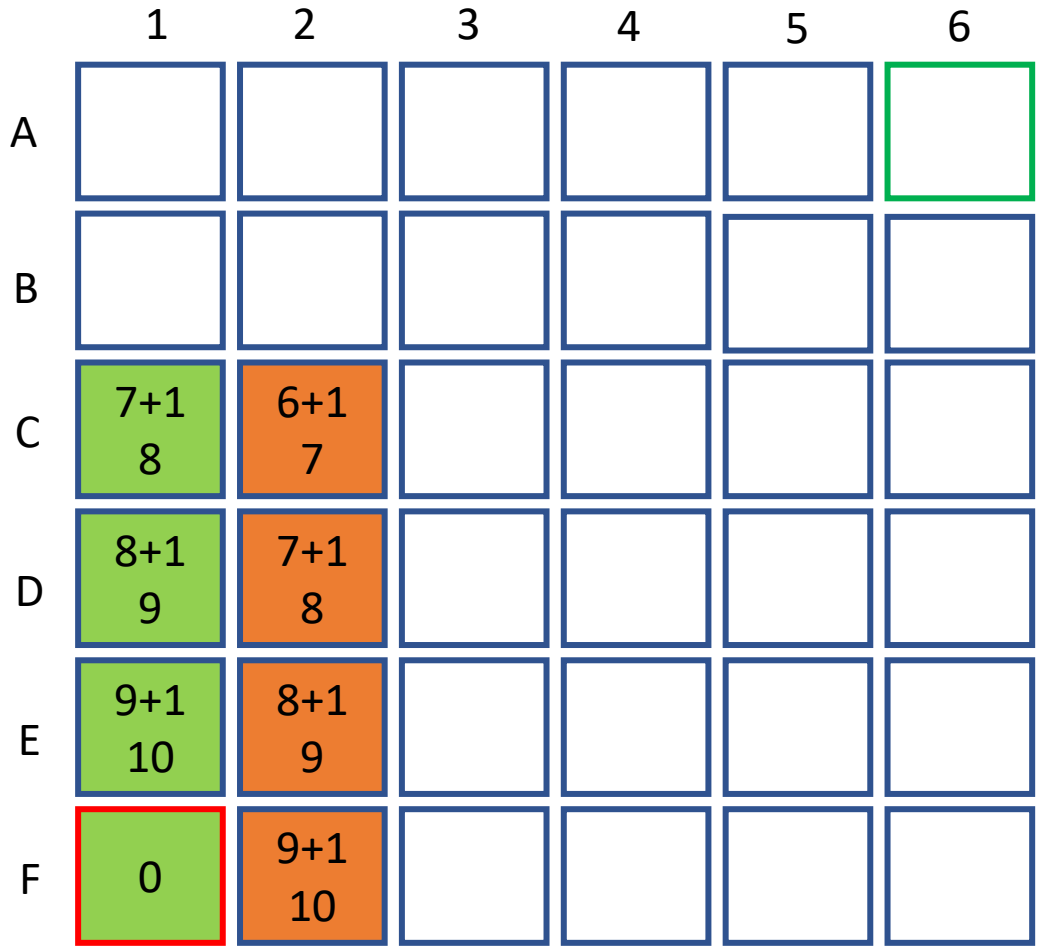
Figura 1



	1	2	3	4	5	6
A	1	1	1	1	1	1
B	■	■	1	■	1	1
C	1	1	1	■	1	1
D	1	1	1	1	1	1
E	1	1	1	■	■	■
F	1	1	1	1	1	1

Figura 2

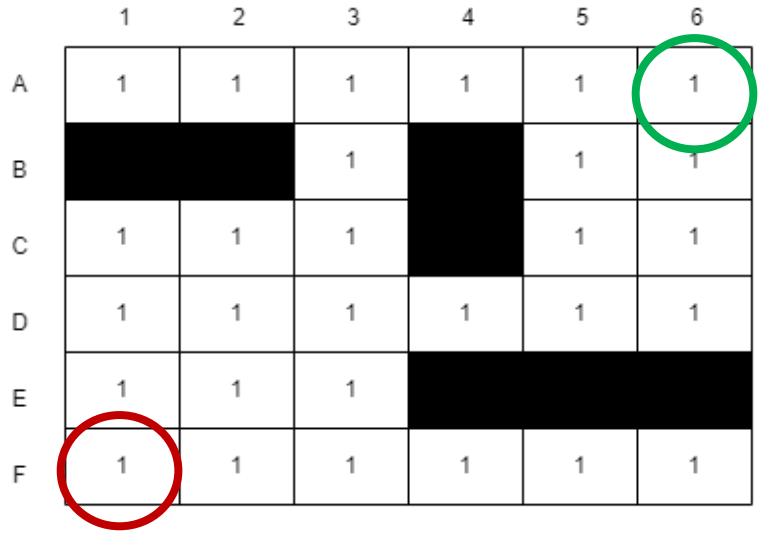
	1	2	3	4	5	6
A	5	4	3	2	1	0
B	6	5	4	3	2	1
C	7	6	5	4	3	2
D	8	7	6	5	4	3
E	9	8	7	6	5	4
F	10	9	8	7	6	5



manhattan((x1, y1), (x2, y2)) = |x1 — x2| + |y1 — y2|

7. Supondo um labirinto como o apresentado na Figura 1, onde todos os passos têm peso 1 e as paredes (pontos escuros) tem valor infinitamente altos, qual o percurso o algoritmo de Path Finding A* escolherá para o personagem localizado no ponto F1 e com o objetivo localizado no ponto A6? Apresente todos os passos de execução do algoritmo A* e o caminho escolhido no final. Considere os valores de heurística apresentados na Figura 2.

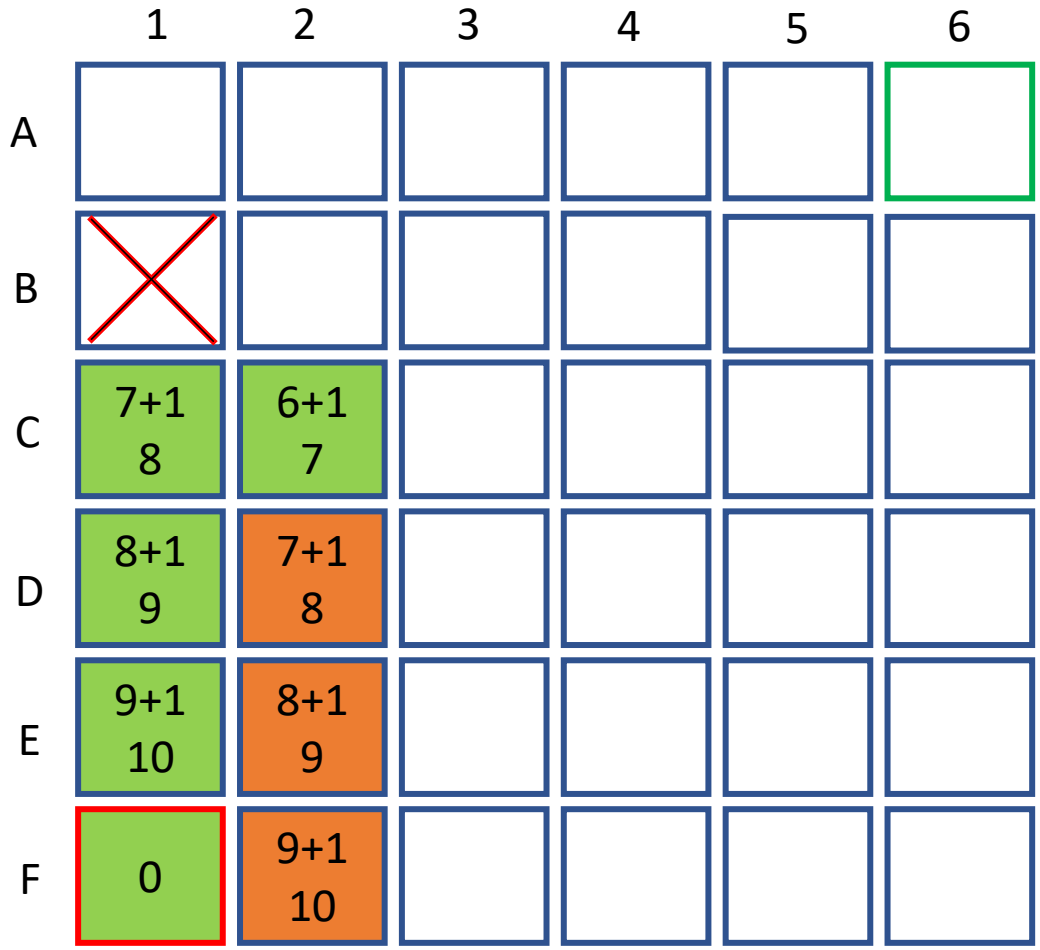
Figura 1



	1	2	3	4	5	6
A	1	1	1	1	1	1
B	■	■	1	■	1	1
C	1	1	1	■	1	1
D	1	1	1	1	1	1
E	1	1	1	■	■	■
F	1	1	1	1	1	1

Figura 2

	1	2	3	4	5	6
A	5	4	3	2	1	0
B	6	5	4	3	2	1
C	7	6	5	4	3	2
D	8	7	6	5	4	3
E	9	8	7	6	5	4
F	10	9	8	7	6	5



manhattan((x1, y1), (x2, y2)) = |x1 — x2| + |y1 — y2|

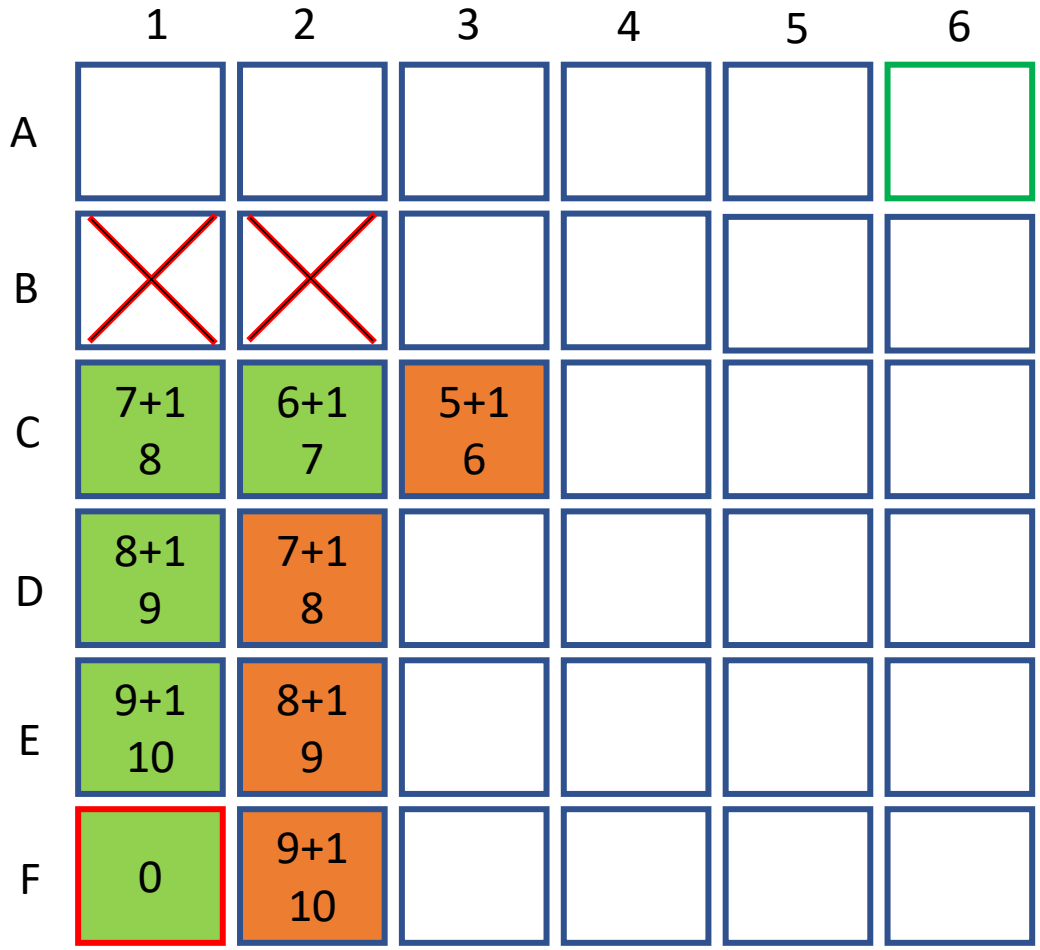
7. Supondo um labirinto como o apresentado na Figura 1, onde todos os passos têm peso 1 e as paredes (pontos escuros) tem valor infinitamente altos, qual o percurso o algoritmo de Path Finding A* escolherá para o personagem localizado no ponto F1 e com o objetivo localizado no ponto A6? Apresente todos os passos de execução do algoritmo A* e o caminho escolhido no final. Considere os valores de heurística apresentados na Figura 2.

