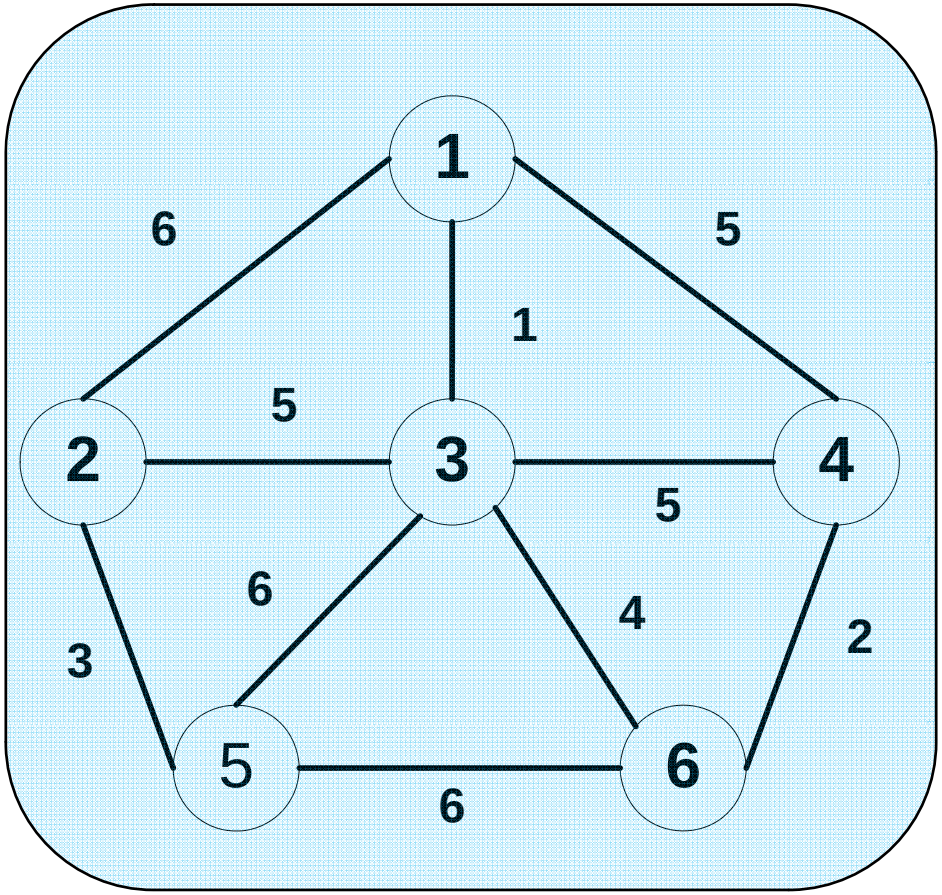
Δένδρα επικάλυψης ελάχιστου κόστους

Αλγόριθμος Kruskal

Αλγόριθμος Kruskal

- Ξεκινάμε από ένα δάσος από **n** δένδρα, κάθε ένα δένδρο εκφυλισμένο σε ένα μεμονωμένο κόμβο.
- Σε κάθε επανάληψη, προσθέτουμε τη πλευρά με το μικρότερο κόστος και η οποία **δεν δημιουργεί κύκλο** με τις ήδη επιλεγμένες πλευρές.
- Σταματάμε όταν έχουμε ένα δένδρο επικάλυψης (γράφος συνεκτικός) ή όταν δεν ευρίσκουμε πλέον πλευρά να προσθέσουμε (ο γράφος δεν είναι συνεκτικός)

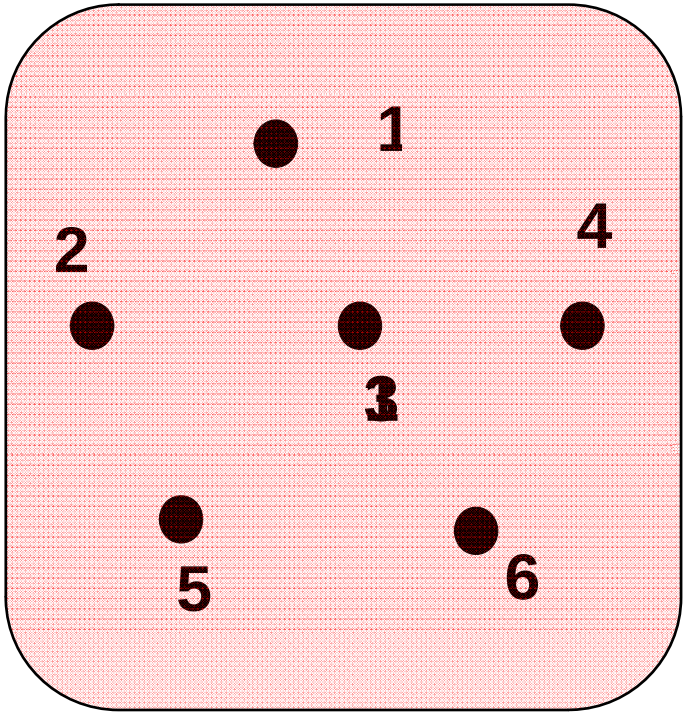
Αλγόριθμος Kruskal - Παράδειγμα



Διάταξη πλευρών κατά αύξουσα τάξη

[1,3] : κόστος 1
[4,6] : κόστος 2
[2,5] : κόστος 3
[3,6] : κόστος 4
[1,4] : κόστος 5
[3,4] : κόστος 5
[2,3] : κόστος 5
[1,2] : κόστος 6
[3,5] : κόστος 6
[5,6] : κόστος 6

