Prática 10

Q1 $(\underline{21.3-1})$

for $i = 1$ to 16
    MakeSet(x_i) ⎤ I

for $i = 1$ to 15 by 2
    Union(x_i, x_{i+1}) ⎤ II

for $i = 1$ to 13 by 4
    Union(x_i, x_{i+2}) ⎤ III

Union(x_1, x_5)
Union(x_{11}, x_{13}) } IV

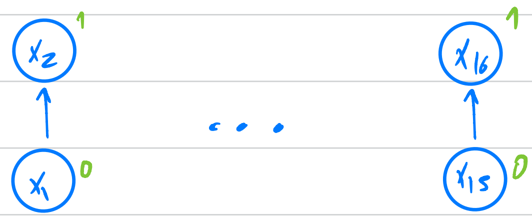
Union(x_1, x_{10}) V

FindSet(x_2) VI

FindSet(x_9) VII

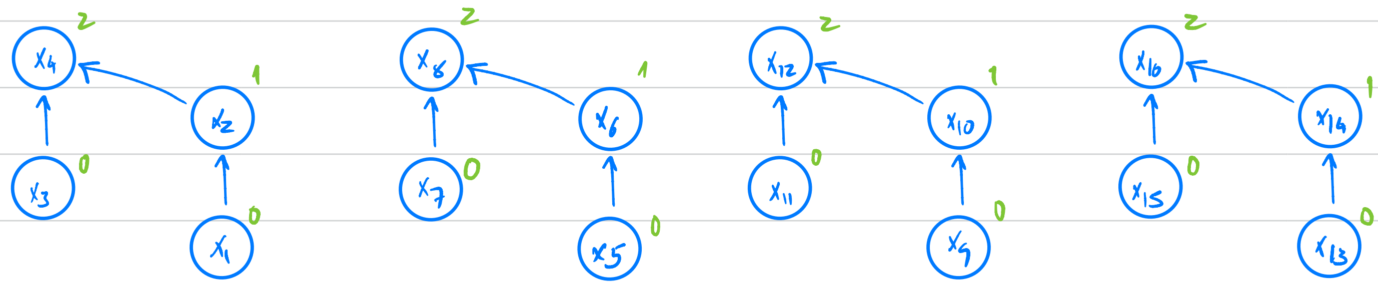
I

II



III

Union(x_1, x_3)
Union(x_5, x_7)
⋮
Union(x_{13}, x_{15})



Q1 (21.3-1)

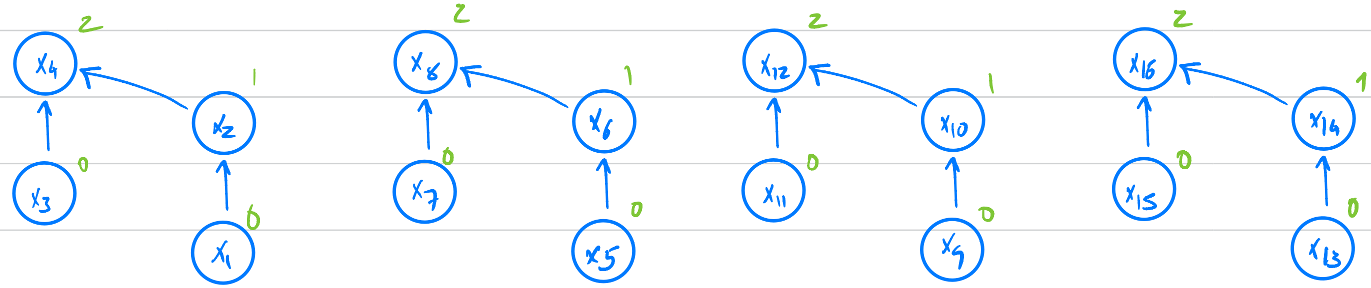
Union(x_1, x_5)
Union(x_{11}, x_{13})
} IV