Figura 1

	1	2	3	4	5	6
A	1	1	1	1	1	1
B	■	■	1	■	1	1
C	1	1	1	■	1	1
D	1	1	1	1	1	1
E	1	1	1	■	■	■
F	1	1	1	1	1	1

Figura 2

	1	2	3	4	5	6
A	5	4	3	2	1	0
B	6	5	4	3	2	1
C	7	6	5	4	3	2
D	8	7	6	5	4	3
E	9	8	7	6	5	4
F	10	9	8	7	6	5



manhattan((x1, y1), (x2, y2)) = |x1 — x2| + |y1 — y2|

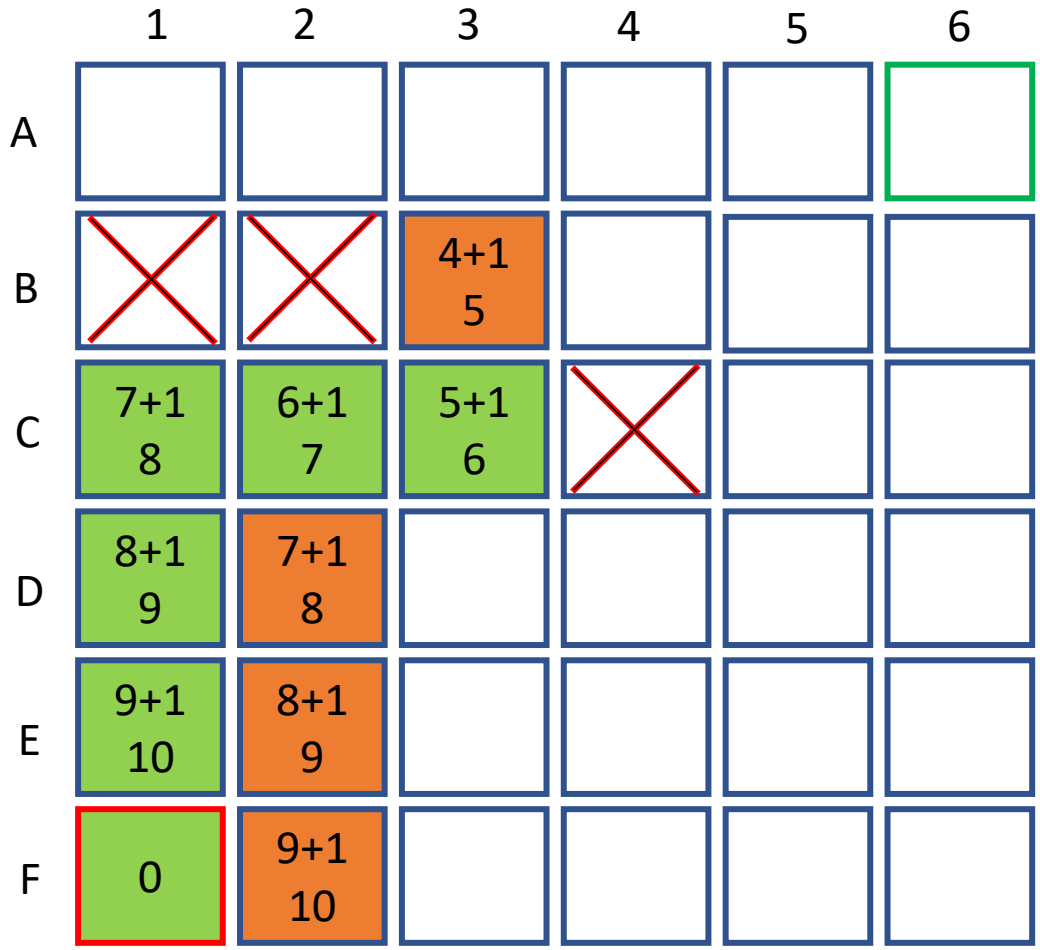
7. Supondo um labirinto como o apresentado na Figura 1, onde todos os passos têm peso 1 e as paredes (pontos escuros) tem valor infinitamente altos, qual o percurso o algoritmo de Path Finding A* escolherá para o personagem localizado no ponto F1 e com o objetivo localizado no ponto A6? Apresente todos os passos de execução do algoritmo A* e o caminho escolhido no final. Considere os valores de heurística apresentados na Figura 2.

Figura 1

	1	2	3	4	5	6
A	1	1	1	1	1	1
B	■	■	1	■	1	1
C	1	1	1	■	1	1
D	1	1	1	1	1	1
E	1	1	1	■	■	■
F	1	1	1	1	1	1

Figura 2

	1	2	3	4	5	6
A	5	4	3	2	1	0
B	6	5	4	3	2	1
C	7	6	5	4	3	2
D	8	7	6	5	4	3
E	9	8	7	6	5	4
F	10	9	8	7	6	5



manhattan((x1, y1), (x2, y2)) = |x1 — x2| + |y1 — y2|

7. Supondo um labirinto como o apresentado na Figura 1, onde todos os passos têm peso 1 e as paredes (pontos escuros) tem valor infinitamente altos, qual o percurso o algoritmo de Path Finding A* escolherá para o personagem localizado no ponto F1 e com o objetivo localizado no ponto A6? Apresente todos os passos de execução do algoritmo A* e o caminho escolhido no final. Considere os valores de heurística apresentados na Figura 2.

Figura 1

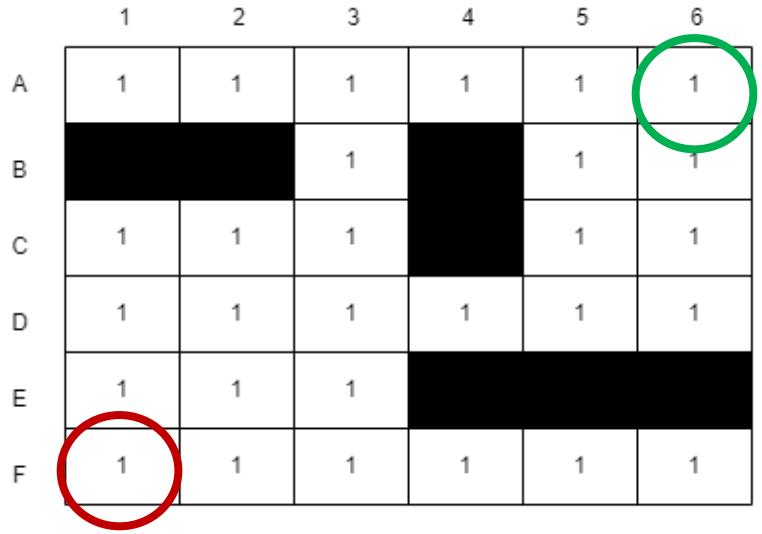
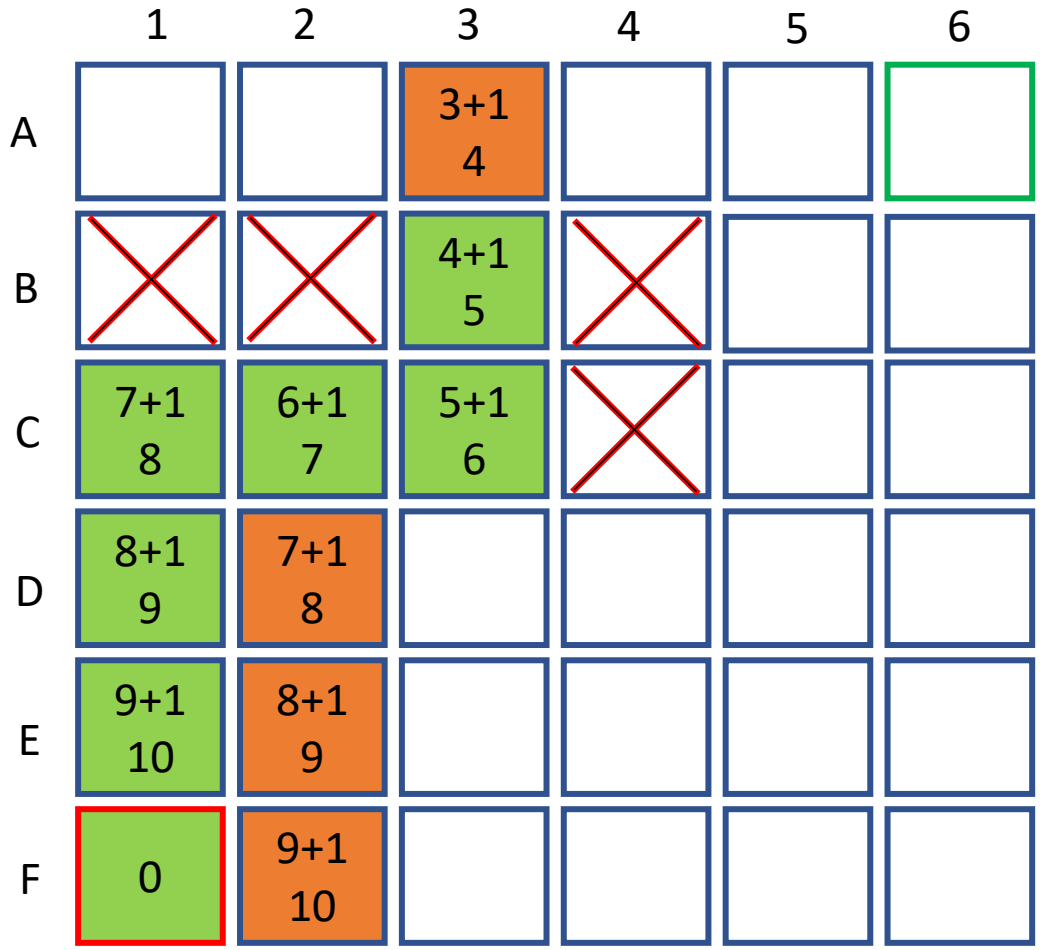


Figura 2

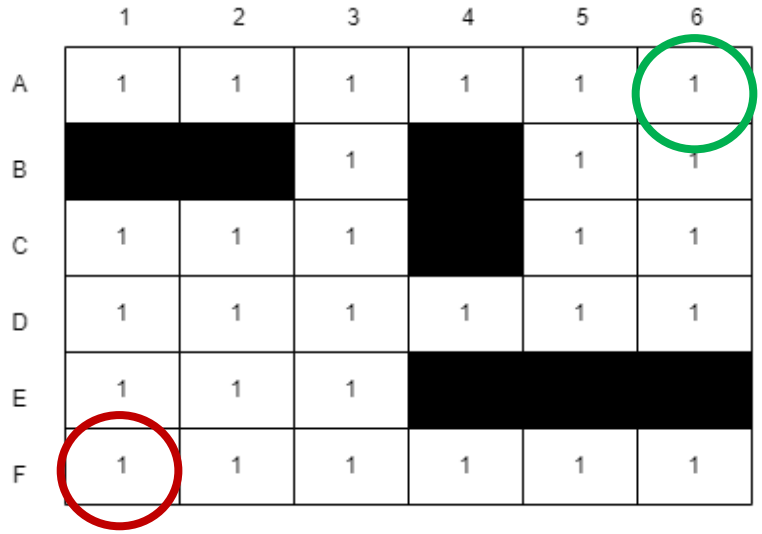
	1	2	3	4	5	6
A	5	4	3	2	1	0
B	6	5	4	3	2	1
C	7	6	5	4	3	2
D	8	7	6	5	4	3
E	9	8	7	6	5	4
F	10	9	8	7	6	5



manhattan((x1, y1), (x2, y2)) = |x1 — x2| + |y1 — y2|

7. Supondo um labirinto como o apresentado na Figura 1, onde todos os passos têm peso 1 e as paredes (pontos escuros) tem valor infinitamente altos, qual o percurso o algoritmo de Path Finding A* escolherá para o personagem localizado no ponto F1 e com o objetivo localizado no ponto A6? Apresente todos os passos de execução do algoritmo A* e o caminho escolhido no final. Considere os valores de heurística apresentados na Figura 2.

