Algoritmos e Estruturas de Dados

Marcelo Lobosco
DCC/UFJF

O Problema do Caminho Mínimo

Segunda Parte - Aula 02

Agenda

- Grafos
  - O problema do caminho mínimo
    - Algoritmo de Dijkstra
    - Algoritmo de Bellman-Moore
    - Algoritmo de Bellman-Moore-d'Esopo
    - Algoritmo de Floyd

O Problema do Caminho Mínimo

- Dado um grafo G = (N, A, P), onde P = (p_{ij} = peso da aresta e_{ij}), encontrar o caminho de menor peso entre:
  - Dois vértices s e t
  - Entre um vértice s até j, para todo j ≠ s
  - Entre cada par de vértices (i,j) de G

Algoritmo de Dijkstra

- Algoritmo clássico para determinar caminhos mínimos
- Dijkstra, 1959
- Devemos considerar os pesos não negativos ($p_{ij} \geq 0, \forall (i,j) \in A$)
- Duas fases: iniciação e iterações

Algoritmo de Dijkstra

- Iniciação

  Para todo i ≠ s faça

    dist (i) = ∞

    final (i) = false

    pred (i) = -1

  Dist (s) = 0

  Final (s) = true

  Recente = s // último nó que recebeu final = true

Algoritmo de Dijkstra

- Iterações

Enquanto final (t) = false faça

Para todo nó i adjacente ao nó recente com final(i) false faça

NewLabel = dist (Recente) + $P_{recente, i}$

Se NewLabel < dist (i)