Union(x_1, x_{10}) V

FindSet(x_2) VI

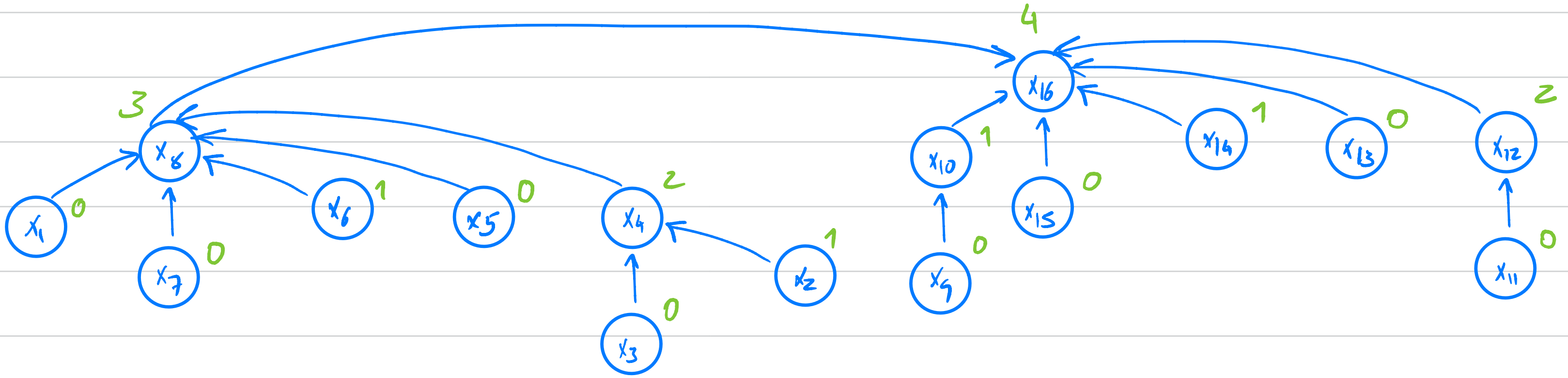
FindSet(x_9) VII

III



IV

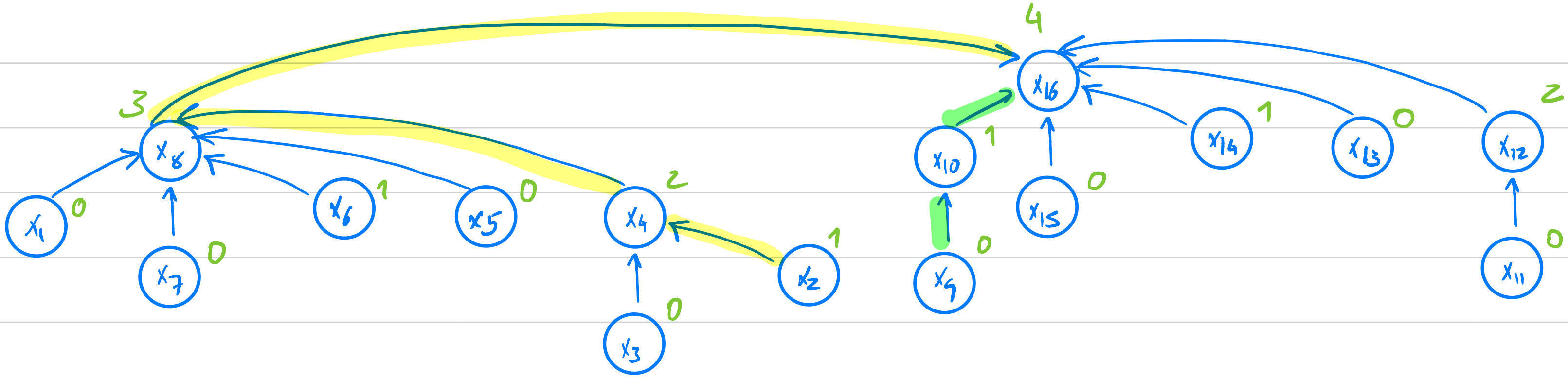
V



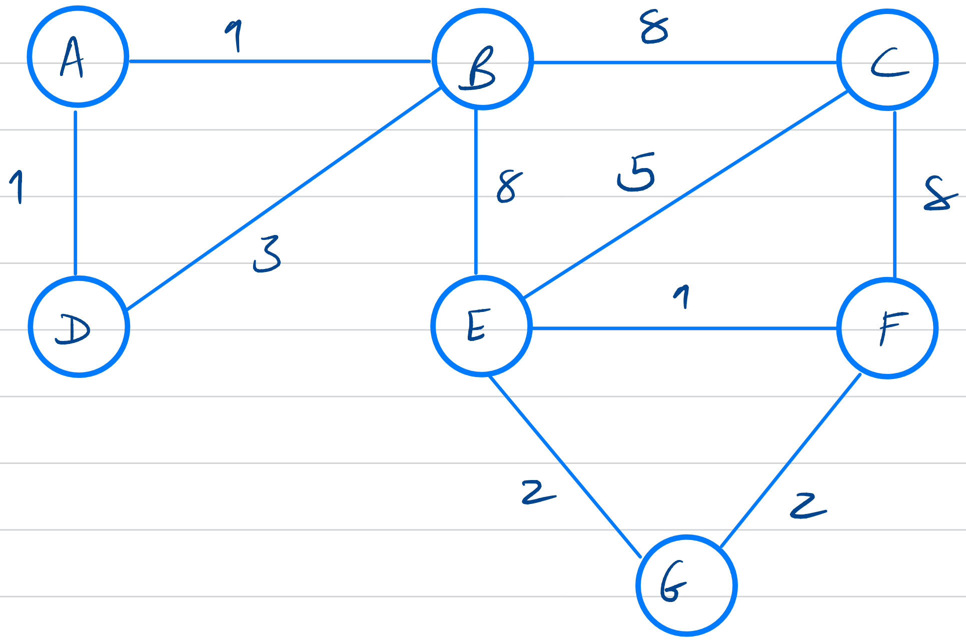
Q1 (21.3-1)

FindSet(x_2) VI

FindSet(x_9) VII



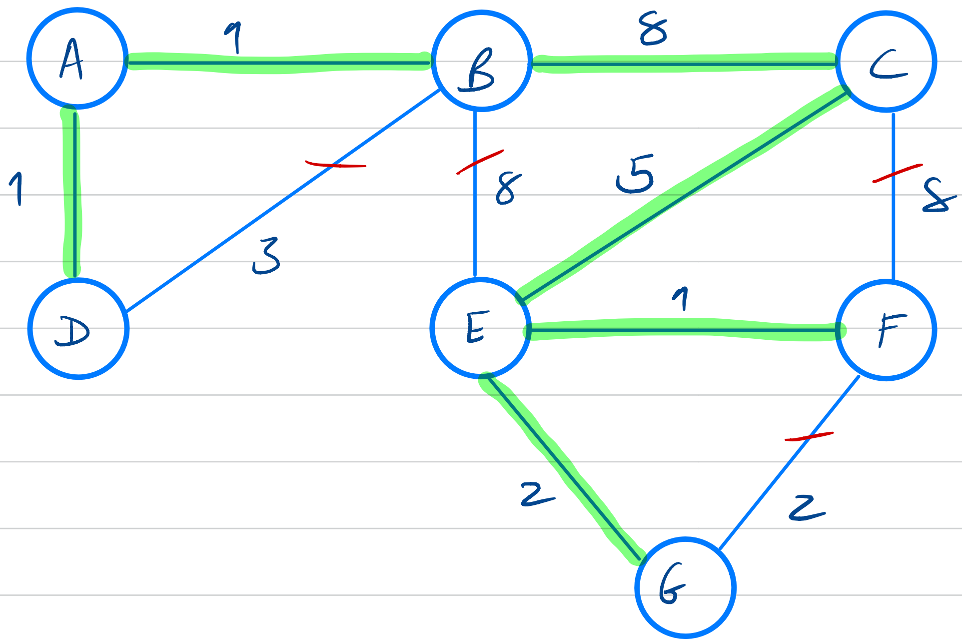
Q2 (TI 06/07 I.3)