Figura 1



	1	2	3	4	5	6
A	1	1	1	1	1	1
B	■	■	1	■	1	1
C	1	1	1	■	1	1
D	1	1	1	1	1	1
E	1	1	1	■	■	■
F	1	1	1	1	1	1

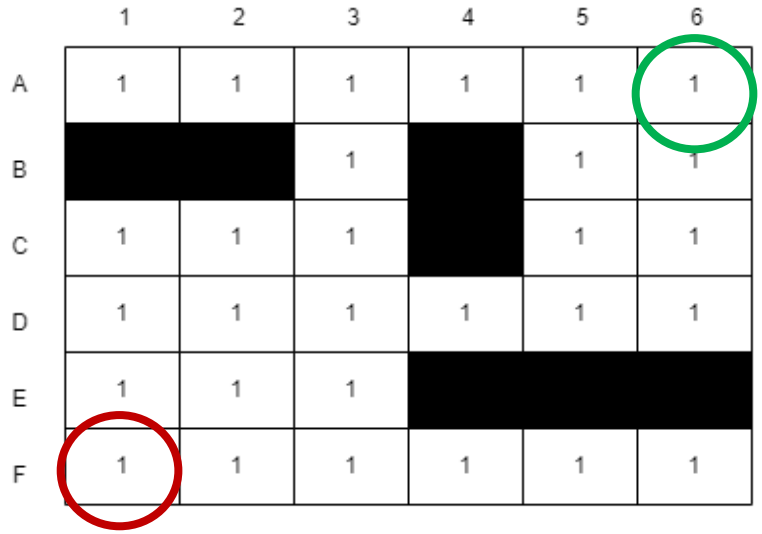
Figura 2

	1	2	3	4	5	6
A	5	4	3	2	1	0
B	6	5	4	3	2	1
C	7	6	5	4	3	2
D	8	7	6	5	4	3
E	9	8	7	6	5	4
F	10	9	8	7	6	5

manhattan((x1, y1), (x2, y2)) = |x1 — x2| + |y1 — y2|

7. Supondo um labirinto como o apresentado na Figura 1, onde todos os passos têm peso 1 e as paredes (pontos escuros) tem valor infinitamente altos, qual o percurso o algoritmo de Path Finding A* escolherá para o personagem localizado no ponto F1 e com o objetivo localizado no ponto A6? Apresente todos os passos de execução do algoritmo A* e o caminho escolhido no final. Considere os valores de heurística apresentados na Figura 2.

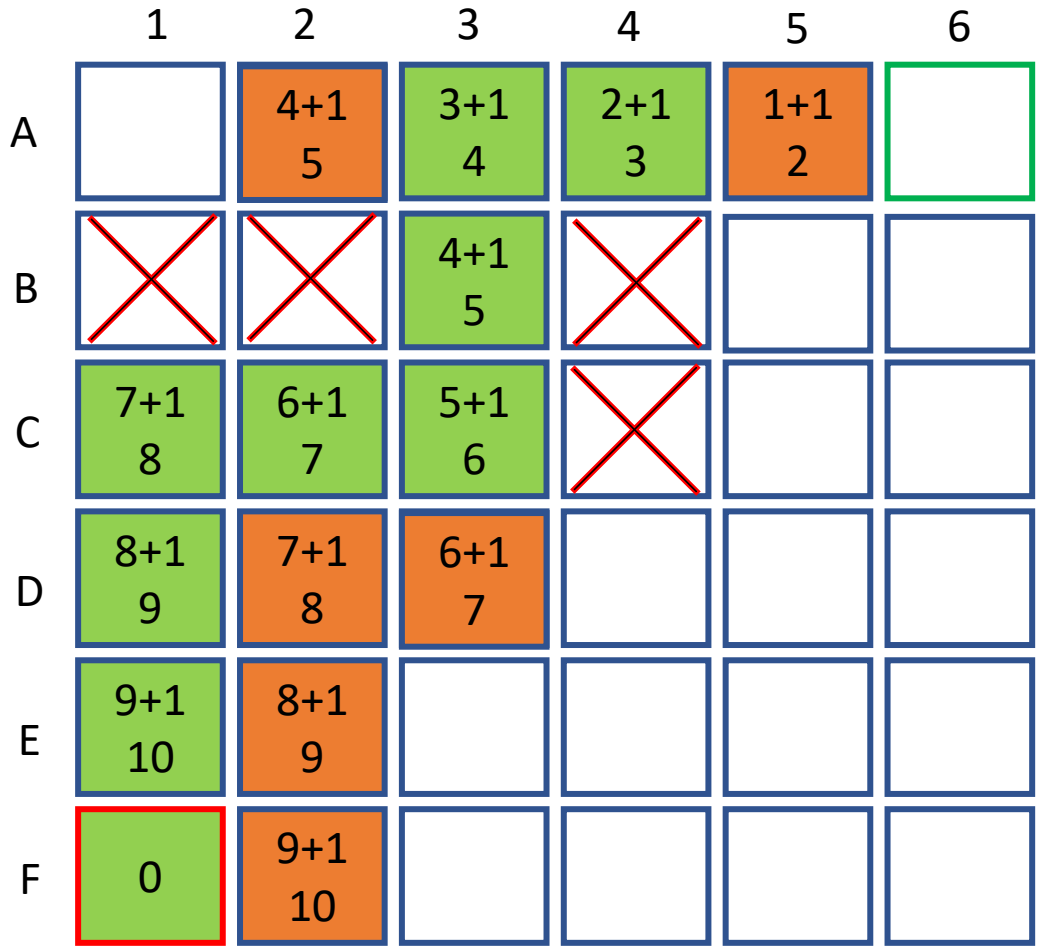
Figura 1



	1	2	3	4	5	6
A	1	1	1	1	1	1
B	■	■	1	■	1	1
C	1	1	1	■	1	1
D	1	1	1	1	1	1
E	1	1	1	■	■	■
F	1	1	1	1	1	1

Figura 2

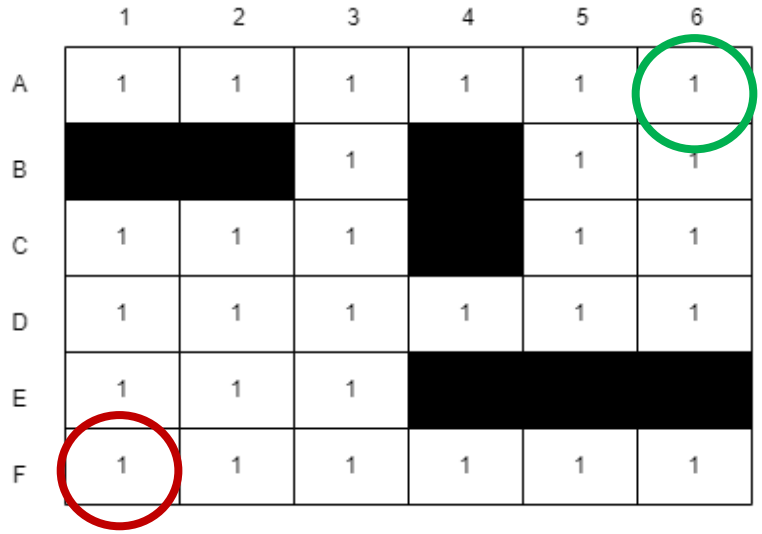
	1	2	3	4	5	6
A	5	4	3	2	1	0
B	6	5	4	3	2	1
C	7	6	5	4	3	2
D	8	7	6	5	4	3
E	9	8	7	6	5	4
F	10	9	8	7	6	5



manhattan((x1, y1), (x2, y2)) = |x1 — x2| + |y1 — y2|

7. Supondo um labirinto como o apresentado na Figura 1, onde todos os passos têm peso 1 e as paredes (pontos escuros) tem valor infinitamente altos, qual o percurso o algoritmo de Path Finding A* escolherá para o personagem localizado no ponto F1 e com o objetivo localizado no ponto A6? Apresente todos os passos de execução do algoritmo A* e o caminho escolhido no final. Considere os valores de heurística apresentados na Figura 2.