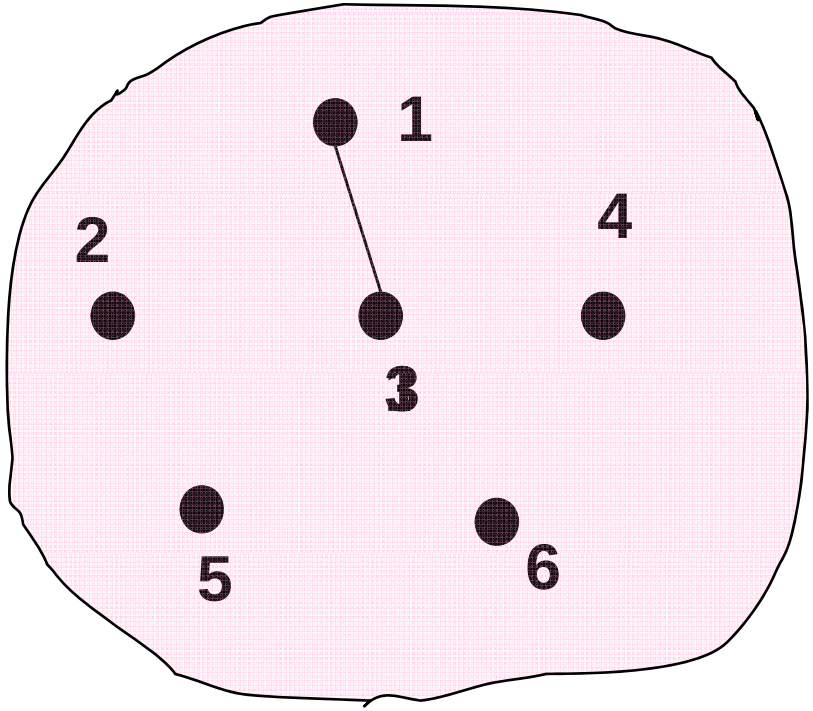
Αριθμός συνιστωσών n = αριθμός μεμονομένων κόμβων



Διάταξη πλευρών κατά αύξουσα τάξη

[1,3] : κόστος 1
[4,6] : κόστος 2
[2,5] : κόστος 3
[3,6] : κόστος 4
[1,4] : κόστος 5
[3,4] : κόστος 5
[2,3] : κόστος 5
[1,2] : κόστος 6
[3,5] : κόστος 6
[5,6] : κόστος 6

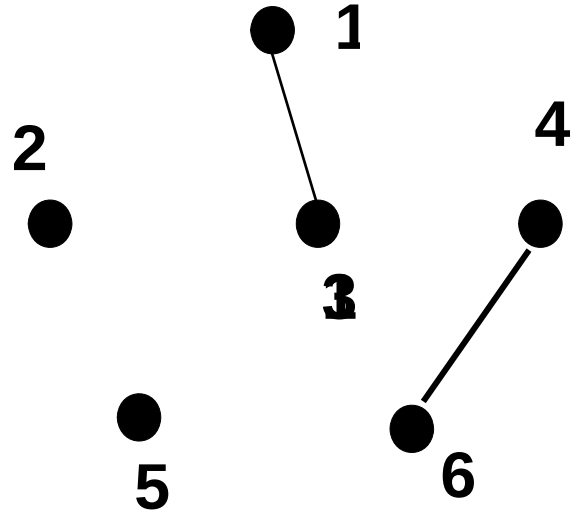
Αλγόριθμος Kruskal: βήμα 1



Διάταξη πλευρών
κατά αύξουσα τάξη

[1,3] : *κόστος* 1
[4,6] : *κόστος* 2
[2,5] : *κόστος* 3
[3,6] : *κόστος* 4
[1,4] : *κόστος* 5
[3,4] : *κόστος* 5
[2,3] : *κόστος* 5
[1,2] : *κόστος* 6
[3,5] : *κόστος* 6
[5,6] : *κόστος* 6

