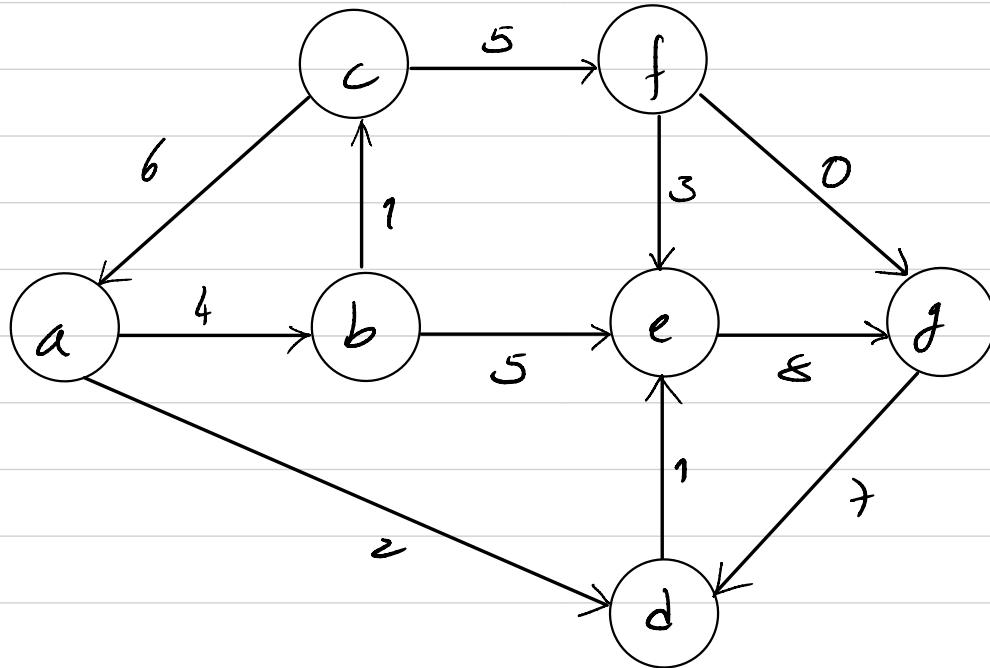

Prática 4

Caminhos mais curtos de origem

mínica



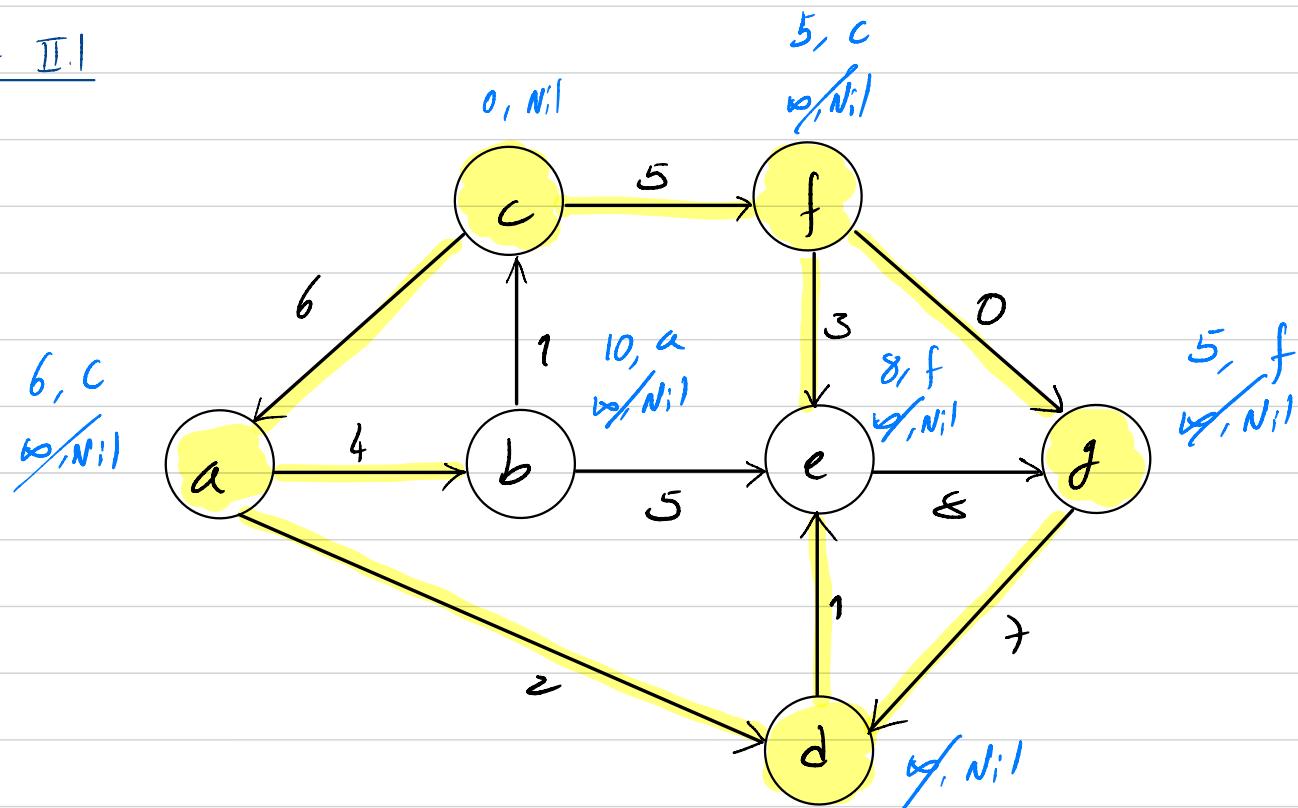
T1 06/07 II.1



Fonre: c

(até nostenrem apenas 2 vértices
na fila de prioridade)