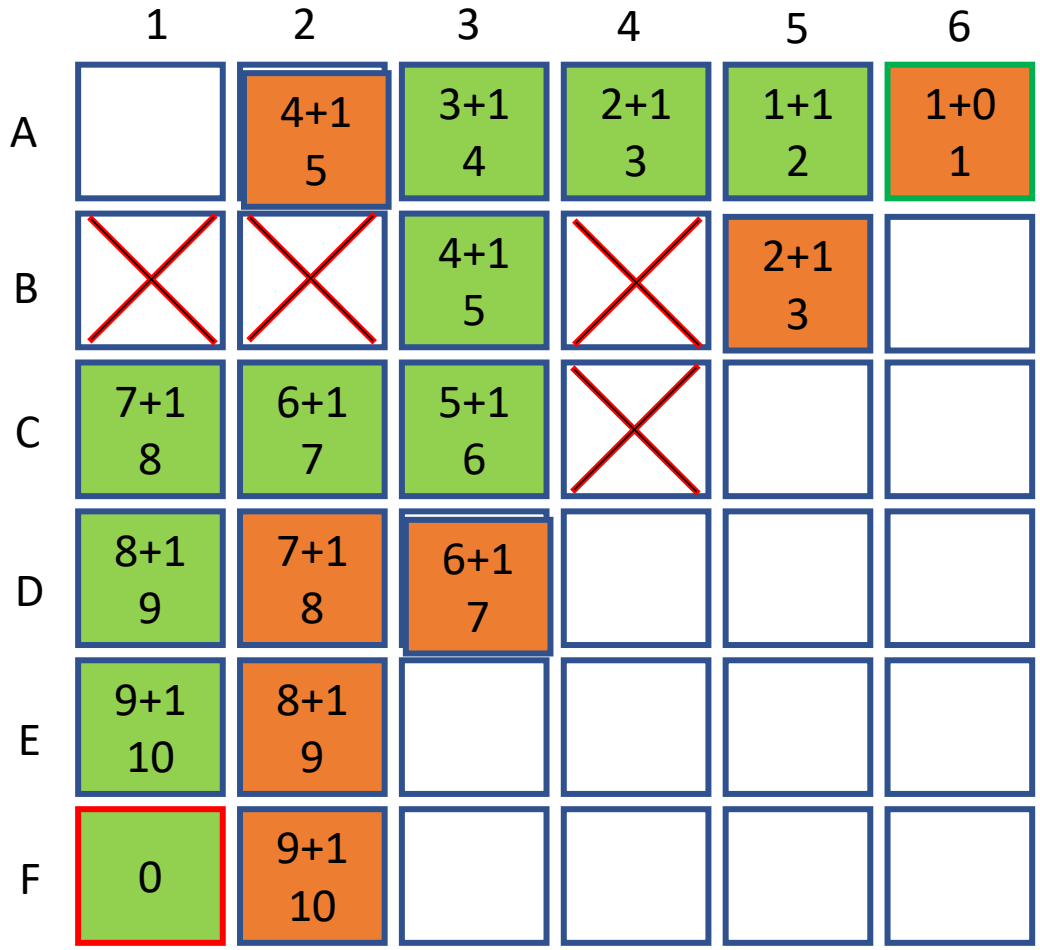
Figura 1



	1	2	3	4	5	6
A	1	1	1	1	1	1
B	■	■	1	■	1	1
C	1	1	1	■	1	1
D	1	1	1	1	1	1
E	1	1	1	■	■	■
F	1	1	1	1	1	1

Figura 2

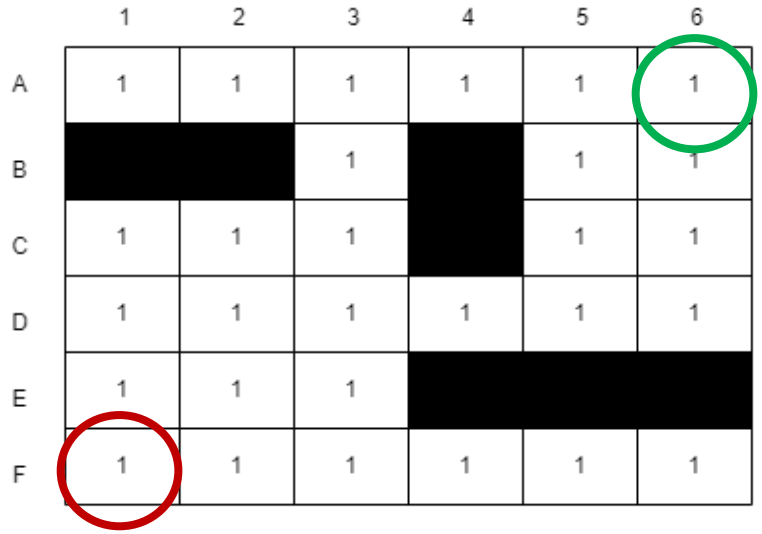
	1	2	3	4	5	6
A	5	4	3	2	1	0
B	6	5	4	3	2	1
C	7	6	5	4	3	2
D	8	7	6	5	4	3
E	9	8	7	6	5	4
F	10	9	8	7	6	5



manhattan((x1, y1), (x2, y2)) = |x1 — x2| + |y1 — y2|

7. Supondo um labirinto como o apresentado na Figura 1, onde todos os passos têm peso 1 e as paredes (pontos escuros) tem valor infinitamente altos, qual o percurso o algoritmo de Path Finding A* escolherá para o personagem localizado no ponto F1 e com o objetivo localizado no ponto A6? Apresente todos os passos de execução do algoritmo A* e o caminho escolhido no final. Considere os valores de heurística apresentados na Figura 2.

Figura 1



	1	2	3	4	5	6
A	1	1	1	1	1	1
B	■	■	1	■	1	1
C	1	1	1	■	1	1
D	1	1	1	1	1	1
E	1	1	1	■	■	■
F	1	1	1	1	1	1

Figura 2

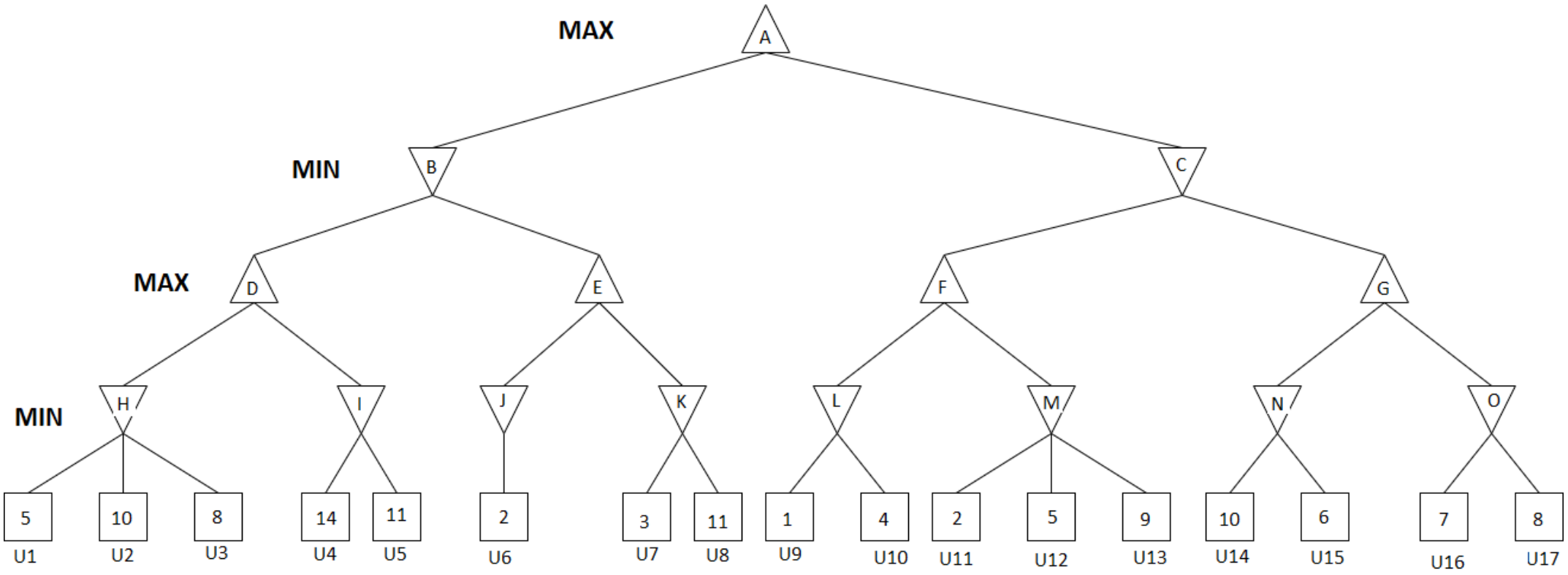
	1	2	3	4	5	6
A	5	4	3	2	1	0
B	6	5	4	3	2	1
C	7	6	5	4	3	2
D	8	7	6	5	4	3
E	9	8	7	6	5	4
F	10	9	8	7	6	5

	1	2	3	4	5	6
A		4+1 5	3+1 4	2+1 3	1+1 2	1+0 1
B	✕	✕	4+1 5	✕	2+1 3	
C	7+1 8	6+1 7	5+1 6	✕		
D	8+1 9	7+1 8	6+1 7			
E	9+1 10	8+1 9				
F	0	9+1 10				

manhattan((x1, y1), (x2, y2)) = |x1 — x2| + |y1 — y2|

8. Supondo o diagrama a seguir:

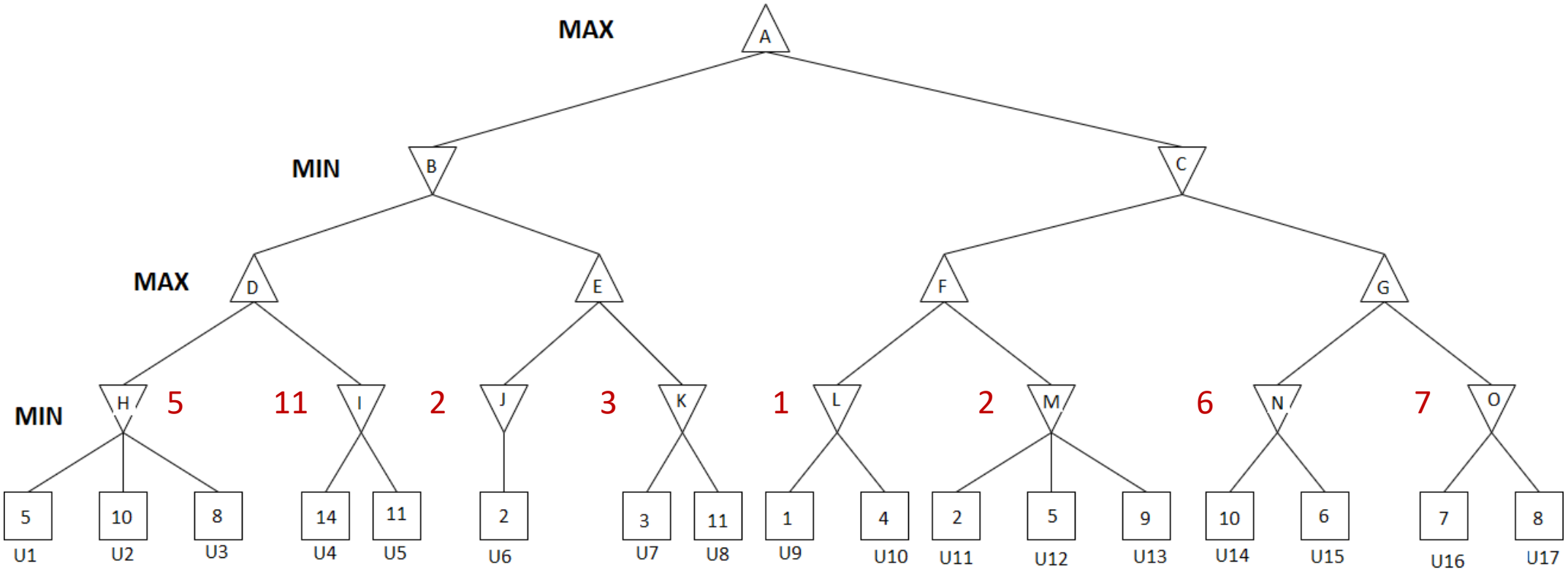
a) Execute o Algoritmo MINIMAX sem Poda alfa-beta
b) Execute o Algoritmo MINIMAX com Poda alfa-beta
c) Identifique qual a principal diferença entre o resultado dos algoritmos MINIMAX com e sem Poda alfa-beta.



8. Supondo o diagrama a seguir:

a) Execute o Algoritmo MINIMAX sem Poda alfa-beta
b) Execute o Algoritmo MINIMAX com Poda alfa-beta
c) Identifique qual a principal diferença entre o resultado dos algoritmos MINIMAX com e sem Poda alfa-beta.

