Δένδρα επικάλυψης ελάχιστου κόστους

(Minimum cost spanning trees)

Δένδρα επικάλυψης μέγιστου βάρους

(Maximum weight spanning trees)

Δέντρα

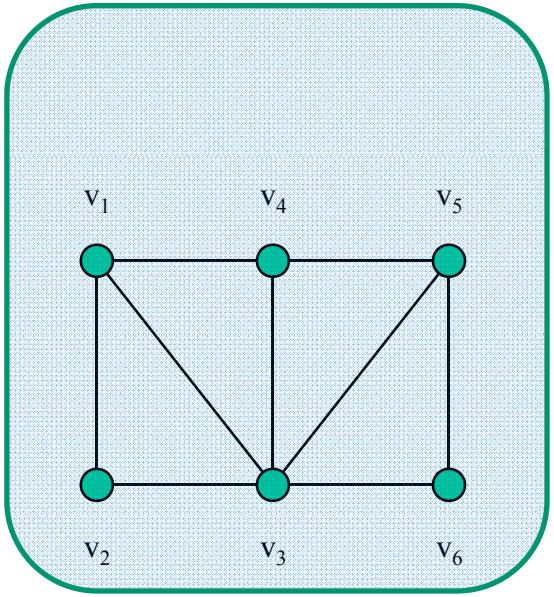
Στη τρέχουσα γλώσσα, ένα δένδρο είναι ένας γράφος G με «**ακριβώς αυτό που πρέπει από πλευρές για να είναι συνεκτικός**»

Ορισμοί (ισοδύναμοι)

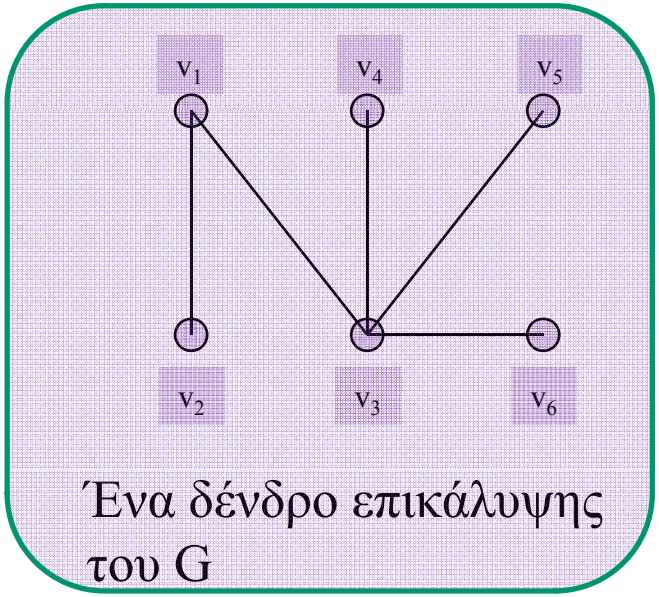
- G είναι **συνεκτικός χωρίς κύκλο**
- G είναι **χωρίς κύκλο και έχει n-1 πλευρές** (όπου n η τάξη του G)
- G είναι **χωρίς κύκλο και εισαγωγή μιας πλευράς** δημιουργεί ένα μόνο κύκλο

Δένδρα επικάλυψης ελάχιστου κόστους: ΔΕΕΚ

Ένα δένδρο επικάλυψης ενός γράφου είναι ένα δένδρο που συνδέει όλους τους κόμβους