Αλγόριθμος Kruskal: βήμα 2



Διάταξη πλευρών
κατά αύξουσα τάξη

[1,3] : *κόστος* 1

[4,6] : *κόστος* 2

[2,5] : *κόστος* 3

[3,6] : *κόστος* 4

[1,4] : *κόστος* 5

[3,4] : *κόστος* 5

[2,3] : *κόστος* 5

[1,2] : *κόστος* 6

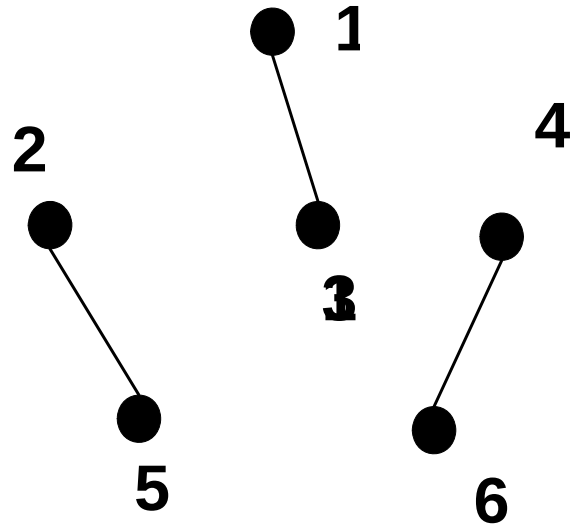
[3,5] : *κόστος* 6

[5,6] : *κόστος* 6

Αλγόριθμος Kruskal: βήμα 3

Διάταξη πλευρών
κατά αύξουσα τάξη

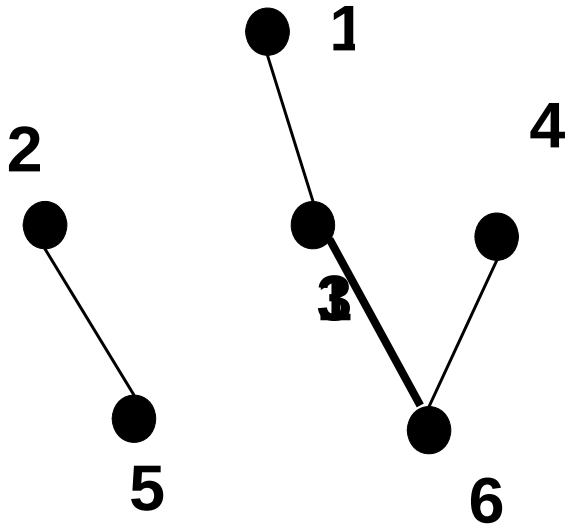
[1,3] : *κόστος* 1
[4,6] : *κόστος* 2
[2,5] : *κόστος* 3
[3,6] : *κόστος* 4
[1,4] : *κόστος* 5
[3,4] : *κόστος* 5
[2,3] : *κόστος* 5
[1,2] : *κόστος* 6
[3,5] : *κόστος* 6
[5,6] : *κόστος* 6



Αλγόριθμος Kruskal: βήμα 4

Διάταξη πλευρών
κατά αύξουσα τάξη

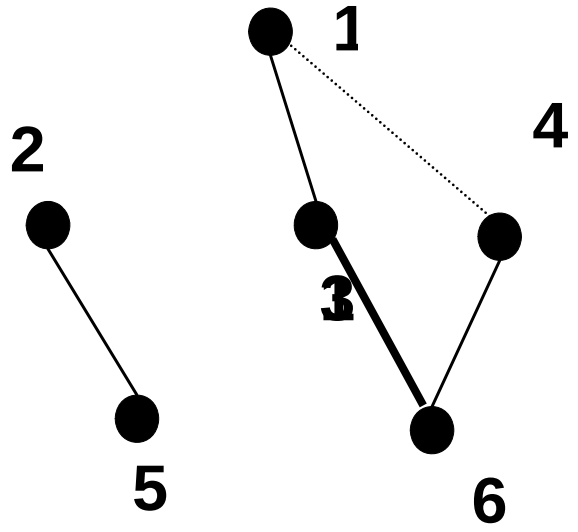
[1,3] : *κόστος* 1
[4,6] : *κόστος* 2
[2,5] : *κόστος* 3
[3,6] : *κόστος* 4
[1,4] : *κόστος* 5
[3,4] : *κόστος* 5
[2,3] : *κόστος* 5
[1,2] : *κόστος* 6
[3,5] : *κόστος* 6
[5,6] : *κόστος* 6



Αλγόριθμος Kruskal: βήμα 5

Διάταξη πλευρών
κατά αύξουσα τάξη

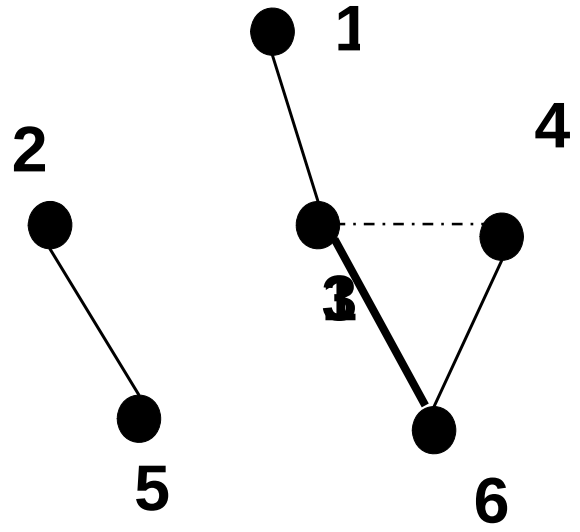
[1,3] : *κόστος* 1
[4,6] : *κόστος* 2
[2,5] : *κόστος* 3
[3,6] : *κόστος* 4
[1,4] : *κόστος* 5
[3,4] : *κόστος* 5
[2,3] : *κόστος* 5
[1,2] : *κόστος* 6
[3,5] : *κόστος* 6
[5,6] : *κόστος* 6



Αλγόριθμος Kruskal: βήμα 6

Διάταξη πλευρών
κατά αύξουσα τάξη

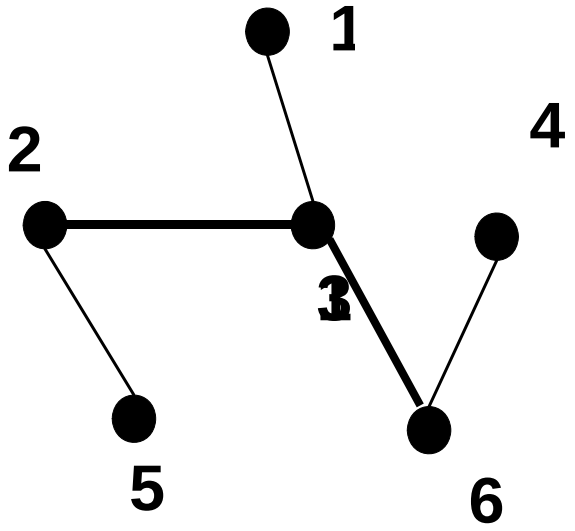
[1,3] : *κόστος* 1
[4,6] : *κόστος* 2
[2,5] : *κόστος* 3
[3,6] : *κόστος* 4
[1,4] : *κόστος* 5
[3,4] : *κόστος* 5
[2,3] : *κόστος* 5
[1,2] : *κόστος* 6
[3,5] : *κόστος* 6
[5,6] : *κόστος* 6



Αλγόριθμος Kruskal: βήμα 7

Διάταξη πλευρών
κατά αύξουσα τάξη

[1,3] : *κόστος* 1
[4,6] : *κόστος* 2
[2,5] : *κόστος* 3
[3,6] : *κόστος* 4
[1,4] : *κόστος* 5
[3,4] : *κόστος* 5
[2,3] : *κόστος* 5
[1,2] : *κόστος* 6
[3,5] : *κόστος* 6
[5,6] : *κόστος* 6