TI 06/07 II.1



a: $6, C$

b: $10, a$

c: $0, N_i$

d: $8, a$

e: $8, f$

f: $5, C$

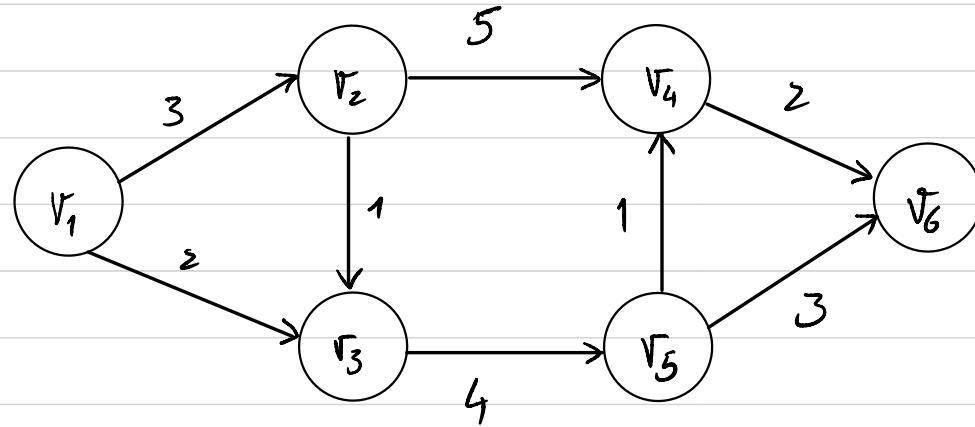
g: $5, f$

ω/N_i

$12, g$

$8, a$

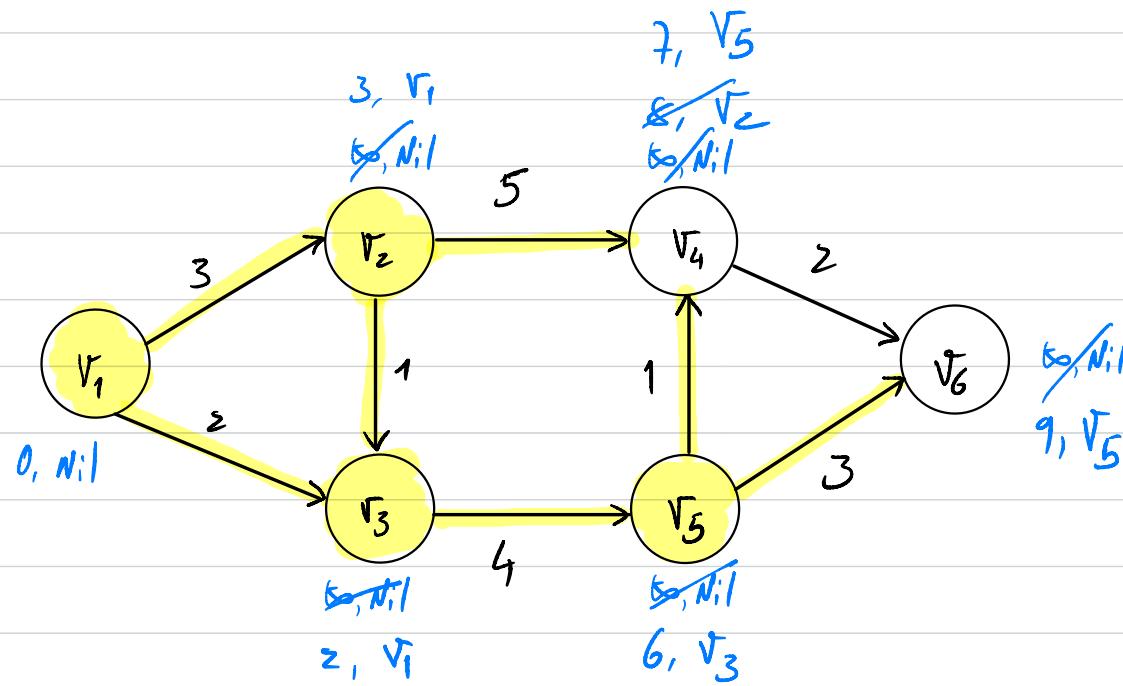
TI 08/09 II.3 -



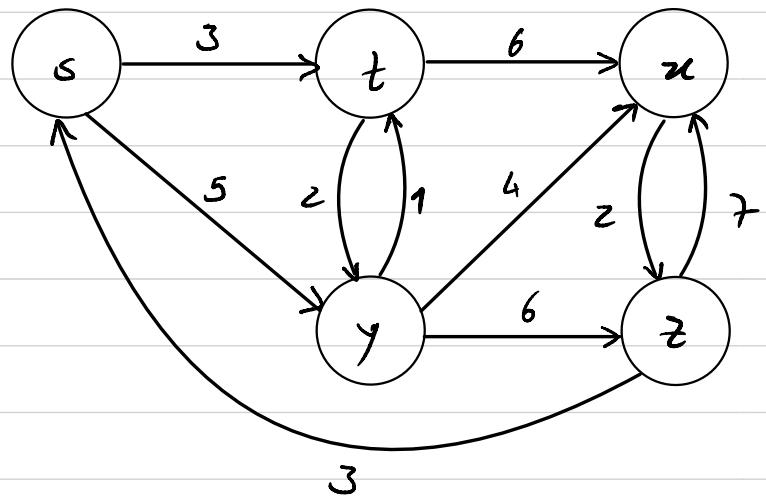
Fonkt: v_1

(at v_5 sei relaxed)