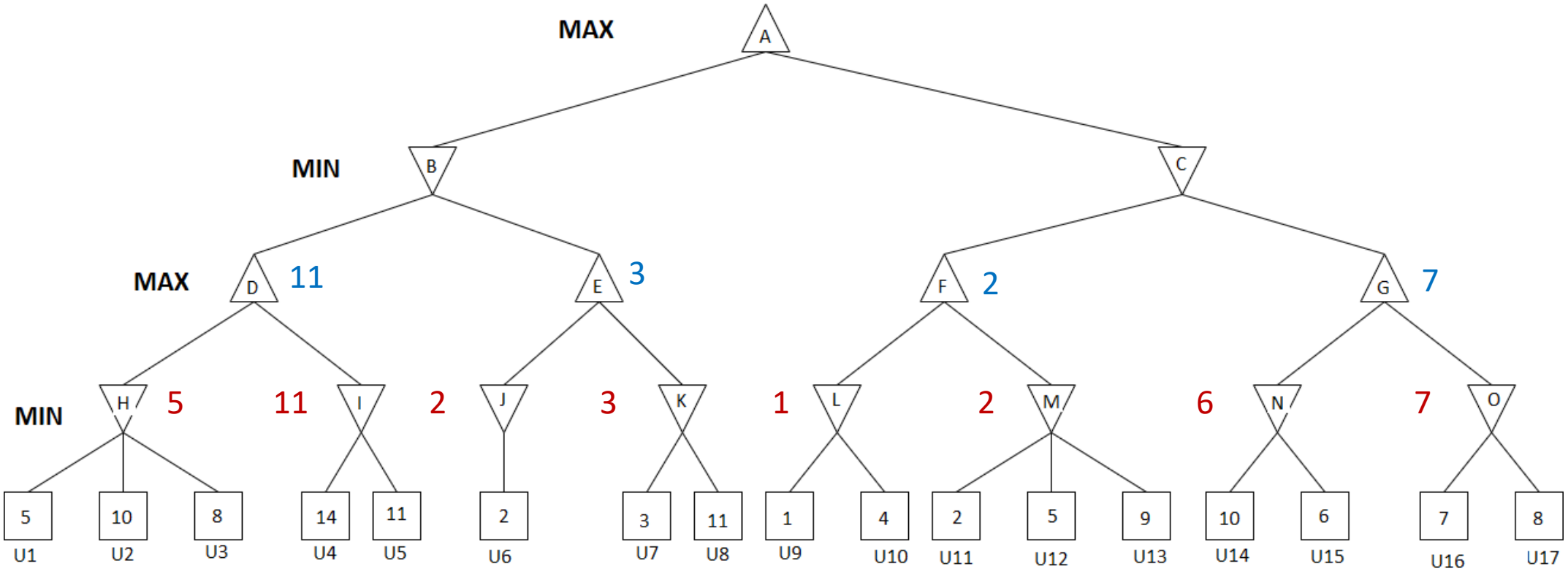
Sem poda



8. Supondo o diagrama a seguir:

a) Execute o Algoritmo MINIMAX sem Poda alfa-beta
b) Execute o Algoritmo MINIMAX com Poda alfa-beta
c) Identifique qual a principal diferença entre o resultado dos algoritmos MINIMAX com e sem Poda alfa-beta.

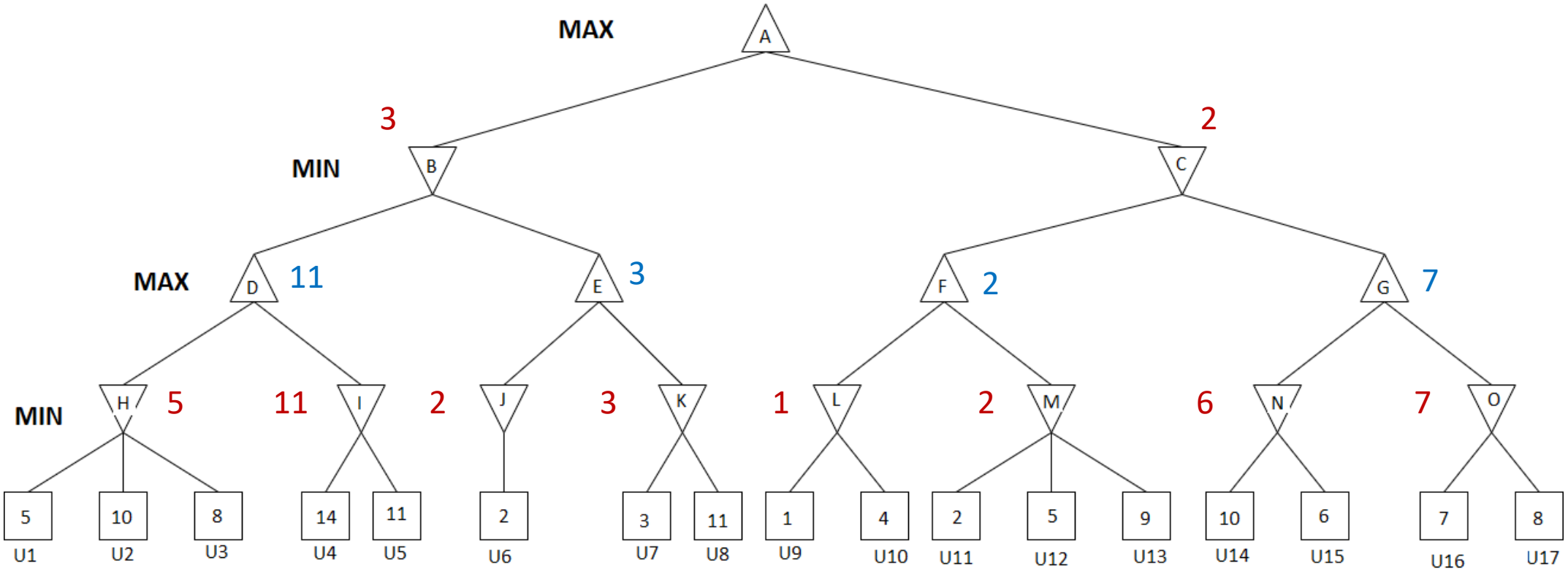
Sem poda



8. Supondo o diagrama a seguir:

a) Execute o Algoritmo MINIMAX sem Poda alfa-beta
b) Execute o Algoritmo MINIMAX com Poda alfa-beta
c) Identifique qual a principal diferença entre o resultado dos algoritmos MINIMAX com e sem Poda alfa-beta.

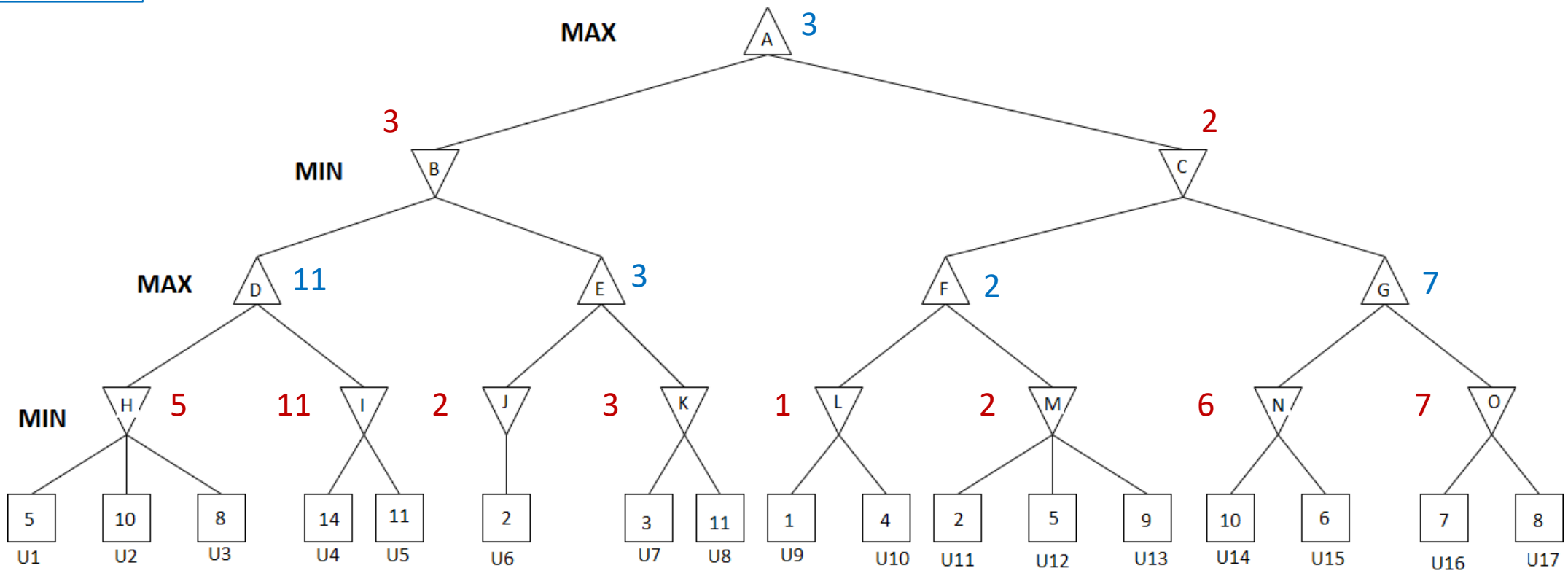
Sem poda



8. Supondo o diagrama a seguir:

a) Execute o Algoritmo MINIMAX sem Poda alfa-beta
b) Execute o Algoritmo MINIMAX com Poda alfa-beta
c) Identifique qual a principal diferença entre o resultado dos algoritmos MINIMAX com e sem Poda alfa-beta.

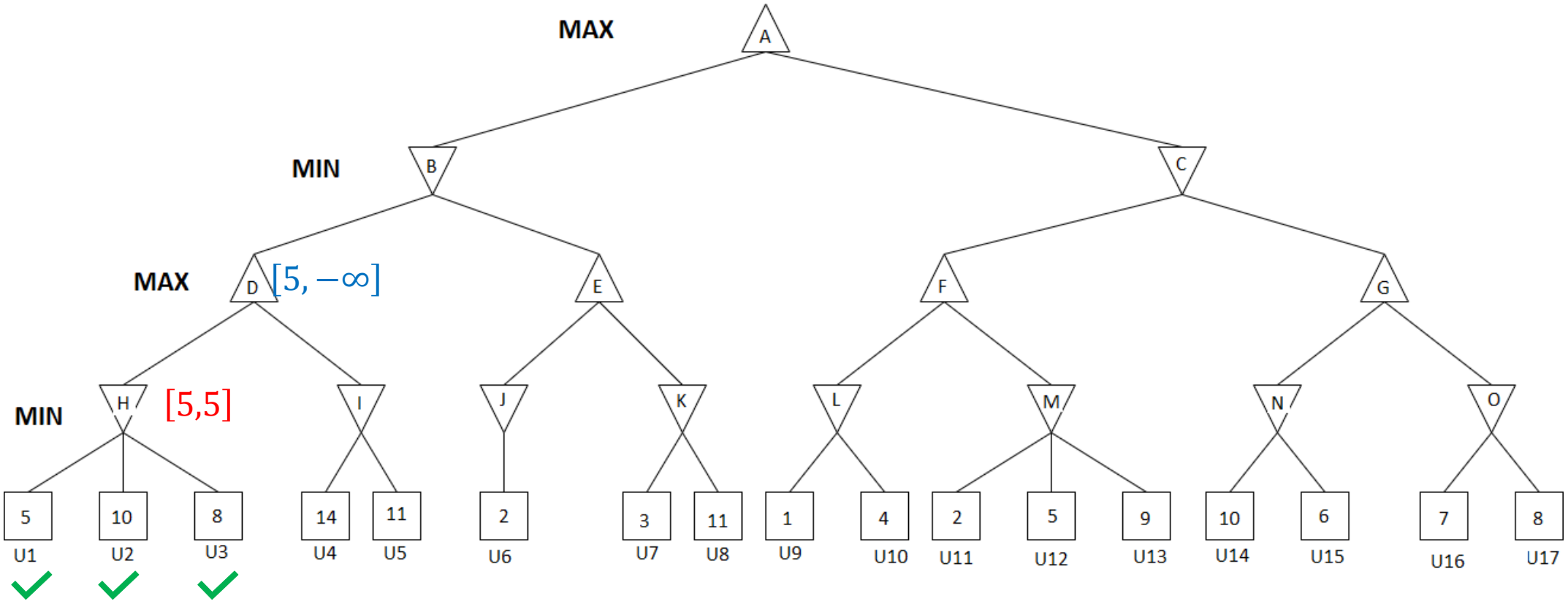
Sem poda



8. Supondo o diagrama a seguir:

a) Execute o Algoritmo MINIMAX sem Poda alfa-beta
b) Execute o Algoritmo MINIMAX com Poda alfa-beta
c) Identifique qual a principal diferença entre o resultado dos algoritmos MINIMAX com e sem Poda alfa-beta.