- Kruskal
- Prim

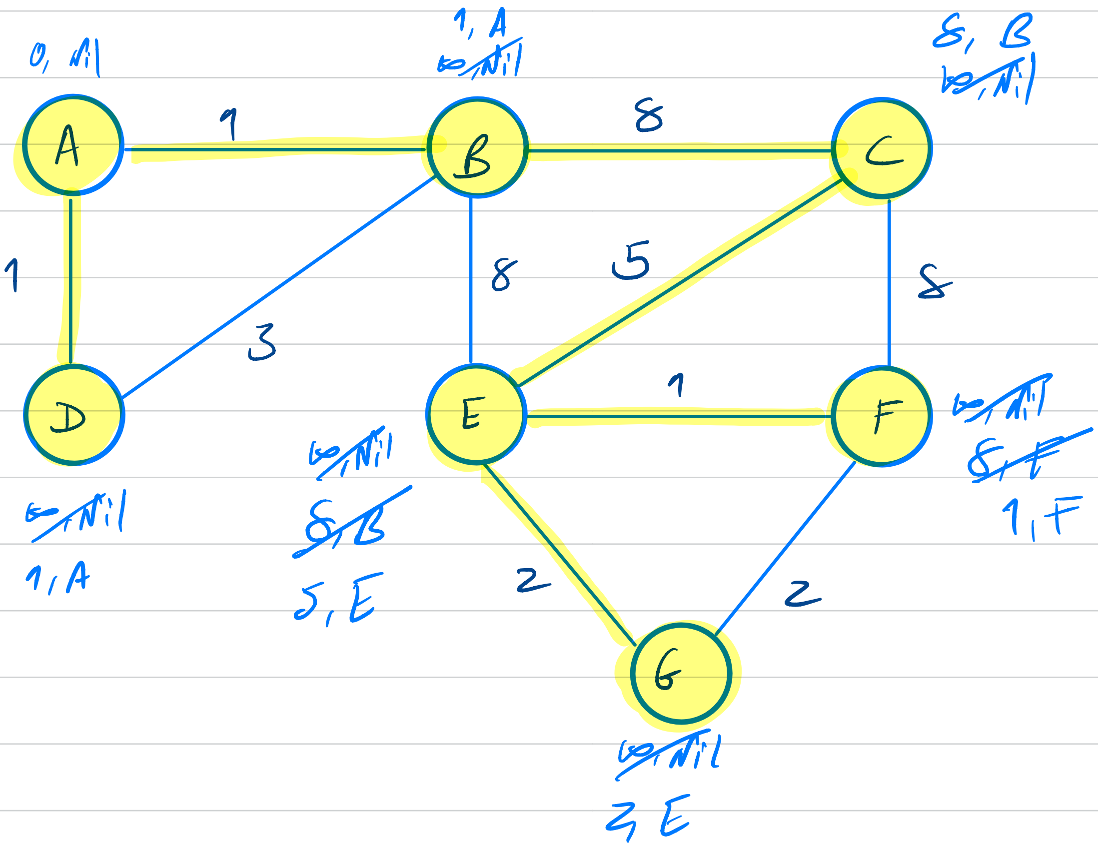
Q2 (TI 06/07 I.3)

- Kruskal
- Prim



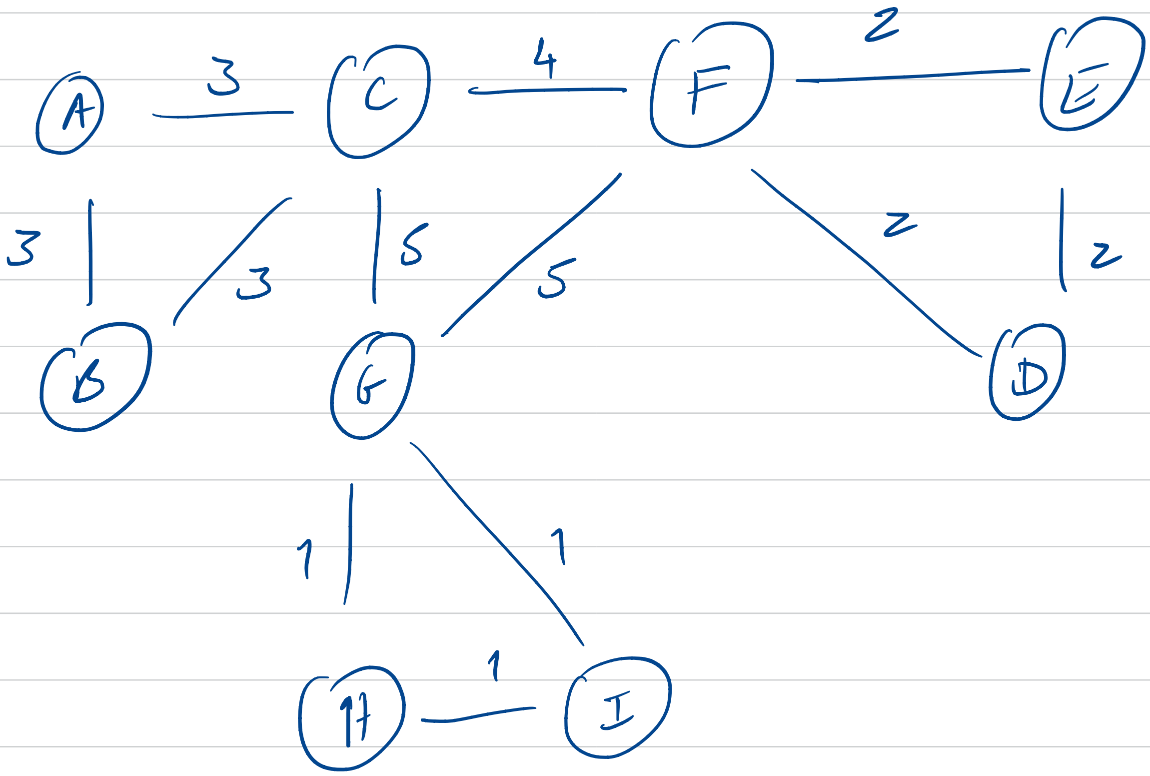
$$W(T) = 1+1+1+2+5+8$$
$$= 18$$

Q2 (TI 06/07 I.3)