TI 08/09 II.3 -

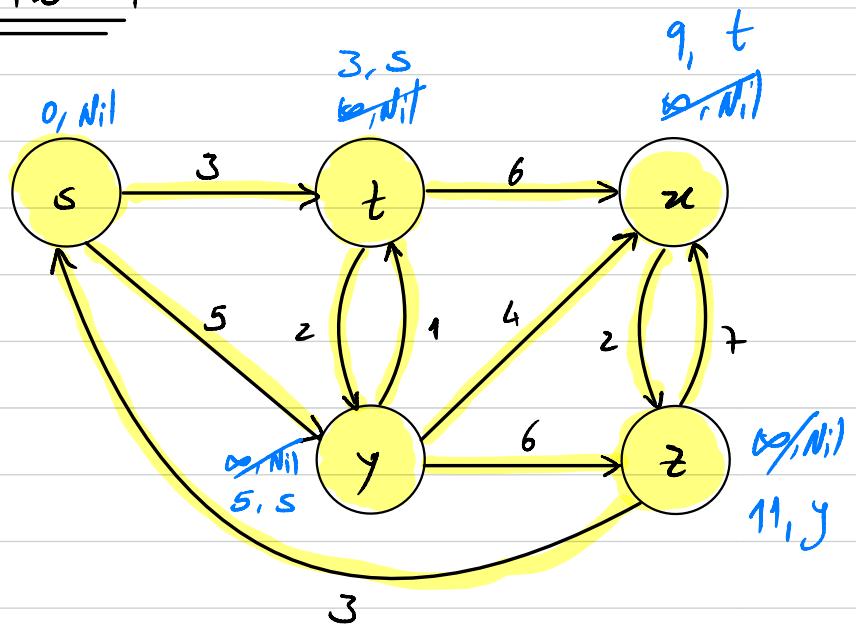


Ex. 24.3 - 1



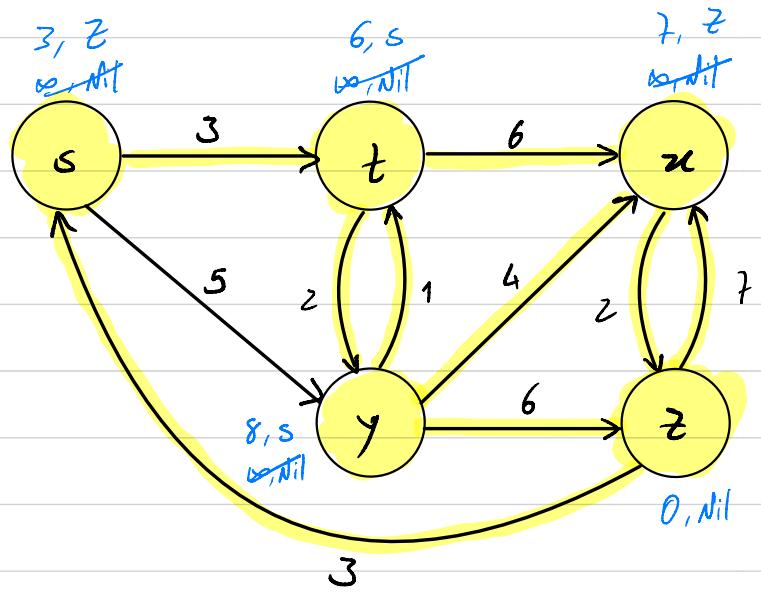
Sources: s, z

Ex. 24.3 - 1



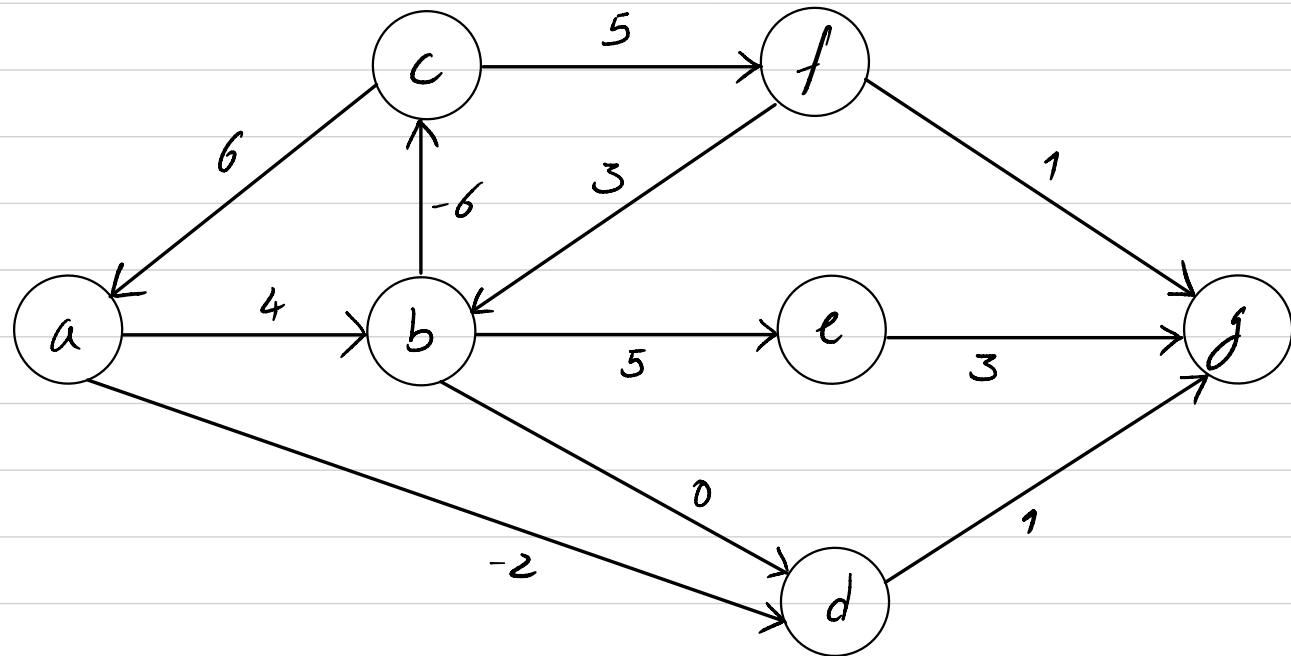
Sources : s, z

Ex. 24.3 - 1



Sources: s, z

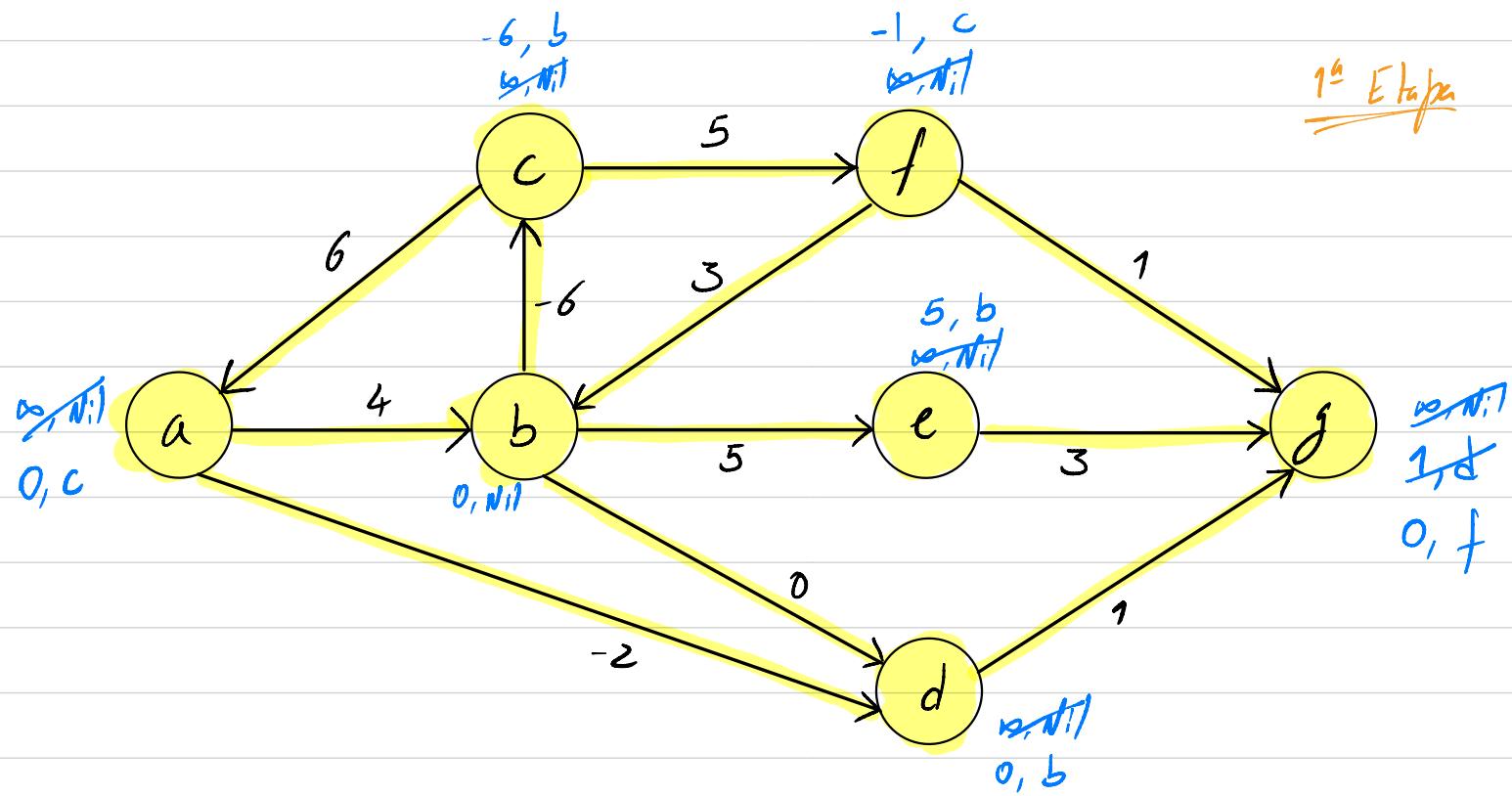
R1 06/07 II.1



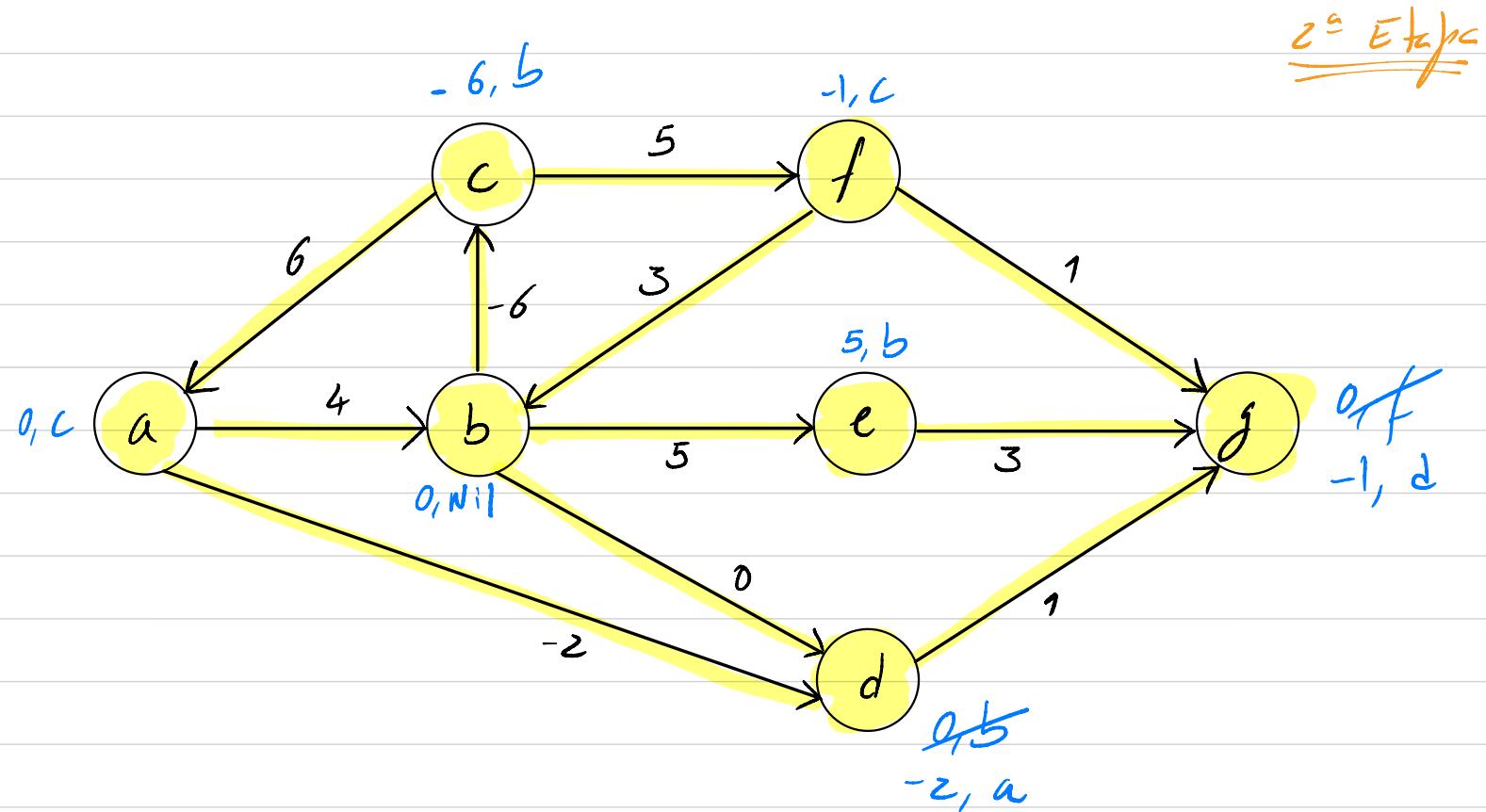