dist (i) = NewLabel

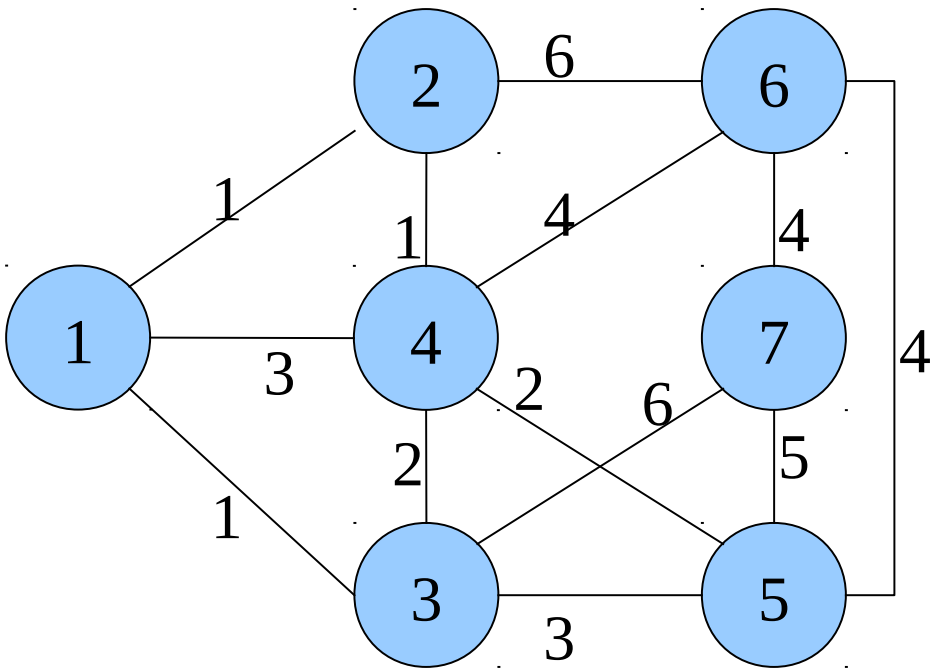
pred(i) = recente

Seja y o vértice com menor rótulo temporário, tal que dist (y) ≠ ∞

final(y) = true

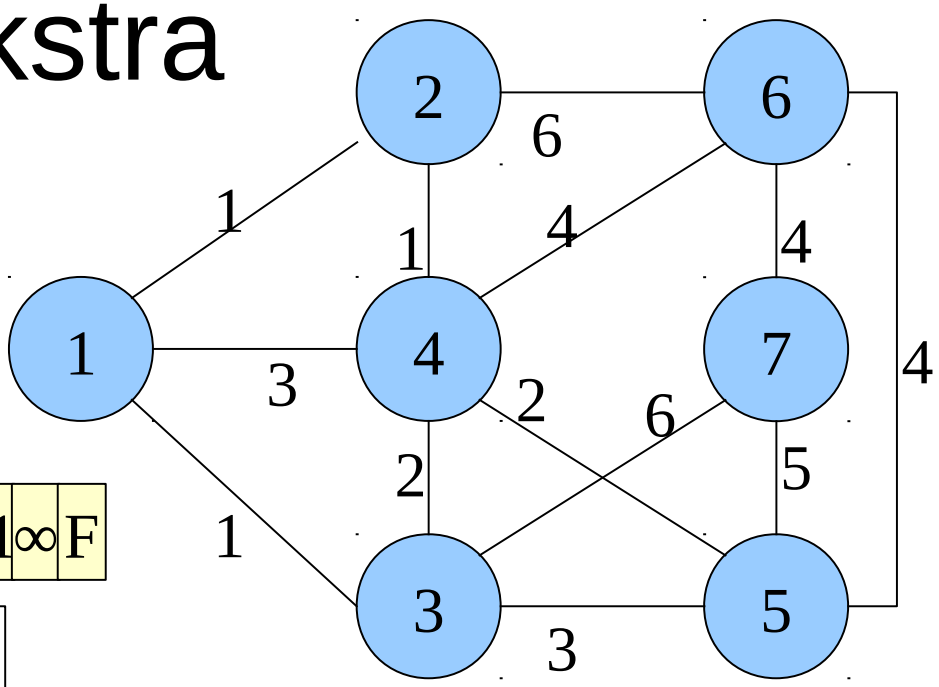
recente = y

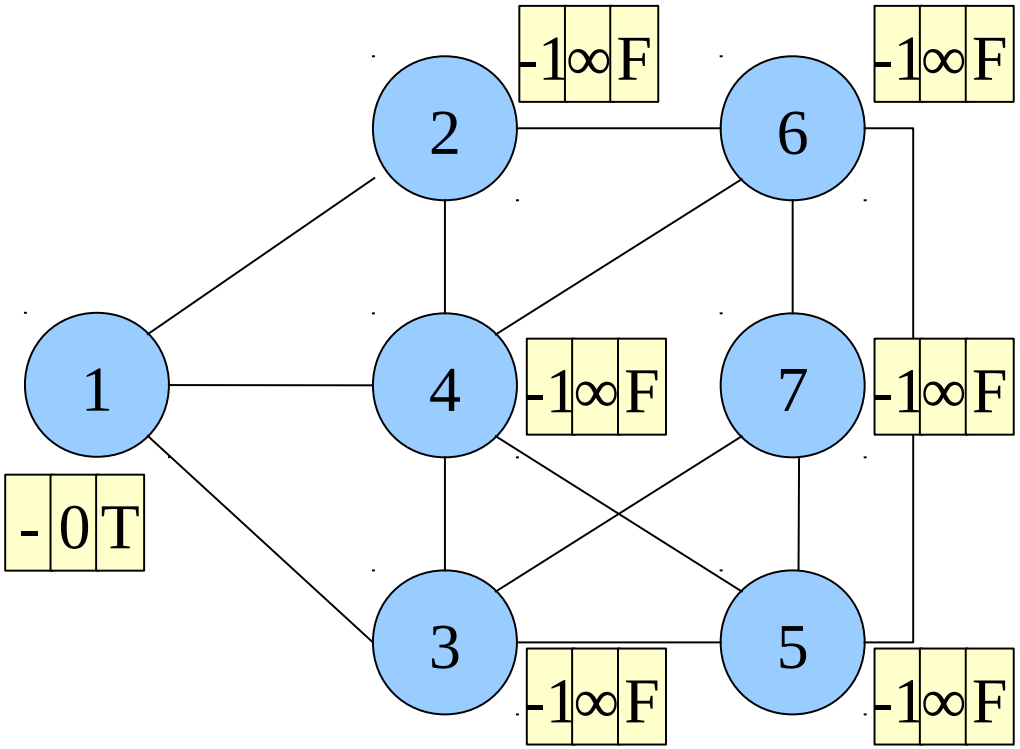
Algoritmo de Dijkstra



Algoritmo de Dijkstra

Encontrar caminho mínimo entre 1 e 7

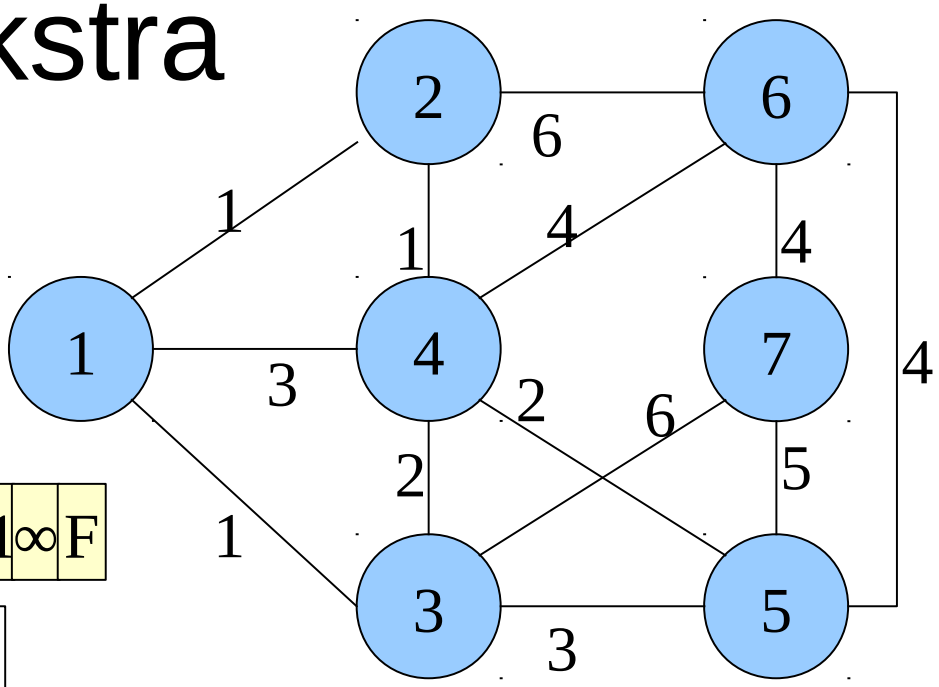


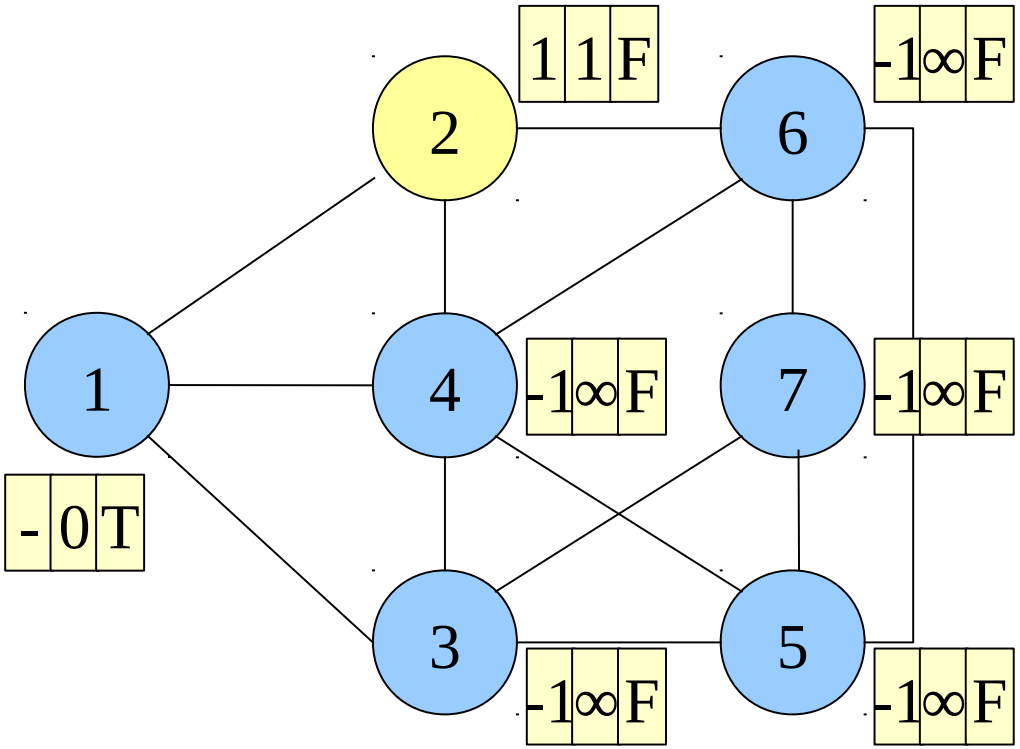


Recente = 1

Algoritmo de Dijkstra

Encontrar caminho mínimo entre 1 e 7





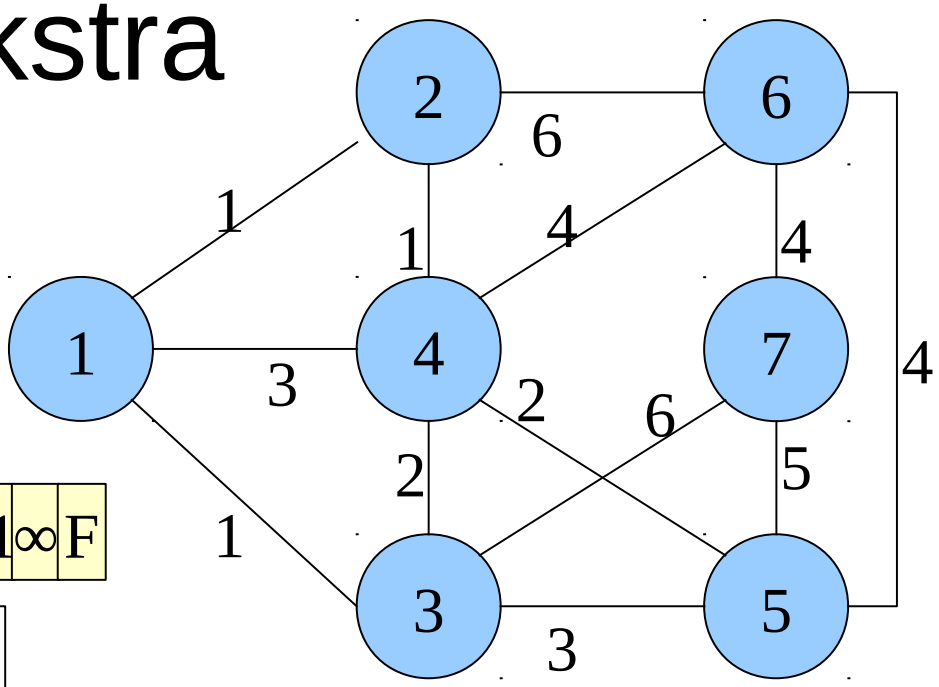
i = 2

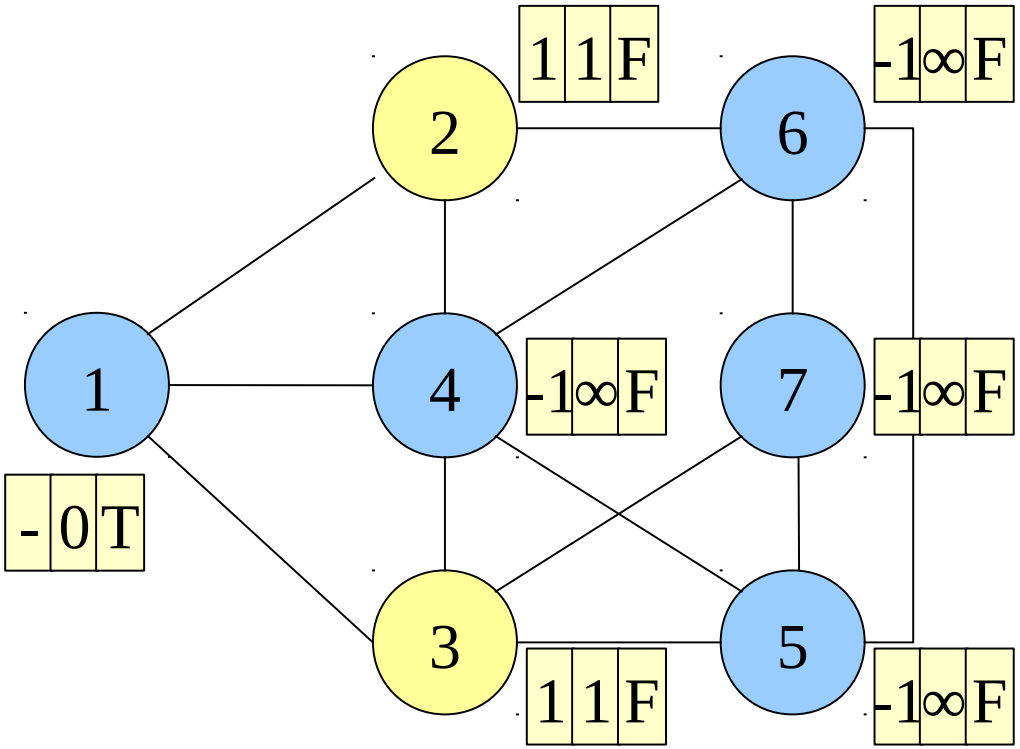
Newlabel = 0 + 1 = 1 < ∞

Recente = 1

Algoritmo de Dijkstra

Encontrar caminho mínimo entre 1 e 7

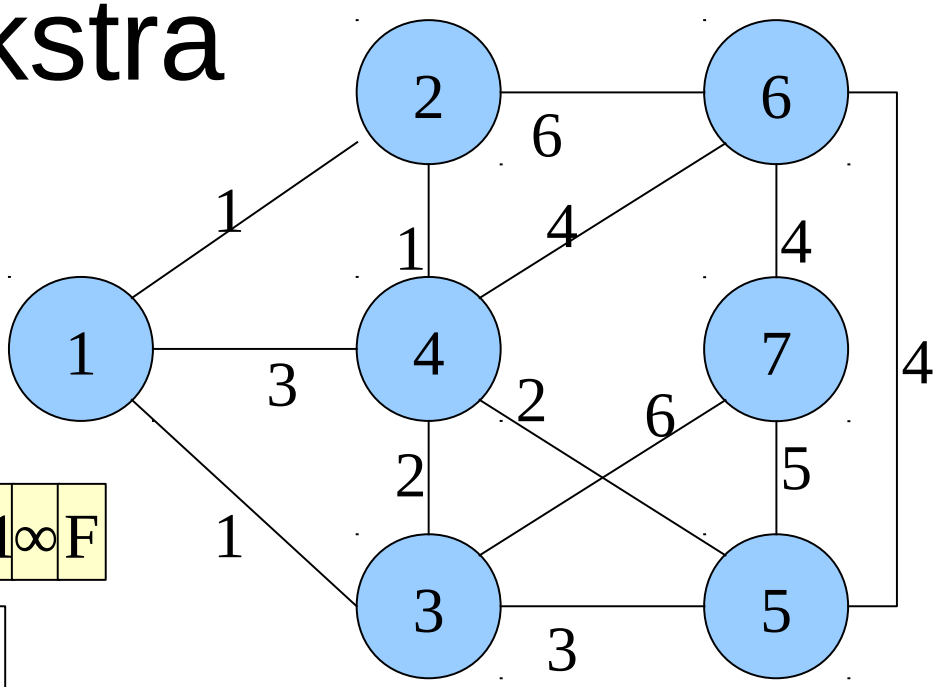


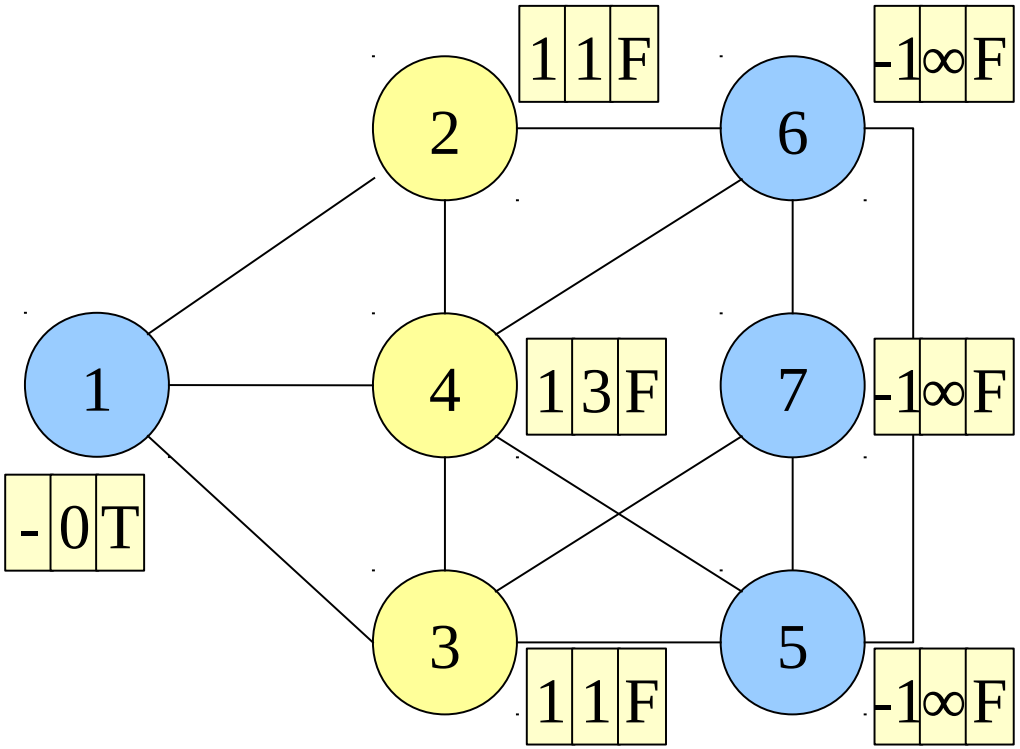


i = 3
Newlabel = 0 + 1 = 1 < ∞
Recente = 1

Algoritmo de Dijkstra

Encontrar caminho mínimo entre 1 e 7

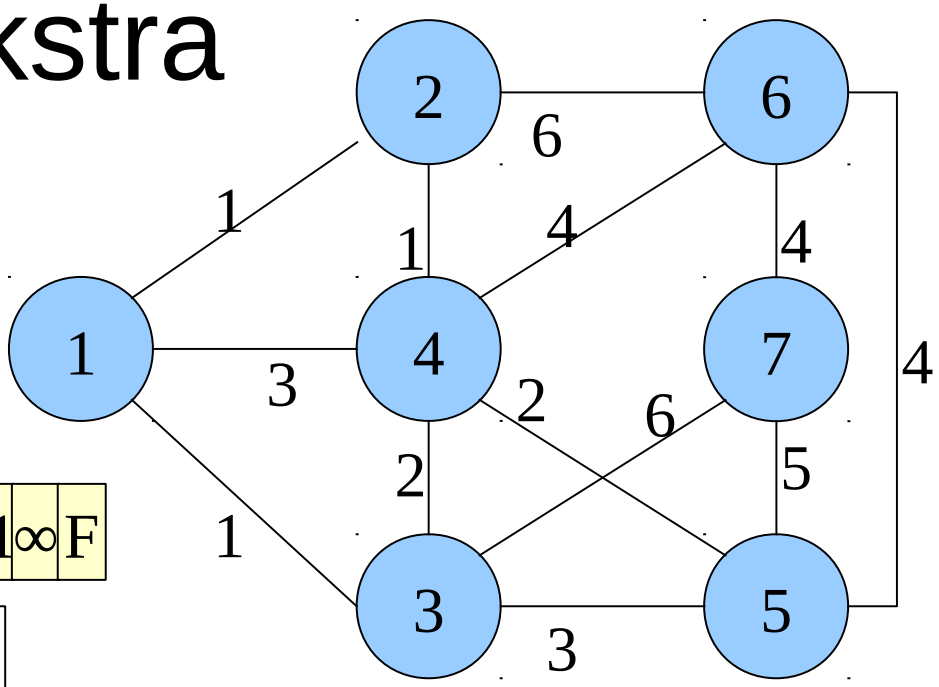