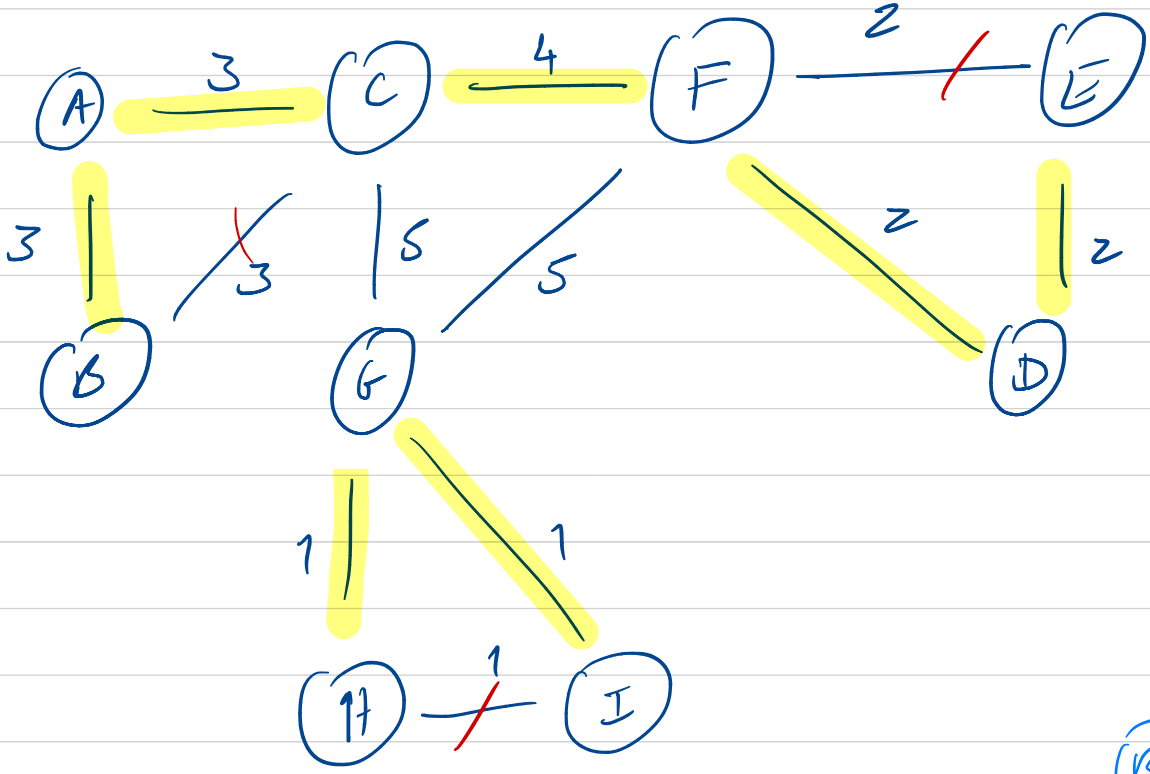


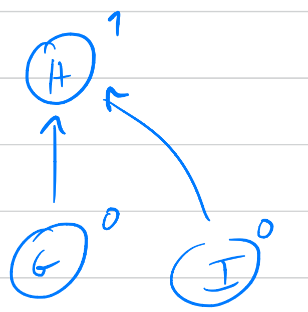
- Kruskal
- Prim

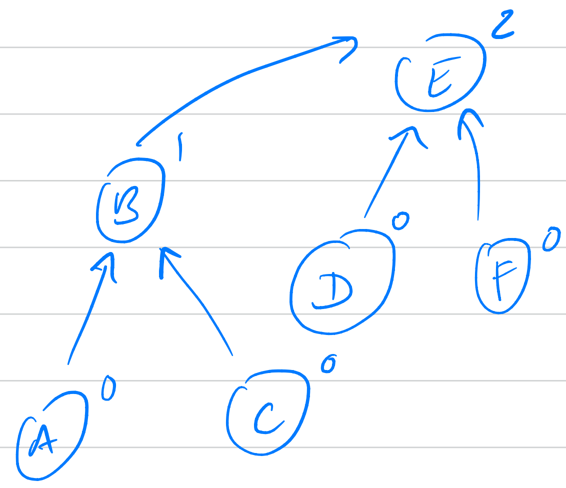
Q3 (EE 20/21 I.b)



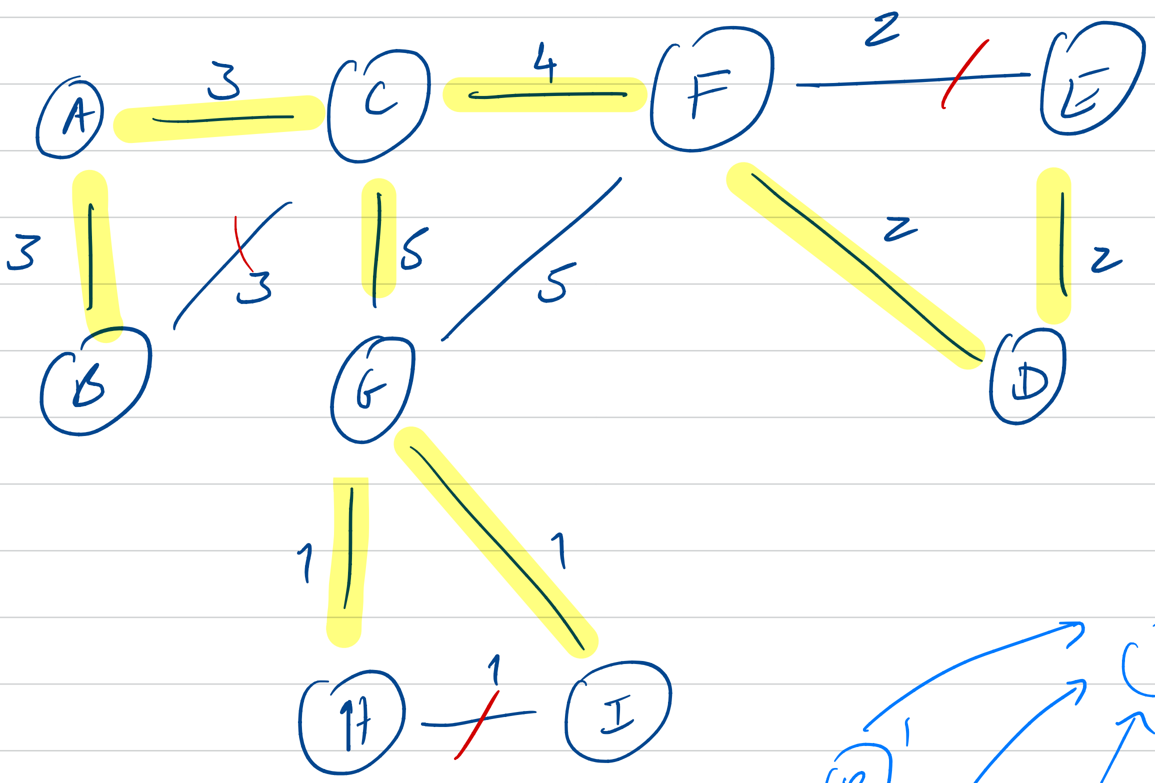
Q3 (EE 20/21 I.b)