Fonte: b

↳ Ordem lexicográfica

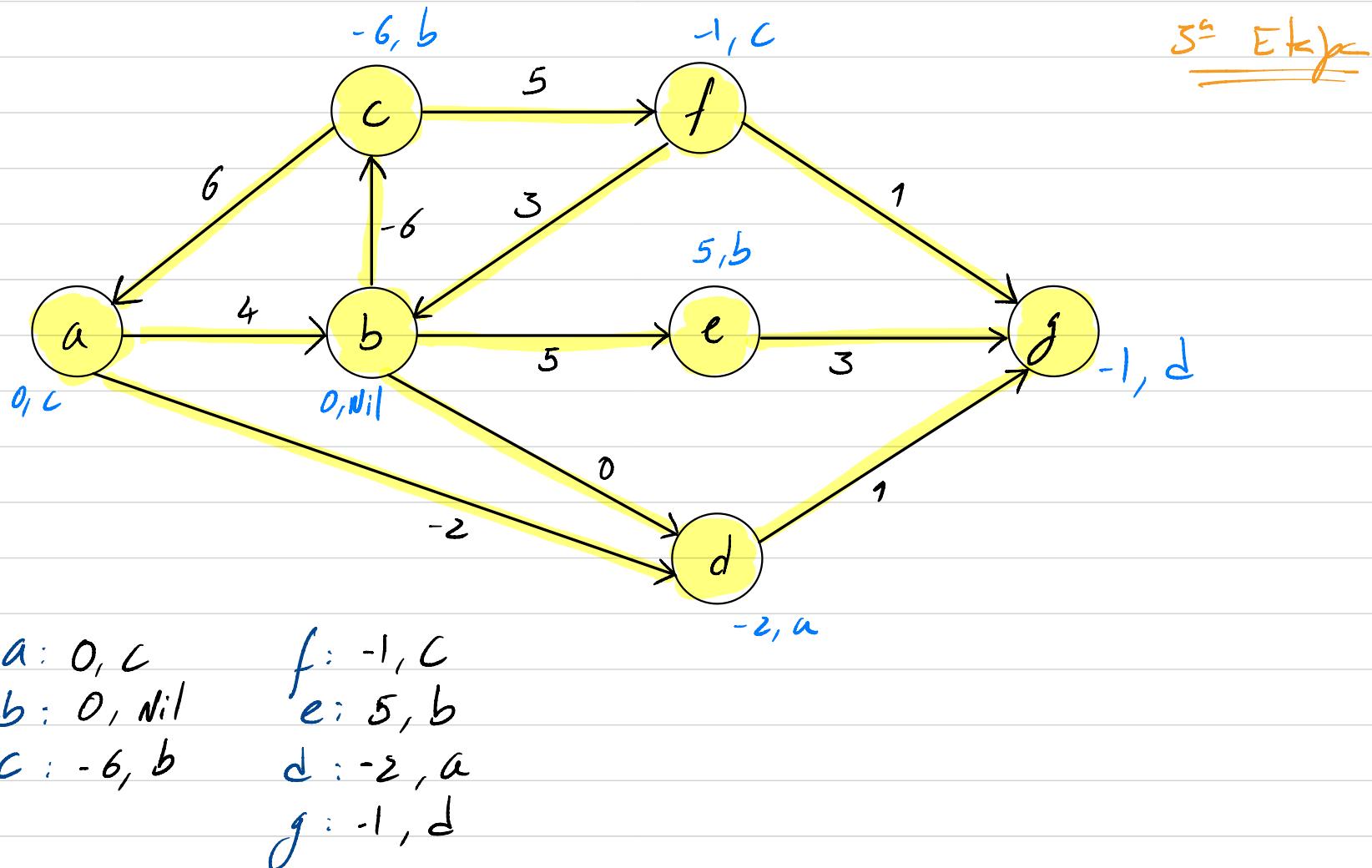
R1 06/07 II.1



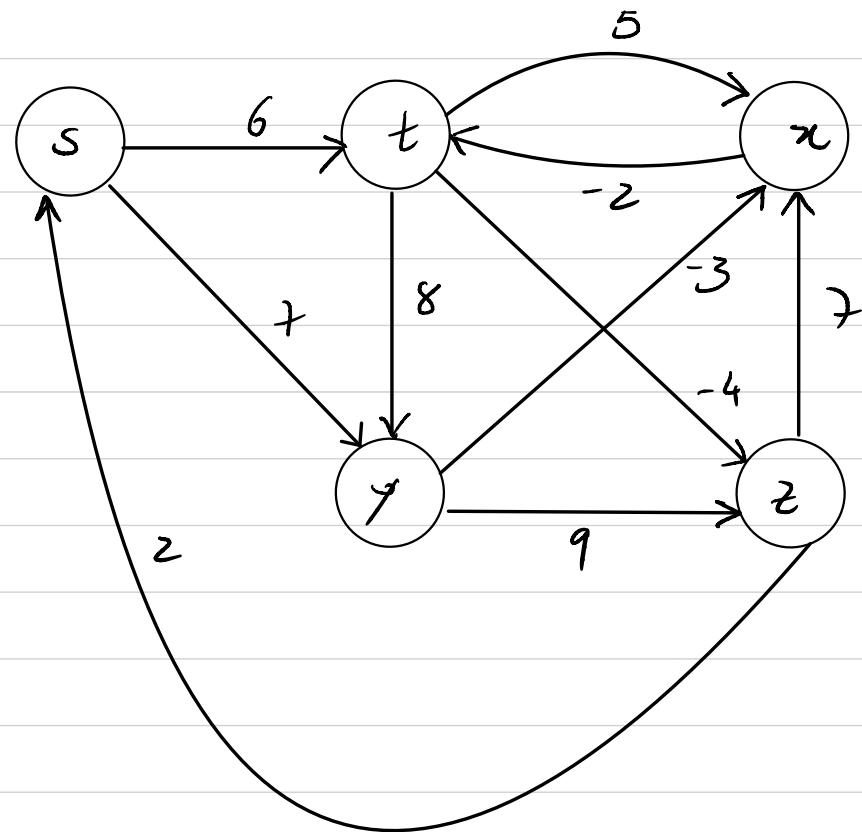
R1 06/07 II.1



R1 06/07 II.1



Ex 24.1-1



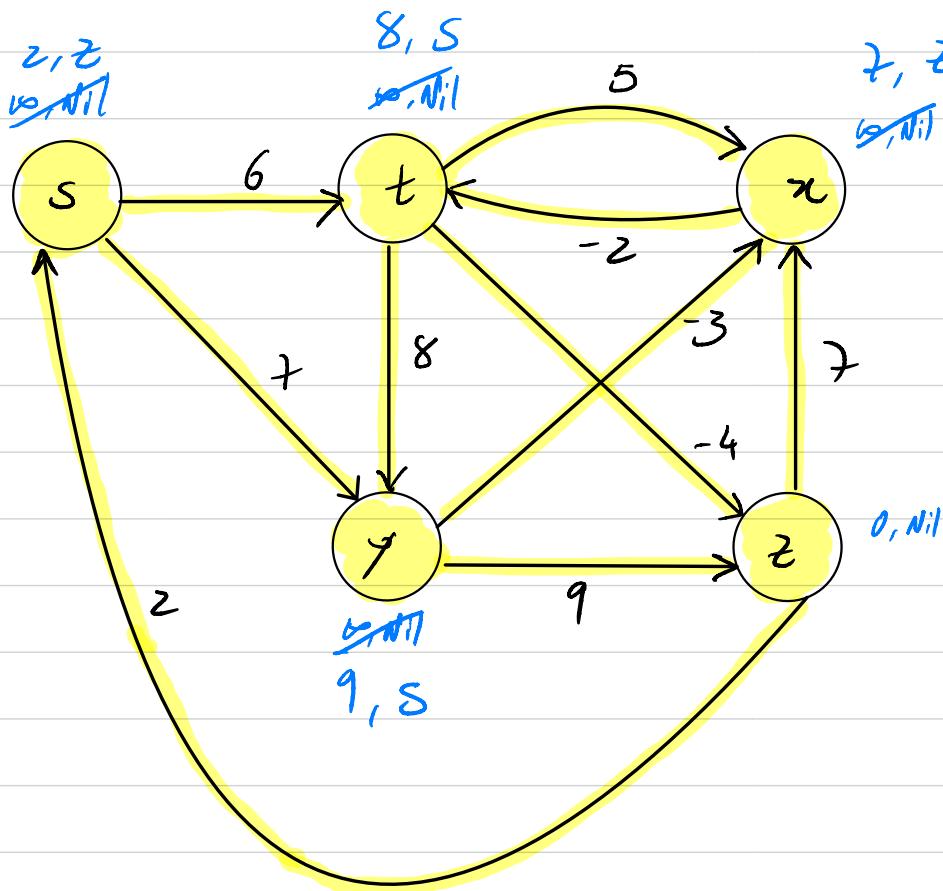
Order of relaxations:

t
n
y
z
s

Source : z

Ex 24.1-1

1^a Etapa



Order of relaxations:

t

n

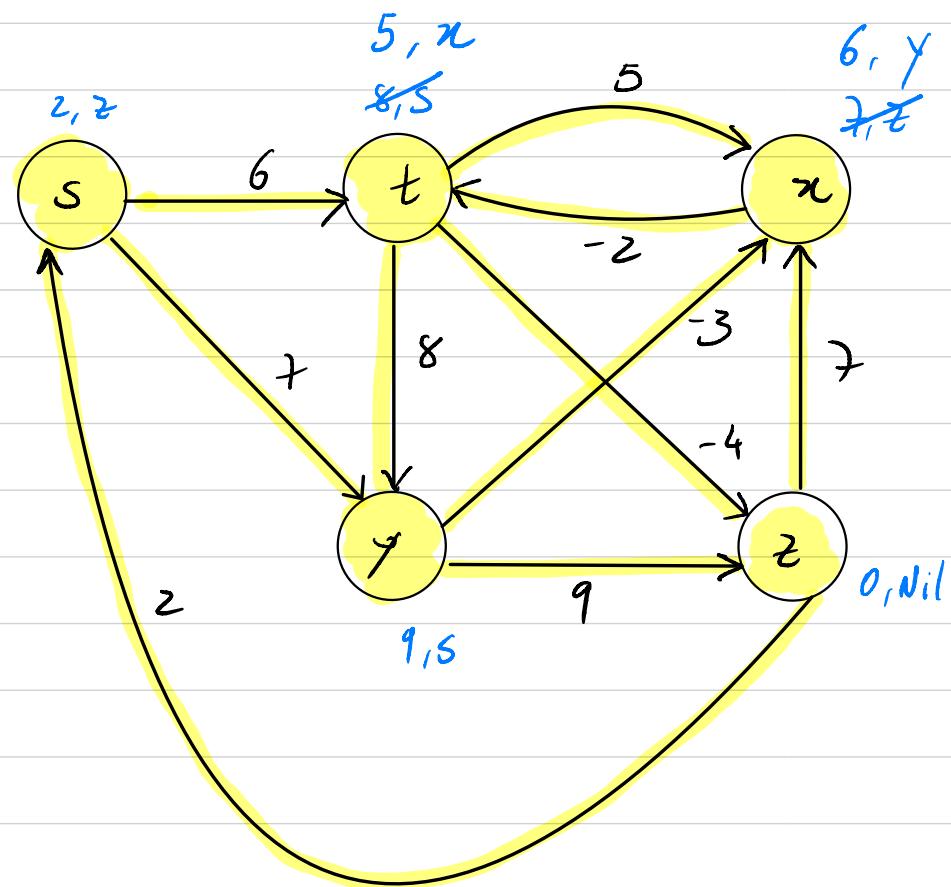
y
z

s

Source: z

Ex 24.1-1

z^o Etapa



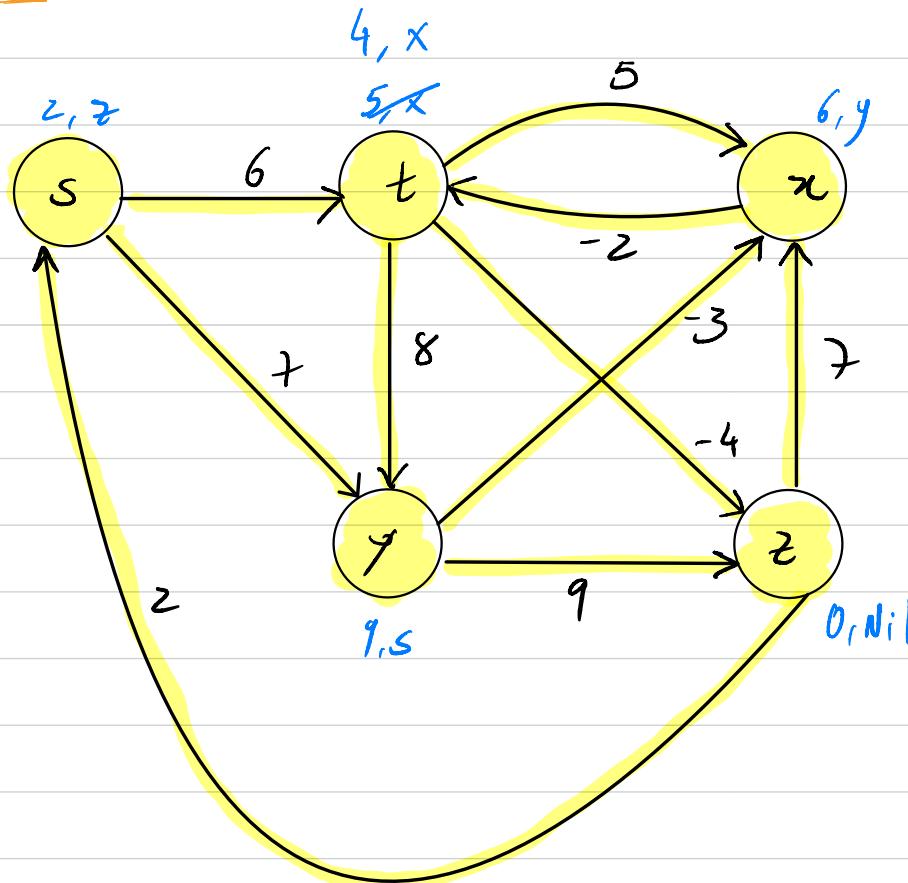
Order of relaxations:

t
n
y
z
s

Source : z

Ex 24.1-1

3rd Etapa

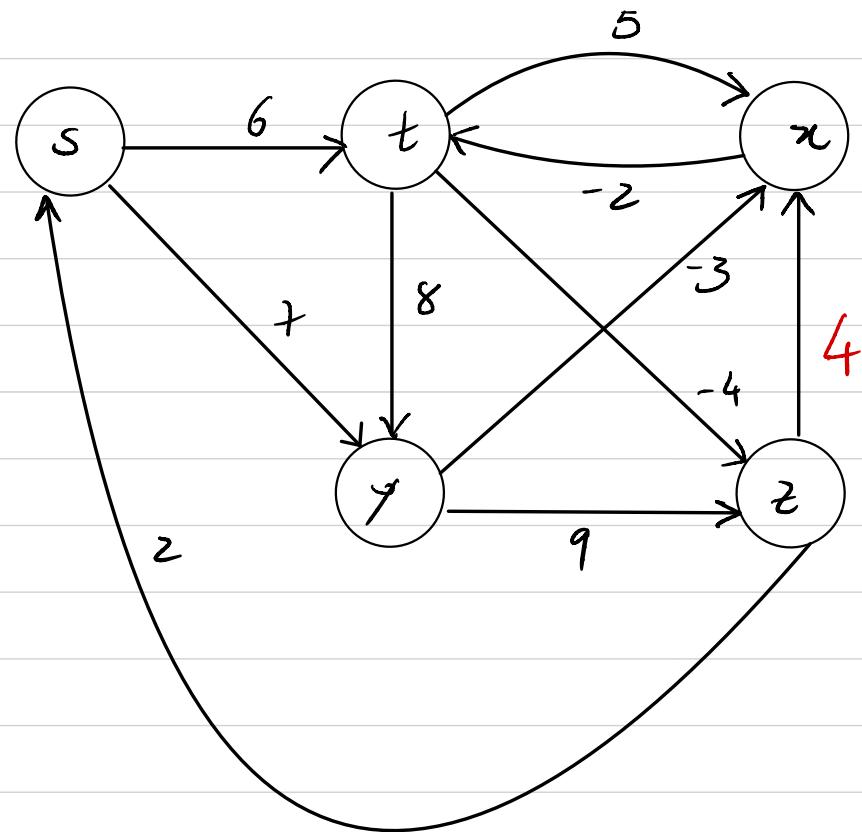


Order of relaxations:

t
n
y
z
s

Source : z

Ex 24.1-1

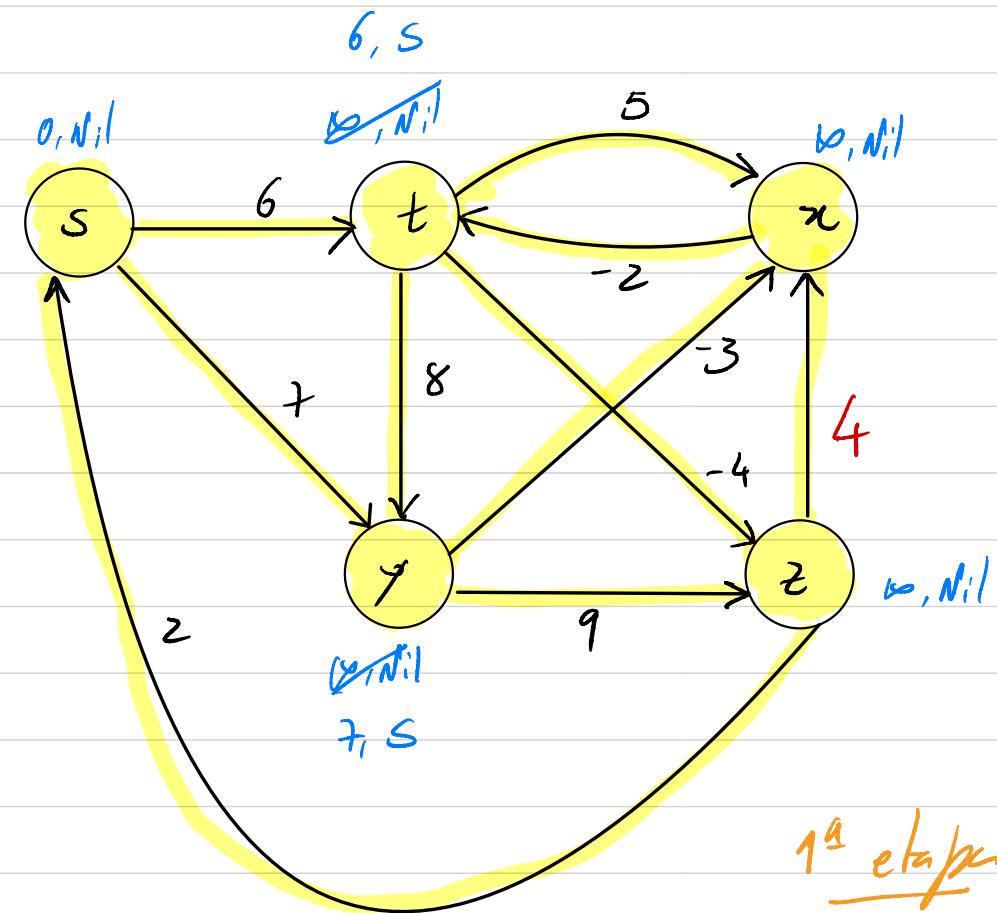


Order of relaxations:

t
n
y
z
s

Source : S

Ex 24.1-1

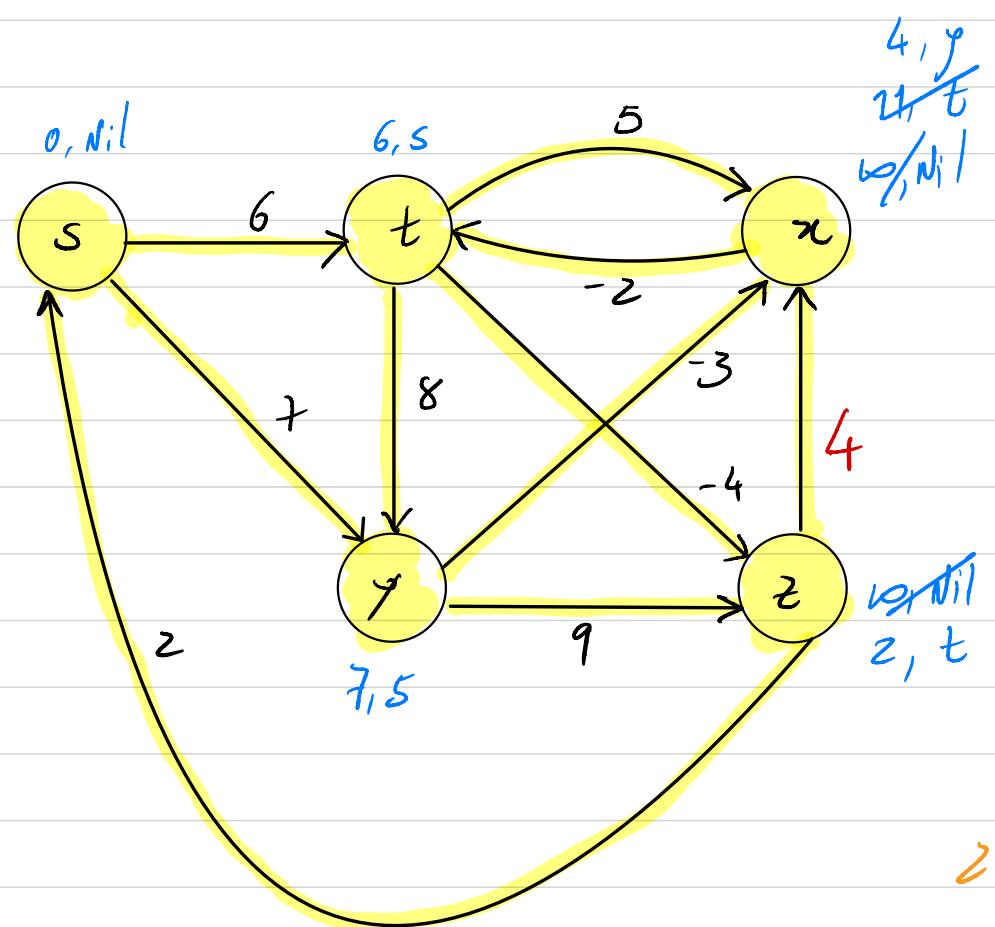


Order of relaxations:

t
 n
 y
 z
 s

Source : s

Ex 24.1-1



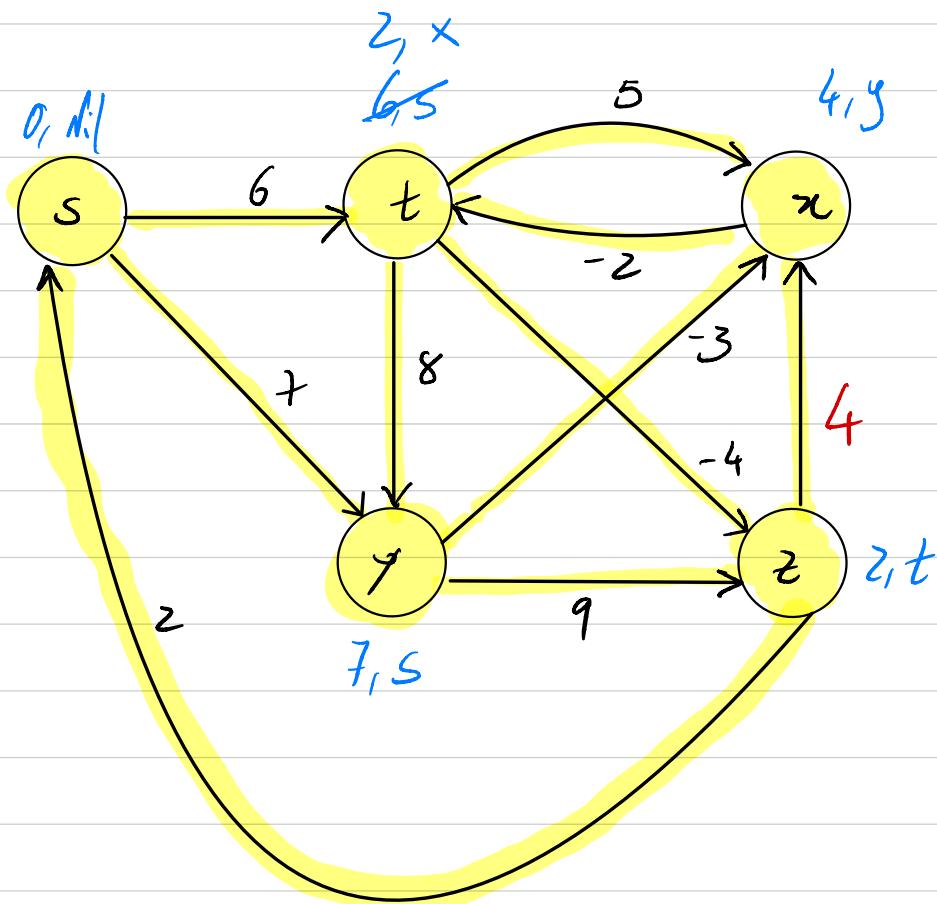
Order of relaxations:

2
n
y
z
s

Sorce : S

\mathcal{L}^S ekpc

Ex 24.1-1



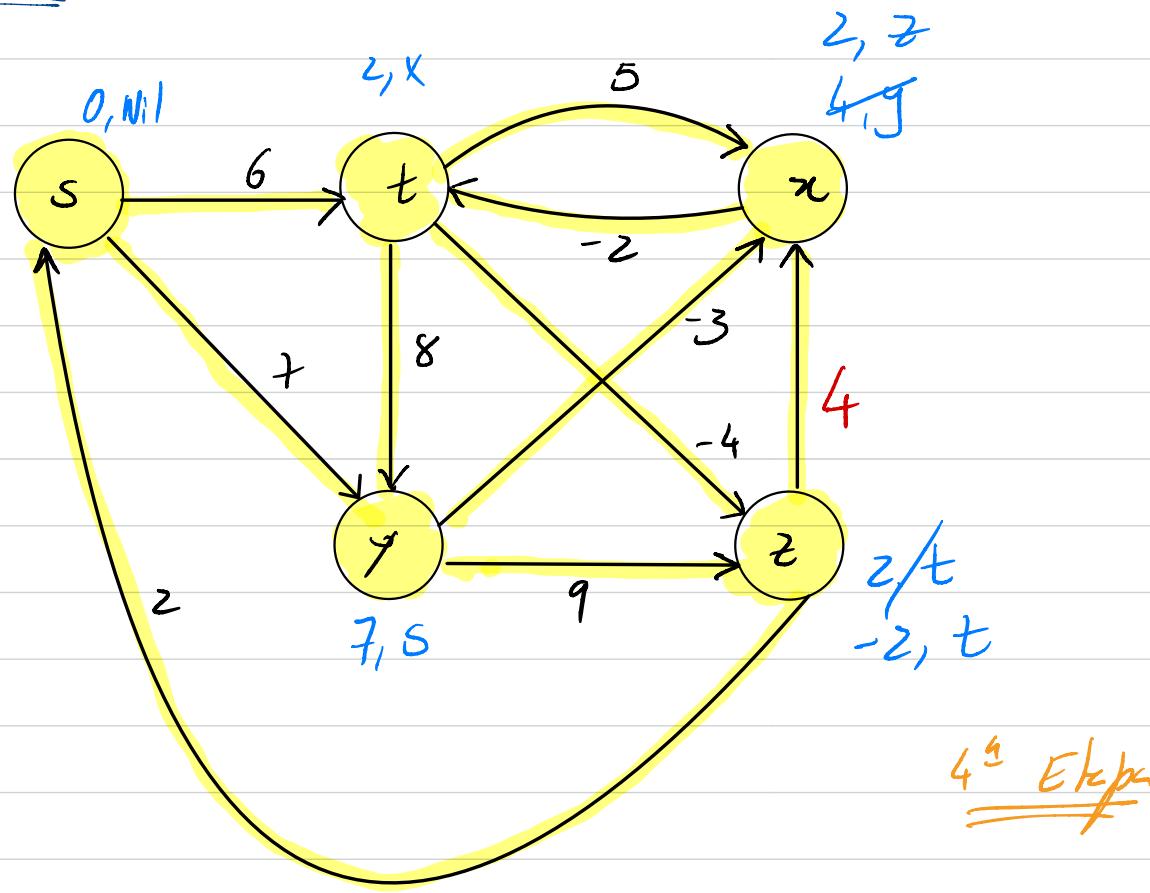
Order of relaxations:

t
n
y
z
s

Source : s

3rd Elap

Ex 24.1-1

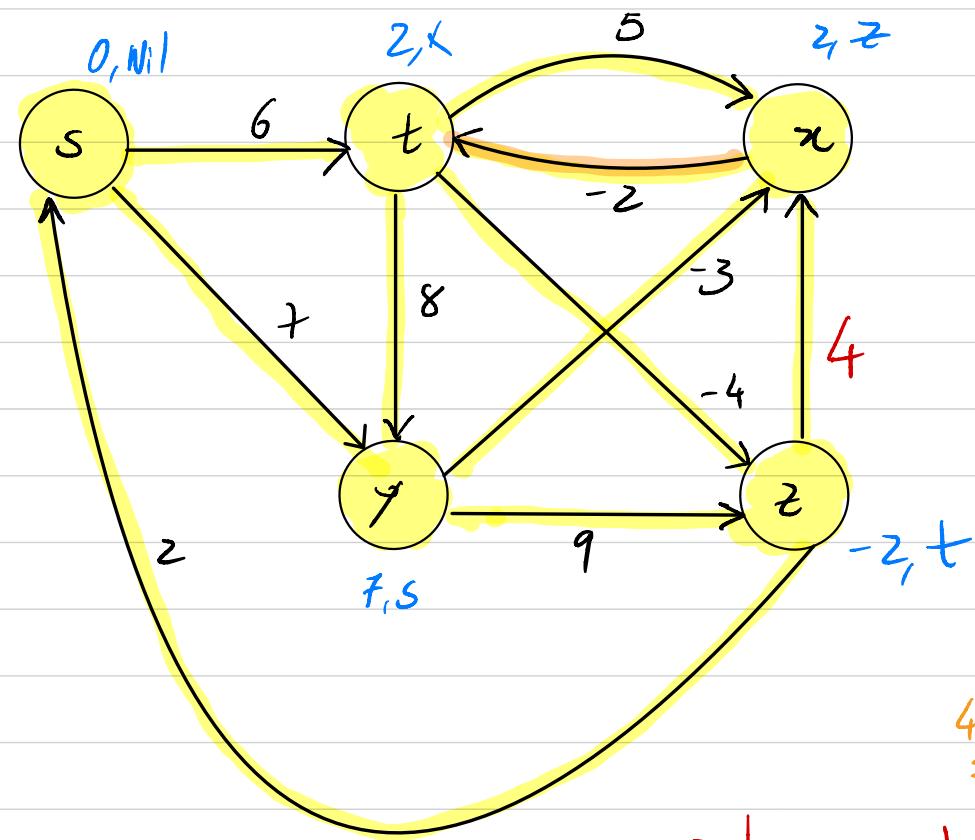


Order of relaxations:

t
n
y
z
s

Source: s

Ex 24.1-1



Order of relaxations:

t
n
y
z
s

Source: s

4th Eliza

• Detectar el ciclo negativo:

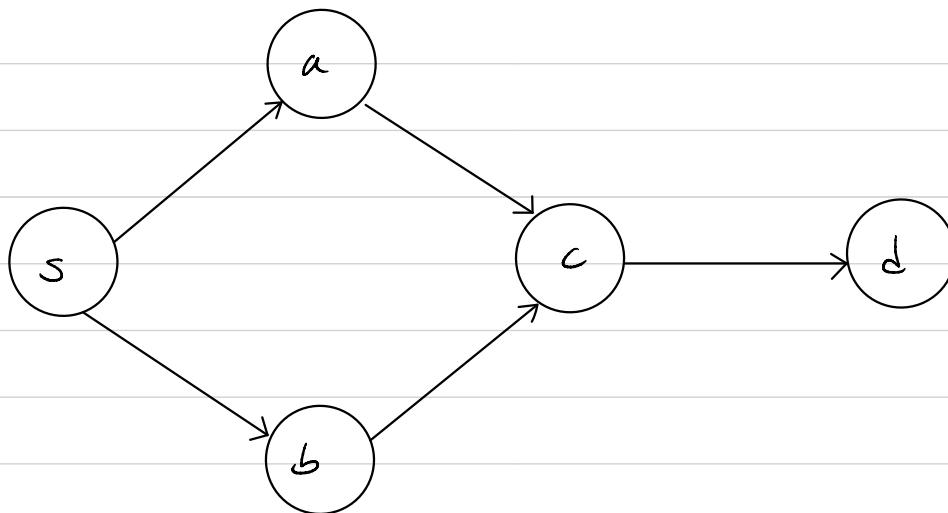
- Encuentra arco (u, v) tal que:
 $d[v] > d[u] + w(u, v)$

Anco (x, t)

$$\underbrace{d[x] + w(x, t)}_0 < \underbrace{d[t]}_2 \quad \text{X}$$

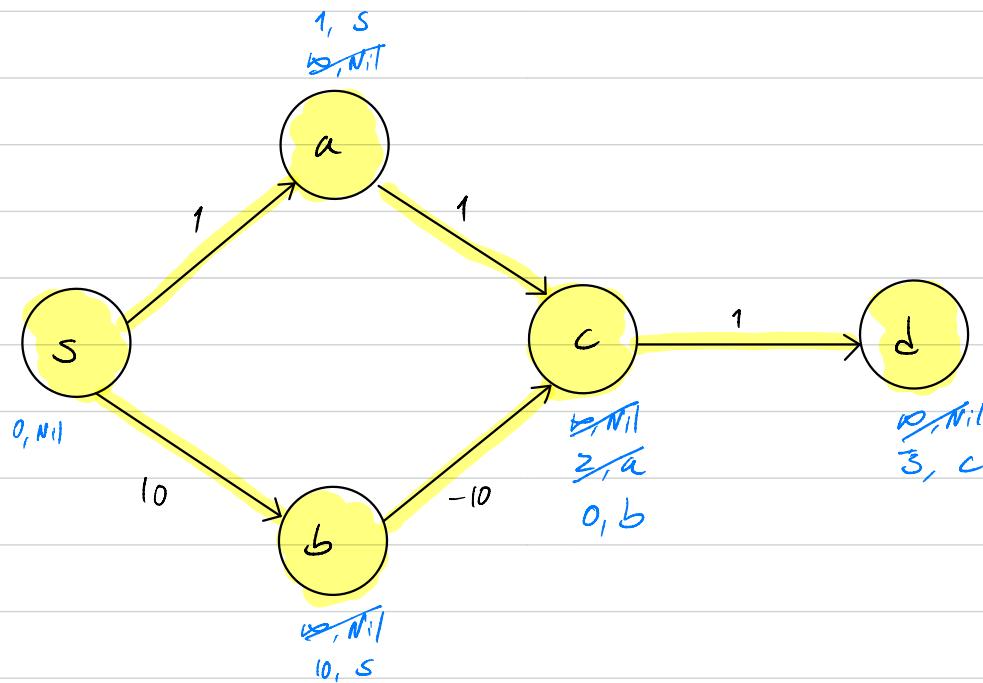
24.3.2

- Algoritmo de Dijkstra não funciona quando o grafo tem arestas com pesos negativos



24.3.2

- Algoritmo de Dijkstra não funciona quando o grafo tem arestas com pesos negativos



Algoritmo de Dijkstra - Invariante

$$\forall v \in S. \delta(s, v) = v.d$$

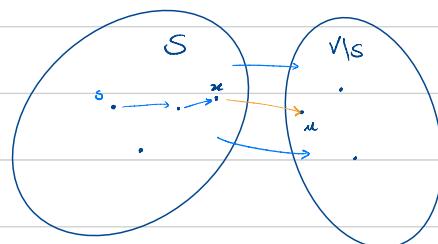
$$\wedge S = V \setminus Q$$

- 2 casos a considerar:

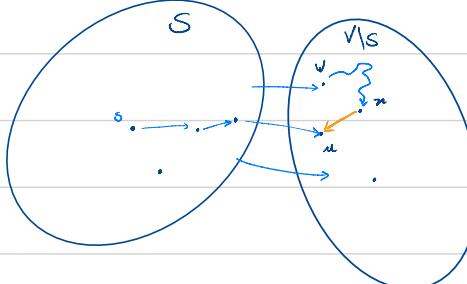
I $x \in S$

- m é tal que:

$$m.d = \min \{ u.d \mid u \in V \setminus Q \}$$



II $x \notin S$



- Há que provar que:

$$m.d = \delta(s, u)$$

- Suponhamos que $m.d \neq \delta(s, u)$

- Existe um caminho p que liga s a u tal que $s \xrightarrow{p} u \wedge w(p) = m.d$

- Seja x o predecessor de u no caminho mais curto entre s e u .