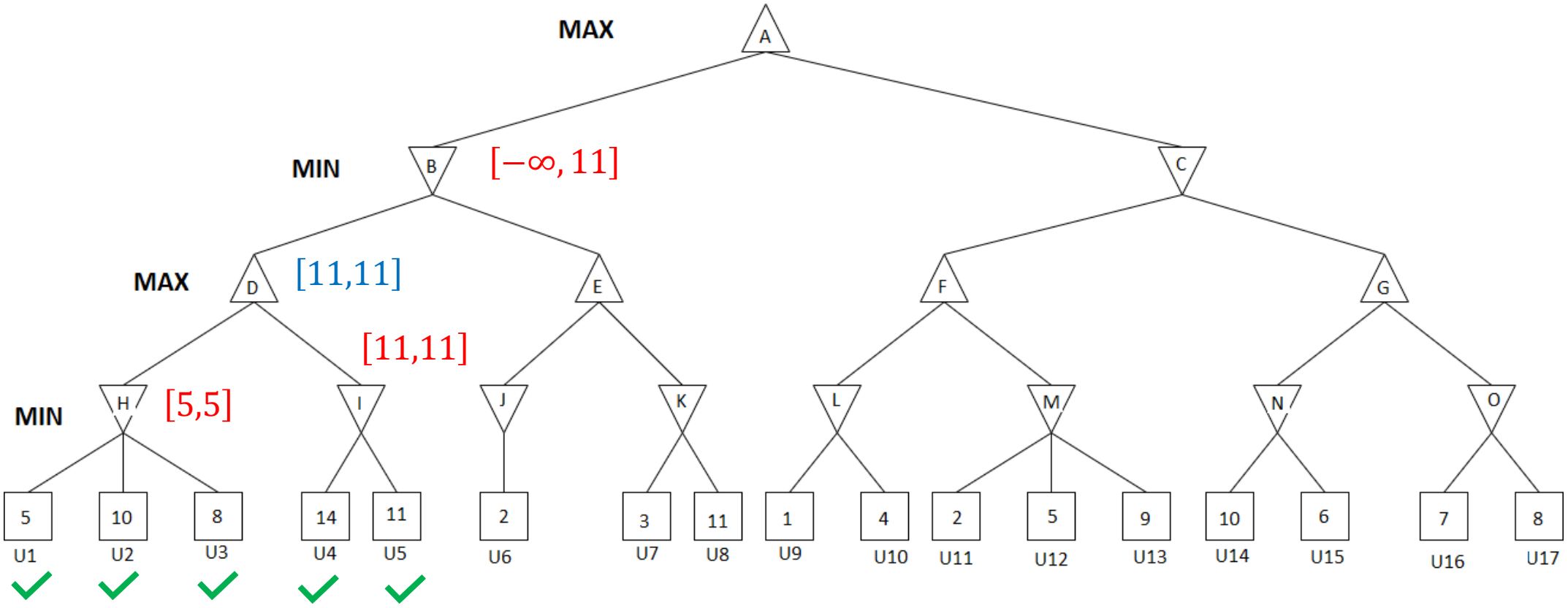
Com poda



8. Supondo o diagrama a seguir:

a) Execute o Algoritmo MINIMAX sem Poda alfa-beta
b) Execute o Algoritmo MINIMAX com Poda alfa-beta
c) Identifique qual a principal diferença entre o resultado dos algoritmos MINIMAX com e sem Poda alfa-beta.

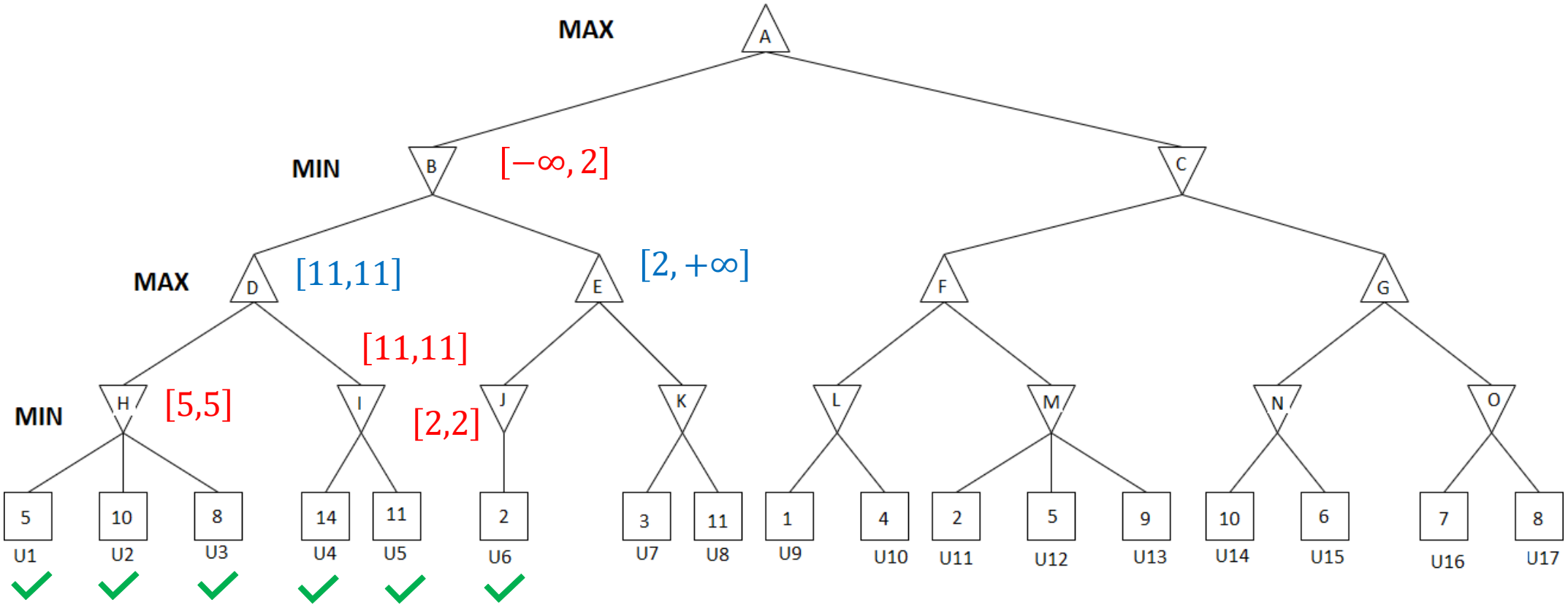
Com poda



8. Supondo o diagrama a seguir:

a) Execute o Algoritmo MINIMAX sem Poda alfa-beta
b) Execute o Algoritmo MINIMAX com Poda alfa-beta
c) Identifique qual a principal diferença entre o resultado dos algoritmos MINIMAX com e sem Poda alfa-beta.

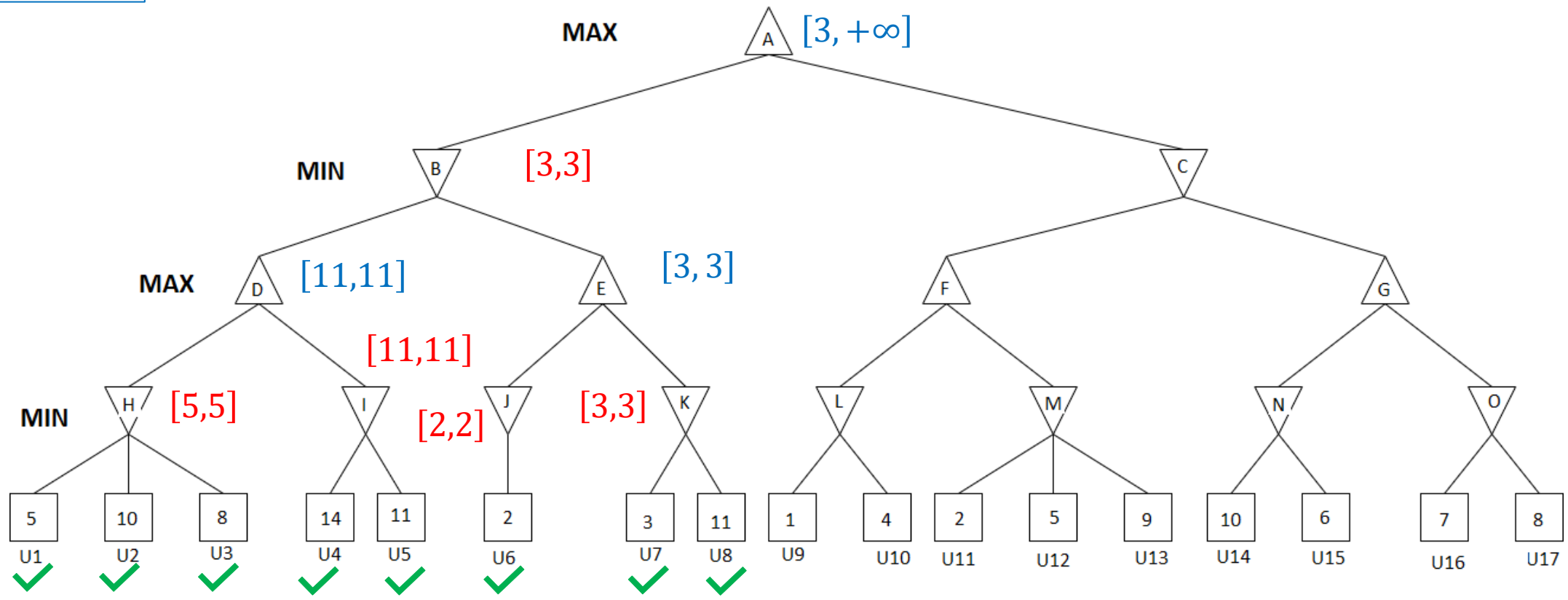
Com poda



8. Supondo o diagrama a seguir:

a) Execute o Algoritmo MINIMAX sem Poda alfa-beta
b) Execute o Algoritmo MINIMAX com Poda alfa-beta
c) Identifique qual a principal diferença entre o resultado dos algoritmos MINIMAX com e sem Poda alfa-beta.