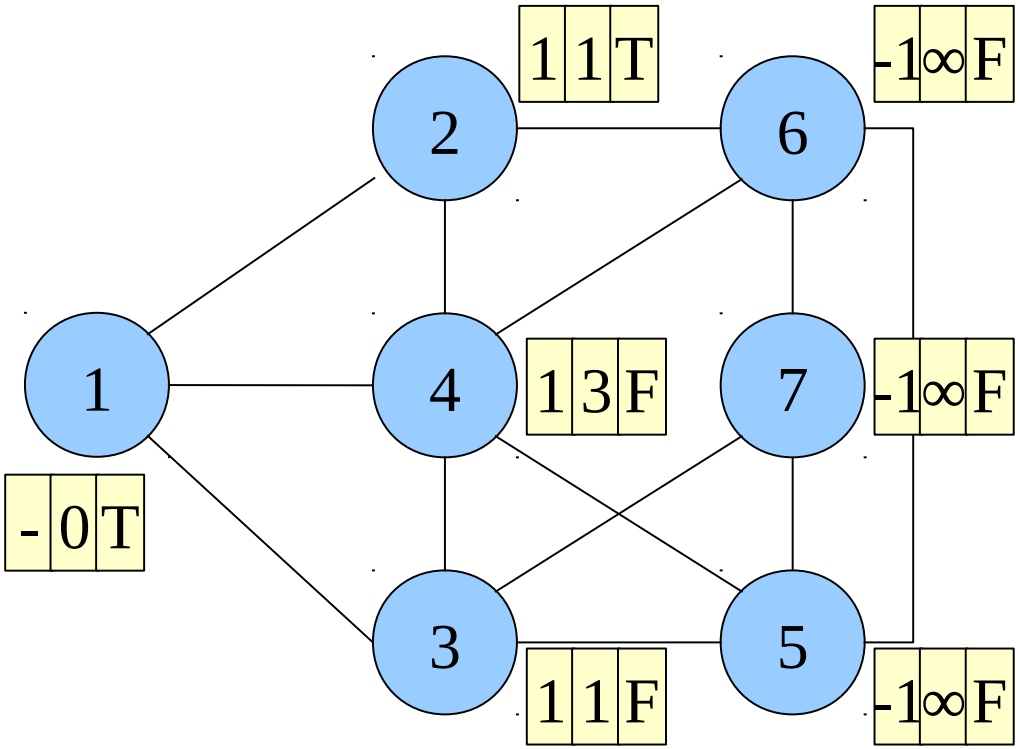


i = 4
Newlabel = 0 + 3 = 3 < ∞
Recente = 1

Algoritmo de Dijkstra

Encontrar caminho mínimo entre 1 e 7

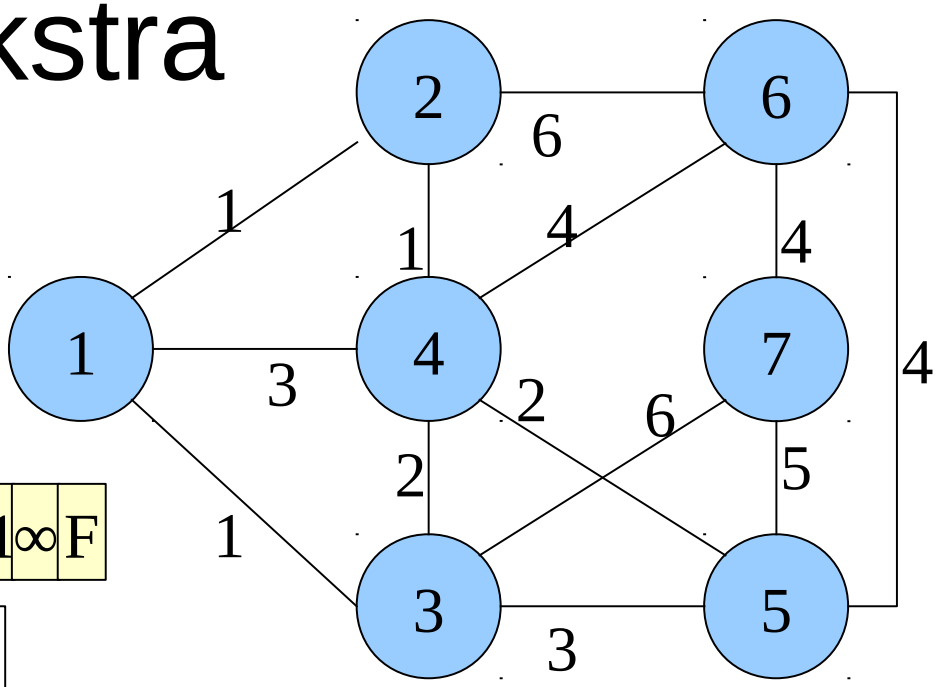


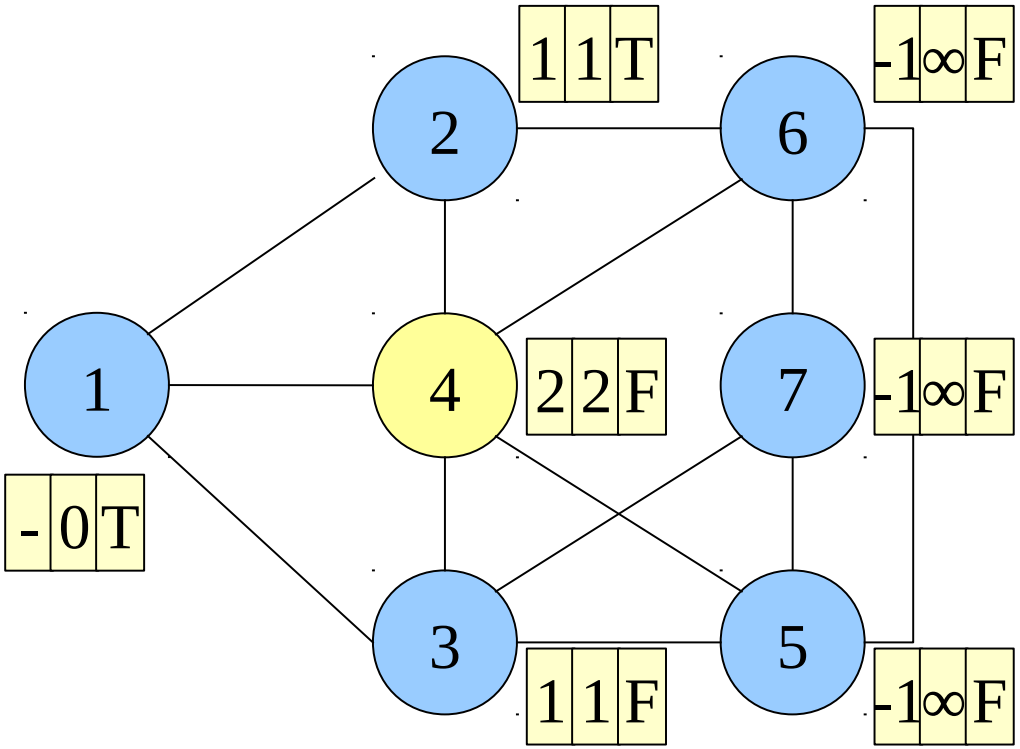


Recente = 2

Algoritmo de Dijkstra

Encontrar caminho mínimo entre 1 e 7

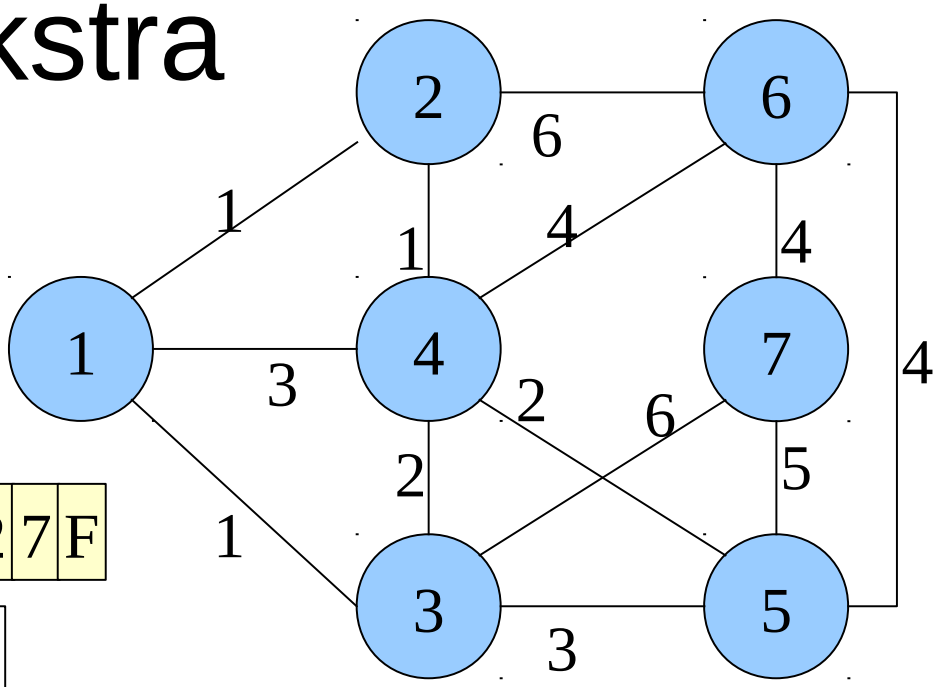


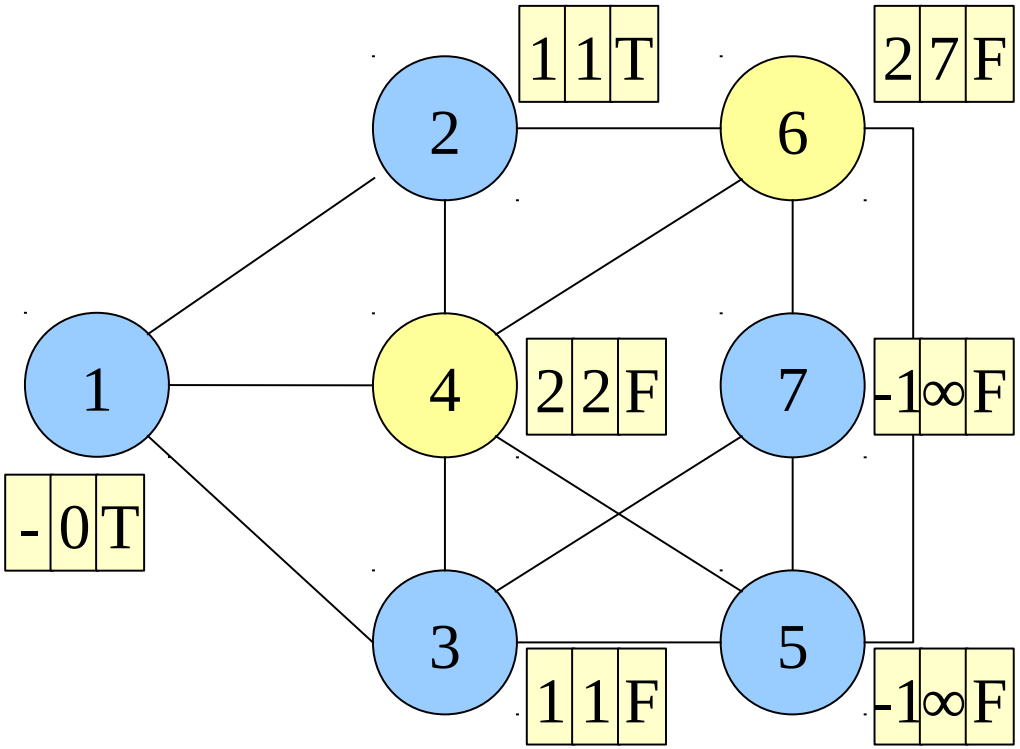


i = 4
Newlabel = 1 + 1 = 2 < 3
Recente = 2

Algoritmo de Dijkstra

Encontrar caminho mínimo entre 1 e 7

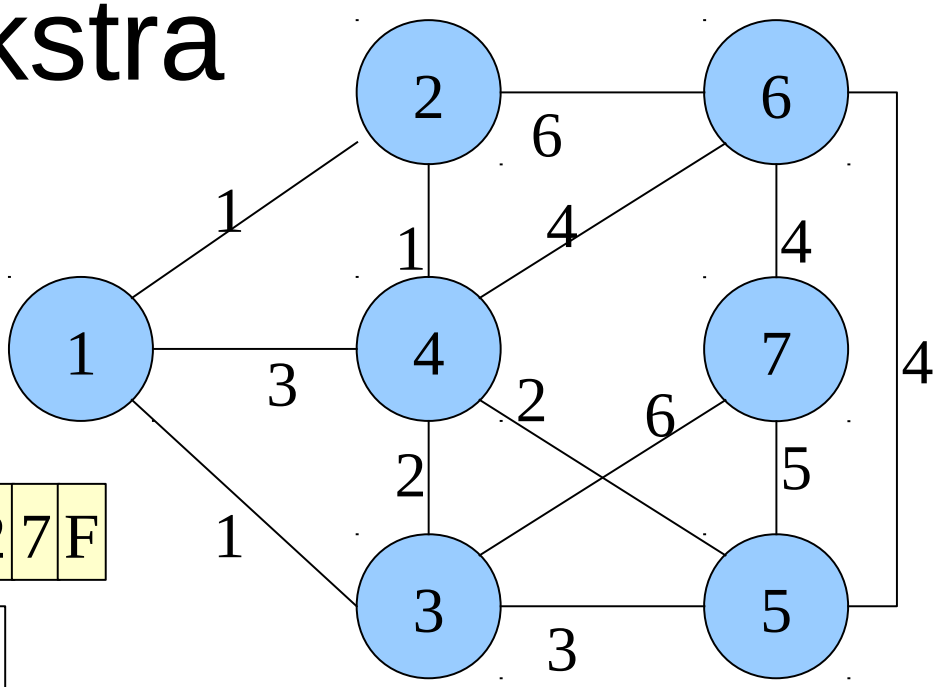


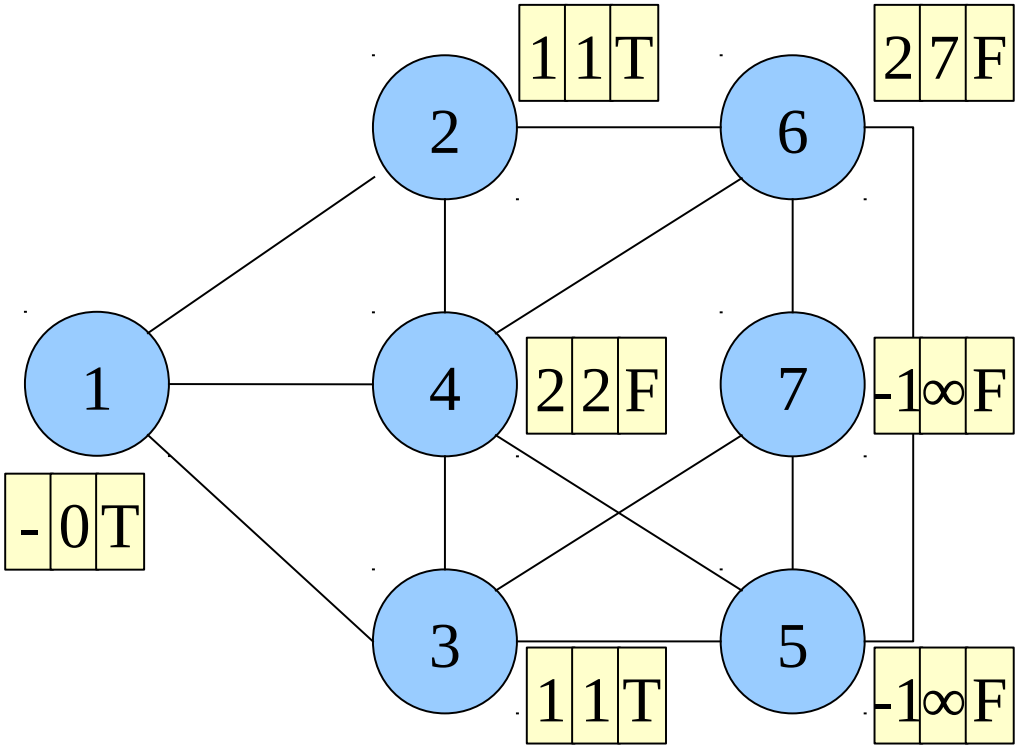


i = 6
Newlabel = 1 + 6 = 7 < ∞
Recente = 2

Algoritmo de Dijkstra

Encontrar caminho mínimo entre 1 e 7

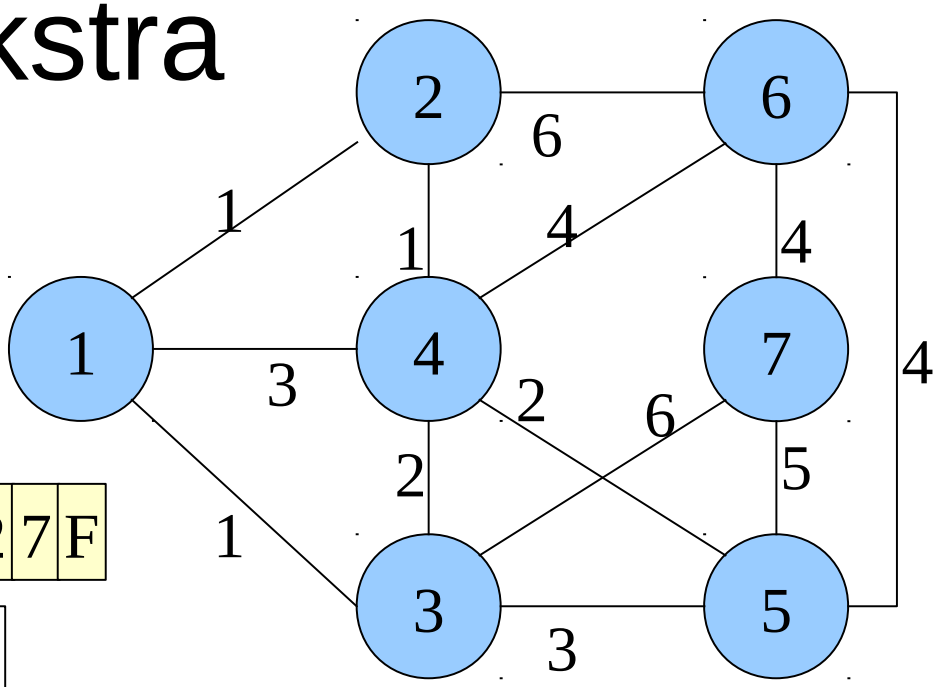


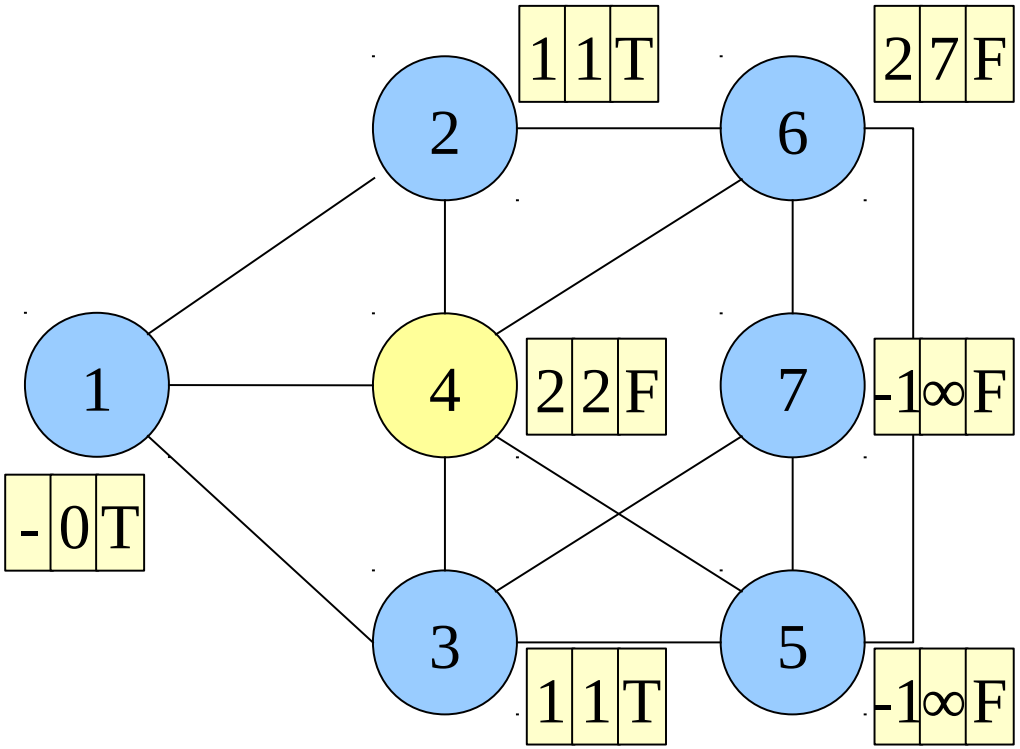


Recente = 3

Algoritmo de Dijkstra

Encontrar caminho mínimo entre 1 e 7

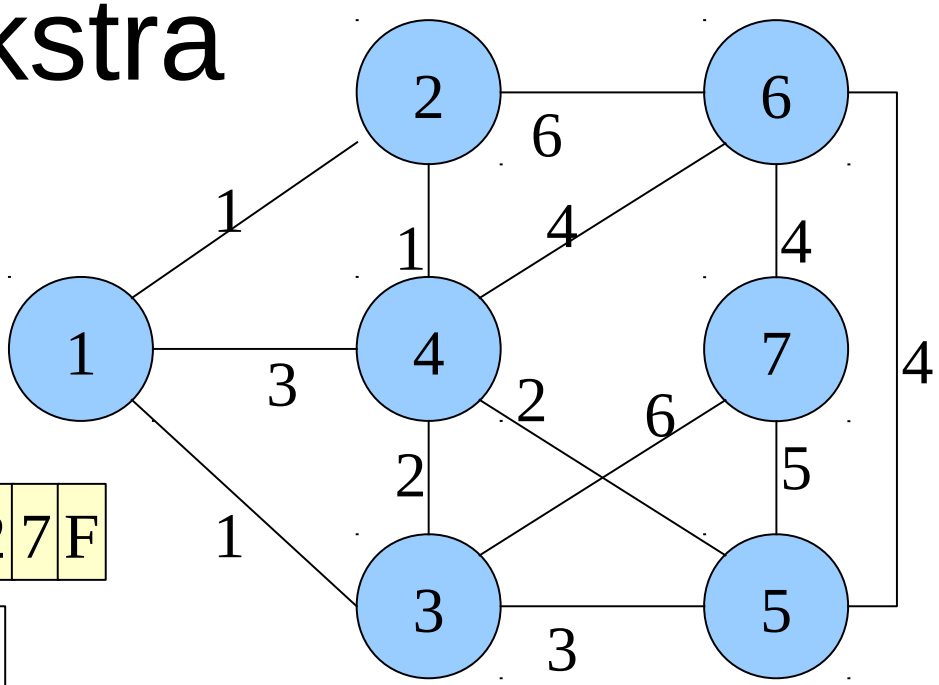


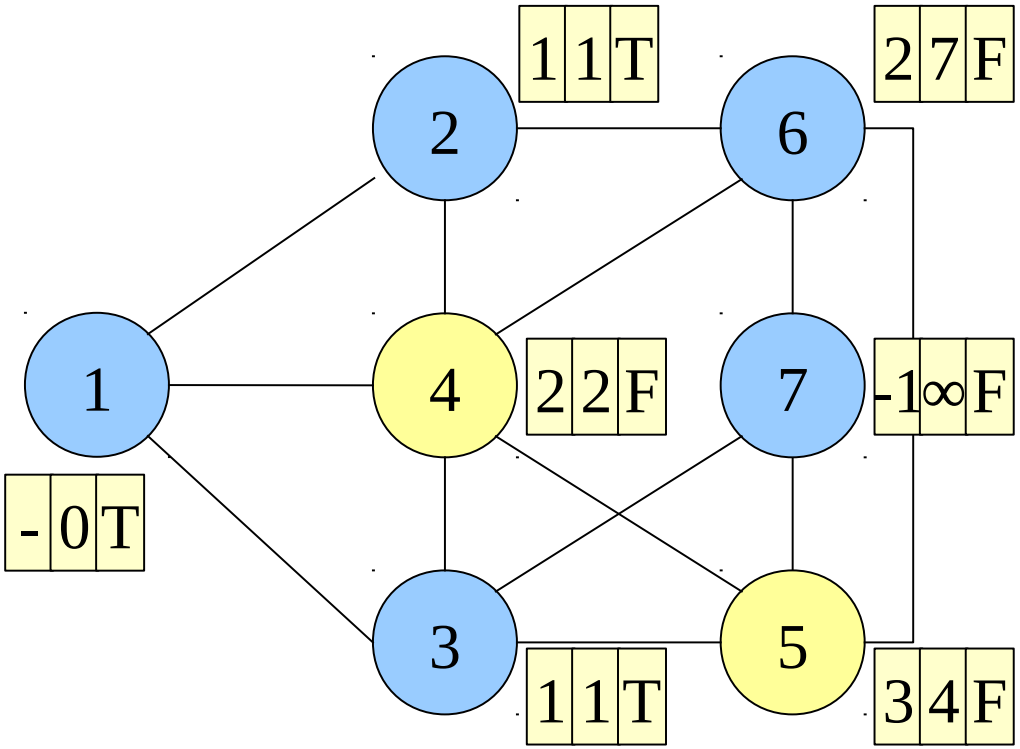