Algoritmo de Dijkstra - Invariante

$$\forall v \in S. \delta(s, v) = v.d$$

$$\wedge S = V \setminus Q$$

• m é tal que:

$$m.d = \min \{ u.d \mid u \in V \setminus Q \}$$

• Há que provar que:

$$m.d = \delta(s, m)$$

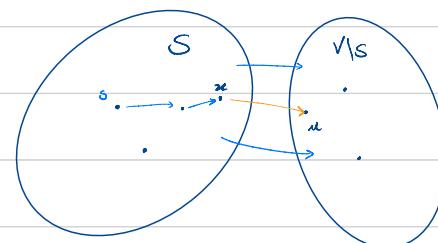
• Suponhamos que $m.d \neq \delta(s, m)$

- Existe um caminho p que liga s a m tal que $s \xrightarrow{p} m \wedge w(p) = m.d$

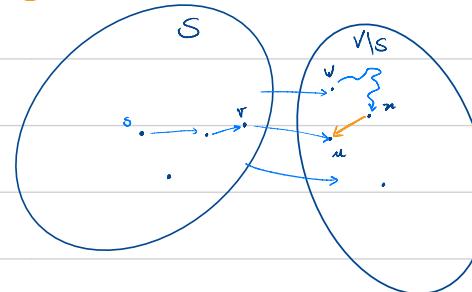
- Seja x o predecessor de m no caminho mais curto entre s e m .

- 2 casos a considerar:

I $x \in S$



II $x \notin S$



• O arco (x, m) foi relaxado

de modo que temos que:

$$d[m] \leq d[x] + w(x, m)$$

$$\Rightarrow d[m] \leq \delta(s, x) + w(x, m)$$

$$\Rightarrow d[m] \leq \delta(s, m)$$

$$\Rightarrow d[m] = \delta(s, m)$$

$$\delta(s, m) = \delta(s, v) + \delta(v, x) + w(x, m)$$

$$= d[w] + \underbrace{\delta(w, x) + w(x, m)}_{\geq 0}$$

$$\delta(s, m) \geq d[w]$$

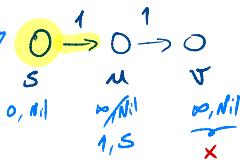
$$d[m] > \delta(s, m)$$

$$d[m] > d[w]$$

∴

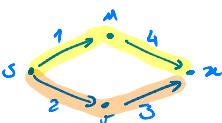
R1 08/09 II.2 Considere os algoritmos para o cálculo de caminhos mais curtos. Indique se cada uma das seguintes afirmações é verdadeira (V) ou falsa (F).

1. O algoritmo de Bellman-Ford permite detectar ciclos negativos.
2. Se a relaxação dos arcos de um grafo dirigido e acíclico for efectuada de acordo com a ordenação topológica dos respectivos vértices, é possível determinar os caminhos mais curtos de fonte única em tempo $\Theta(V + E)$.
3. No algoritmo de Dijkstra, quando um vértice u é extraído da fila de prioridade, $d[u]$ e $\pi[u]$ já têm o respectivo valor final, mesmo em grafos contendo arcos com peso negativo.
4. O algoritmo de Dijkstra produz os valores finais correctos, mesmo que o ciclo principal seja executado apenas $|V| - 2$ vezes.
5. Se num grafo existir mais do que um componente fortemente ligado (SCC), têm obrigatoriamente que existir dois vértices u e v , tal que $\delta(u, v) = \infty$.
6. Os caminhos mais curtos obedecem sempre à desigualdade triangular.
7. Em grafos em que os pesos dos arcos sejam todos diferentes e inteiros positivos, existe apenas um caminho mais curto entre qualquer par de vértices.
8. O tempo de execução do algoritmo de Bellman-Ford é $O(VE^2)$.



R1 08/09 II.2 Considere os algoritmos para o cálculo de caminhos mais curtos. Indique se cada uma das seguintes afirmações é verdadeira (V) ou falsa (F).

1. O algoritmo de Bellman-Ford permite detectar ciclos negativos. ✓
2. Se a relaxação dos arcos de um grafo dirigido e acíclico for efectuada de acordo com a ordenação topológica dos respectivos vértices, é possível determinar os caminhos mais curtos de fonte única em tempo $\Theta(V + E)$. ✓
3. No algoritmo de Dijkstra, quando um vértice u é extraído da fila de prioridade, $d[u]$ e $\pi[u]$ já têm o respectivo valor final, mesmo em grafos contendo arcos com peso negativo. F
4. O algoritmo de Dijkstra produz os valores finais correctos, mesmo que o ciclo principal seja executado apenas $|V| - 2$ vezes. F
5. Se num grafo existir mais do que um componente fortemente ligado (SCC), têm obrigatoriamente que existir dois vértices u e v , tal que $\delta(u, v) = \infty$. ✓
6. Os caminhos mais curtos obedecem sempre à desigualdade triangular. ✓
7. Em grafos em que os pesos dos arcos sejam todos diferentes e inteiros positivos, existe apenas um caminho mais curto entre qualquer par de vértices. F
8. O tempo de execução do algoritmo de Bellman-Ford é $O(VE^2)$. T



24.3-3

Dijkstra(G, s)

Initialize Single Source(G, s)

$Q := \text{new MinQueue}(G.V)$

$S = \{\}$

while ($! Q.\text{empty}()$) {

$u := Q.\text{extractMin}();$

$S := S \cup \{u\};$

 for each $v \in G.\text{Adj}[u]$

 Relax(t, w, u, v)

}

$|Q| > 1$

- O algoritmo ainda estávnia conecto?

Invariante do algoritmo de Dijkstra:

- $\forall u \in S. \quad d[u] = \delta(s, u)$
 $\wedge S = V \setminus Q$

- Seja u o vértice q ficou em Q :
 $\forall v \in V \setminus \{u\}. \quad d[v] = \delta(s, v)$

- A operação de relax das arestas q partem de u não altera o valor de d para nenhum vértice em V .

24.3-4

- Verificam se caminhos mais curtos estão bem calculados

- Input: $d[J]$ e $\pi[J]$

- ① Verificar que $d[J]$ e $\pi[J]$ estão corretos na origem

- $d[s] = 0 \wedge \pi[s] = \text{Nil}$

- $O(1)$

- ② Verificar se o subgrafo induzido por π é uma árvore

- $G_\pi = (V_\pi, E_\pi)$

$$V_\pi = \left\{ v \mid \pi[v] \neq \text{Nil} \right\} \cup \{s\}$$

$$E_\pi = \left\{ (\pi[v], v) \mid \pi[v] \neq \text{Nil} \right\}$$

- $\text{DFS-Visit}(G_\pi, s)$

encontra todos os vértices em V_π e não encontra nenhum arco para baixo

- $O(V+E)$

24.3-4

- Verificam se caminhos mais curtos estão bem calculados

- Input: $d[J]$ e $\pi[J]$

(3) Verificam que $d[J]$ e $\pi[J]$ são consistentes:

- $\forall v \in V. \pi[v] \neq \text{Nil} \Rightarrow d[v] = d[\pi[v]] + w(\pi[v], v)$
- $\forall v \in V \setminus \{s\}. \pi[v] = \text{Nil} \Leftrightarrow d[v] = \infty$
- $O(V)$

(4) Verificam se d satisfaz a desigualdade triangular

- $\forall (u, v) \in E. d[v] \leq d[u] + w(u, v)$
- $O(V+E)$

26.3 - 4

- Prova do que o procedimento proposto está correto

- Suponhamos que existe $m \in \mathbb{N}$ que $d[m] \neq S(s, m)$

Por ① sabemos que $m \neq s$.

Admitimos, sem perda de generalidade, $\exists d[\pi[m]] = S(s, \pi[m])$

- Há dois casos a considerar:

① $\pi[m]$ é um predecessor de m num caminho mais curto

\bar{g} liga s a m :

$$\begin{aligned}d[m] &= d[\pi[m]] + w(\pi[m], m) \\&= S(s, \pi[m]) + w(\pi[m], m) \\&= S(s, m) \quad \therefore\end{aligned}$$

26.3 - 4

• Prova do que o procedimento proposto está correto

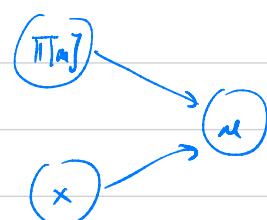
• Suponhamos que existe $m \neq l$, que $d[s] \neq S(s, m)$

Por ① sabemos que $m \neq s$.

Admitimos, sem perda de generalidade, q $d[\pi[m]] = S(s, T[m])$

• Há dois casos a considerar:

② $\pi[m]$ não é predecessor de m num caminho mais curto que liga s a m . Seja x esse predecessor. Admitimos, sem perda de generalidade, que $d[x]$ está bem calculado.



$$\begin{aligned} d[m] &= d[\pi[m]] + w(\pi[m], m) \\ &= S(s, \pi[m]) + w(\pi[m], m) \\ &> S(s, m) \end{aligned}$$

$$\begin{aligned} d[m] &\leq d[x] + w(x, m) \\ &= S(s, x) + w(x, m) \\ &= S(s, m) \end{aligned}$$

} $\therefore =$

24.1-4

Bellman Ford'(G, s)

Initialize Single Source (G, s)

for $i=1$ to $|G.V|-1$

| for each $(u, v) \in G.E$

| | relax($u, v, G.u$)

let marked = \emptyset

for each $(u, v) \in G.E$

| if $v.d > u.d + w(u, v)$

| | marked = marked $\cup \{v\}$

for each $v \in \text{marked}$

mark all nodes reachable from v with ∞

modified DFS