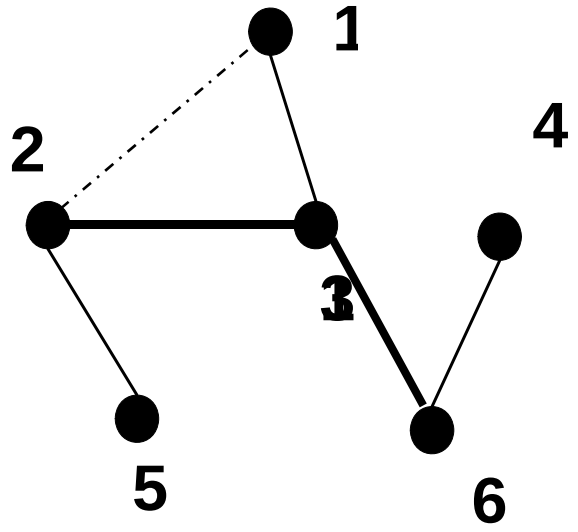


δέντρο επικάλυψης: τέλος

Αλγόριθμος Kruskal: βήμα 8

Διάταξη πλευρών
κατά αύξουσα τάξη

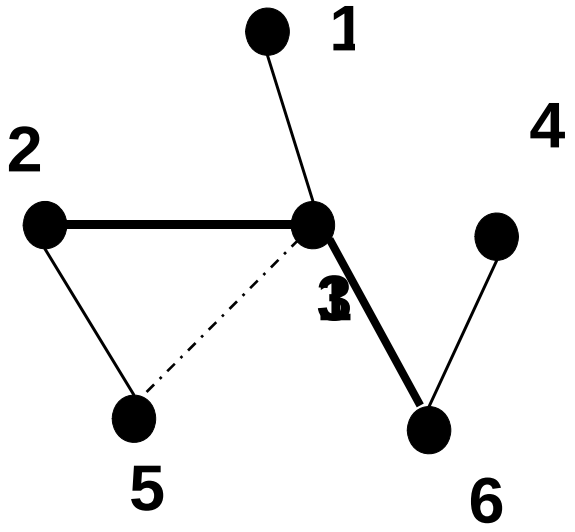
[1,3] : *κόστος* 1
[4,6] : *κόστος* 2
[2,5] : *κόστος* 3
[3,6] : *κόστος* 4
[1,4] : *κόστος* 5
[3,4] : *κόστος* 5
[2,3] : *κόστος* 5
[1,2] : *κόστος* 6
[3,5] : *κόστος* 6
[5,6] : *κόστος* 6



Αλγόριθμος Kruskal: βήμα 9

Διάταξη πλευρών
κατά αύξουσα τάξη

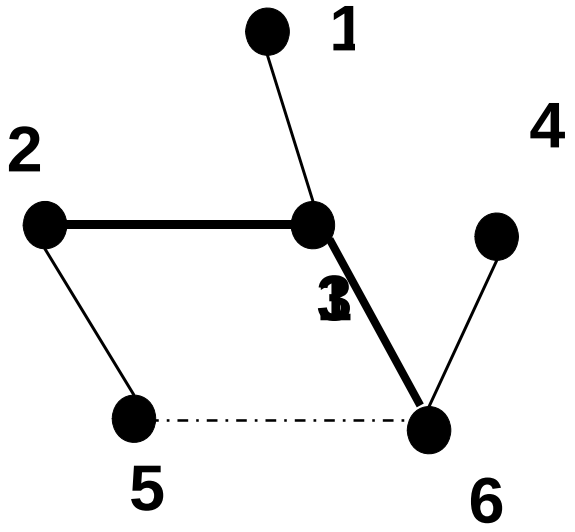
[1,3] : *κόστος* 1
[4,6] : *κόστος* 2
[2,5] : *κόστος* 3
[3,6] : *κόστος* 4
[1,4] : *κόστος* 5
[3,4] : *κόστος* 5
[2,3] : *κόστος* 5
[1,2] : *κόστος* 6
[3,5] : *κόστος* 6
[5,6] : *κόστος* 6