i = 4
Newlabel = 1 + 2 = 3 > 2
Recente = 3

Algoritmo de Dijkstra

Encontrar caminho mínimo entre 1 e 7

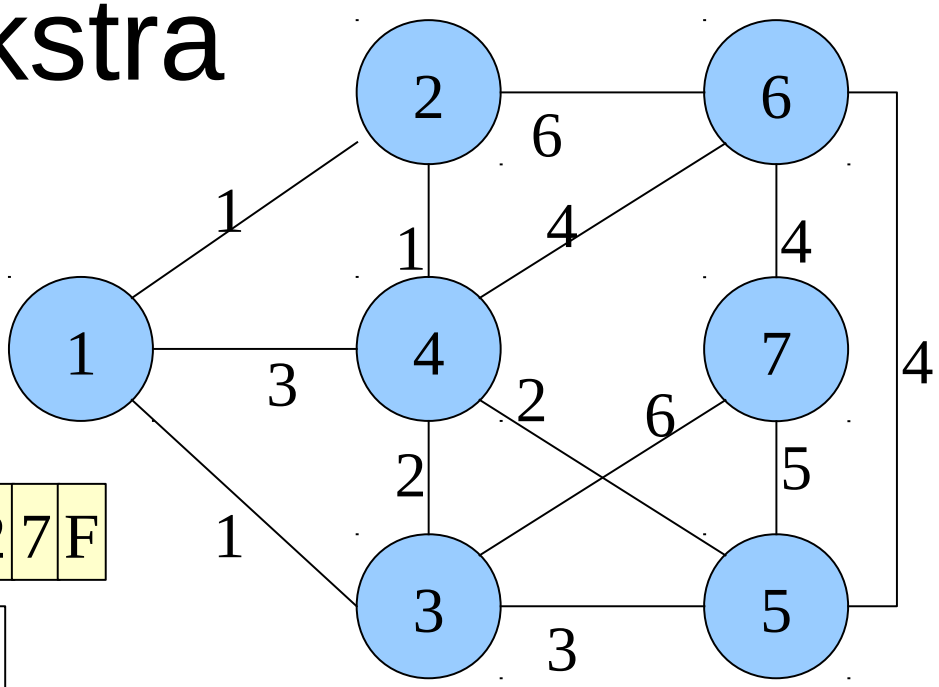


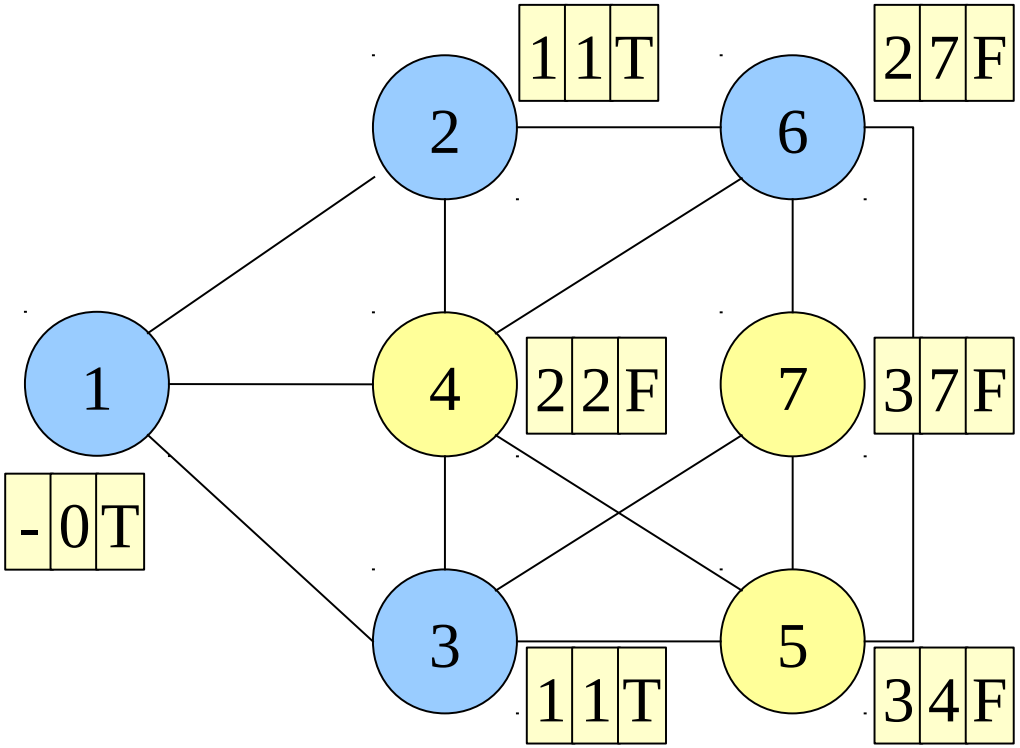


i = 5
Newlabel = 1 + 3 = 4 < ∞
Recente = 3

Algoritmo de Dijkstra

Encontrar caminho mínimo entre 1 e 7

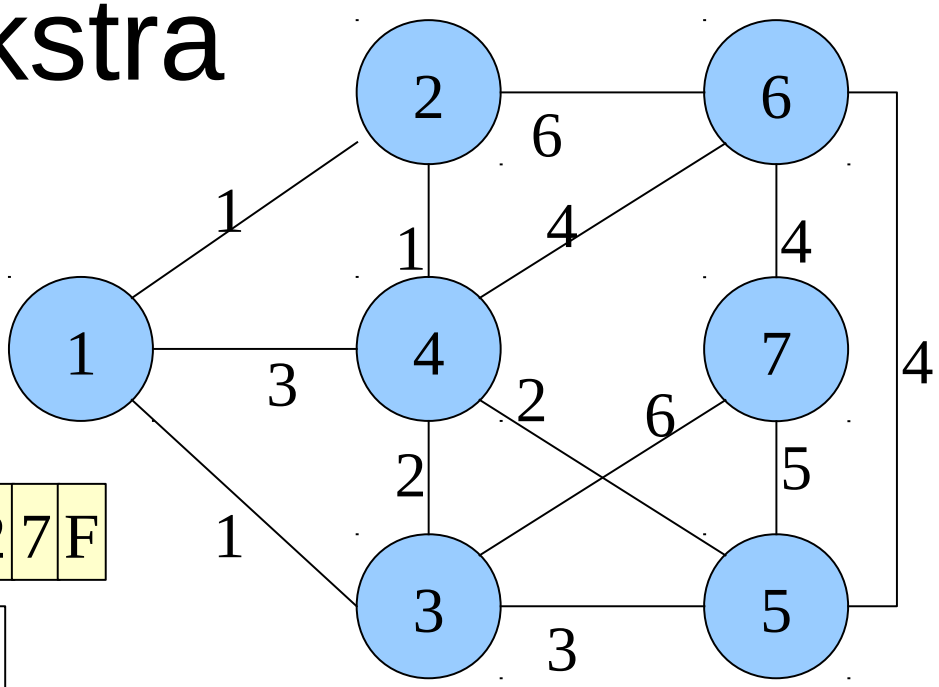


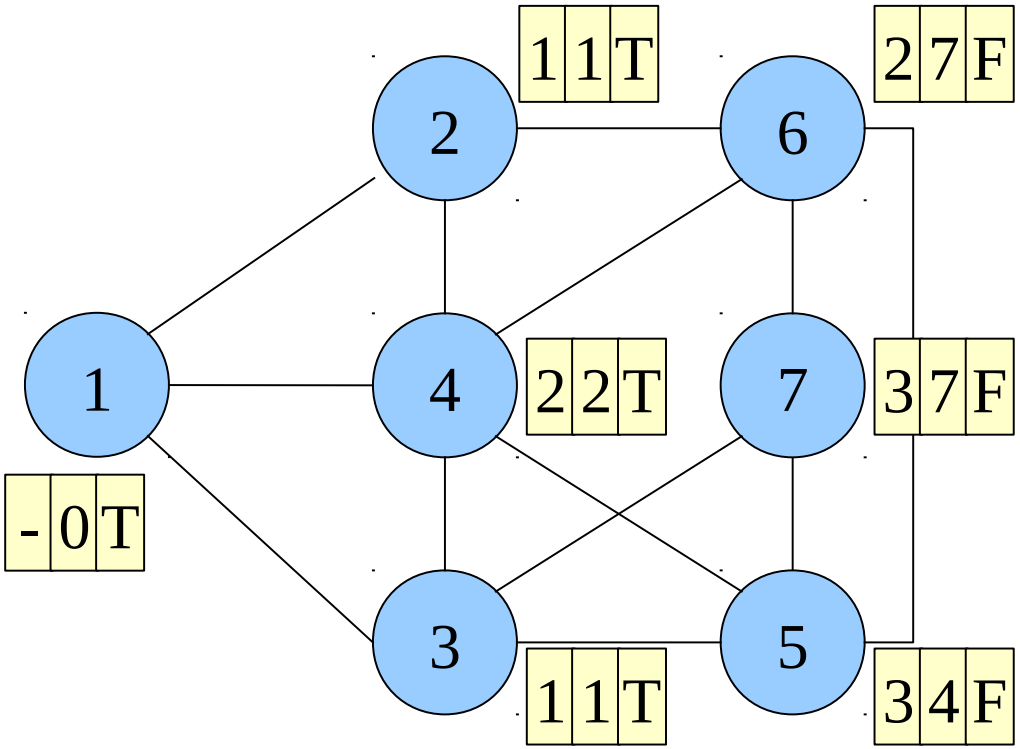


i = 7
Newlabel = 1 + 6 = 7 < ∞
Recente = 3

Algoritmo de Dijkstra

Encontrar caminho mínimo entre 1 e 7

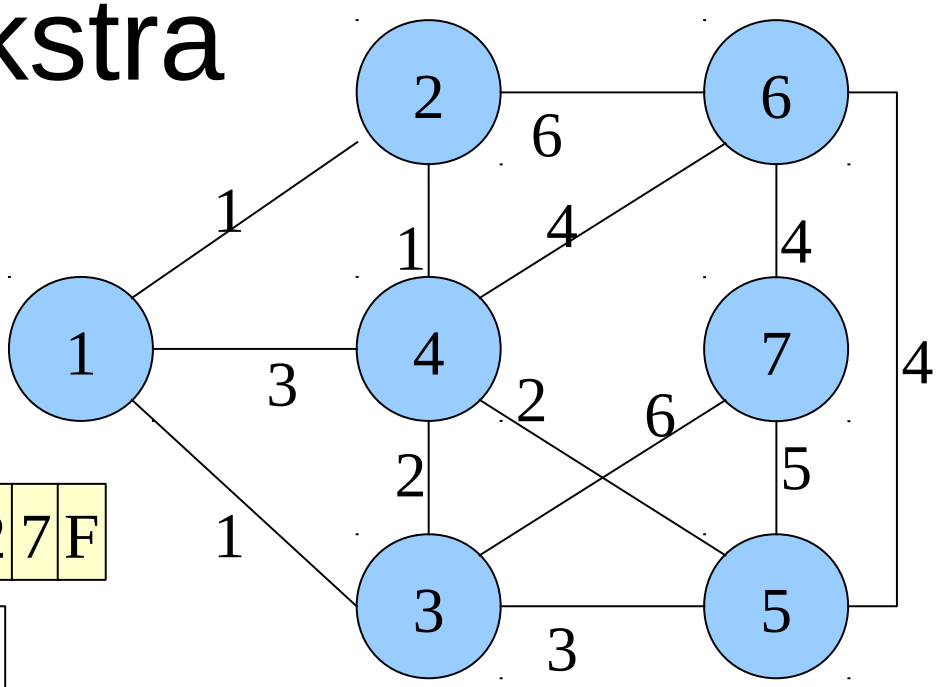


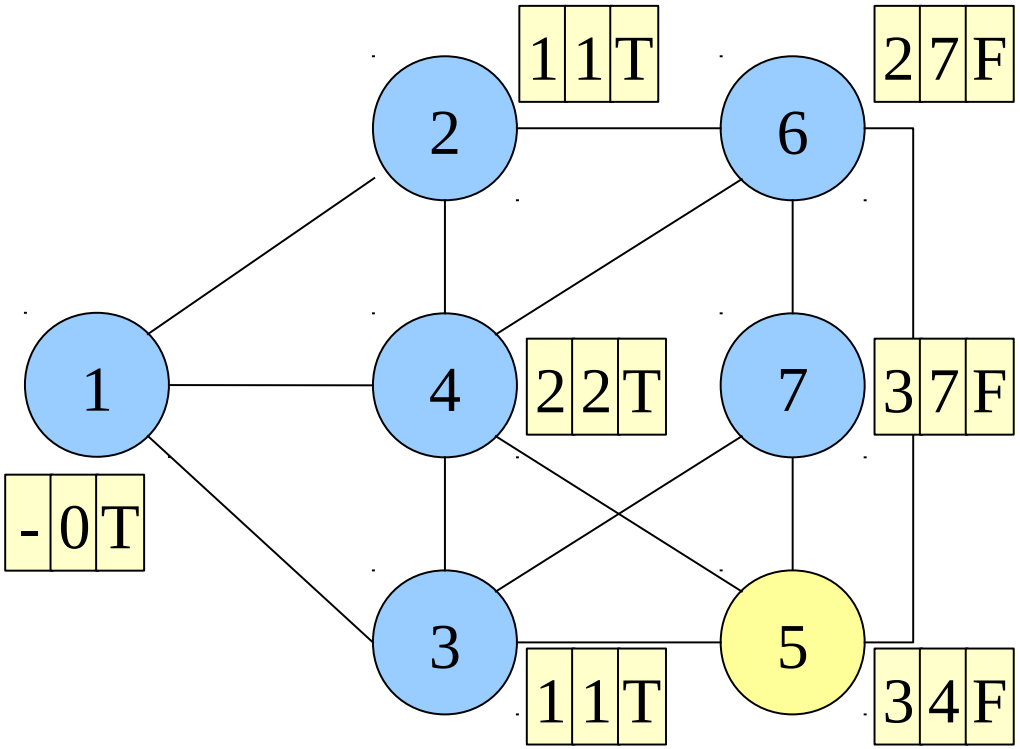


Recente = 4

Algoritmo de Dijkstra

Encontrar caminho mínimo entre 1 e 7

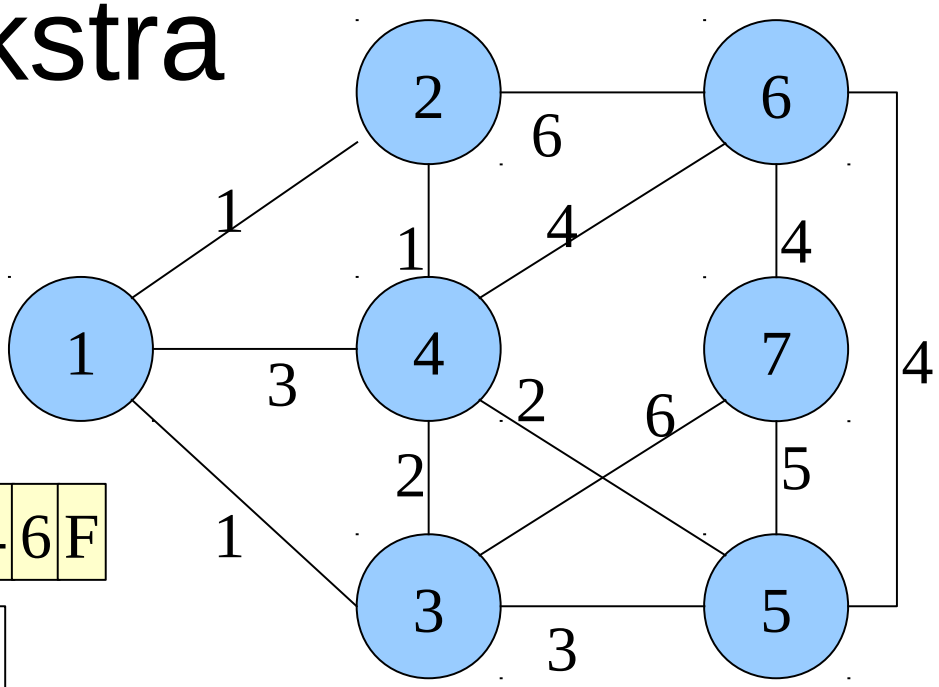


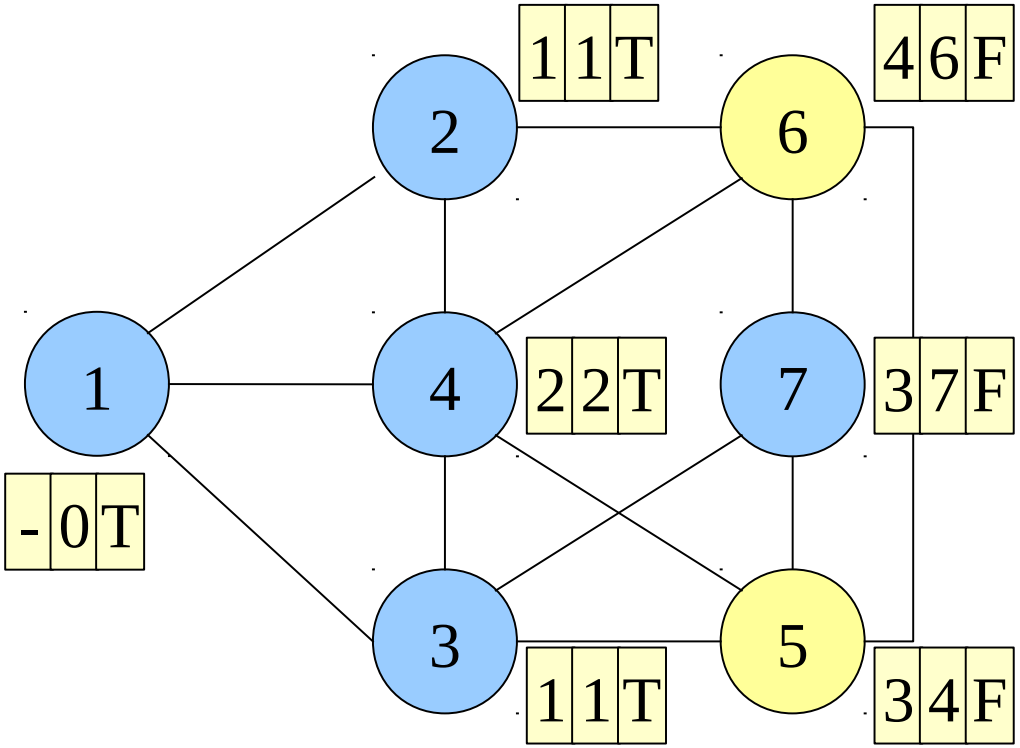


i = 5
Newlabel = 2 + 2 = 4 =4
Recente = 4

Algoritmo de Dijkstra

Encontrar caminho mínimo entre 1 e 7

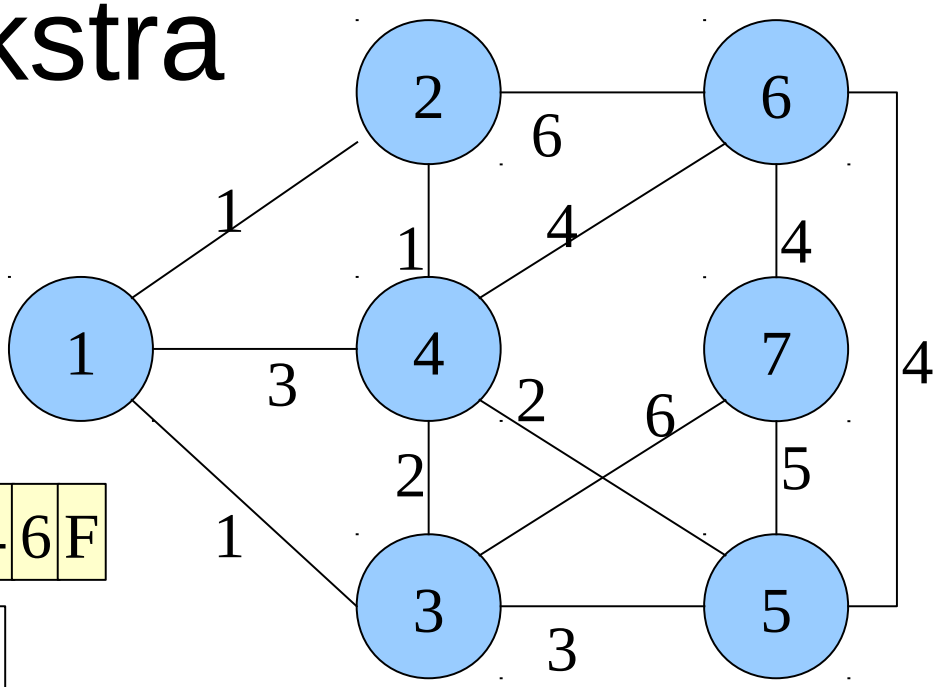


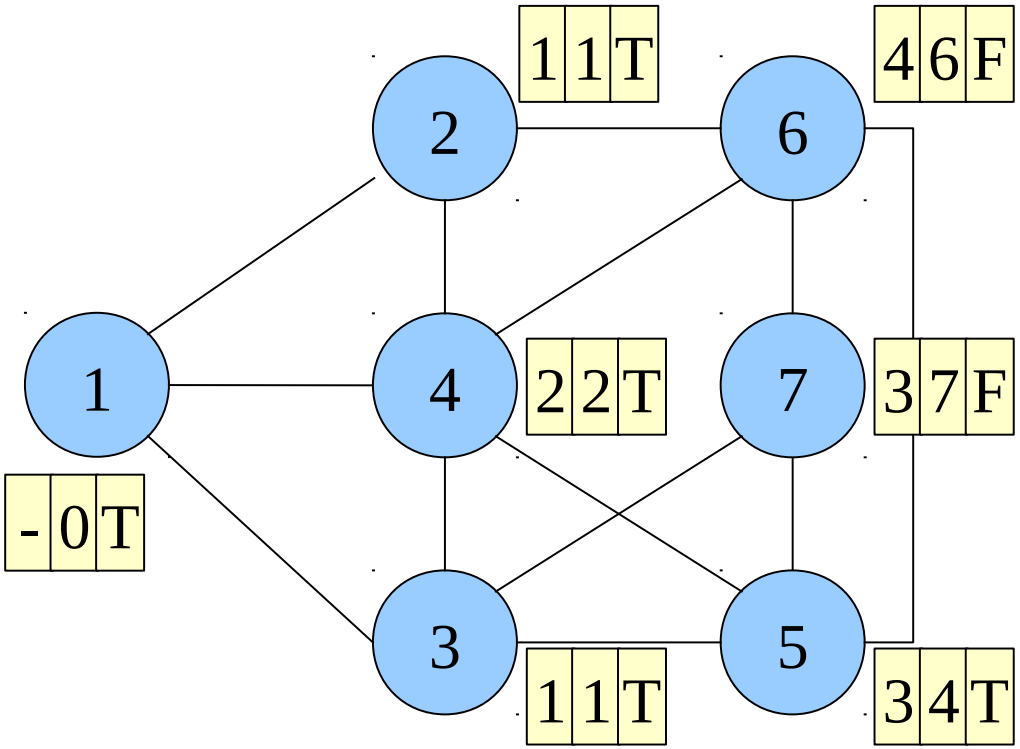