Ένας γράφος G



Ένα δένδρο επικάλυψης του G

Δένδρα επικάλυψης ελάχιστου κόστους

Έστω G=(V, E, W) ένας απλός συνεκτικός γράφος με βάρη στις ακμές

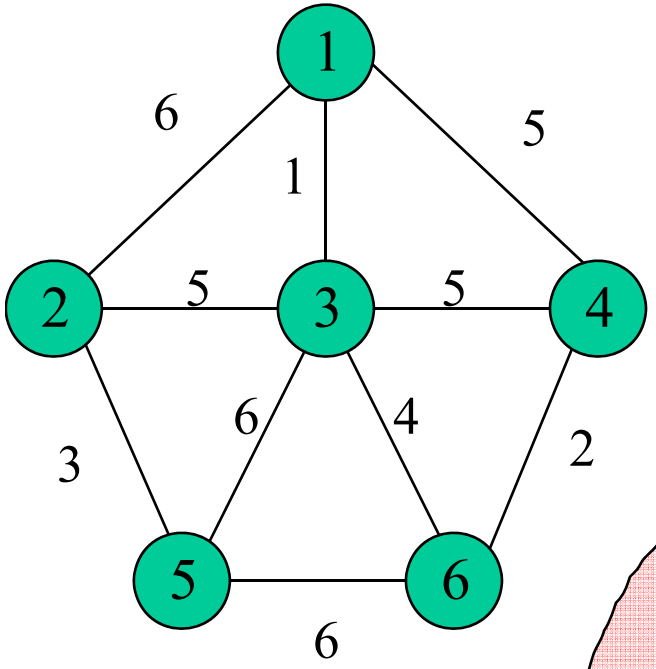
Ζητείται δένδρο επικάλυψης του G, του οποίου το συνολικό κόστος των πλευρών είναι ελάχιστο (ΔΕΕΚ)

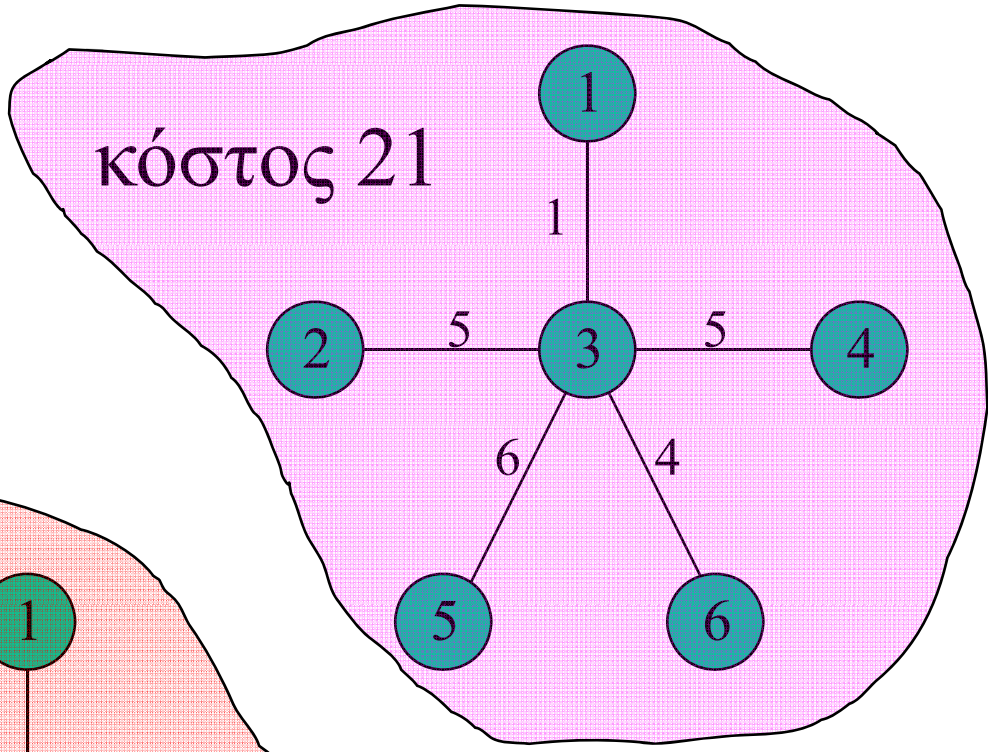
(minimum cost spanning tree)