Αλγόριθμος Kruskal: βήμα 10

Διάταξη πλευρών
κατά αύξουσα τάξη

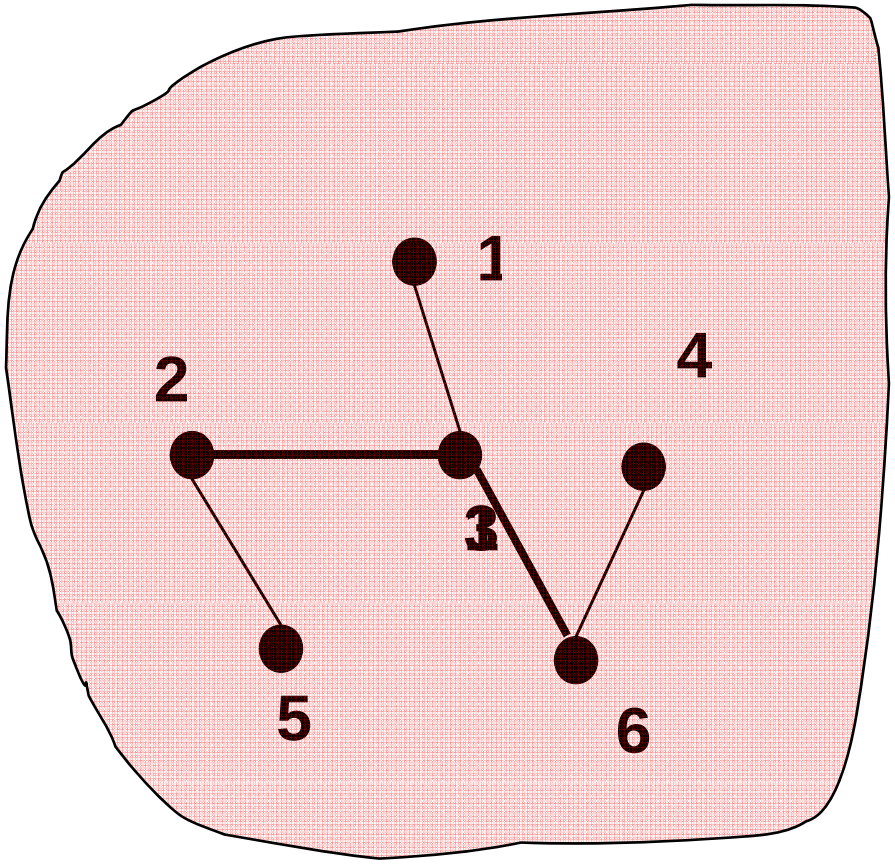
[1,3] : *κόστος* 1
[4,6] : *κόστος* 2
[2,5] : *κόστος* 3
[3,6] : *κόστος* 4
[1,4] : *κόστος* 5
[3,4] : *κόστος* 5
[2,3] : *κόστος* 5
[1,2] : *κόστος* 6
[3,5] : *κόστος* 6
[5,6] : *κόστος* 6



Αλγόριθμος Kruskal: βήμα 10

Διάταξη πλευρών
κατά αύξουσα τάξη

[1,3] : *κόστος* 1
[4,6] : *κόστος* 2
[2,5] : *κόστος* 3
[3,6] : *κόστος* 4
[1,4] : *κόστος* 5
[3,4] : *κόστος* 5
[2,3] : *κόστος* 5
[1,2] : *κόστος* 6
[3,5] : *κόστος* 6
[5,6] : *κόστος* 6



Αλγόριθμος Kruskal

```
T ← ∅
while (|T| < n-1) and (E ≠ ∅) do
    e ← smallest edge in E;
    E ← E - {e};
    if (T ∪ {e} has no cycle) then
        T ← T ∪ {e}
if (|T| < n-1) then
    write "network disconnected";
```

(*****)

Πολυπλκότητα: O(mlogn)
Εξακρίβωση σχηματισμού κύκλου σε O(logn)

Αλγόριθμος Kruskal - Δάσος Επικάλυψης

- Αν ο γράφος είναι μη συνεκτικός, για να βρούμε ένα δάσος επικάλυψης με τον αλγόριθμο Prim θα πρέπει να επαναλάβουμε τον αλγόριθμο σε κάθε συνεκτική συνιστώσα

- **Ο αλγόριθμος Kruskal ευρίσκει απ' ευθείας ένα δάσος επικάλυψης ελάχιστου κόστους.**

Αλγόριθμος Kruskal - Πολυπλοκότητα

Ταξινόμηση πλευρών: O(m logn)
(m αριθμός πλευρών)

Εξακρίβωση σχηματισμού κύκλου: O(logn)

Επαναλήψεις: m

O(mlogn) + O(mlogn) =O(mlogn)

δεδομένου ότι $m < n^2$

Σύγκριση των 2 αλγορίθμων

$$\text{Prim} : O(n^2)$$

$$\text{Kruskal} = O(m \log n)$$

$$\pi\upsilon\kappa\nu o i \rightarrow m = O(n^2)$$

$$\Downarrow$$

$$\text{Prim} \rightarrow O(n^2) < O(n^2 \log n)$$

$$\underline{\text{Αραιοί}} \rightarrow m = O(n)$$

$$O(n^2) > O(n \log n) \rightarrow \underline{\text{Kruskal}}$$

Θεώρημα βέλτιστου για το ΔΕΕΚ

$G = (V, E)$ **G συνεκτικός**

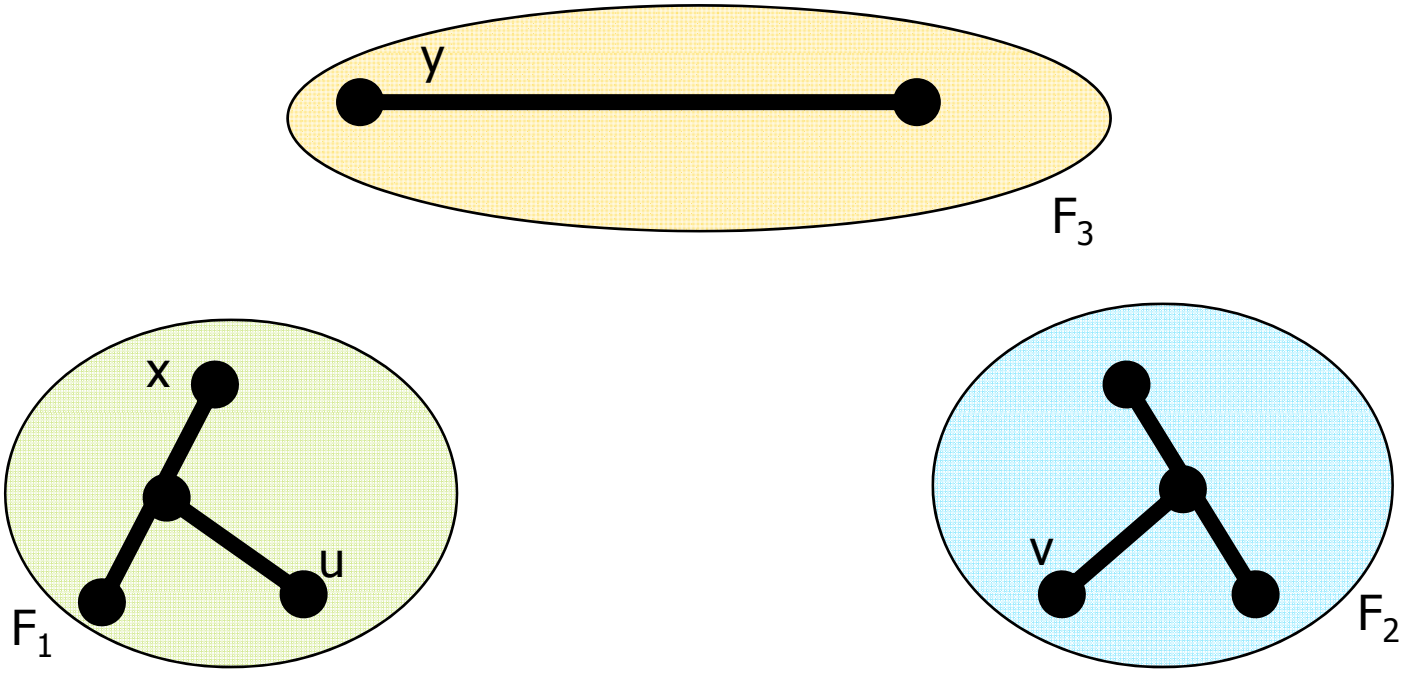
Έστω $F=(F_1, F_2, \ldots F_k)$ $\mu\varepsilon$ $F_i = (V_i, E_i)$ ένα δάσος του γράφου και