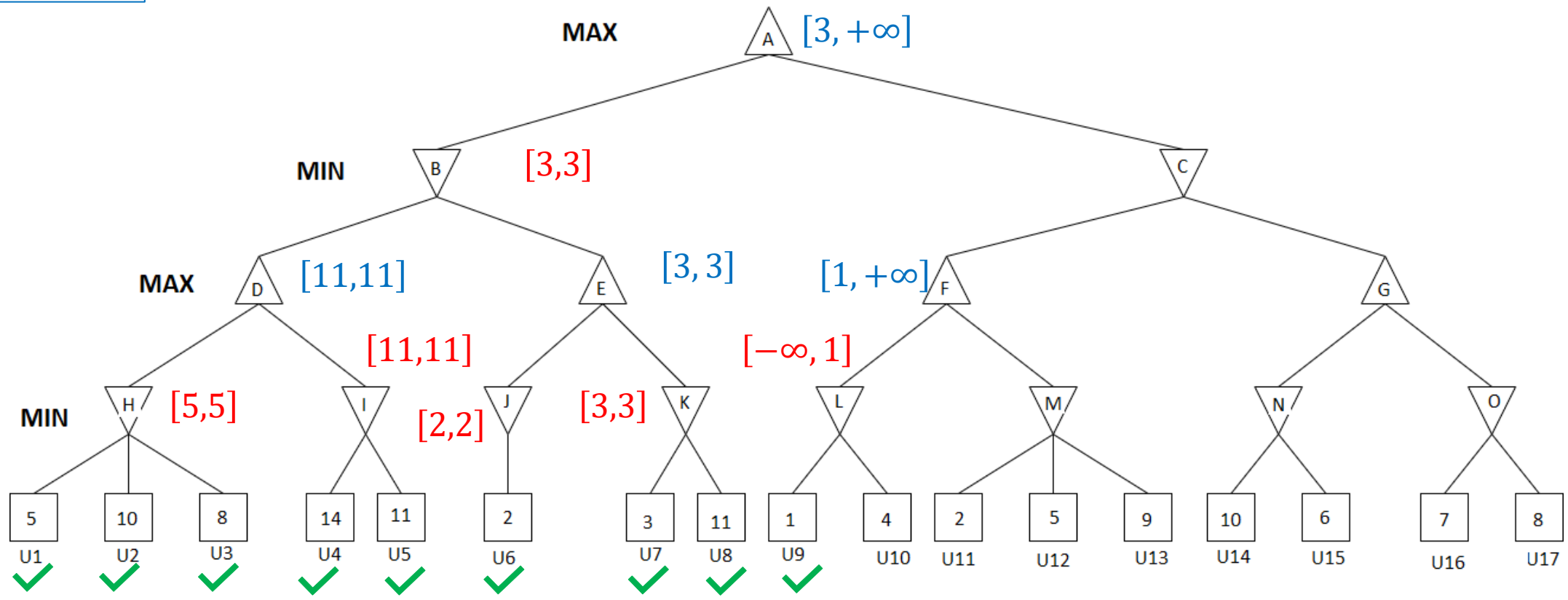
Com poda



8. Supondo o diagrama a seguir:

a) Execute o Algoritmo MINIMAX sem Poda alfa-beta
b) Execute o Algoritmo MINIMAX com Poda alfa-beta
c) Identifique qual a principal diferença entre o resultado dos algoritmos MINIMAX com e sem Poda alfa-beta.

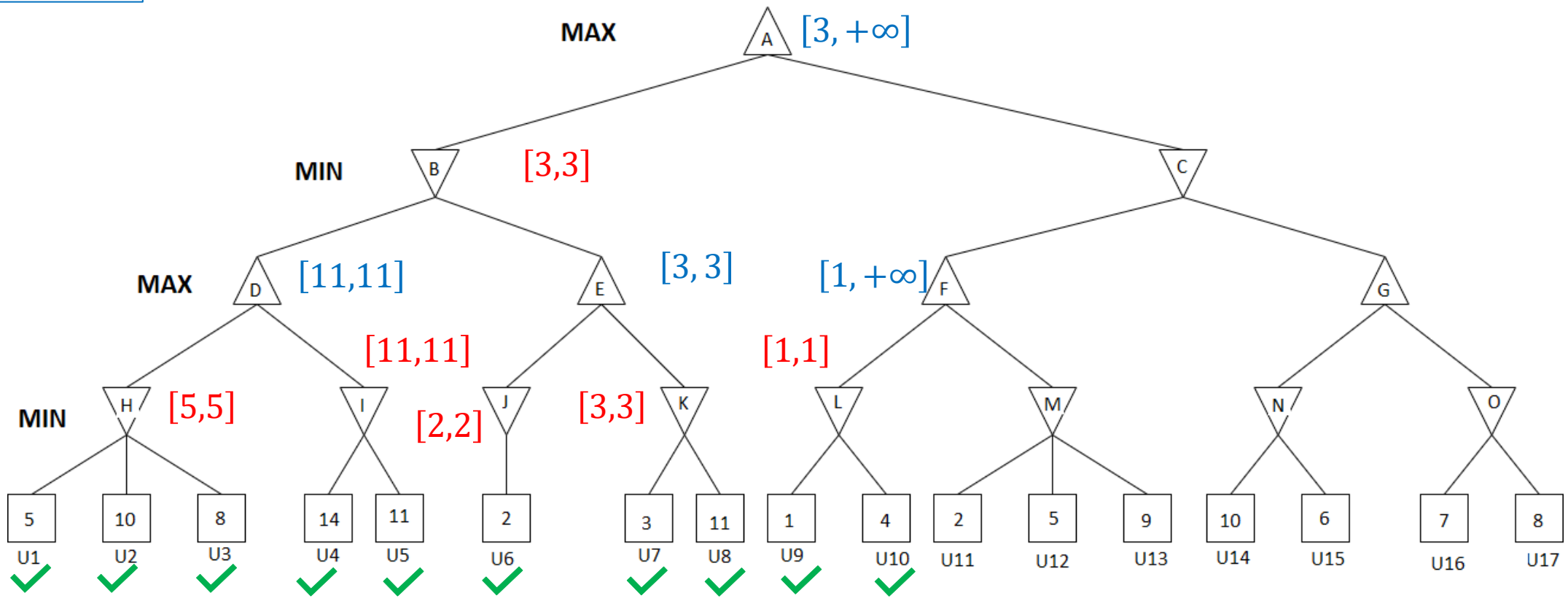
Com poda



8. Supondo o diagrama a seguir:

a) Execute o Algoritmo MINIMAX sem Poda alfa-beta
b) Execute o Algoritmo MINIMAX com Poda alfa-beta
c) Identifique qual a principal diferença entre o resultado dos algoritmos MINIMAX com e sem Poda alfa-beta.

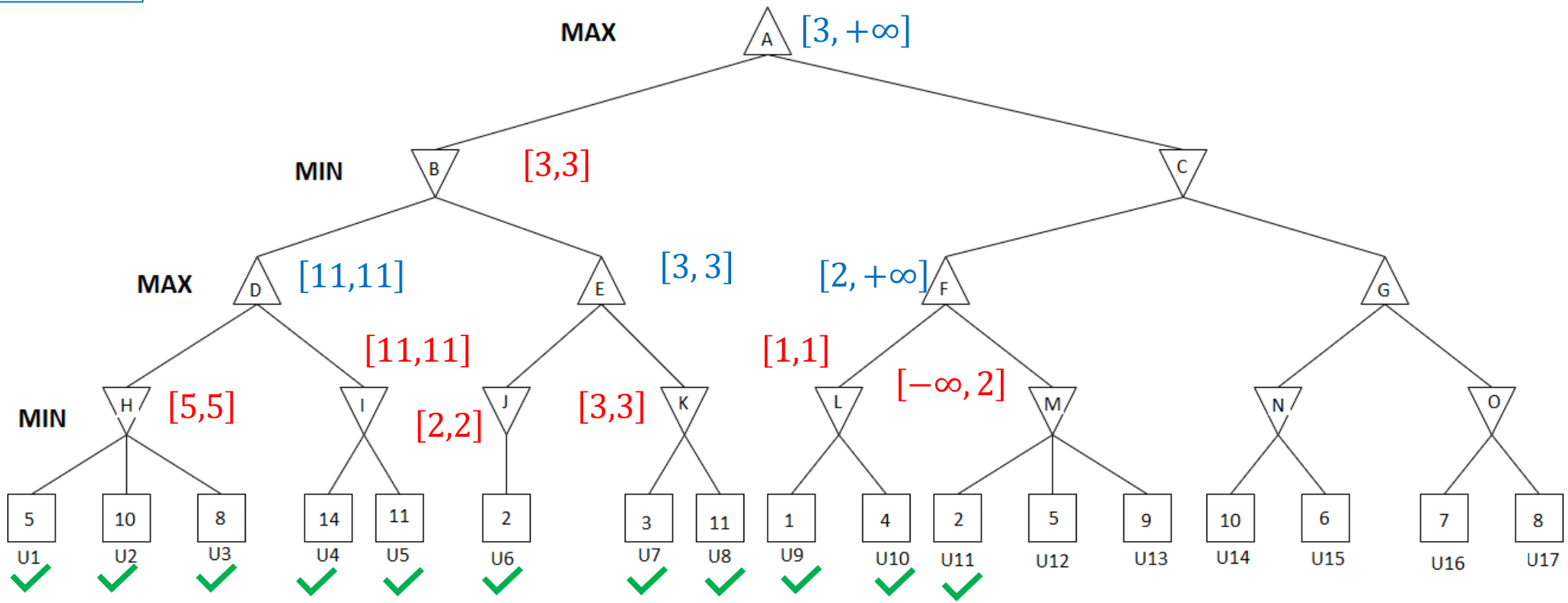
Com poda



8. Supondo o diagrama a seguir:

a) Execute o Algoritmo MINIMAX sem Poda alfa-beta
b) Execute o Algoritmo MINIMAX com Poda alfa-beta
c) Identifique qual a principal diferença entre o resultado dos algoritmos MINIMAX com e sem Poda alfa-beta.

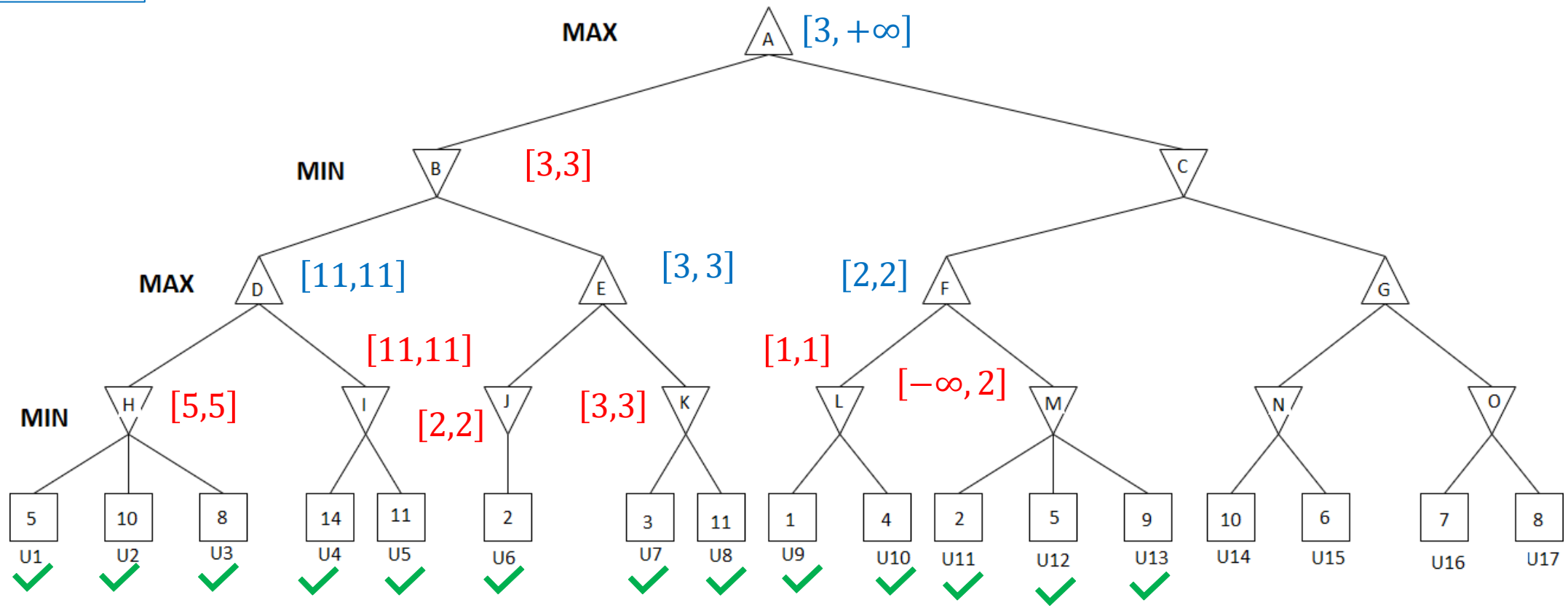
Com poda



8. Supondo o diagrama a seguir:

a) Execute o Algoritmo MINIMAX sem Poda alfa-beta
b) Execute o Algoritmo MINIMAX com Poda alfa-beta
c) Identifique qual a principal diferença entre o resultado dos algoritmos MINIMAX com e sem Poda alfa-beta.