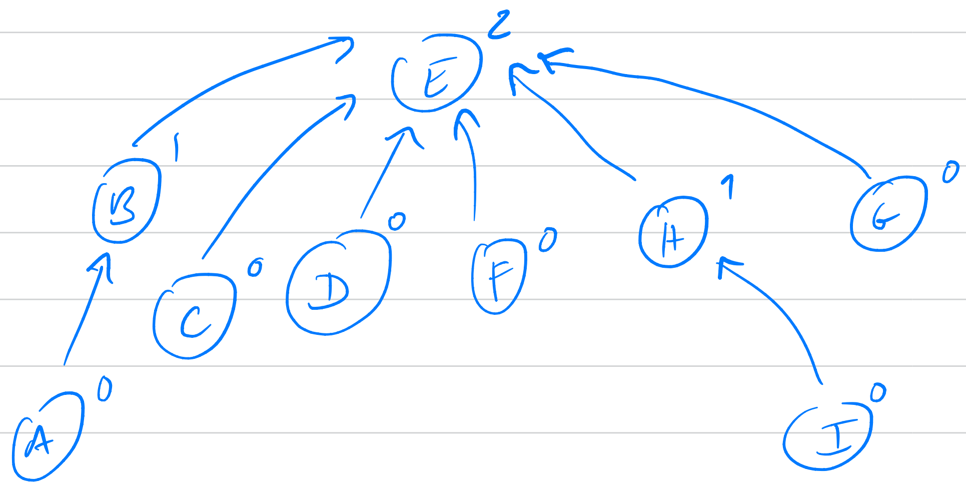






Q3 (EE 20/21 I.b)



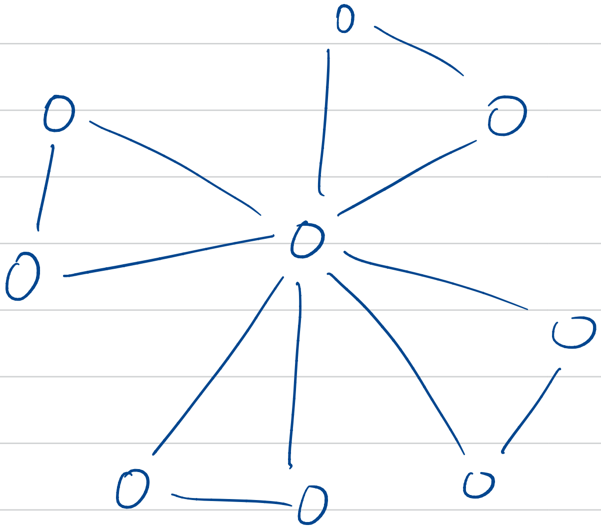


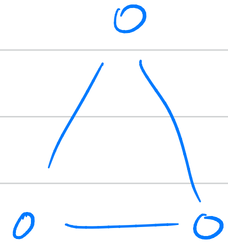
Nº de árvores:

$3^3 \times 2 = 54$

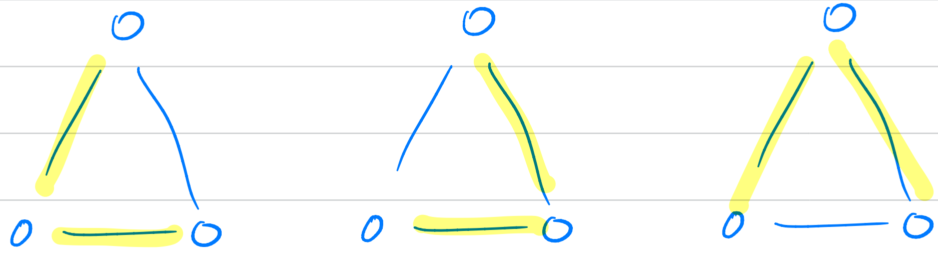
$w(T) = \underline{\underline{21}}$

Q4 (T1 08/09 II.1)





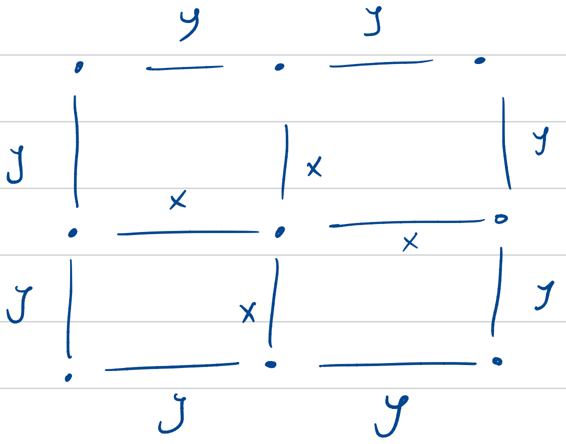
Quantas MSTs?
3



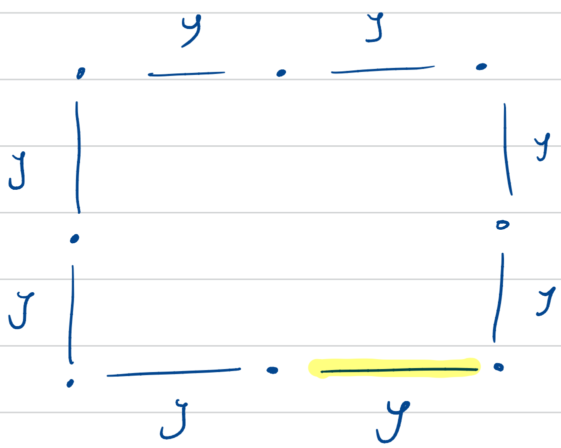
• Todos os arcos pesam o mesmo

Total: $3^4 = 81$

Q5 (R1 08/09 II.1)