i = 6
Newlabel = 2 + 4 = 6 < 7
Recente = 4

Algoritmo de Dijkstra

Encontrar caminho mínimo entre 1 e 7

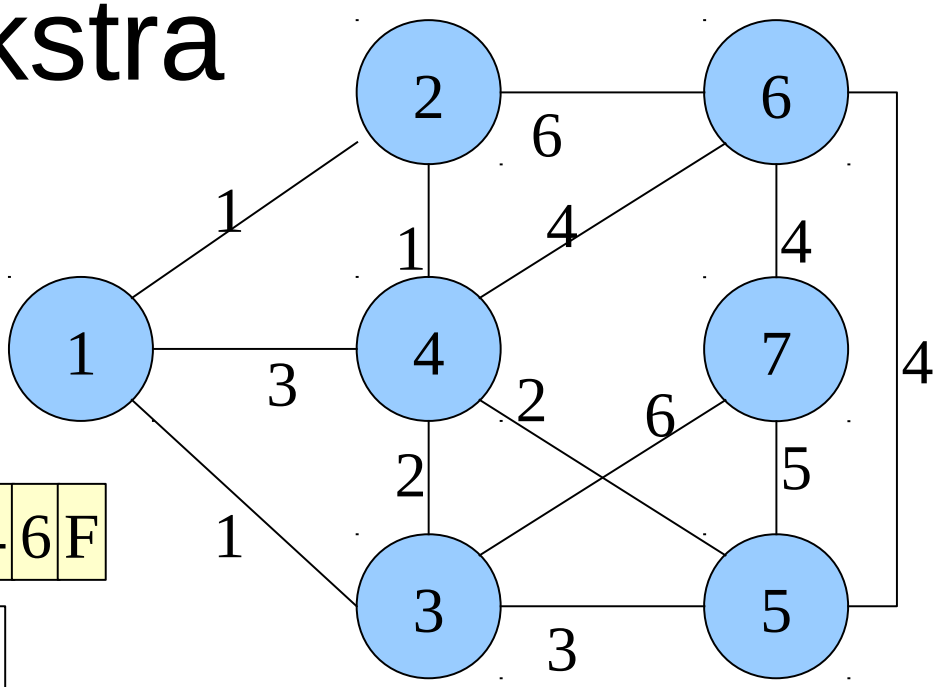


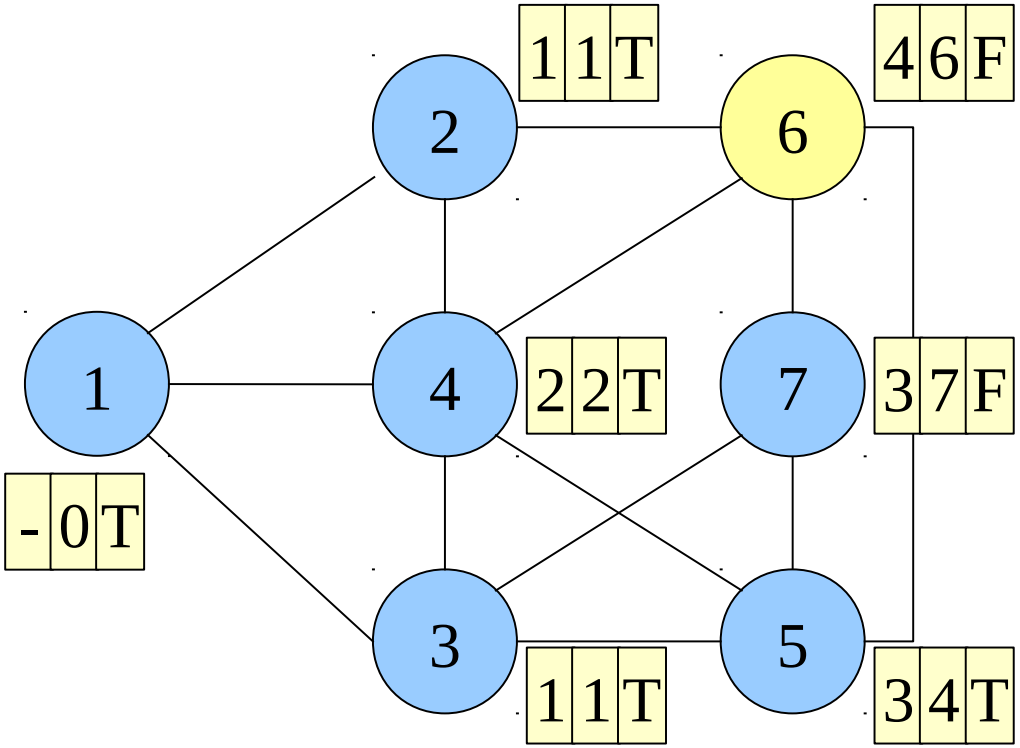


Recente = 5

Algoritmo de Dijkstra

Encontrar caminho mínimo entre 1 e 7

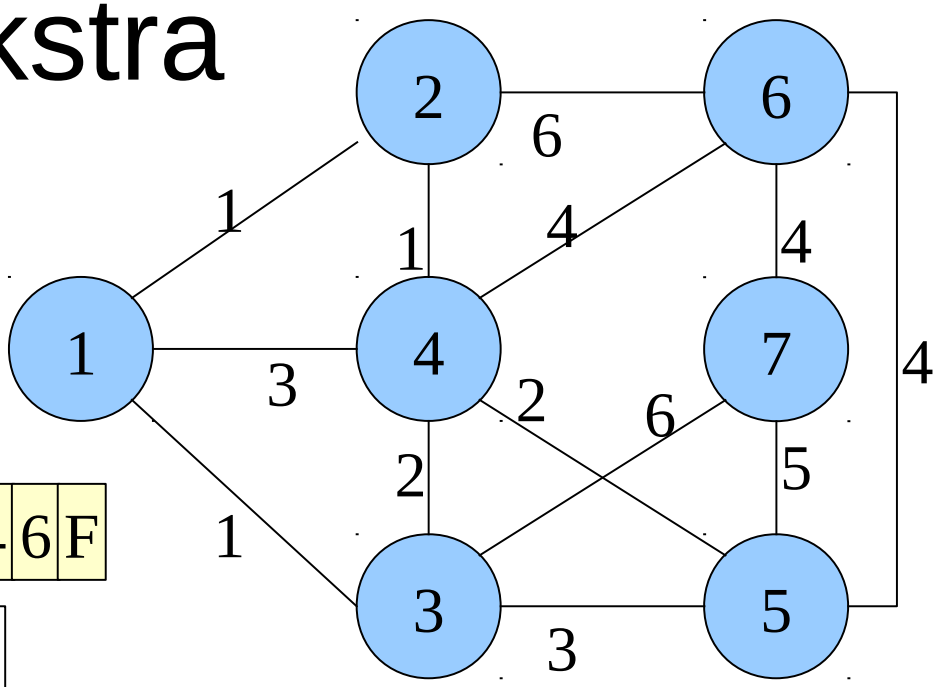


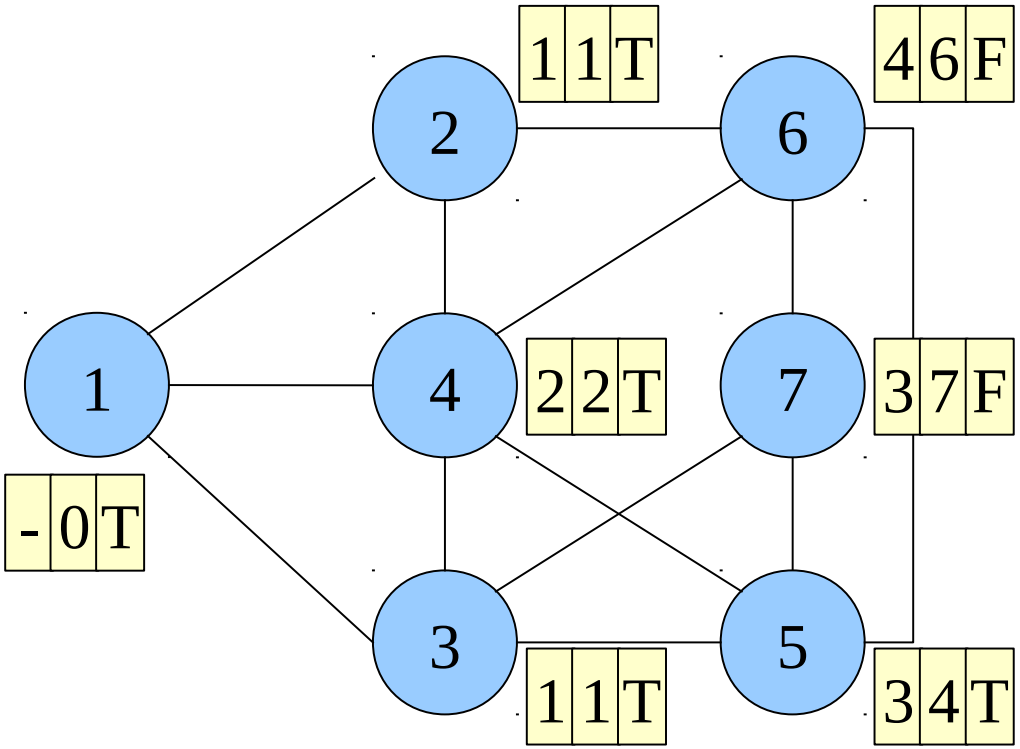


i = 6
Newlabel = 4 + 4 = 8 > 6
Recente = 5

Algoritmo de Dijkstra

Encontrar caminho mínimo entre 1 e 7

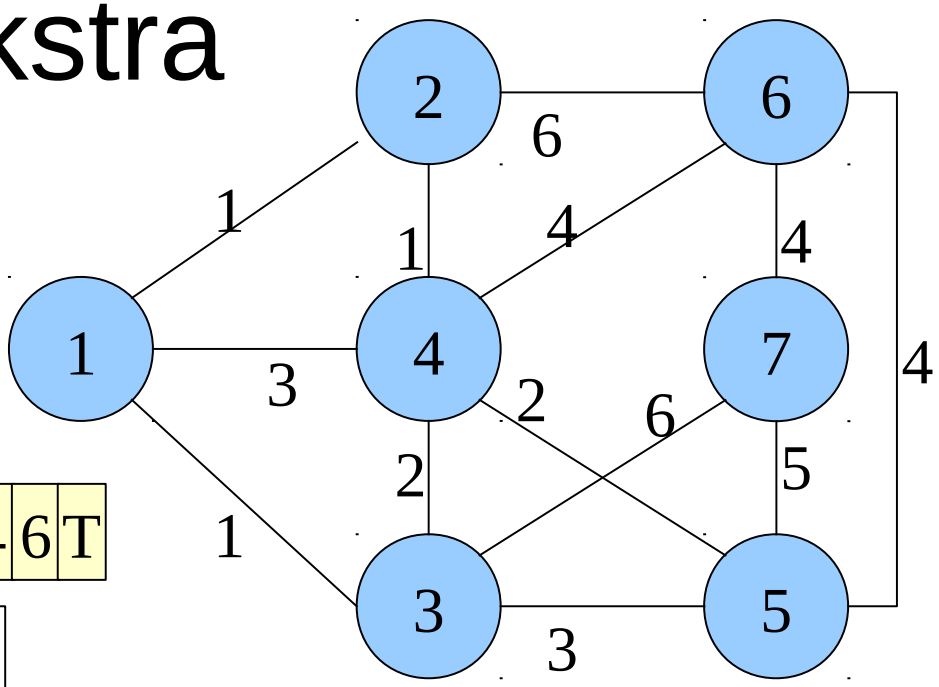


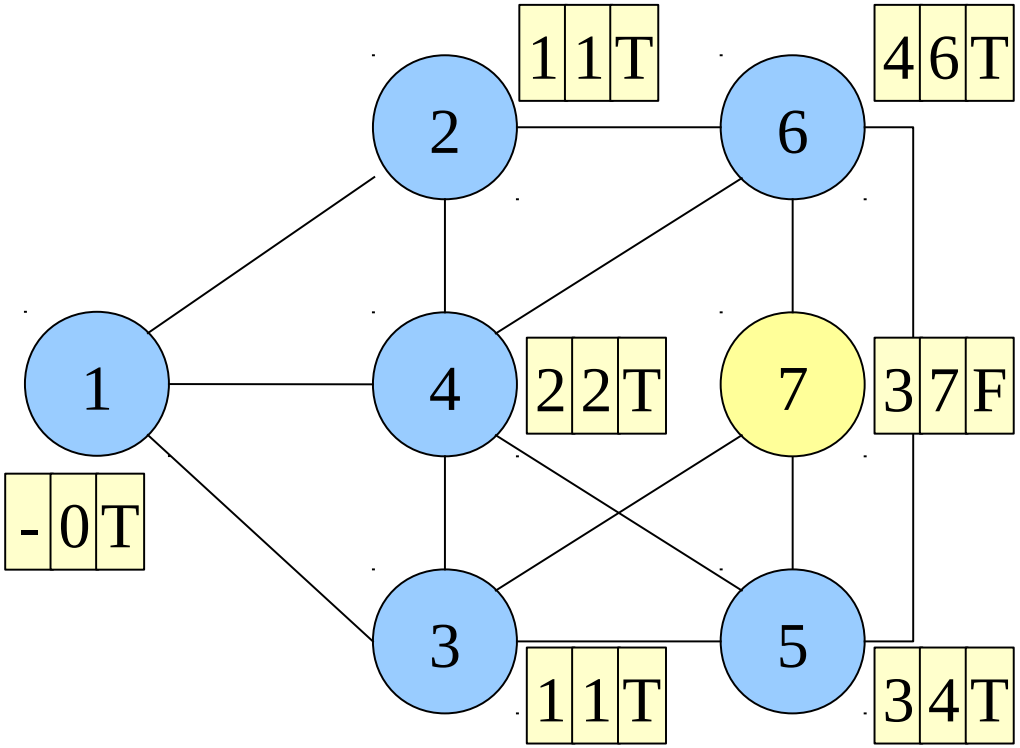


Recente = 6

Algoritmo de Dijkstra

Encontrar caminho mínimo entre 1 e 7

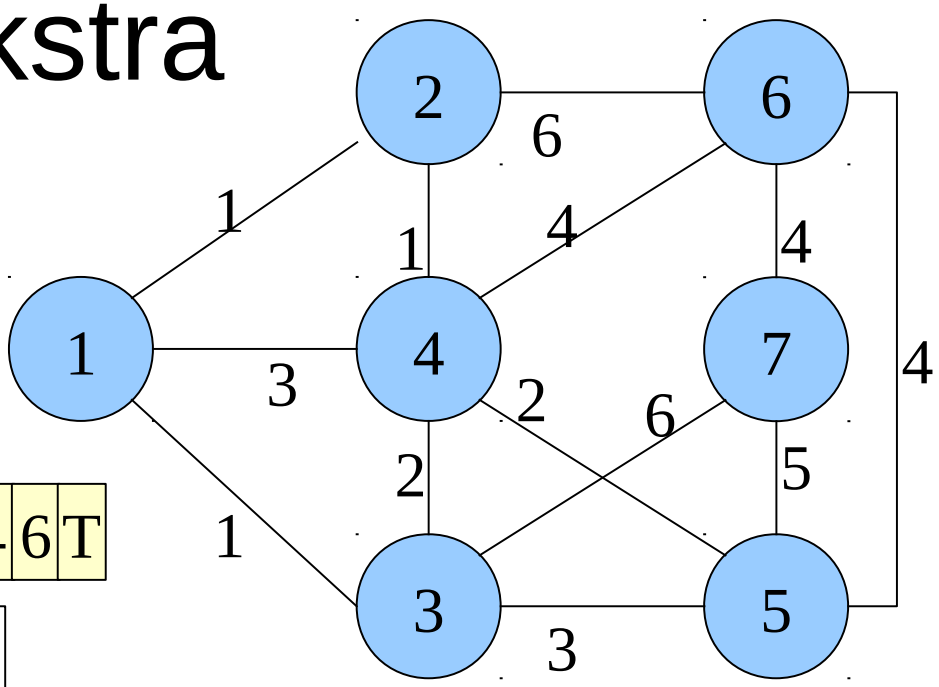


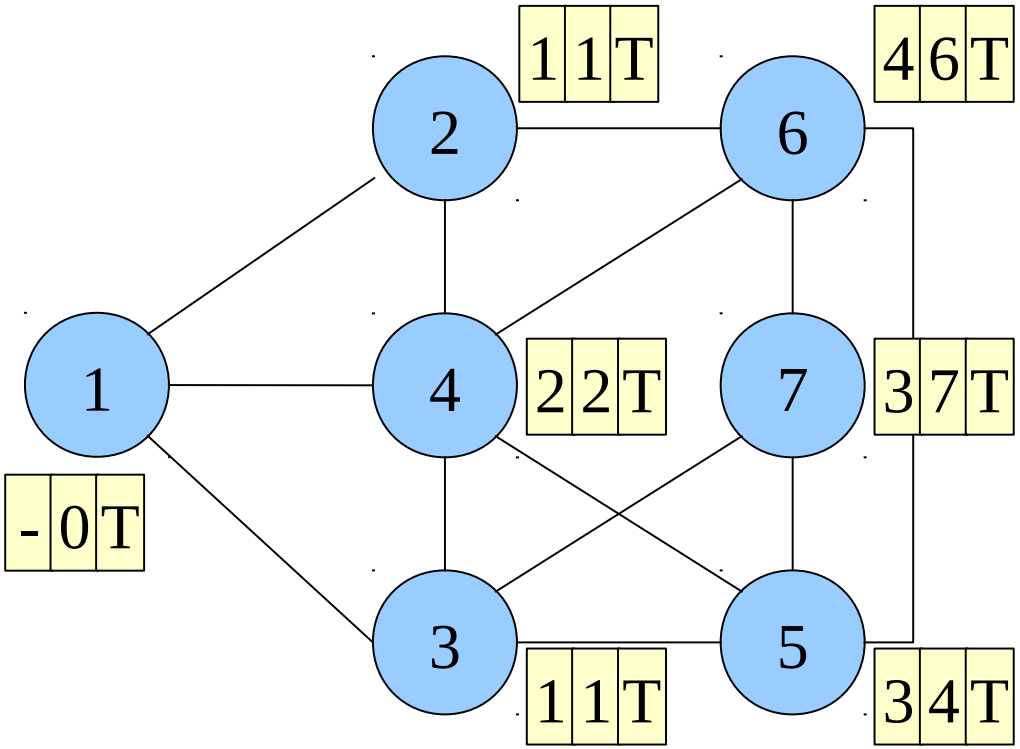


i = 7
Newlabel = 6 + 4 =10 > 7
Recente = 6

Algoritmo de Dijkstra

Encontrar caminho mínimo entre 1 e 7

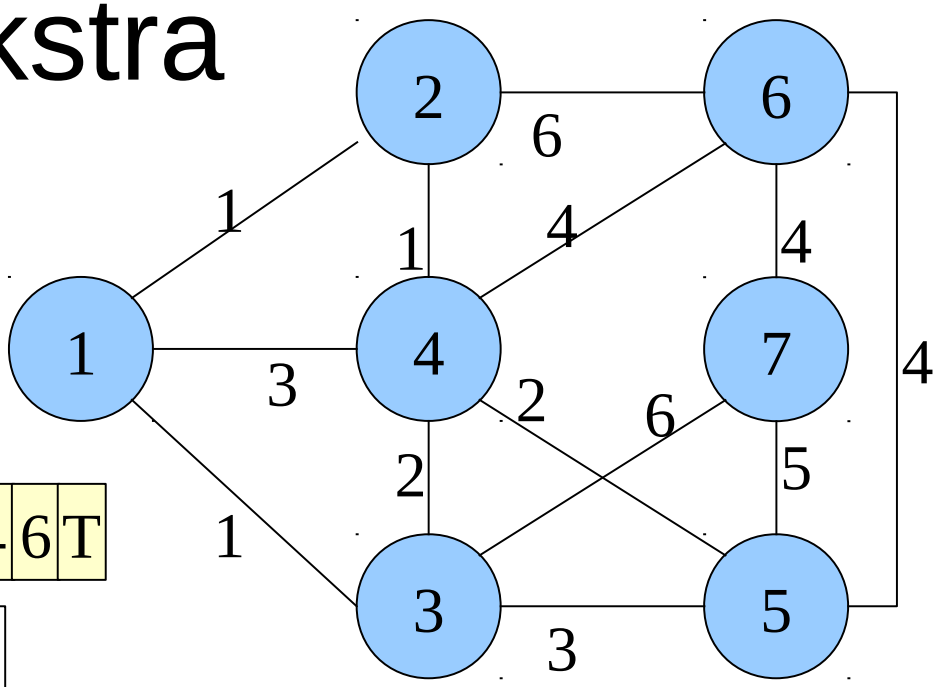


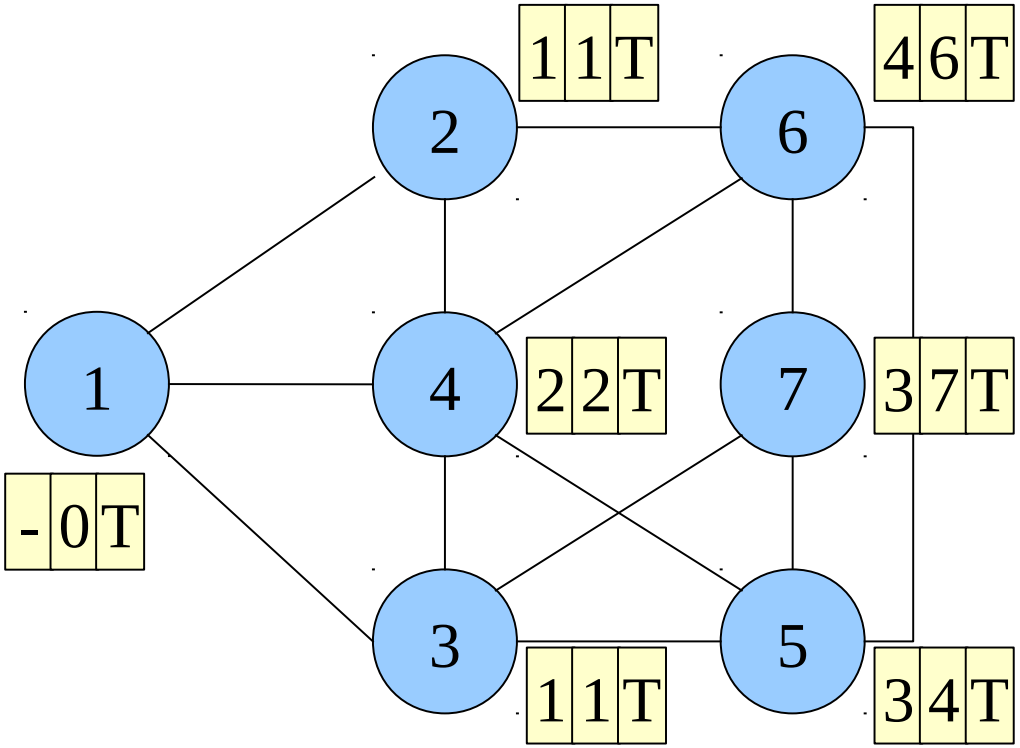


Recente = 7
(Fim do Algoritmo)

