Prática 07

Ex. 21.3-1

for $i=1$ to 16
  Make-Set(x_i) — For 1

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

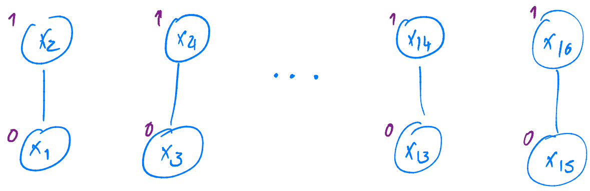
for $i=1$ to 13 by 4
  Union(x_i, x_{i+2}) — For 3

Union(x_1, x_5)
Union(x_{11}, x_{13})
Union(x_1, x_{10}) — Unions

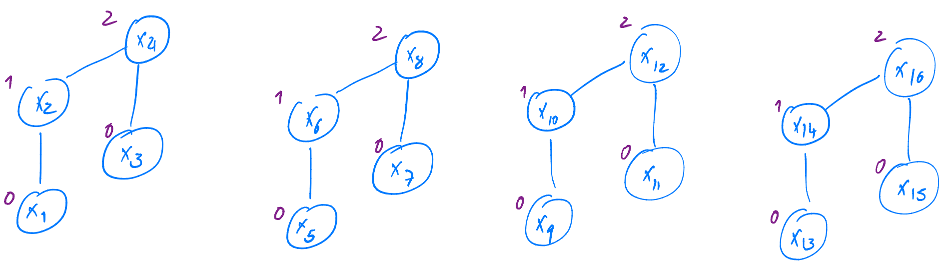
Find Set(x_2)
Find Set(x_9) — Finds

- After For 1:

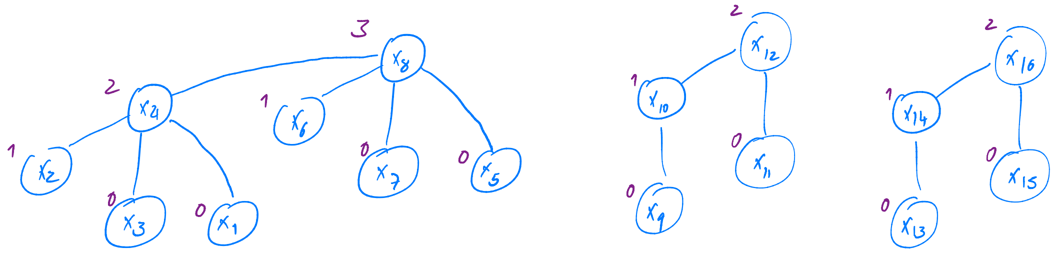
- After For 2:



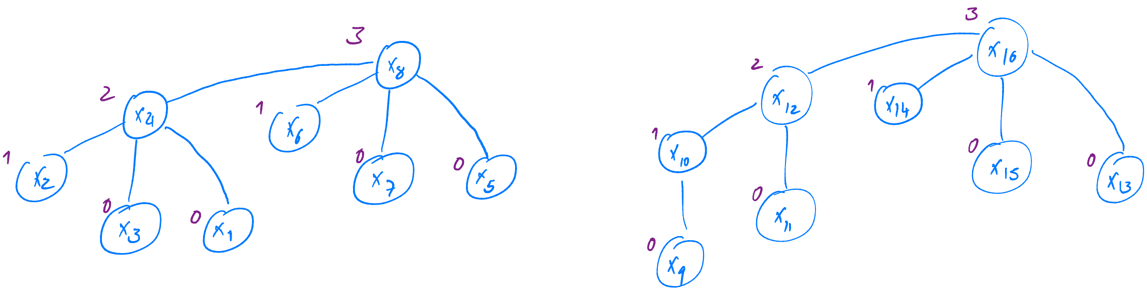
- After For 3:
  $U(1,3)$, $U(5,7)$, $U(9,11)$, $U(13,15)$