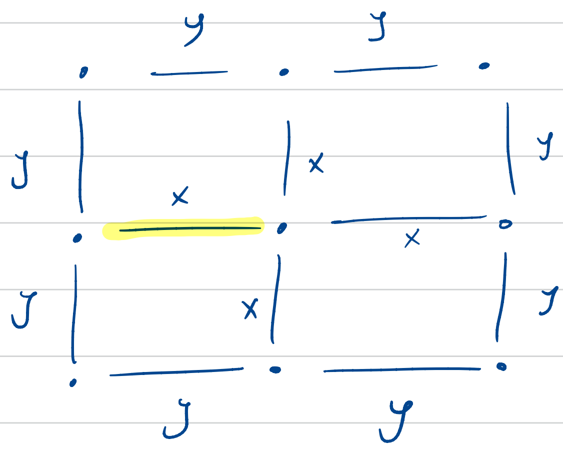


$x > y$



8 MSTs possíveis

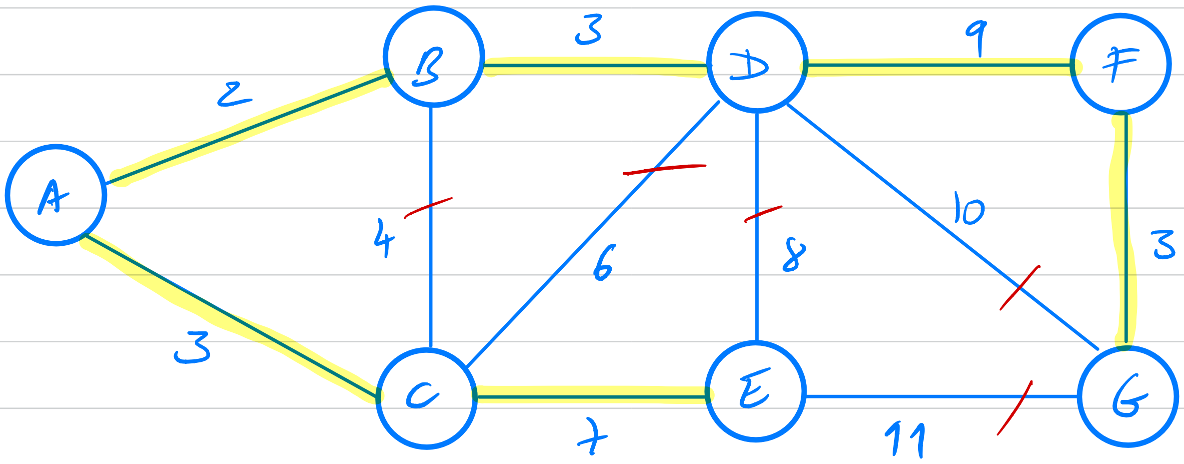
Qual o arco q pode ser removido?



Nº total de MSTs: $8 \times 4 = 32$

↳ Qual o arco interior q vai ser mantido?

Q6 (TI 07/08 - II.1)



- Uma MST
- Peso da MST
- nº de MSTs ⇒ 1

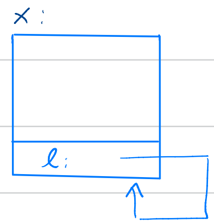
$$W(T) = 2 + 3 + 3 + 3 + 9 + 7$$
$$= 2 + 9 + 9 + 7$$
$$= \underline{27}$$

Q7 (CLRS Ex 21.3-4)

MakeSet(x)
$x.p = x$
$x.Rank = 0$

$\longrightarrow$

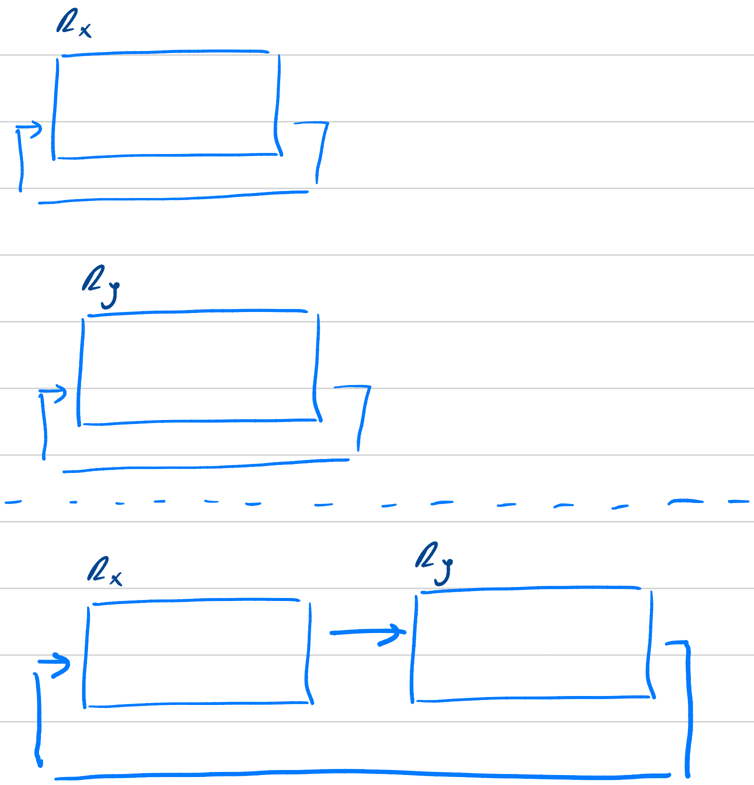
MakeSet(x)
$x.p = x$
$x.Rank = 0$
$x.l = x$



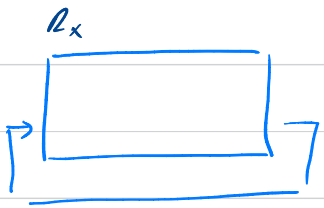
Union(x, y)
let R_x = FindSet(x)
let R_y = FindSet(y)
if ($R_x == R_y$) return
if ($R_x.Rank > R_y.Rank$)
$R_y.p := R_x$
else if ($R_y.Rank > R_x.Rank$)
$R_x.p := R_y$
else
$R_y.p := R_x$
$R_y.Rank := R_x.Rank + 1$

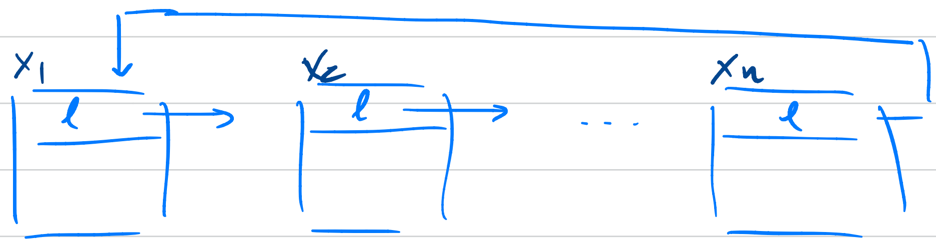
$\longrightarrow$

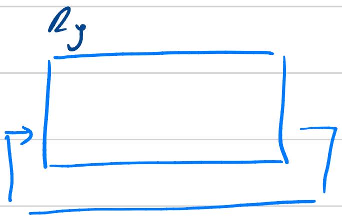
Union(x, y)
let R_x = FindSet(x)
let R_y = FindSet(y)
if ($R_x == R_y$) return
if ($R_x.Rank > R_y.Rank$)
$R_y.p := R_x$
ExtendList(R_x, R_y)
else if ($R_y.Rank > R_x.Rank$)
$R_x.p := R_y$
ExtendList(R_y, R_x)
else
$R_y.p := R_x$
$R_y.Rank := R_x.Rank + 1$
ExtendList(R_x, R_y)