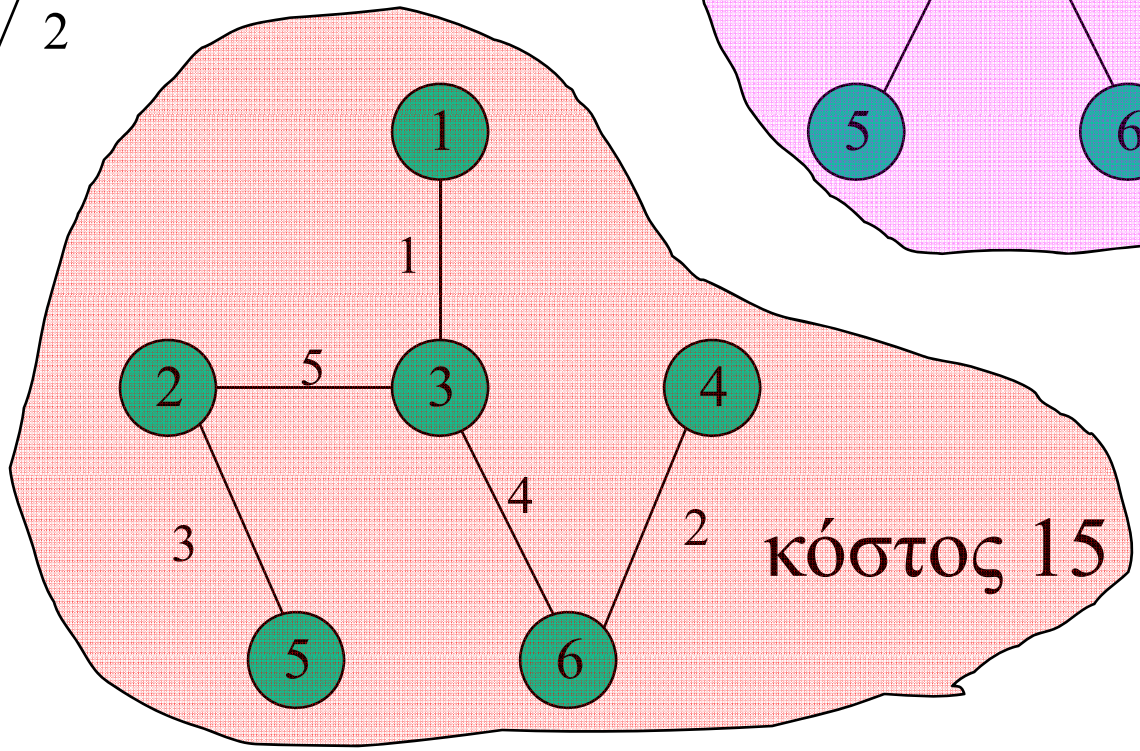
Ζητείται δένδρο επικάλυψης του G, του οποίου το συνολικό κόστος των πλευρών είναι μέγιστο (ΔΕΜΚ)

(maximum cost spanning tree)