$[u, v]$ η πλευρά με το ελάχιστο κόστος έχουσα ένα άκρο στο **V1**

Τότε, ανάμεσα σε όλα τα δένδρα επικάλυψης που περιέχουν αυτό το δάσος, υπάρχει ένα, το καλύτερο το οποίο περιέχει αυτή την πλευρά.

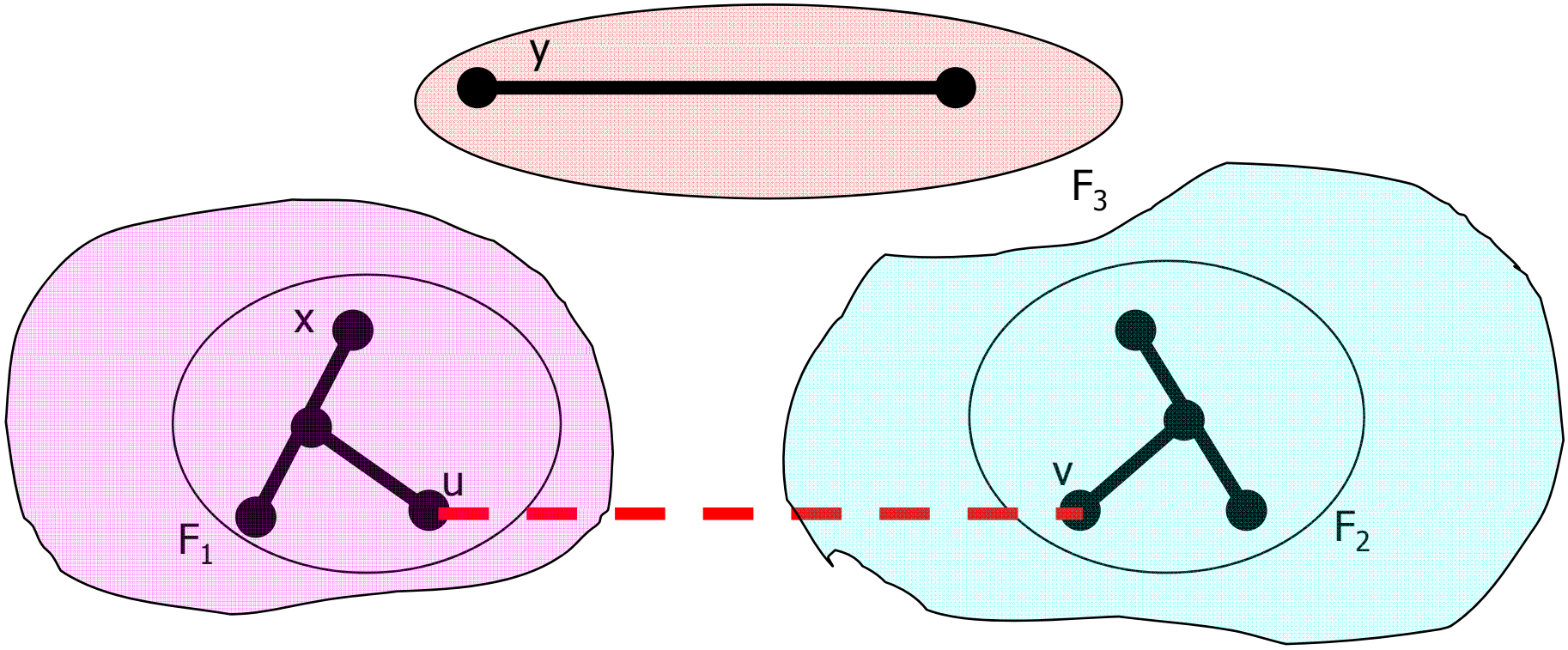
Θεώρημα βέλτιστου για το ΔΕΕΚ - Απόδειξη

Έστω $T = \bigcup_{i=1}^{k} E_i$ οι πλευρές του δάσους F.