Com poda

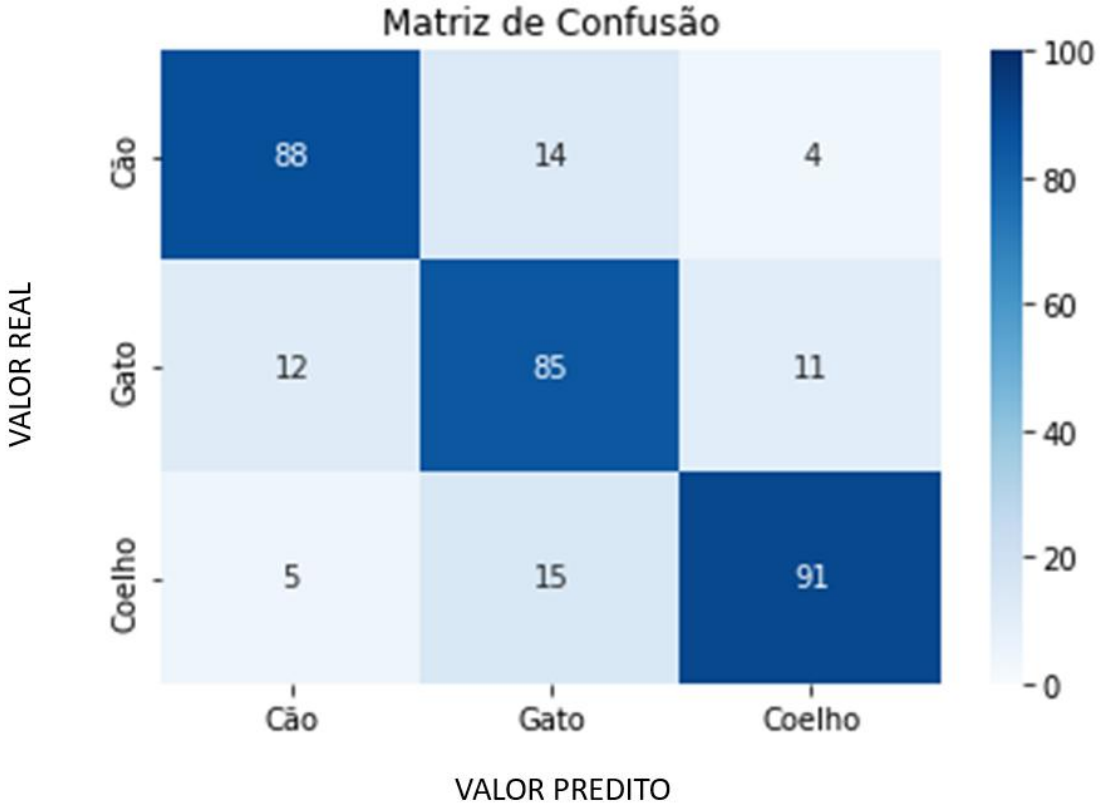


9. Com relação a Matriz de Confusão a seguir, responda as perguntas:

a) Quantas e quais são as classes?

b) Quantos são os dados de teste?

c) Para cada classe, apresente a quantidade real de dados de teste e a quantidade de predição.



a) 3 classes (Cão, Gato e Coelho)

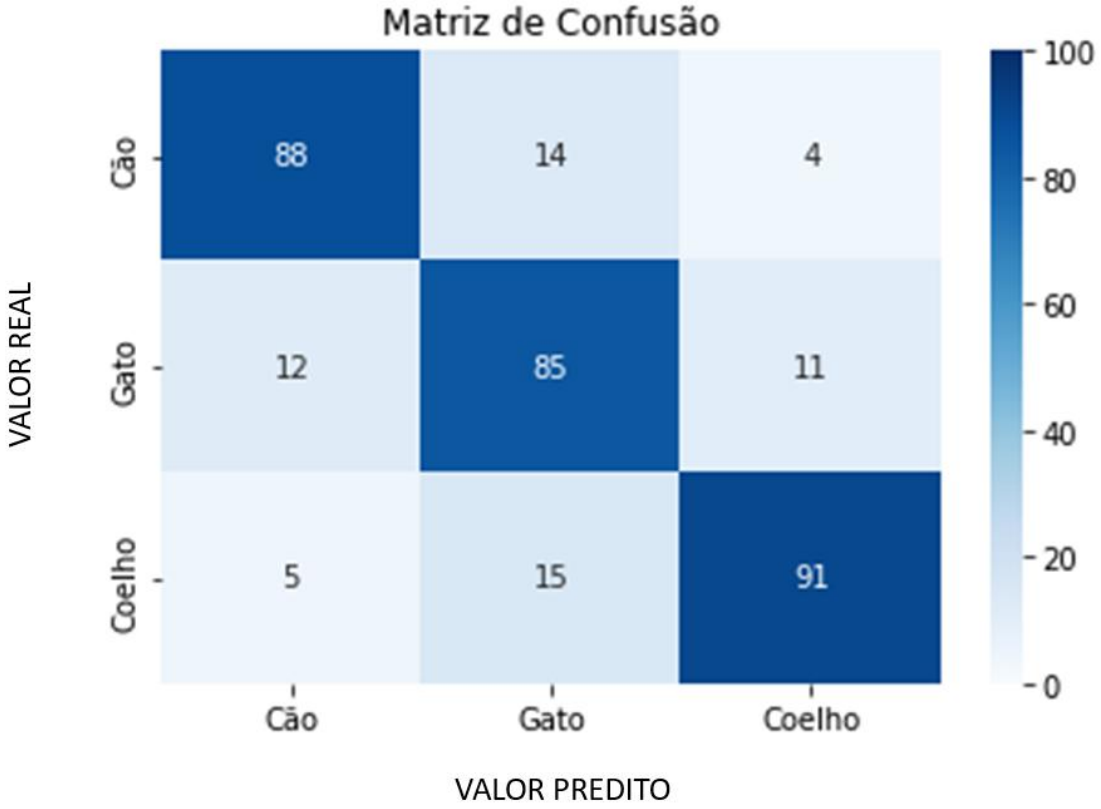
b) 88+14+4+12+85+11+5+15+91 = 325

9. Com relação a Matriz de Confusão a seguir, responda as perguntas:

a) Quantas e quais são as classes?

b) Quantos são os dados de teste?

c) Para cada classe, apresente a quantidade real de dados de teste e a quantidade de predição (correta e classes que a predição foram incorretas).



CÃO:

Real = 88+14+4 = 106

Predito = 88+12+5 = 105

Predito Corretamente = 88

Realmente era GATO = 12

Realmente era COELHO = 5

GATO:

Real = 12+85+11 = 108

Predito = 15+85+14 = 114

Predito Corretamente = 85

Realmente era CÃO = 14

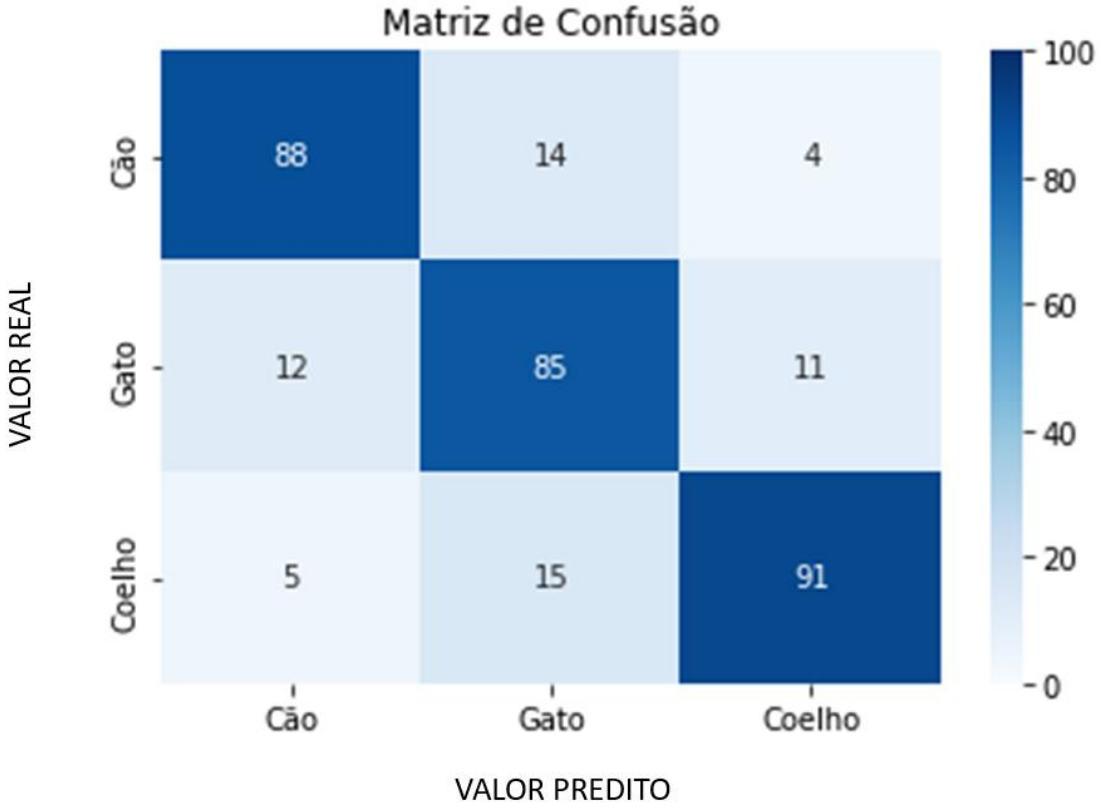
Realmente era COELHO = 15

9. Com relação a Matriz de Confusão a seguir, responda as perguntas:

a) Quantas e quais são as classes?

b) Quantos são os dados de teste?

c) Para cada classe, apresente a quantidade real de dados de teste e a quantidade de predição (correta e classes que a predição foram incorretas).



COELHO:

Real = 5+15+91 = 111

Predito = 91+11+4 = 106

Predito Corretamente = 91

Realmente era GATO = 11

Realmente era COELHO = 4

10. Construa a Matriz de Confusão para o teste de um modelo de classificação de objetos, sabendo que:

- Existem quatro classes (Classe A, Classe B, Classe C e Classe D)
- Foram utilizados 100 dados de teste (25 de cada classe)
- Para a Classe A: foram preditos 25 objetos (dos quais 25 são mesmo da Classe A)
- Para a Classe B: foram preditos 26 objetos (dos quais 25 são mesmo da Classe B e 1 da Classe D)
- Para a Classe C: foram preditos 23 objetos (dos quais 23 são mesmo da Classe C)
- Para a Classe D: foram preditos 26 objetos (dos quais 24 são mesmo da Classe D e 2 da Classe C)

10. Construa a Matriz de Confusão para o teste de um modelo de classificação de objetos, sabendo que:

- Existem quatro classes (Classe A, Classe B, Classe C e Classe D)
- Foram utilizados 100 dados de teste (25 de cada classe)
- Para a Classe A: foram preditos 25 objetos (dos quais 25 são mesmo da Classe A)
- Para a Classe B: foram preditos 26 objetos (dos quais 25 são mesmo da Classe B e 1 da Classe D)
- Para a Classe C: foram preditos 23 objetos (dos quais 23 são mesmo da Classe C)
- Para a Classe D: foram preditos 26 objetos (dos quais 24 são mesmo da Classe D e 2 da Classe C)

Predição

Real	**Classe A**	**Classe B**	**Classe C**	**Classe D**	
Classe A	25	0	0	0	25
Classe B	0	25	0	0	25
Classe C	0	0	23	2	25
Classe D	0	1	0	24	25

Daniel Nogueira

dnogueira@ipca.pt