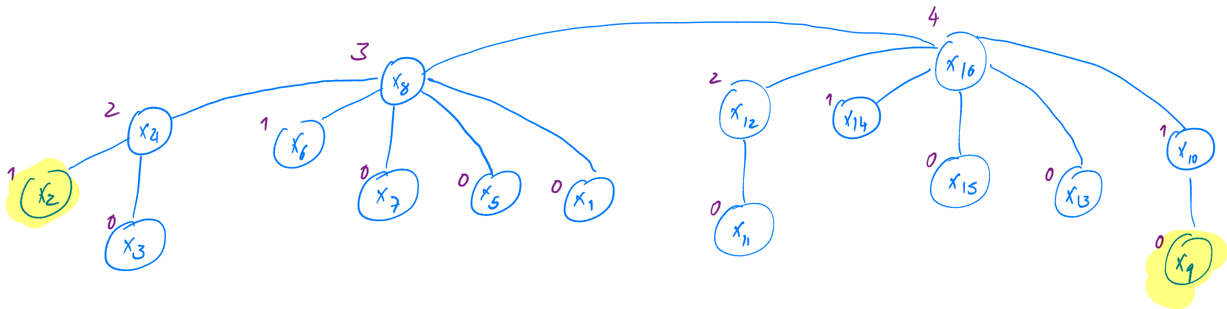
- $U(1,5)$



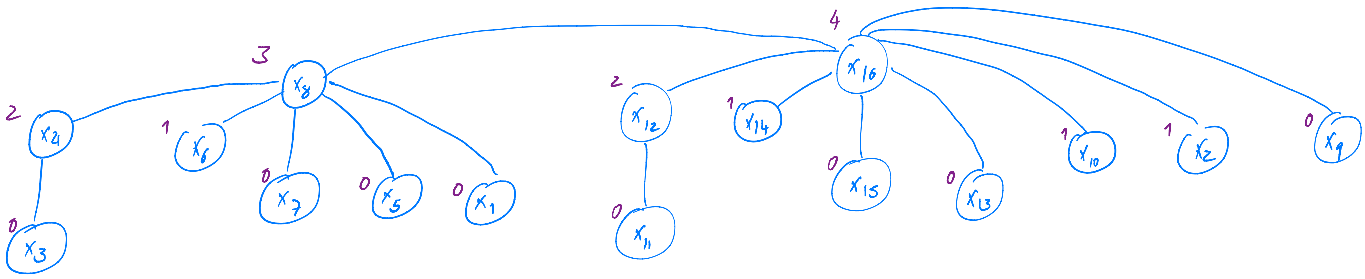
• U(11, 13):



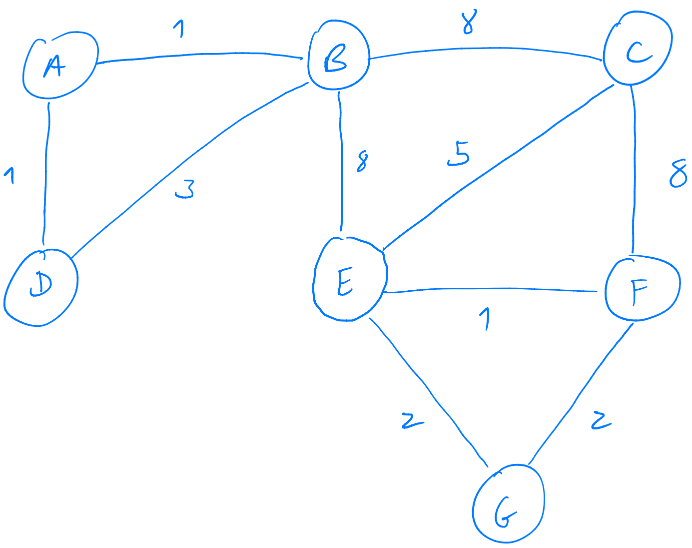
• U(1, 10)



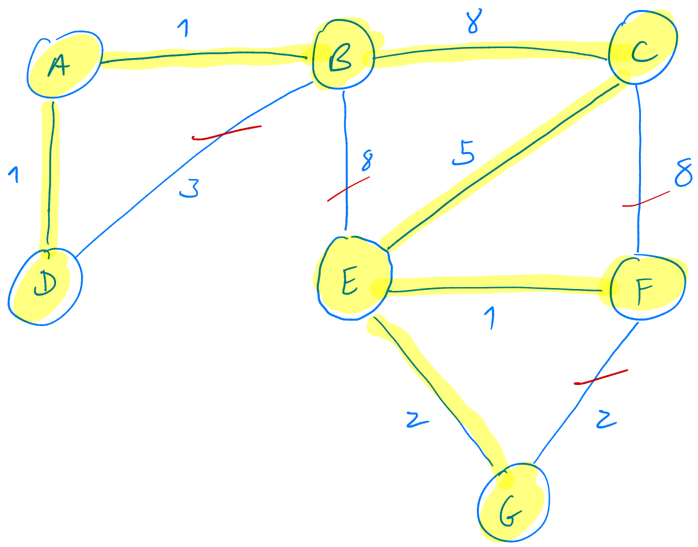
• FS(2), FS(9)