Algoritmo de Dijkstra

Encontrar caminho mínimo entre 1 e 7





Caminho Mínimo = 1 → 3 → 7
Distância Mínima = 7

O Problema do Caminho Mínimo

- Encontrar o caminho mínimo entre dois vértices distintos do grafo: algoritmo de Dijkstra
- Encontrar o caminho mínimo entre um vértice s e todos os vértices j, com j ≠ s: adaptar Dijkstra modificando o seu critério de parada para
  - Enquanto existir nó x com final (y) = false faça
- Encontrar o caminho mínimo entre cada par de vértices (u,v) de G: repetir n vezes o algoritmo
- O algoritmo de Dijkstra exige que todos os pesos sejam maiores ou iguais à zero

Algoritmo de Bellman-Moore

- Utilizado para encontrar caminhos mínimos numa rede direcionada com pesos arbitrários
  - Pesos podem ser negativos, *mas não podem existir ciclos negativos*
- Iniciação

  dist (s) = 0

  dist (i) = ∞, ∀ i ≠ s

  pred (i) = -1, ∀i
- Iterações

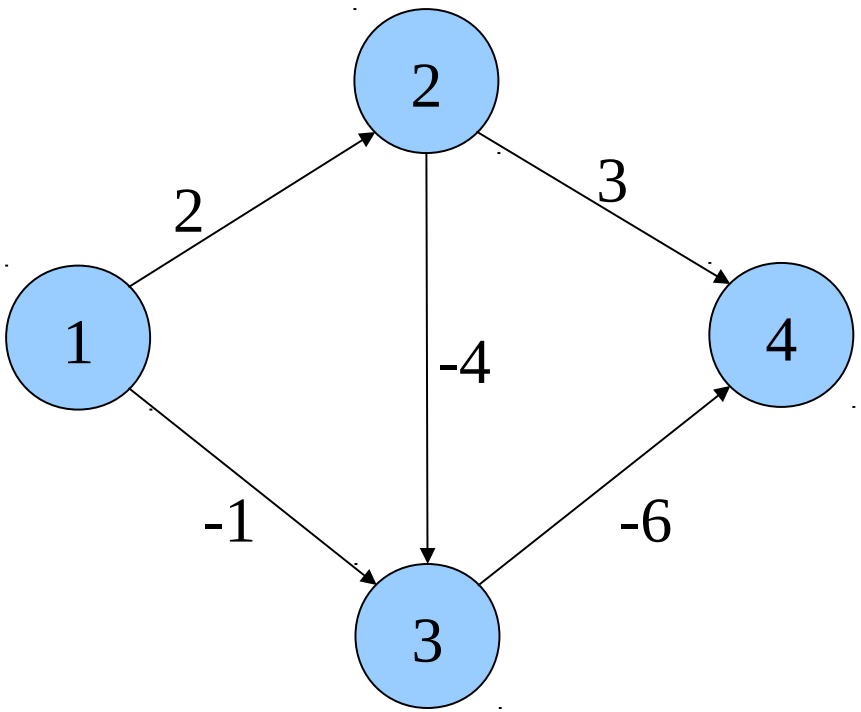
  Para todo vértice u

  Enquanto existir arco (u,v) tal que dist(u) + p_{uv}< dist(v) faça

  dist (v) = dist(u) + p_{uv}

  pred(v) = u

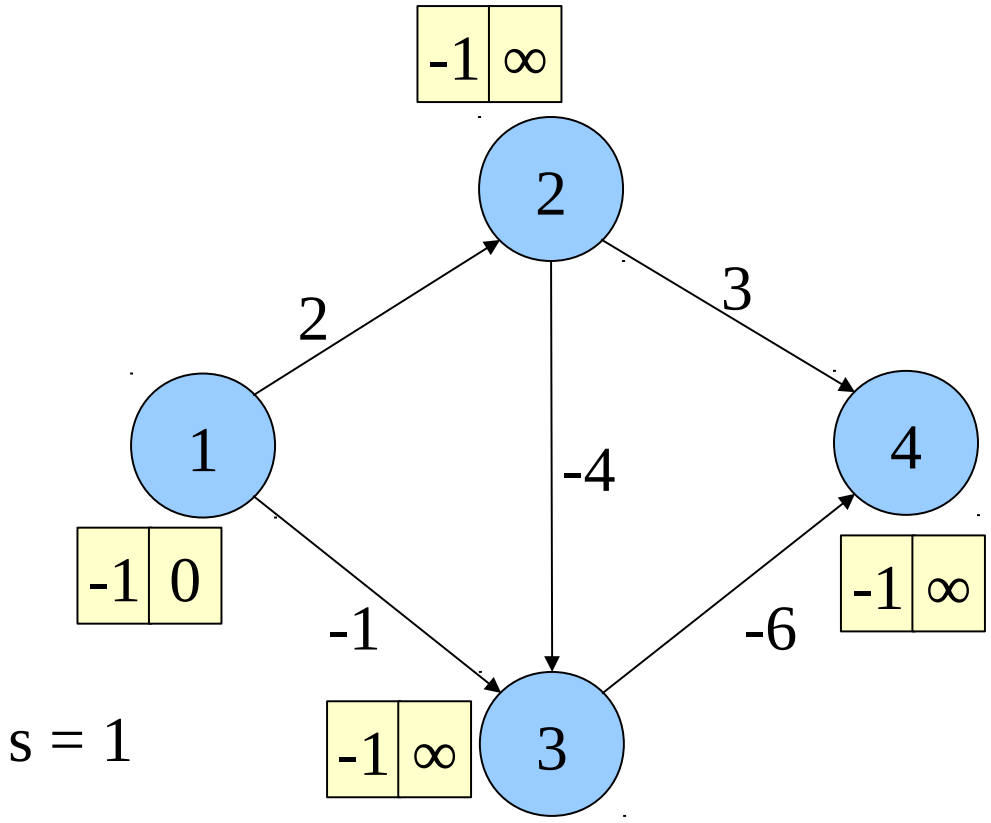
Algoritmo de Bellman-Moore



Encontrar caminho mínimo entre 1 e 4

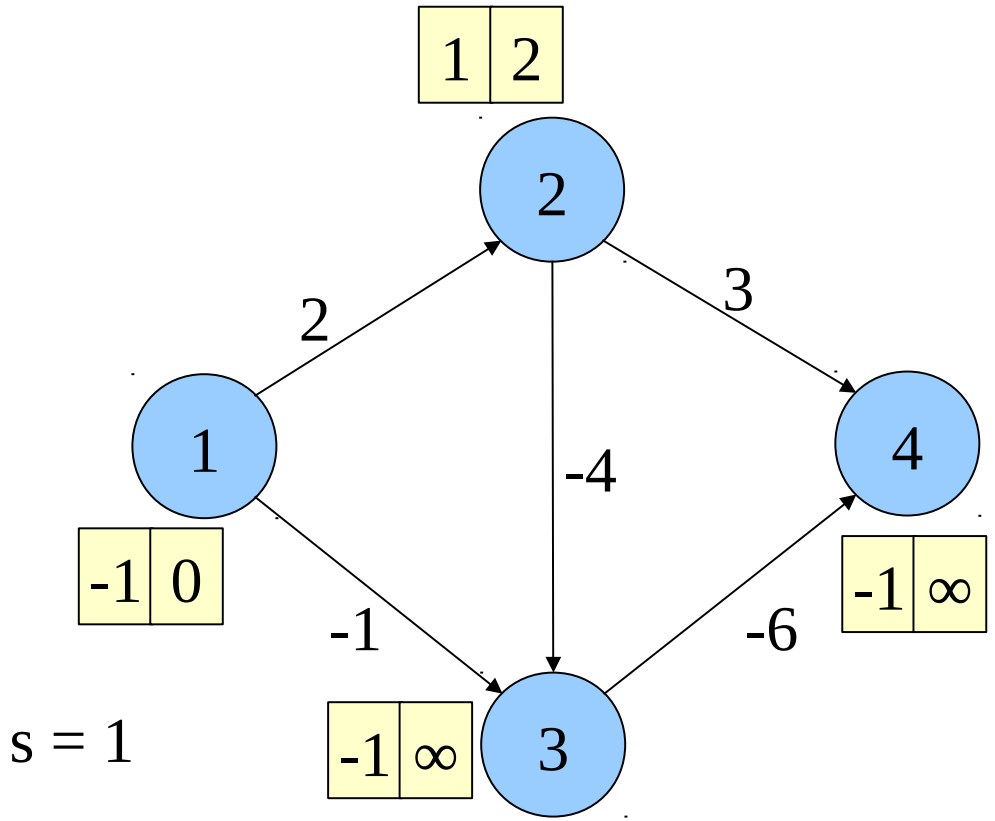
Algoritmo de Bellman-Moore

- Iniciação:
  - s = 1
  - dist(1) = 0
  - dist (2) = ∞
  - dist (3) = ∞
  - dist (4) = ∞



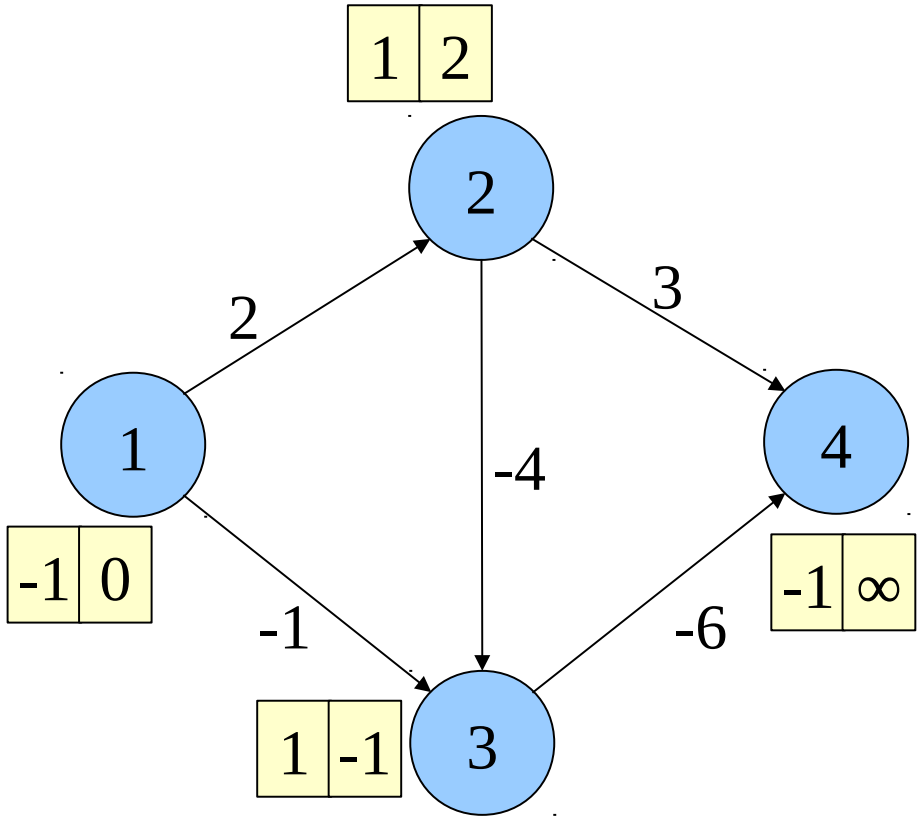
Algoritmo de Bellman-Moore

- u = 1
- (u,v) = (1,2)
  - dist(2) = 2
  - pred(2) = 1
  - dist (3) = ∞
  - dist (4) = ∞



Algoritmo de Bellman-Moore

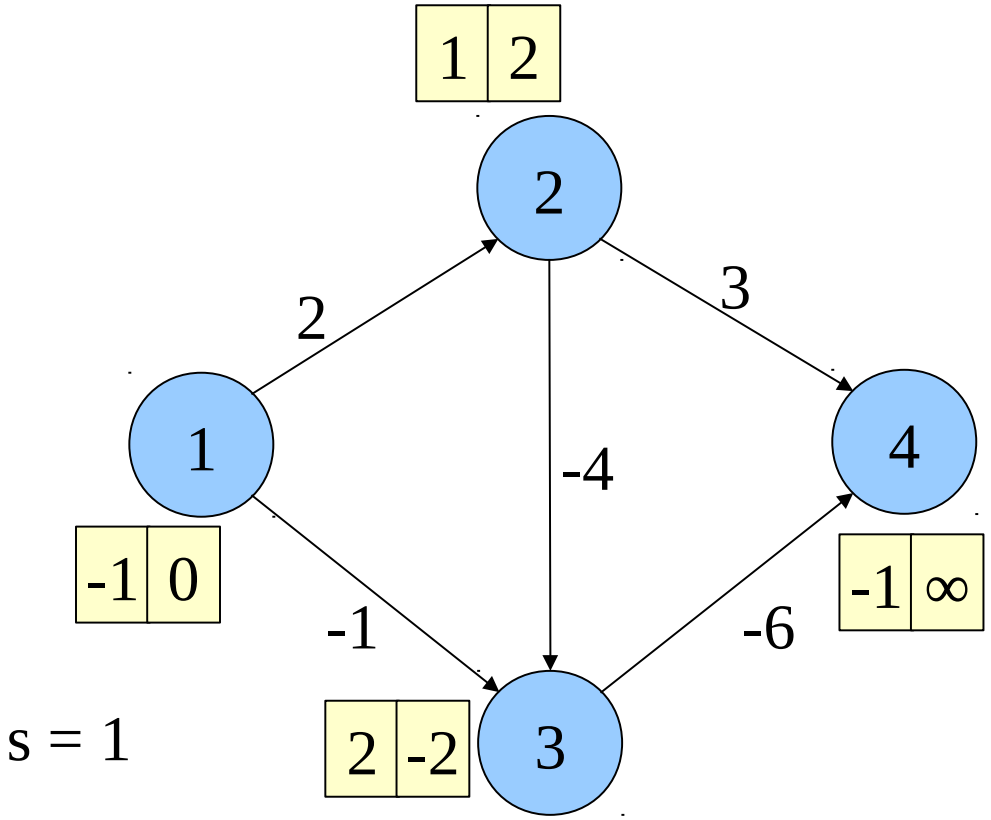
- u = 1
- (u,v) = (1,3)
  - dist(2) = 2
  - dist (3) = -1
  - pred(3) = 1
  - dist (4) = ∞



s = 1

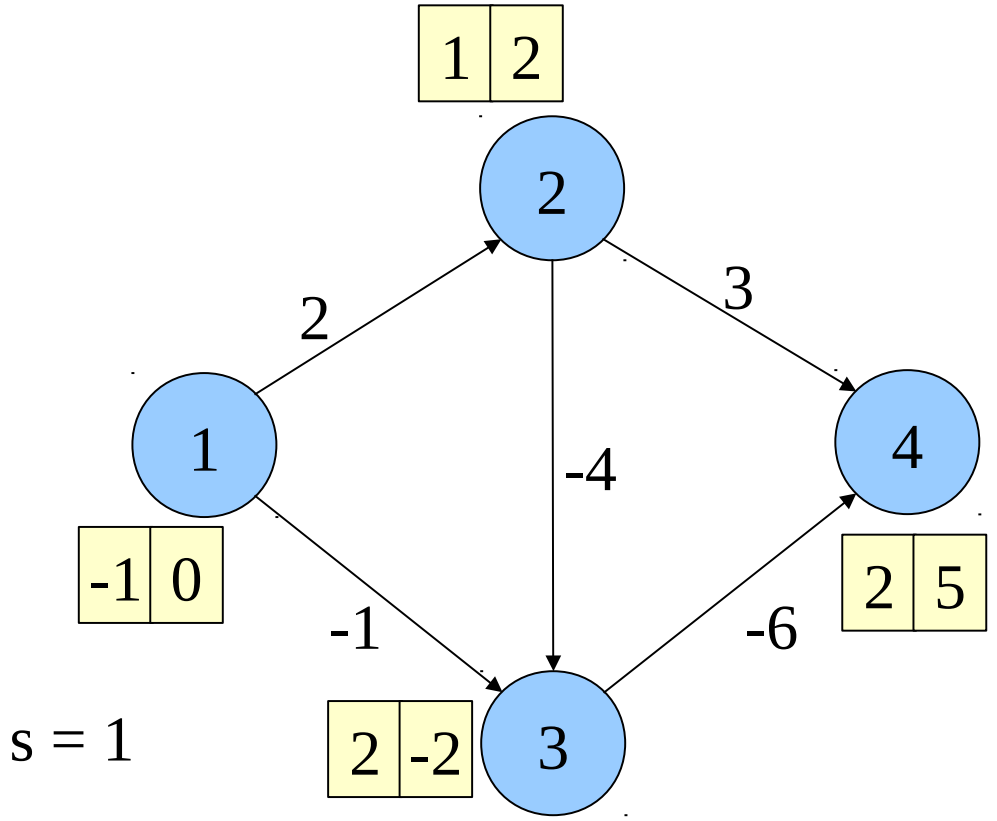
Algoritmo de Bellman-Moore

- u = 2
- (u,v) = (2,3)
  - dist(2) = 2
  - dist (3) = -2
  - pred(3) = 2
  - dist (4) = ∞



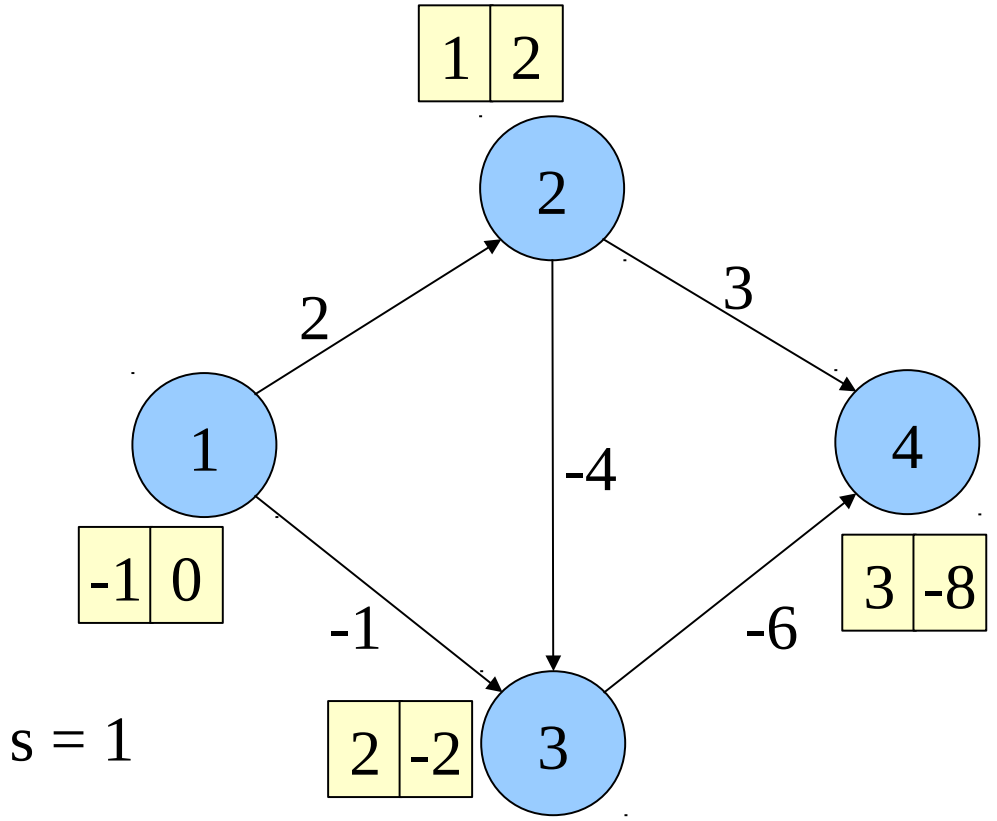
Algoritmo de Bellman-Moore

- u = 2
- (u,v) = (2,4)
  - dist(2) = 2
  - dist (3) = -2
  - dist (4) = 5
  - pred(4) = 2