Δένδρα επικάλυψης ελάχιστου κόστους





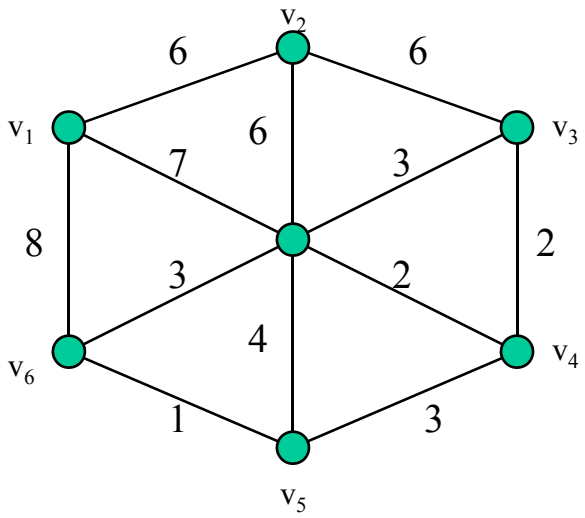


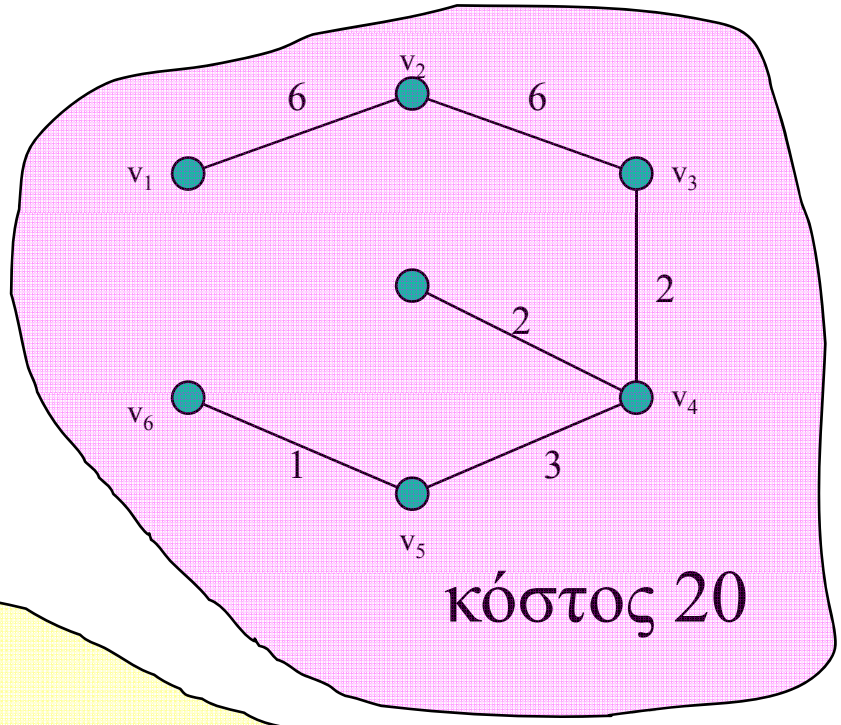
Δένδρα επικάλυψης ελάχιστου κόστους

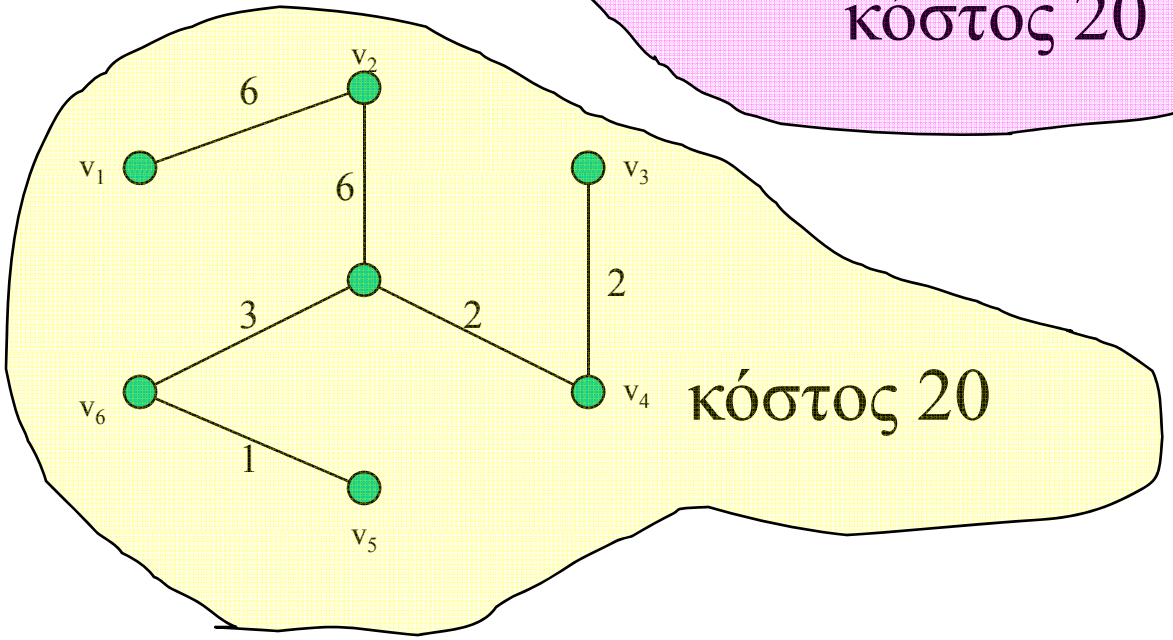
- Αν το βάρος κάθε πλευράς είναι 1, το πρόβλημα συνίσταται στο να βρούμε ένα οποιοδήποτε δένδρο επικάλυψης (κόστους n-1)

- Το ΔΕΕΚ δεν είναι αναγκαία μοναδικό

Δένδρα επικάλυψης ελάχιστου κόστους







Δένδρα επικάλυψης

Σε ένα πλήρη γράφο, ο αριθμός των δένδρων επικάλυψης είναι ακριβώς n^{n-2}, οπου n είναι η τάξη του γράφου (Θεώρημα Cayley)

n	1	2	3	4	5	6	7	8	10
ΔΕΕΚ	1	1	3	16	125	1296	16807	262144	10^8

Το πρόβλημα όμως είναι πολυωνυμικό!