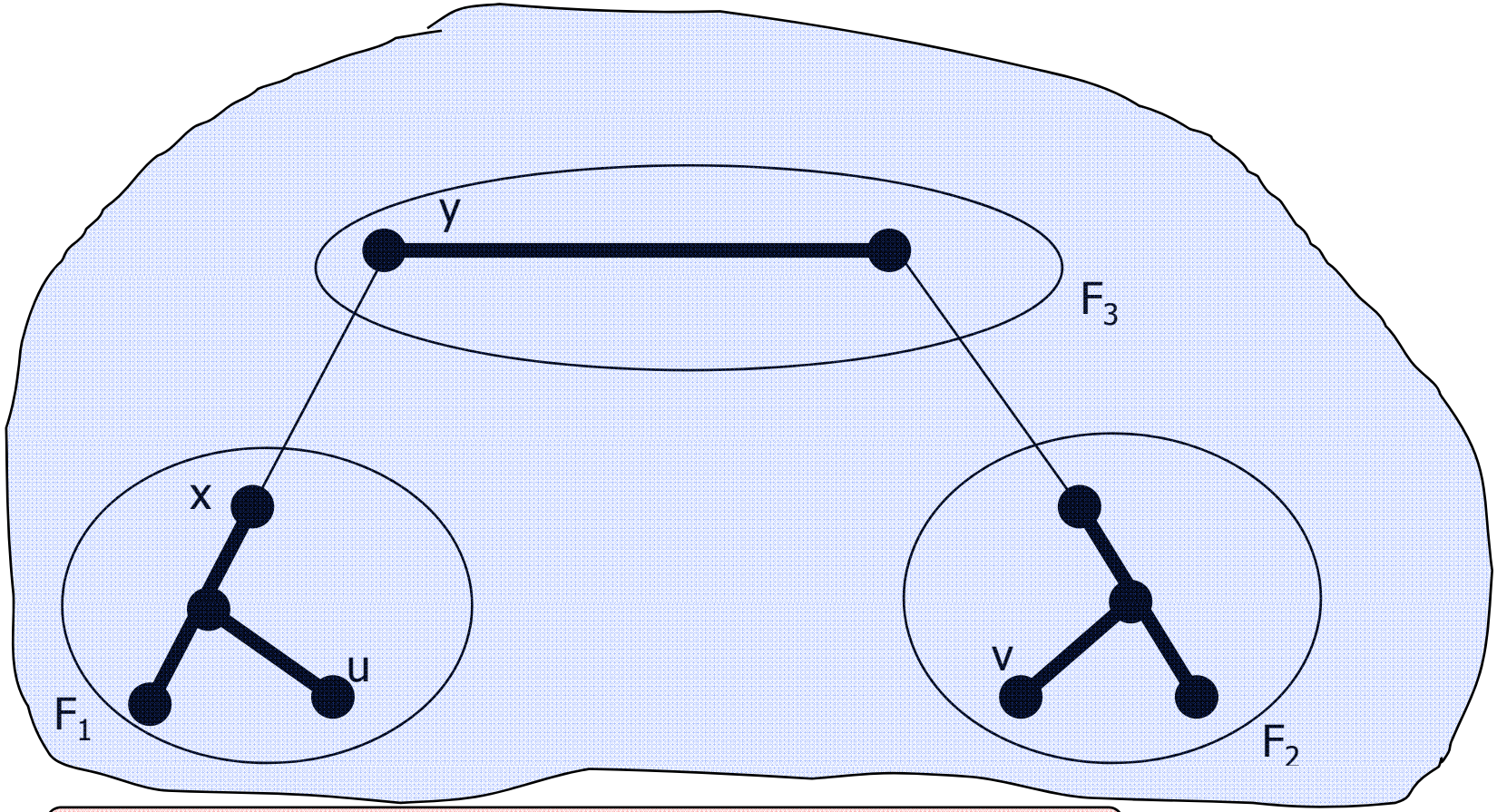


Θεώρημα βέλτιστου για το ΔΕΕΚ - Απόδειξη

Έστω $T = \bigcup_{i=1}^{k} E_i$ οι πλευρές του δάσους F.



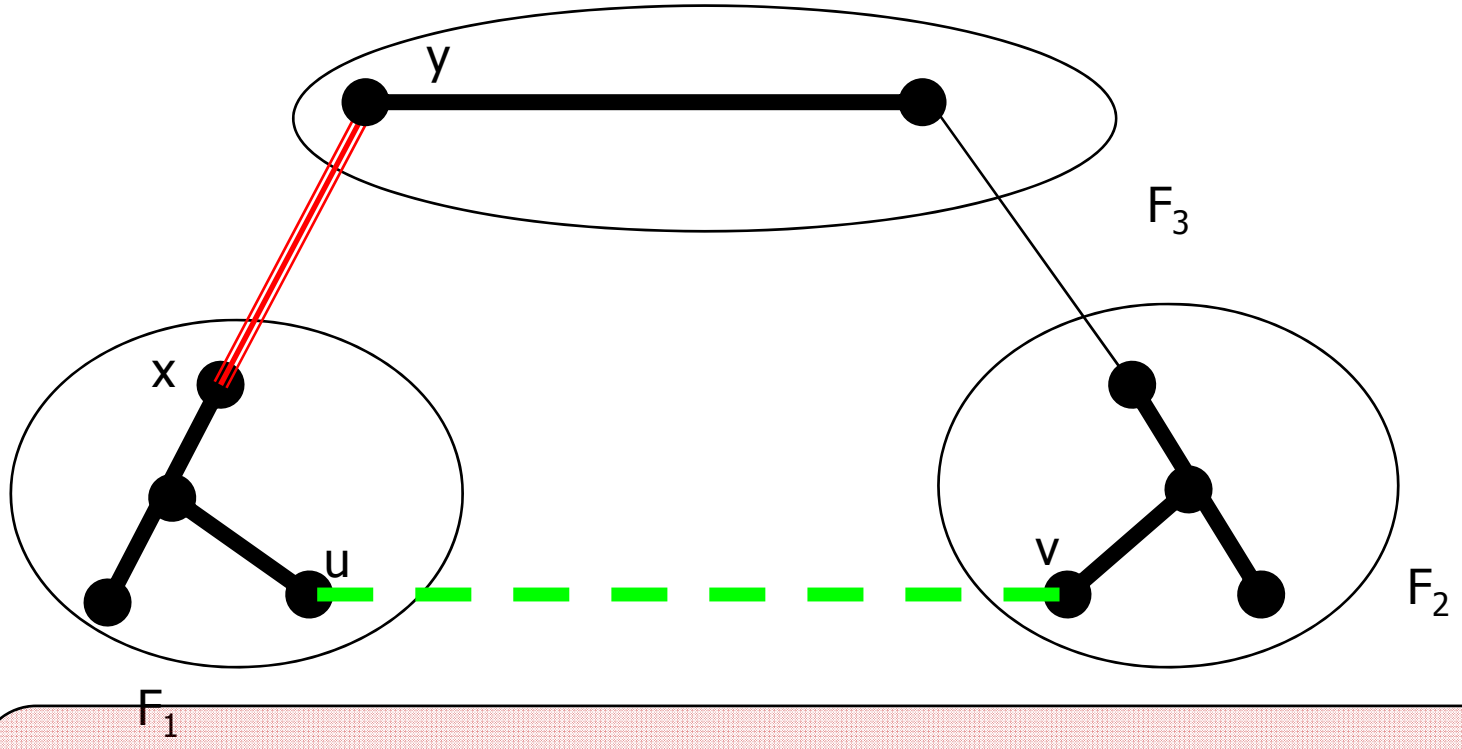
Θεώρημα βέλτιστου για το ΔΕΕΚ - Απόδειξη