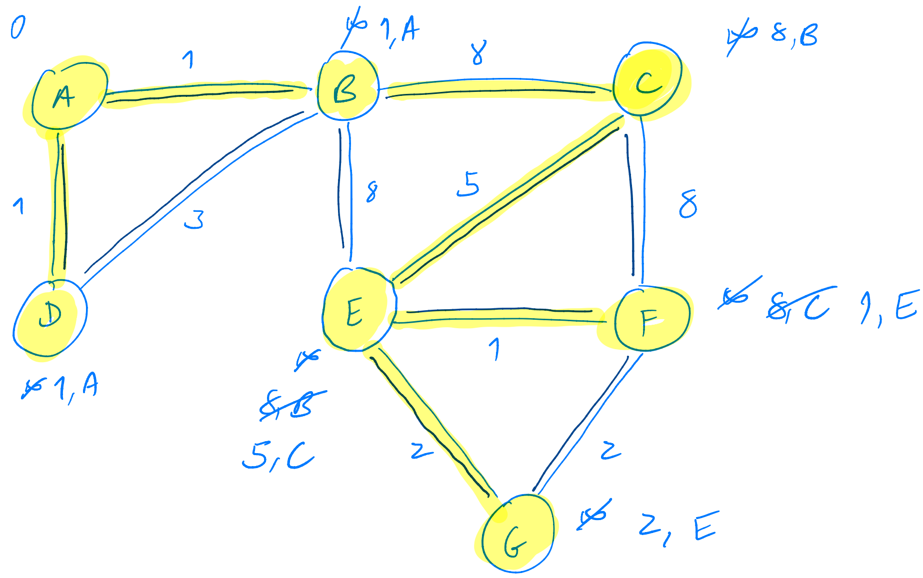
T1 06/07 I.3 -



Kruskal

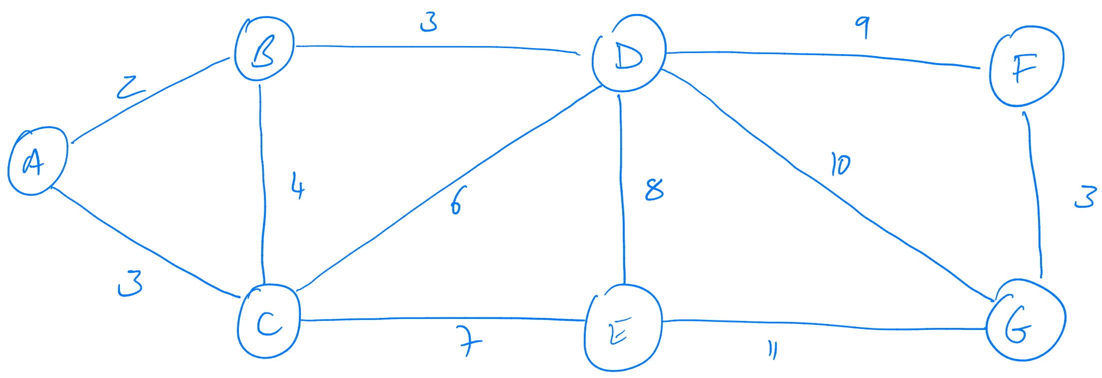


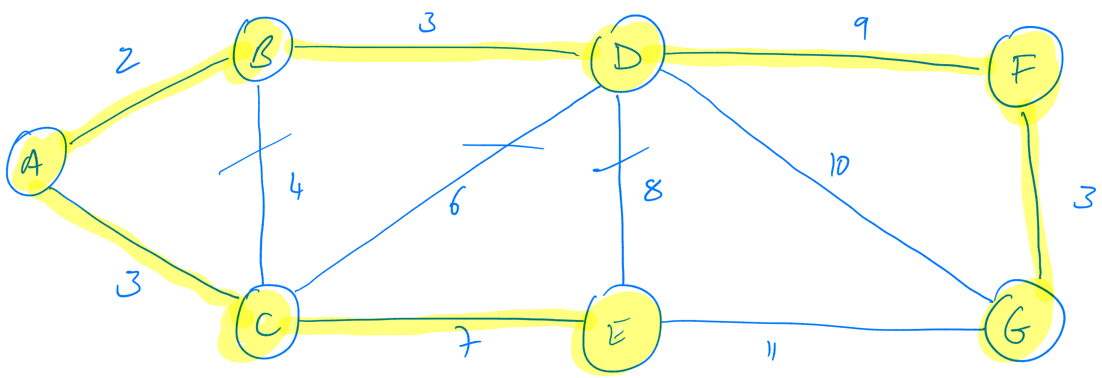
Prim



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

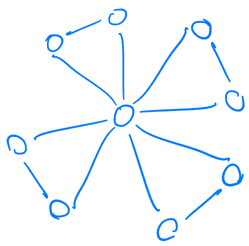
T1 07/08 II.1 -



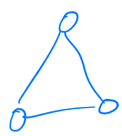


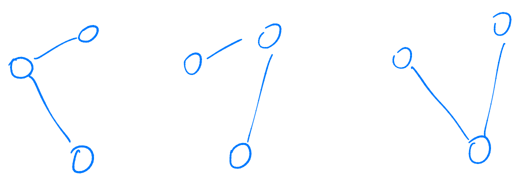
1 MST · $w(T) = 2+3+3+3+7+9 = 27$

T1 08/09 II.1 -



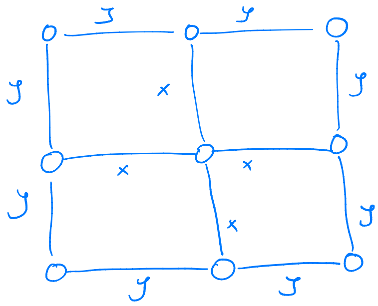
· Para cada triângulo temos 3 casos:





· Temos 4 triângulos: $3 \times 3 \times 3 \times 3 = 3^4 = 81$

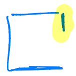
RI 08/09 II.1



$x > y$

• 4 casos:

• Para cada caso, 2 casos:

 & 

• 4 ligações ao centro

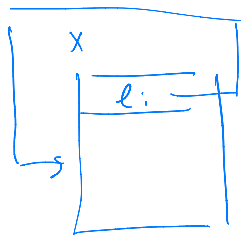
• Total: $4 \times 2 \times 4 = \underline{\underline{32}}$