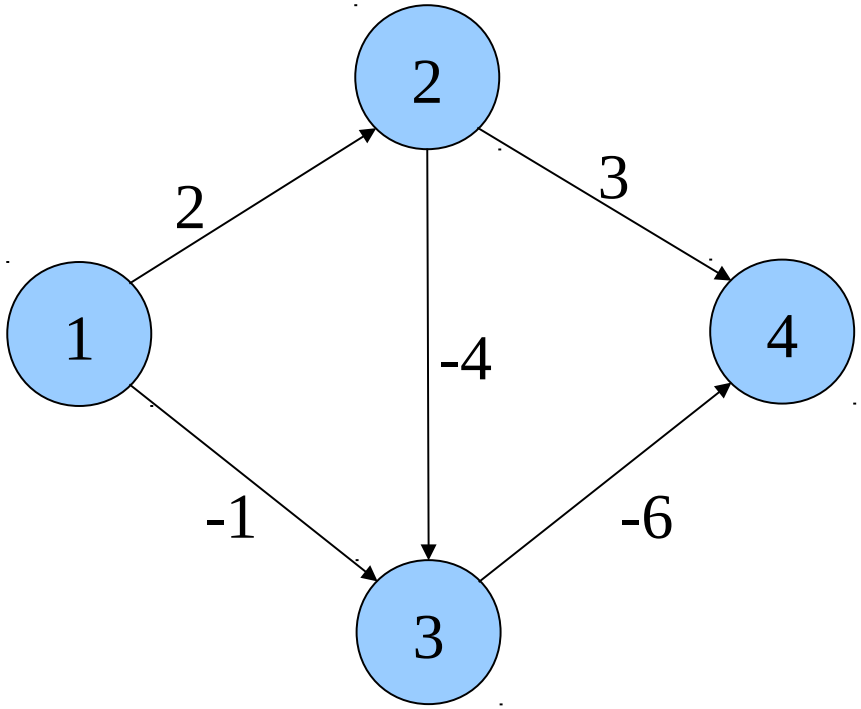
Algoritmo de Bellman-Moore

- u = 3
- (u,v) = (3,4)
  - dist(2) = 2
  - dist (3) = -2
  - dist (4) = -8
  - pred(4) = 3



Algoritmo de Bellman-Moore

- Caminho mínimo: 1 → 2 → 3 → 4
- Custo mínimo: -8



Algoritmo de Bellman-Moore-d'Esopo

- Variação do Algoritmo de Bellman-Moore
- Idéia é construir fila de nós a ser examinado
  - Inicialmente fila Q contém único nó, i = s
  - Numa dada iteração, nó u de Q é examinado
    - u primeiro nó da fila
    - Após examinado, retirado de Q
    - Exame consiste em analisar todos os arcos com origem em u com tipo (u,v)
    - Se peso de s até v for reduzido usando u, adicionamos v na fila Q (caso já não esteja nela)

Algoritmo de Bellman-Moore-d'Esopo

- Observe que v entra na fila Q se dist(v) for reduzido
  - Um determinado nó pode entrar na fila Q várias vezes
    - Cada vez que um caminho mais curto for descoberto entre s e t
    - Na prática não interessante: no pior caso o número de entradas ou saídas de um mesmo nó na fila Q pode ser bastante elevado

Algoritmo de Bellman-Moore-d'Esopo

- Heurística para suprir essa deficiência: colocar um nó v no fim da fila Q se nunca foi analisado antes
  - Caso já tenha sido analisado, colocá-lo no início da fila Q
  - Justificativa: reexaminá-lo imediatamente e reduzir, se for o caso, os valores de todos os nós modificados através deste nó
    - Segundo o autor, isto reduz o número de vezes em que v retorna a fila Q

Algoritmo de Bellman-Moore-d'Esopo

- Para Pesos Arbitrários e sem Ciclos Recursivos
- Iniciação

  Para todo v ≠ s

  dist (v) = ∞

  pred (v) = -1

  Para v = s

  dist (s) = 0

  pred (s) = -1

  Q = {s}

  head = s

Algoritmo de Bellman-Moore-d'Esopo

- Iterações

  Enquanto Q ≠ ∅ faça

    Retire nó head = u de Q

    Para cada aresta (u,v) faça

      Newlabel = dist(u)+p_{uv}

      Se Newlabel < dist(v) então

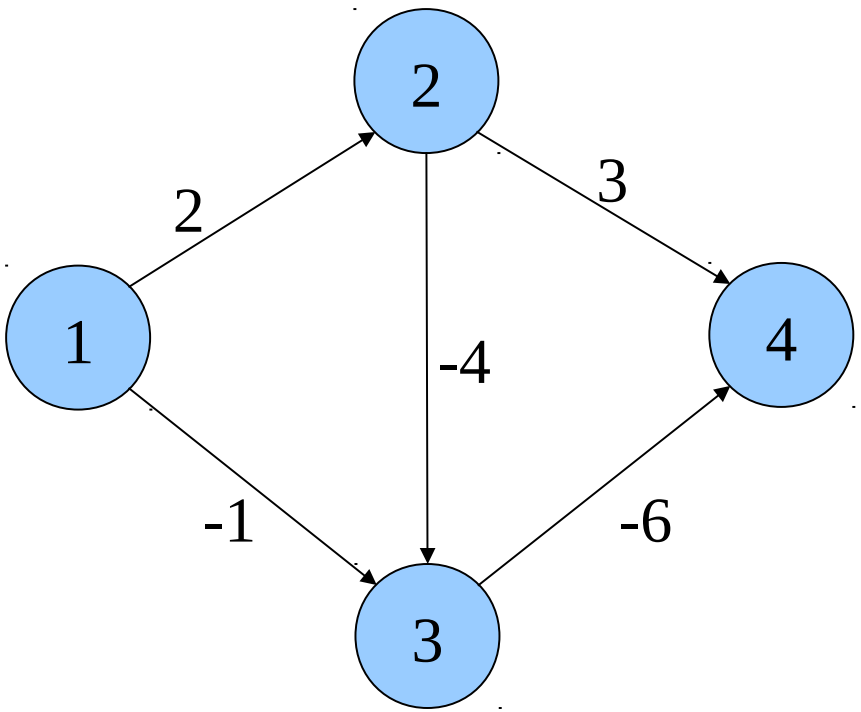
        dist(v) = Newlabel

        pred(v) = u

        Se v nunca esteve em Q antes então insere v no final de Q

        Caso contrário (se v já esteve em Q, mas não está atualmente) inserir v no início de Q

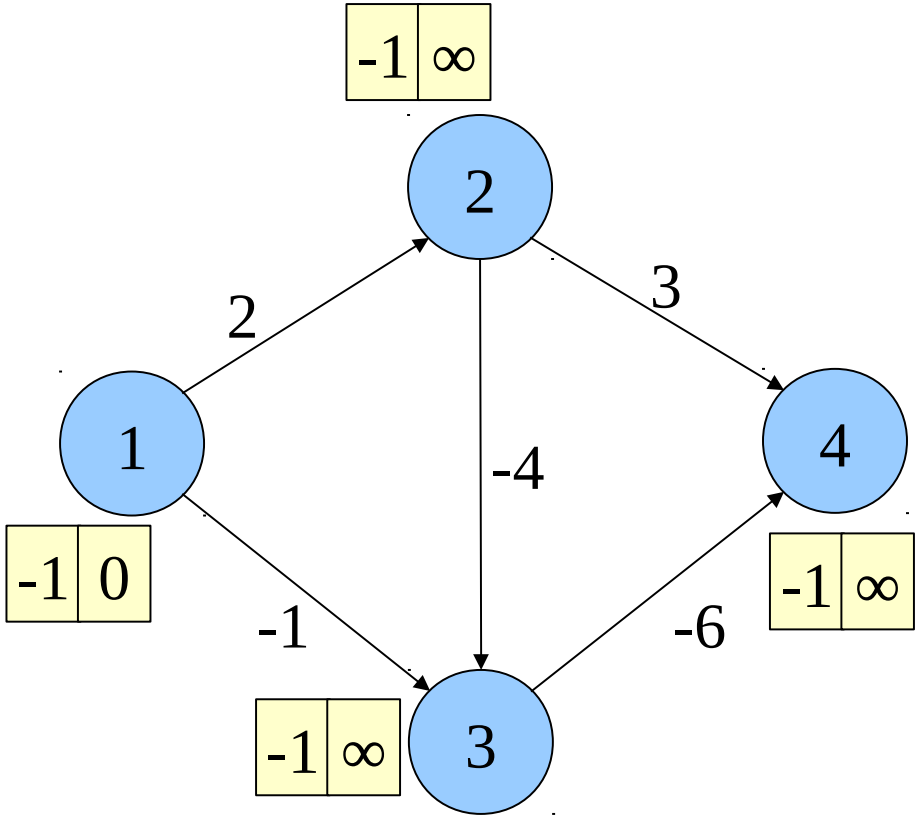
Algoritmo de Bellman-Moore-d'Esopo



Encontrar caminho mínimo entre 1 e 4

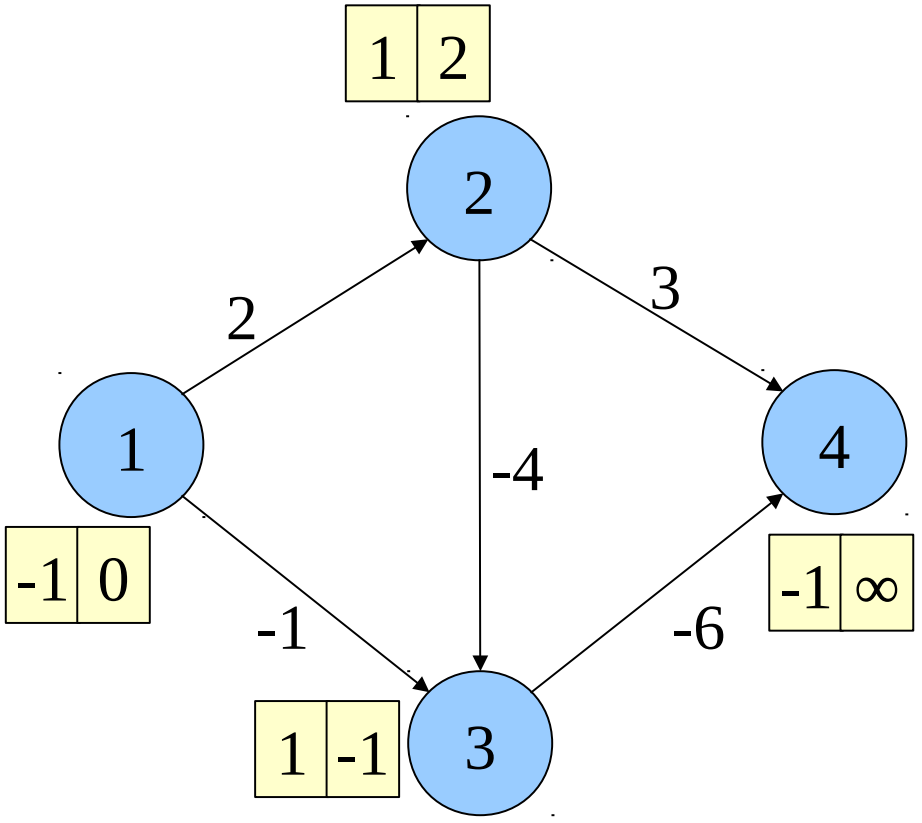
Algoritmo de Bellman-Moore-d'Esopo

- Iniciação:
  - s = 1
  - dist(1) = 0
  - dist (2) = ∞
  - dist (3) = ∞
  - dist (4) = ∞
  - Q = {1}
  - head = 1



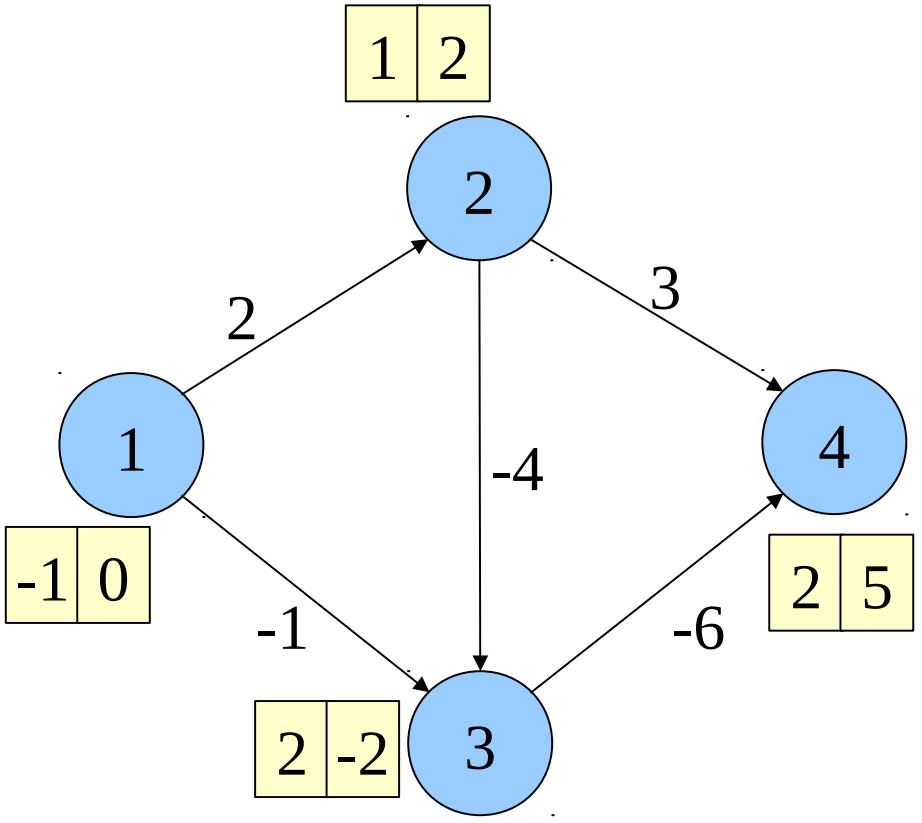
Algoritmo de Bellman-Moore-d'Esopo

- Iterações
  - Q = {2,3}
  - head = 2



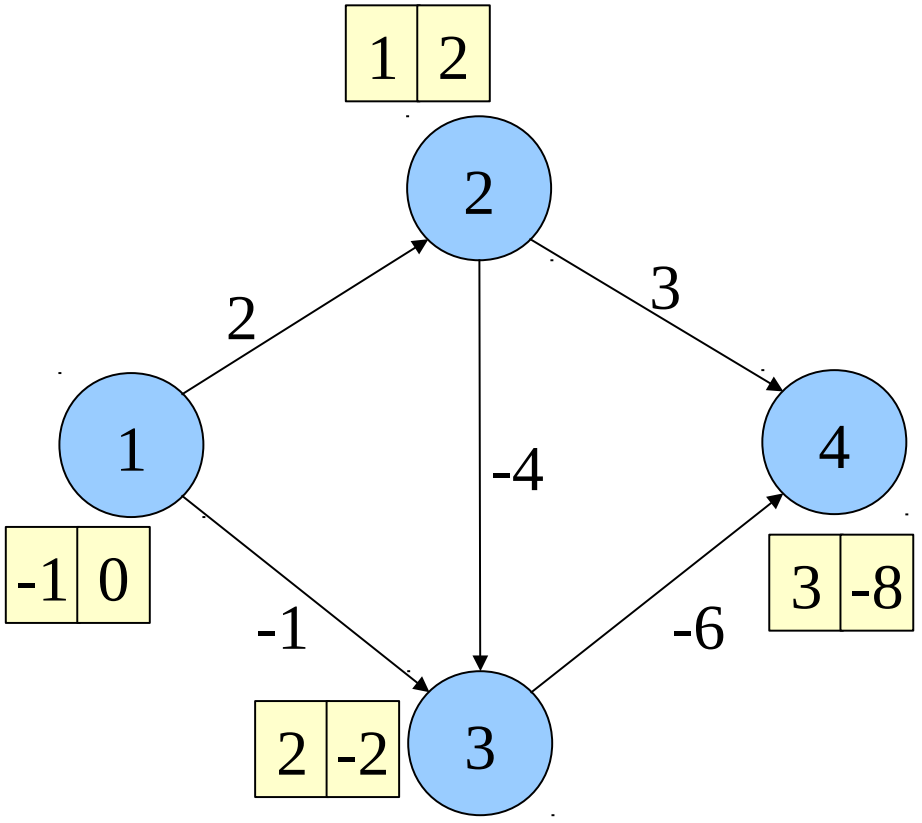
Algoritmo de Bellman-Moore-d'Esopo

- Iterações
  - Q = {3,4}
  - head = 2



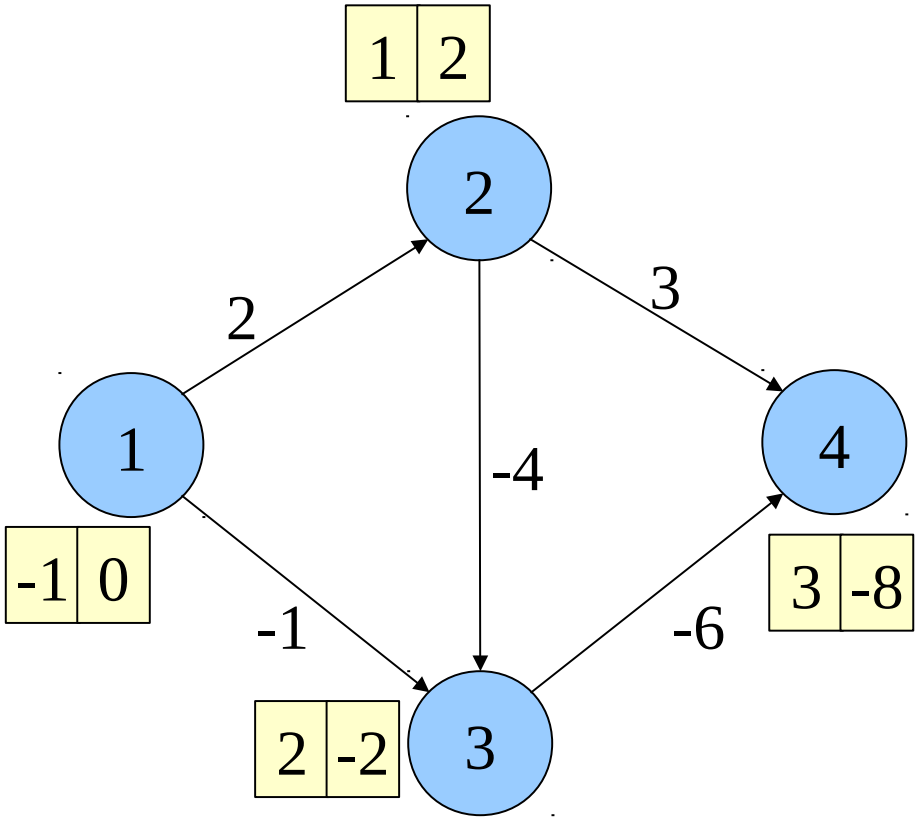
Algoritmo de Bellman-Moore-d'Esopo

- Iterações
  - Q = {4}
  - head = 3



Algoritmo de Bellman-Moore-d'Esopo

- Iterações
  - Q = {}
  - head = 4



Algoritmo de Floyd

- Seja um grafo direcionado representado por uma matriz de adjacências
- A cada etapa calculamos uma matriz de pesos P onde cada elemento P(i,j) representa o valor do caminho mais curto de k arestas entre i e j.
- A cada iteração k atualizamos a matriz, verificando a validade de se incluir k no caminho mínimo de cada par de nós i e j