Ai δέντρο επικάλυψης που περιέχει το T

A δέντρο επικάλυψης περιέχον το T αλλά όχι τη πλευρά [u,v], με K(A) < K(Ai)

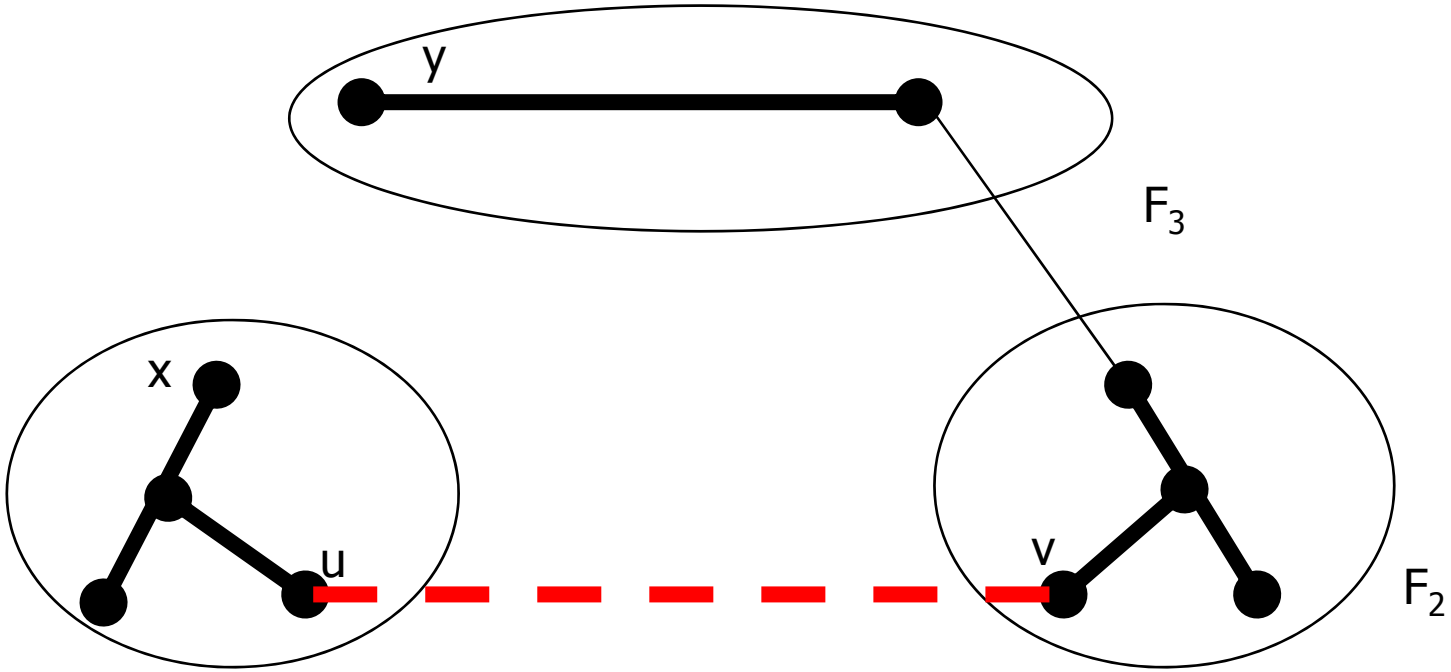
Θεώρημα βέλτιστου για το ΔΕΕΚ - Απόδειξη



Πρόσθεση της πλευράς [u, v] δημιουργεί κύκλο και υπάρχει [x, y] με $x \in V_1$ και $y \in V \setminus V_1$.

$W_{xy} > W_{uv}$ $[u,v] \notin T$.

Θεώρημα βέλτιστου για το ΔΕΕΚ - Απόδειξη



$$A' = (V, E' \cup [u,v] \setminus [x,y]) \qquad W_{xy} > W_{uv}$$

$$\text{Κόστος}(A') < \text{Κόστος}(A)$$

$K(A') < K(A) < K(A_i)$ **αδύνατο**