21.3-4

```
MakeSet(x)
    x.p = x
    x.rank = 0
```

→

```
MakeSet(x)
    x.p = x
    x.rank = 0
    x.l = x
```

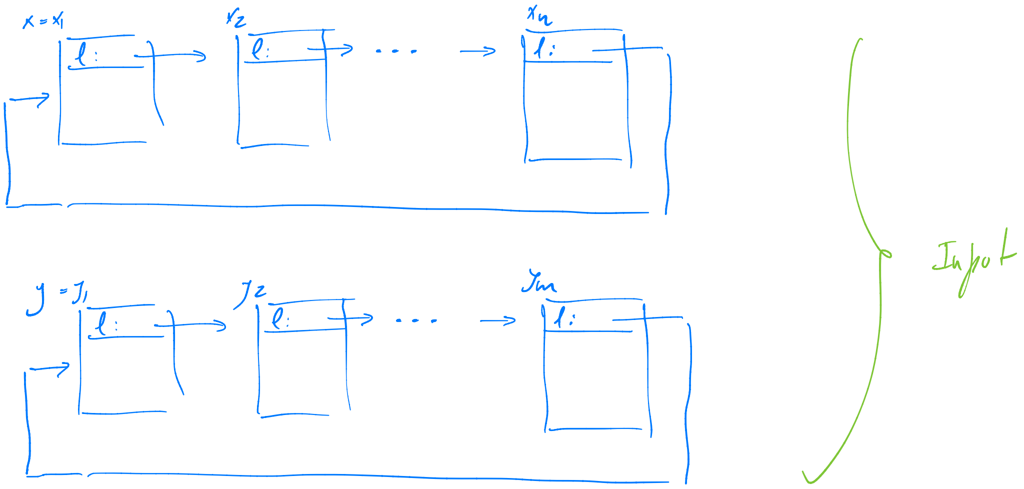


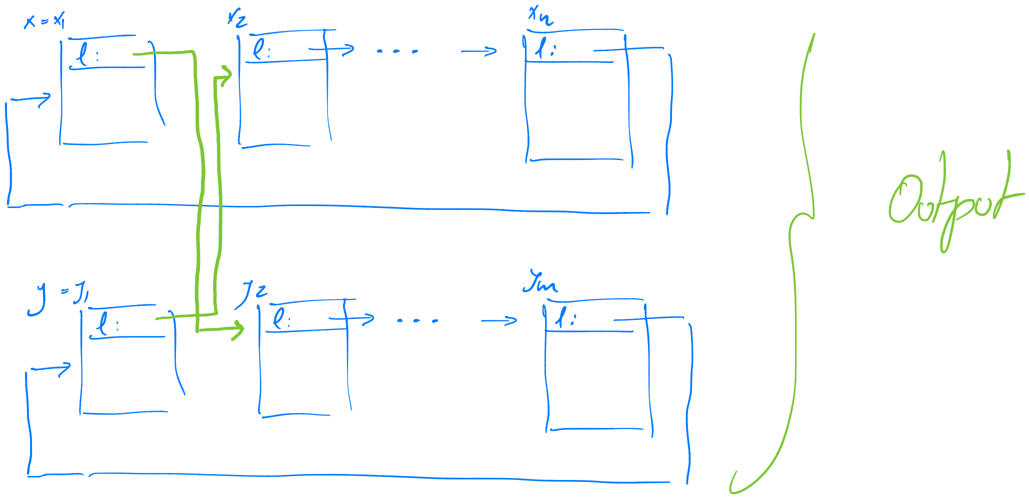

```
Union(x, y)
    let rx = FindSet(x)
    let ry = FindSet(y)
    if (rx == ry) return
    if (rx.rank > ry.rank)
        ry.p = rx
    else if (ry.rank > rx.rank)
        rx.p = ry
    else
        rx.rank = rx.rank + 1
        ry.p = rx
```

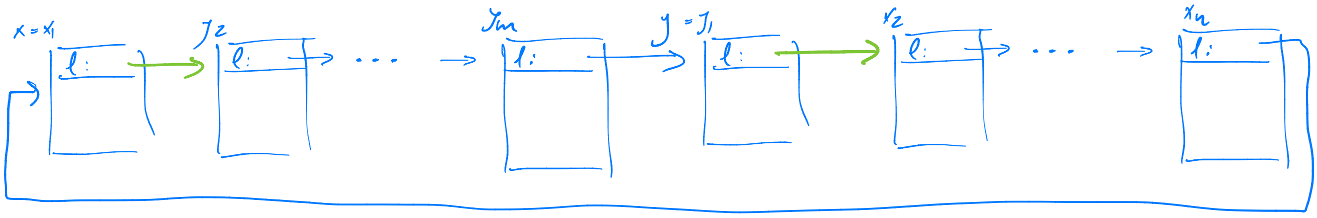
→

```
Union(x, y)
    let rx = FindSet(x)
    let ry = FindSet(y)
    if (rx == ry) return
    if (rx.rank > ry.rank)
        ry.p = rx
        ExtendCList(rx, ry)
    else if (ry.rank > rx.rank)
        rx.p = ry
        ExtendCList(ry, rx)
    else
        rx.rank = rx.rank + 1
        ry.p = rx
        ExtendCList(rx, ry)
```

ExtendCList (x, y)








```
ExtendCList (x, y)
    tmp = x.l ;
    x.l = y.l ;
    y.l = tmp

PrintSet (x)
    y = x
    do {
        Print(y);
        y = y.l
    } while (y ≠ x)
```

23.2-2

```
Prim(G, w, r)
  for each v ∈ G.V
     v.key = ∞
     v.π = Nil
  r.key = 0
  Q = G.V
  while Q ≠ []
    let u = ExtractMin(Q)
    for each v ∈ G.Adj[u]
      if w(u,v) < v.key ∧ v ∈ Q
         v.key = w(u,v)
         v.π = u
```

→

u = -1;

→ matriz com os pesos
→ nº de vértices
→ índice do 1º vértice a ser explorado

```
Prim(M, 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;
  while (j=1 to n-1)
     let w* = min{ w[i] | 1≤i≤n ∧ done[i] = false }      O(n)
     Pick u ∈ { i | 1≤i≤n ∧ w[i] = w* }
     done[u] = true
     for i=1 to n                                          O(n)
        if((M[u,i] < key[i]) && !done[i])
           key[i] = M[u,i]
           π[i] = u
```

$O(n^2)$

23.1-6

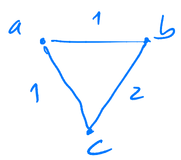
a) Se para qualquer corte num grafo pesado existe um único arco leve que cruza o corte, então o grafo admite uma MST única.