Algoritmo de Floyd

- Iterações k: 0, 1, 2, ..., n

  $p^{(0)}_{ij} \leftarrow p_{ij}$

  $p^{(k)}_{ij} \leftarrow \min (p^{(k-1)}_{ij}; p^{(k-1)}_{ik} + p^{(k-1)}_{kj})$

- Isso significa que na iteração k = 1 o nó 1 é inserido no caminho mínimo de i para j se $p^{(0)}_{ij} > p^{(0)}_{i1} + p^{(0)}_{1j}$

Algoritmo de Floyd

Para k ← 1 até m faça

  Para i ← 1 até n faça

    Para j ← 1 até n faça

      $p_{ij} \leftarrow \min \{ p_{ij}, p_{ik} + p_{kj} \}$

- Notas:
  - O último p_{ij} do algoritmo fornece a distância mínima de i a j
  - O último p_{ii} , i = 1, 2, ..., n é o comprimento do ciclo mínimo passando pelo nó i
  - Para recuperarmos a trajetória do caminho mínimo entre cada par (i,j) necessitamos construir outra seqüência de matrizes $C^k = (C^k_{ij})$, k = 1, 2, ..., n

Algoritmo de Floyd

- Construção da Matriz C
  - Inicialmente

    $C_{ij} \leftarrow i$, se $p_{ij} \neq \infty$

    $C_{ij} \leftarrow 0$, se $p_{ij} = \infty$
  - Na iteração k = l

    Se o nó l foi inserido entre i e j faça

    $C_{ij} \leftarrow C_{lj}$

Algoritmo de Floyd

- No final, um caminho mínimo (i, r_1, r_2, ..., r_q, j) pode ser recuperado usando a matriz C da seguinte forma

  $r_q \leftarrow C_{ij}$

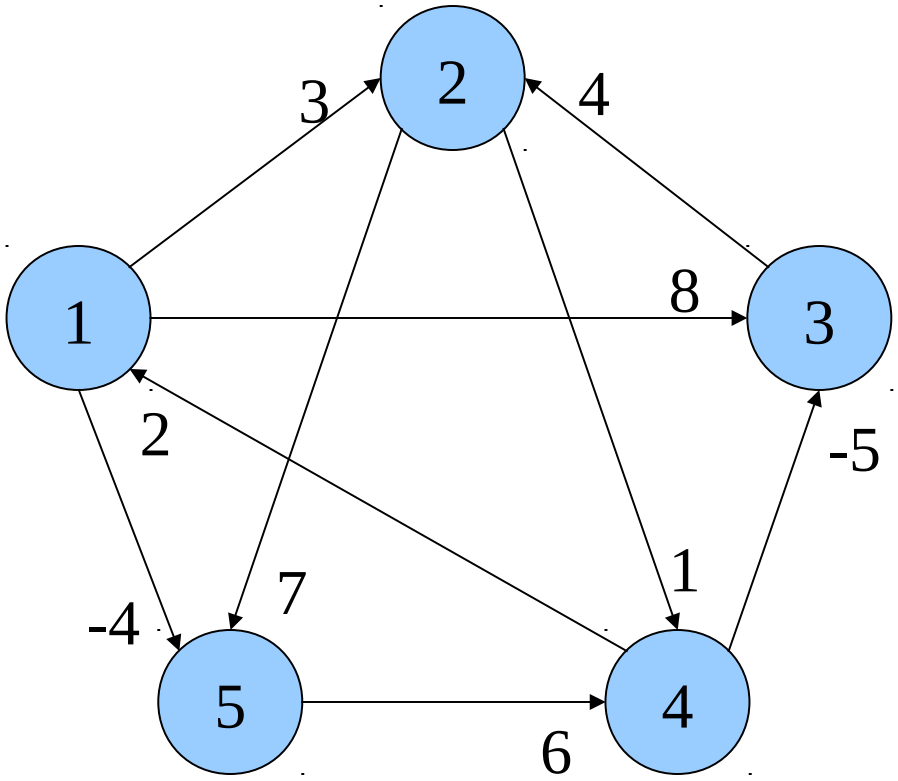
  $r_{q-1} \leftarrow C_{i,rq}$

  $r_{q-2} \leftarrow C_{i,rq-1}$

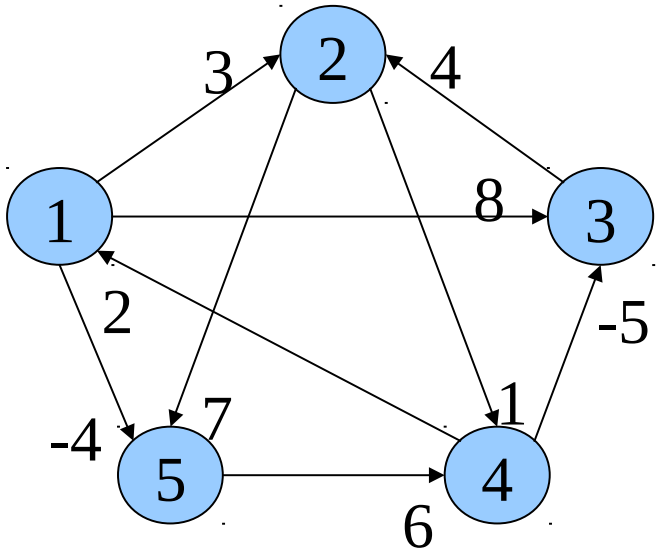
  ...

  $i \leftarrow r_1$

Algoritmo de Floyd



Algoritmo de Floyd



k= 0 (iniciação)

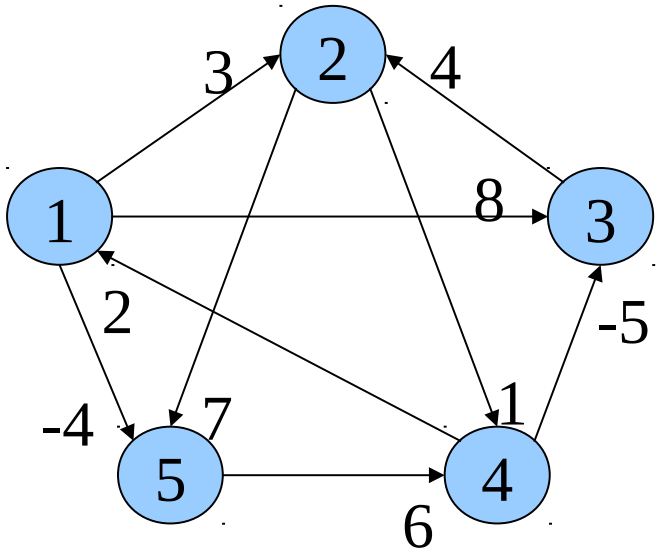
$p^{(0)}=$

	1	2	3	4	5
1	*	3	8	∞	-4
2	∞	*	∞	1	7
3	∞	4	*	∞	∞
4	2	∞	-5	*	∞
5	∞	∞	∞	6	*

$C^{(0)}=$

	1	2	3	4	5
1	0	1	1	0	1
2	0	0	0	2	2
3	0	3	0	0	0
4	4	0	4	0	0
5	0	0	0	5	0

Algoritmo de Floyd



k= 1

$p_{ij} \leftarrow \min \{ p_{ij}, p_{i1} + p_{1j} \}$

$p^{(1)} =$

	1	2	3	4	5
1	*	3	8	∞	-4
2	∞	*	∞	1	7
3	∞	4	*	∞	∞
4	2	5	-5	*	-2
5	∞	∞	∞	6	*

$C^{(1)} =$

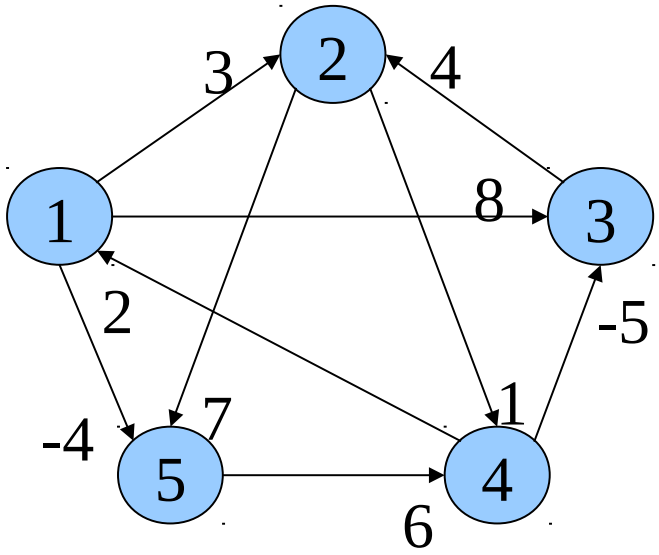
	1	2	3	4	5
1	0	1	1	0	1
2	0	0	0	2	2
3	0	3	0	0	0
4	4	1	4	0	1
5	0	0	0	5	0

$p_{12} \leftarrow \min \{ p_{12}, p_{11} + p_{12} \} = \min \{3, \infty\} = 3$

...

$p_{42} \leftarrow \min \{ p_{42}, p_{41} + p_{12} \} = \min \{\infty, 2+3\} = 5 \Rightarrow C_{42} = C_{12} = 1$

Algoritmo de Floyd



k= 2

$p_{ij} \leftarrow \min \{ p_{ij}, p_{i2} + p_{2j} \}$

$p^{(2)} =$

	1	2	3	4	5
1	*	3	8	4	-4
2	∞	*	∞	1	7
3	∞	4	*	5	11
4	2	5	-5	*	-2
5	∞	∞	∞	6	*

$C^{(2)} =$

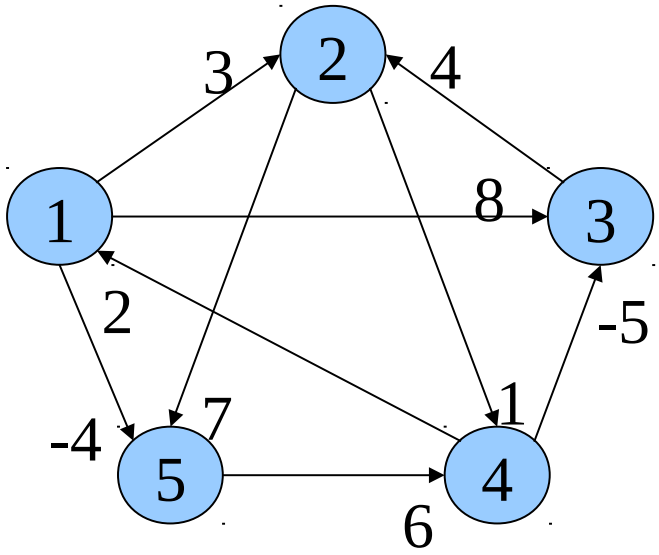
	1	2	3	4	5
1	0	1	1	2	1
2	0	0	0	2	2
3	0	3	0	2	2
4	4	1	4	0	1
5	0	0	0	5	0

$p_{12} \leftarrow \min \{ p_{12}, p_{12} + p_{22} \} = \min \{3, \infty\} = 3$

...

$p_{14} \leftarrow \min \{ p_{14}, p_{12} + p_{24} \} = \min \{\infty, 3+1\} = 4 \Rightarrow C_{14} = C_{24} = 2$

Algoritmo de Floyd



k= 3

$p_{ij} \leftarrow \min \{ p_{ij}, p_{i3} + p_{3j}\}$

$p^{(3)} =$

	1	2	3	4	5
1	*	3	8	4	-4
2	∞	*	∞	1	7
3	∞	4	*	5	11
4	2	-1	-5	*	-2
5	∞	∞	∞	6	*

$C^{(3)} =$

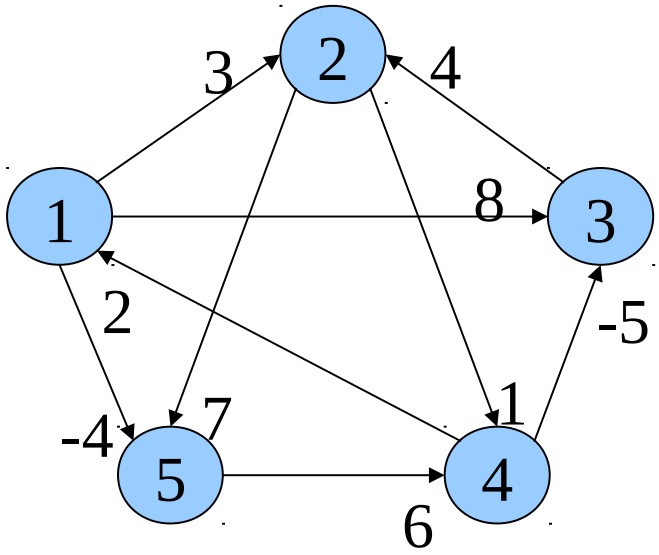
	1	2	3	4	5
1	0	1	1	2	1
2	0	0	0	2	2
3	0	3	0	2	2
4	4	3	4	0	1
5	0	0	0	5	0

$p_{12} \leftarrow \min \{ p_{12}, p_{13} + p_{32}\} = \min \{3,\infty\} = 3$

...

$p_{42} \leftarrow \min \{ p_{42}, p_{43} + p_{32}\} = \min \{5,-5+4\} = -1 \Rightarrow C_{42} = C_{32} = 3$

Algoritmo de Floyd



k= 4

$p_{ij} \leftarrow \min \{ p_{ij}, p_{i4} + p_{4j} \}$

$p^{(4)} =$

	1	2	3	4	5
1	*	3	-1	4	-4
2	3	*	-4	1	-1
3	7	4	*	5	3
4	2	-1	-5	*	-2
5	8	5	1	6	*

$C^{(4)} =$

	1	2	3	4	5
1	0	1	4	2	1
2	4	0	4	2	1
3	4	3	0	2	1
4	4	3	4	0	1
5	4	3	4	5	0

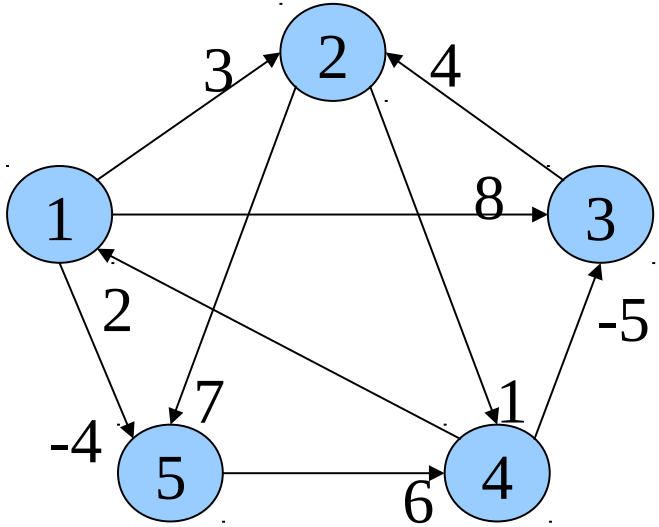
$p_{12} \leftarrow \min \{ p_{12}, p_{14} + p_{42} \} = \min \{3,3\} = 3$

$p_{13} \leftarrow \min \{ p_{13}, p_{14} + p_{43} \} = \min \{8,4-5\} = -1 \Rightarrow C_{13} = C_{43} = 4$

...

$p_{25} \leftarrow \min \{ p_{25}, p_{24} + p_{45} \} = \min \{7,1-2\} = -1 \Rightarrow C_{25} = C_{45} = 1$

Algoritmo de Floyd



k= 5

$p_{ij} \leftarrow \min \{ p_{ij}, p_{i5} + p_{5j} \}$

$p^{(5)} =$

	1	2	3	4	5
1	*	1	-3	2	-4
2	3	*	-4	1	-1
3	7	4	*	5	3
4	2	-1	-5	*	-2
5	8	5	1	6	*

$C^{(5)} =$

	1	2	3	4	5
1	0	3	4	5	1
2	4	0	4	2	1
3	4	3	0	2	1
4	4	3	4	0	1
5	4	3	4	5	0

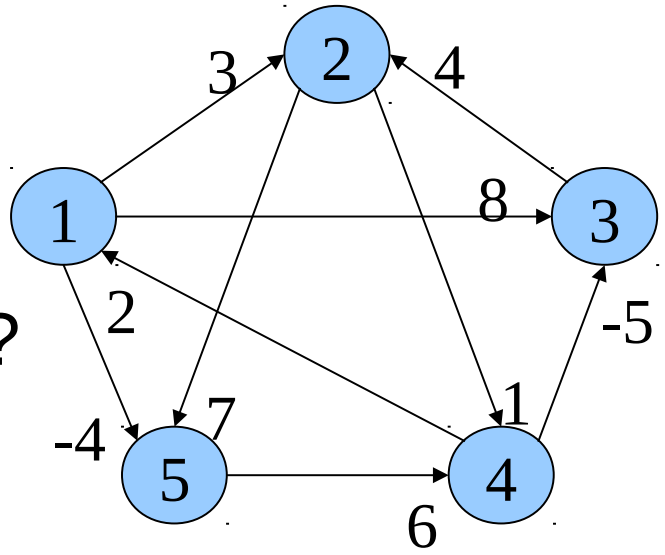
$p_{12} \leftarrow \min \{ p_{12}, p_{15} + p_{52} \} = \min \{3,-4+5\} = 1 \Rightarrow C_{12} = C_{52} = 3$

$p_{13} \leftarrow \min \{ p_{13}, p_{15} + p_{53} \} = \min \{-1,-4+1\} = -3 \Rightarrow C_{13} = C_{53} = 4$

...

Algoritmo de Floyd

Qual o menor caminho entre 5 e 2?



$p^{(5)} =$

	1	2	3	4	5
1	*	1	-3	2	-4
2	3	*	-4	1	-1
3	7	4	*	5	3
4	2	-1	-5	*	-2
5	8	5	1	6	*

$C^{(5)} =$

	1	2	3	4	5
1	0	3	4	5	1
2	4	0	4	2	1
3	4	3	0	2	1
4	4	3	4	0	1
5	4	3	4	5	0

$r_q = C_{ij} = C_{52} = 3$

$r_{q-1} = C_{iq} = C_{53} = 4$

$i = C_{iq-1} = C_{54} = 5$

$5 \rightarrow 4 \rightarrow 3 \rightarrow 2$

Próxima Aula...

- Fluxo Máximo em Redes
  - Introdução
  - Algoritmo de Fork-Fulkerson
  - DMKM