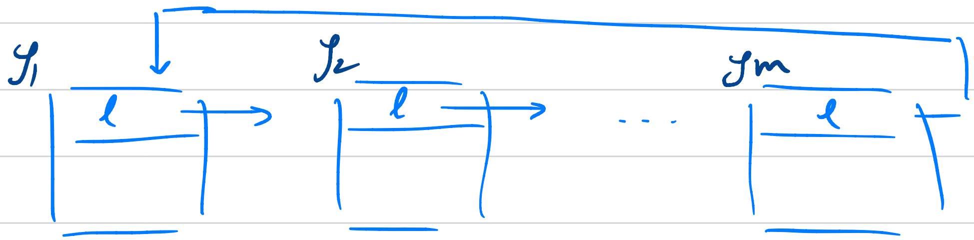


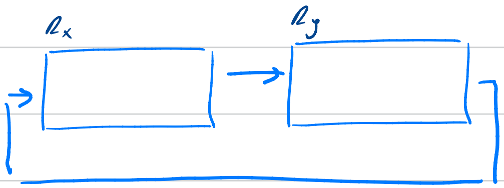
Joining circular lists:



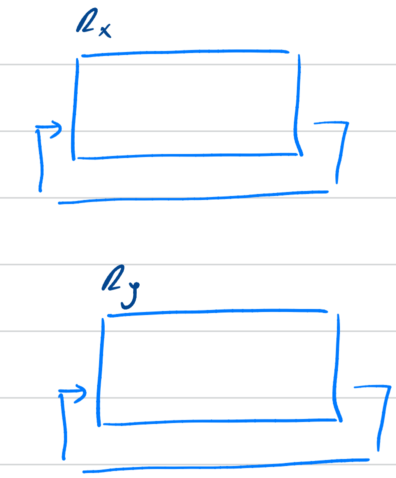


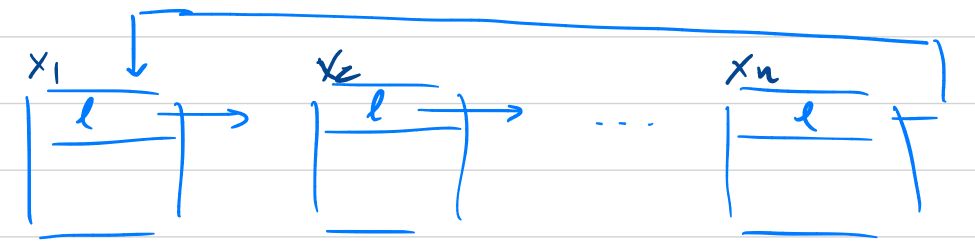


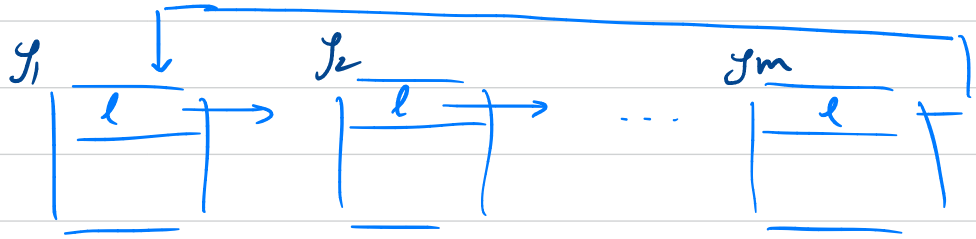


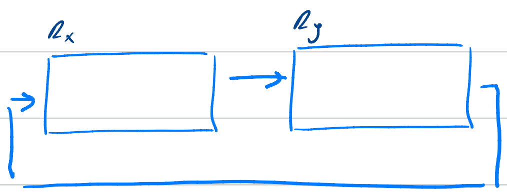


Joining Circular Lists:



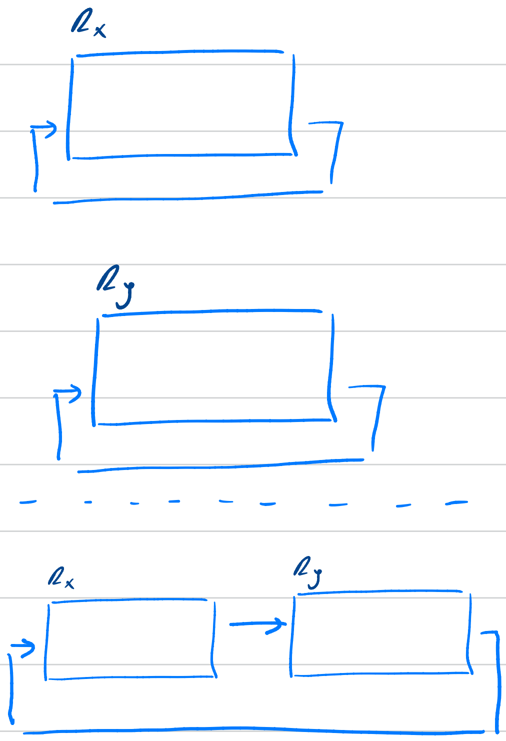


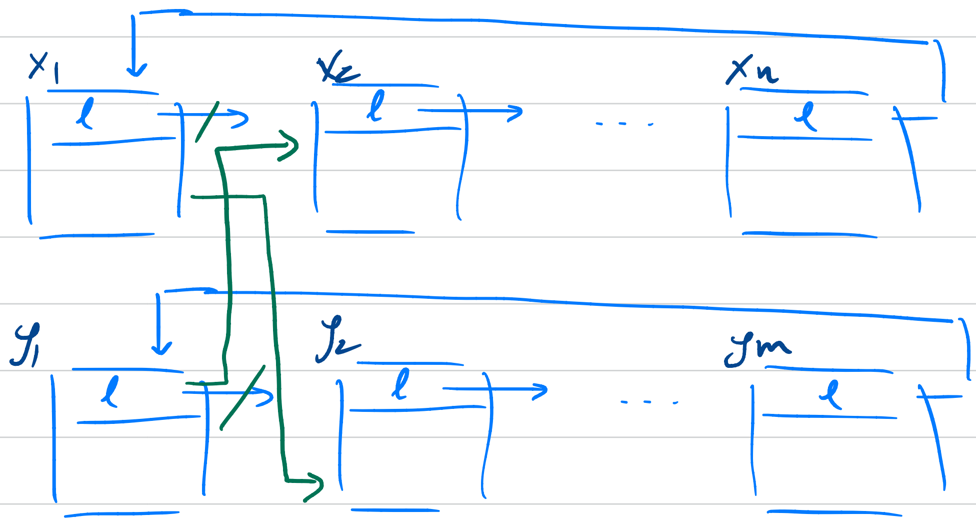


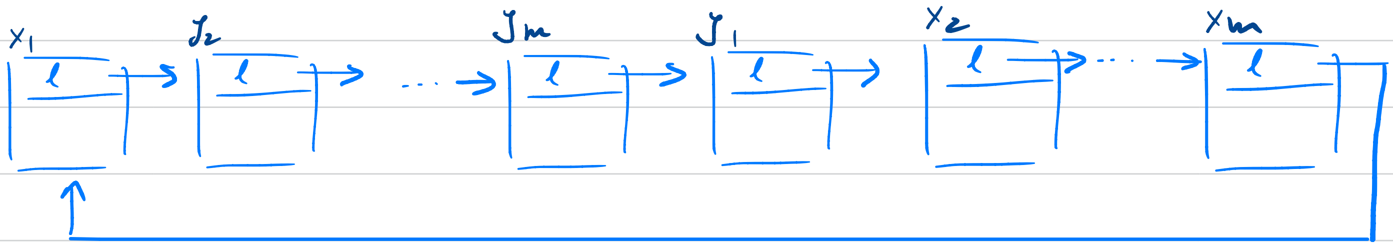



```
PrintSet(x)
    let head = x
    do {
        print(x)
        x = x.l
    } while (x ≠ head)
```

Joining Circular Lists:







ExtendCList(x, y)

$tmp := x.l$

$x.l := y.l$

$y.l := tmp$

Q8 (CLRS 23.2-2)

```
Prim(G, w, r)
  for each v ∈ G.V
     v.key = ∞;  v.π = Nil
  r.key := 0;
  let Q be a min-priority queue with content G.V
  while Q ≠ []
     let u = ExtractMin(Q)
     for each v ∈ G.Adj[u]
        if (v.key > w(u,v)) && v ∈ Q
           v.key := w(u,v);  v.π := u
```

$\Longrightarrow$

matriz de pesos

nº de vértices

vértice de origem

```
Prim(W, n, r)
  let key[1..n] be a new array
  let π[1..n] be a new array
  let done[1..n] be a new array
  for i=1 to n
       key[i] := ∞;  π[i] := Nil;  done[i] := false
  key[r] := 0
  for j=1 to n-1
     let w* = min{ key[i] | done[i] = false }
     Pick u s.t. key[i] = w*
     done[u] := true
     for i=1 to n
        if (W[u,i] < key[i] && done[i] = false)
           key[i] := W[u,i]
           π[i] := u
```

Q8 (CLRS 23.2-2)

```
Prim(G, w, r)
  for each v ∈ G.V
    v.key = ∞; v.π = Nil
  r.key := 0;
  let Q be a min-priority queue with content G.V
  while Q ≠ []
    let u = ExtractMin(Q)
    for each v ∈ G.Adj[u]
      if (v.key > w(u,v)) && v ∈ Q
        v.key := w(u,v); v.π := u
```

⟹

```
Prim(W, n, r)
```

W → matriz de pesos
n → nº de vértices
r → vértice de origem

```
let key[1..n] be a new array
let π[1..n] be a new array
let done[1..n] be a new array
for i=1 to n
    key[i] := ∞; π[i] := Nil; done[i] := false
key[r] := 0
for j=1 to n-1
    let w* = min{ key[i] | done[i] = false }
    Pick u s.t. key[i] = w*
    done[u] := true
    for i=1 to n
        if (W[u,i] < key[i] && done[i] = false)
            key[i] := W[u,i]
            π[i] := u
```

$O(n^2)$

$O(n)$

$O(n)$

Q9 (CLRS 23.1-6)

- Se para qualquer corte num grafo pesado existe existe um único arco leve q̄ cruza o corte então o grafo admite uma única MST.

  a) Provar a implicação
  b) Contra-exemplo para o sentido inverso.

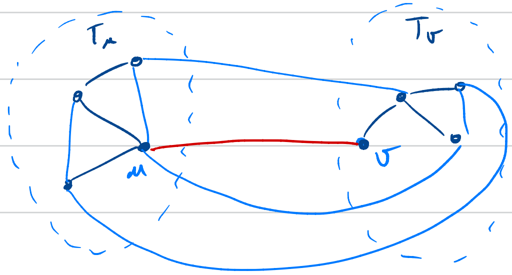
a) $\underbrace{\text{Para todo o corte, existe um único arco leve q̄ cruza o corte}}_{(\dagger)}$ $\Longrightarrow$ O grafo admite uma única MST T.

- Sejam T_1 e T_2 duas quaisquer MSTs de G. Vamos provar que $T_1 = T_2$ dado $(\dagger)$.

- Provar que $T_1 = T_2$ corresponde a provar que:
  $\forall u, v \in V. \ (u,v) \in T_1 \iff (u,v) \in T_2$

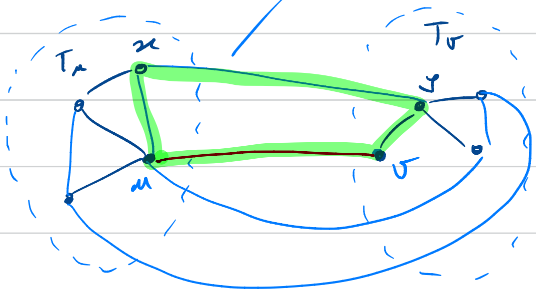
  ↳ Como a prova é simétrica basta provar um sentido da implicação.

23.1-6 Queremos provar que: $\forall u,v \in V.\ (u,v) \in T_1 \Rightarrow (u,v) \in T_2$



- Tome-se qualquer arco (u,v) em T_1.
  Se $(u,v) \in T_2$, não há nada a provar. Admitimos portanto que $(u,v) \notin T_2$.
  A remoção do arco (u,v) induz um corte (T_u, T_v), onde T_u contém os vértices atingíveis a partir de u usando arcos em $T_1 \setminus \{(u,v)\}$ e T_v contém os arcos atingíveis a partir de v.

ciclo q (u,v) introduz em T_2



- Como T_1 é MST concluímos q (u,v) é um arco leve q cruza o corte.

- Como $(u,v) \notin T_2$, a sua inclusão em T_2 dá origem a um caminho circular, p_c, que cruza o corte duas vezes:
  - uma vez pelo arco (u,v) e outra por um arco de T_2; seja (x,y) esse arco

- Como (u,v) é leve, usamos a hipótese p/ concluir que:
  $$w(u,v) < w(x,y).$$

Segue q: $T_2' = (T_2 \setminus \{(x,y)\}) \cup \{(u,v)\}$

é árvore abrangente e $w(T_2') < w(T_2)$.
Concluímos q T_2 não é MST, estabelecendo a contradição.