Prova

- Suponhamos por contradição que o grafo admite duas MSTs diferentes T_1 e T_2.
- $w(T_1) = w(T_2)$ porque são ambas MSTs.
- T_1 e T_2 têm que diferir em pelo menos um arco, logo sabemos q existe um arco (x,y) q está em T_1 e não está em T_2 e um arco (u,v) q está em T_2 e não está em T_1.
  Considere-se o corte: $(\{x,u\}, V \setminus \{x,u\})$
- Há um único arco leve que cruza o corte
  Esse arco não pode ser simultaneamente (x,y) e (u,v)
- Suponhamos q o arco leve é (x,y). Então podemos substituir (u,v) por (x,y) em T_2 e obter uma MST, T_2', com menor peso.

Contradição! ∴

b) O facto de um grafo admitir uma MST única não implica q para qualquer corte haja um único ~~existe um único arco leve a cruzar o corte.~~



$(\{a\}, \{b,c\})$

23.1-8

- Esboço da prova

Suponhamos por contradição que um dado grafo pesado G admite duas MSTs T_1 e T_2 com listas de pesos L_1 e L_2, respectivamente, tais que $L_1 \neq L_2$.

- Para facilitar a prova consideramos os multiconjuntos com os valores de L_1 e os valores de L_2, W_1 e W_2 respectivamente.

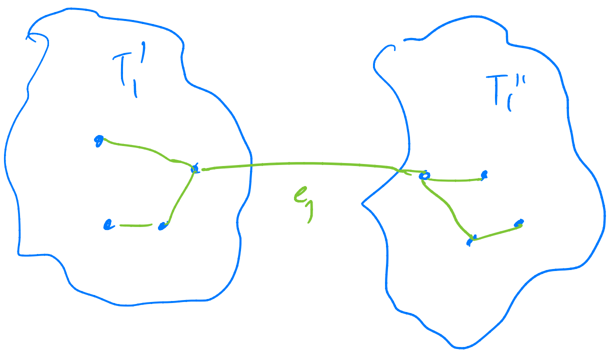
- Consideremos o conjunto: $W^* = W_1 \Delta W_2$ ↳ diferença simétrica

Seja $w^* = \min W^*$

O nº de vezes q w^* ocorre em L_1 é diferente do nº de vezes que w^* ocorre em L_2. Assumimos sem perda de generalidade q o nº de vezes que w^* ocorre em L_1 é $\geq$ ao nº de vezes que w^* ocorre em L_2. Tem portanto de existir um arco de peso w^* q ocorre em T_1 e não ocorre em T_2.

Seja e_1 esse arco.

- Consideremos o corte $(S, V \setminus S)$ induzido por T_1 e e_1 em G.



- $T_1 = T_1' \cup T_1'' \cup \{e_1\}$

- Seja $E(S)$ o conjunto dos arcos de G q cruzam o corte.

- Seja e_2 o arco em $T_2 \cap E(S)$ com peso mínimo

- Há 3 casos a considerar:
  - (I) $w(e_2) < w^*$
  - (II) $w(e_2) > w^*$
  - (III) $w(e_2) = w^*$

(I) $w(e_2) < w^*$

Considere-se a MST:

$$\bar{T}_1 = T_1' \cup T_1'' \cup \{e_2\}$$

Temos que: $w(\bar{T}_1) < w(T_1) \Rightarrow$ Contradição!

(II) $w(e_2) > w^*$

Considere-se a MST:

$$\bar{T}_2 = T_2' \cup T_2'' \cup \{e_1\}$$

Temos que: $w(\bar{T}_2) < w(T_2) \Rightarrow$ Contradição!

(III) $w(e_2) = w^*$

- Considere-se a árvore:

$$\hat{T}_1 = T_1' \cup T_1'' \cup \{e_2\}$$

- $w(\hat{T}_1) = w(T_1)$ e a lista de pesos dos arcos de $\hat{T}_1$ é igual à lista de pesos dos arcos de T_1.

- Repetimos o argumento original para as árvores $\hat{T}_1$ e T_2, observando q as árvores $\hat{T}_1$ e T_2 têm menos arcos de peso w^* diferentes. Como T_2 tem um nº finito de arcos com peso w^* (e inferior a T_1), inevitavelmente vamos acabar por estar no caso (I) ou (II) da prova.