Minimum spanning tree

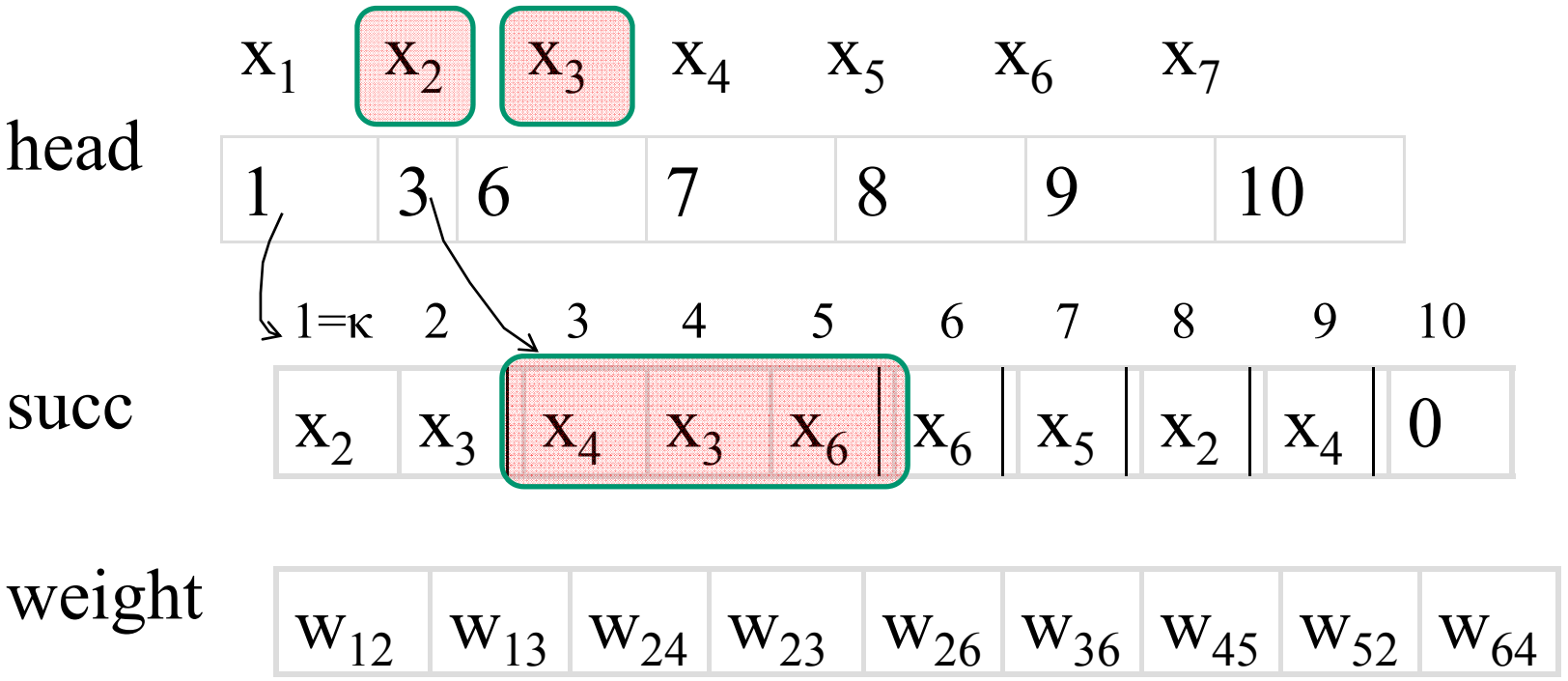
Άσκηση: Αν τα βάρη στο γράφο είναι διαφορετικά τότε το minimum spanning tree είναι μοναδικό.

G=(X,A,W)

X={x_1,x_2,x_3,x_4,x_5,x_6}

(Υπενθύμιση)

Λίστες γειτνίασης με πίνακες



Επόμενοι κόμβοι του x_i

```
for k:=head[ x_i ] to head[ x_{i+1} ]-1
        print(succ[k],w[k])
```

Προβλήματα βέλτιστων μονοπατιών

- Δρομολόγηση μιας οντότητας ανάμεσα σε δύο σημεία ενός δικτύου έτσι ώστε να ελαχιστοποιήσουμε ένα κόστος, μια διάρκεια, κτλ.

Εφαρμογές: Χρονοπρογραμματισμός (διαχείριση εργασιών, προγραμμάτων,...)

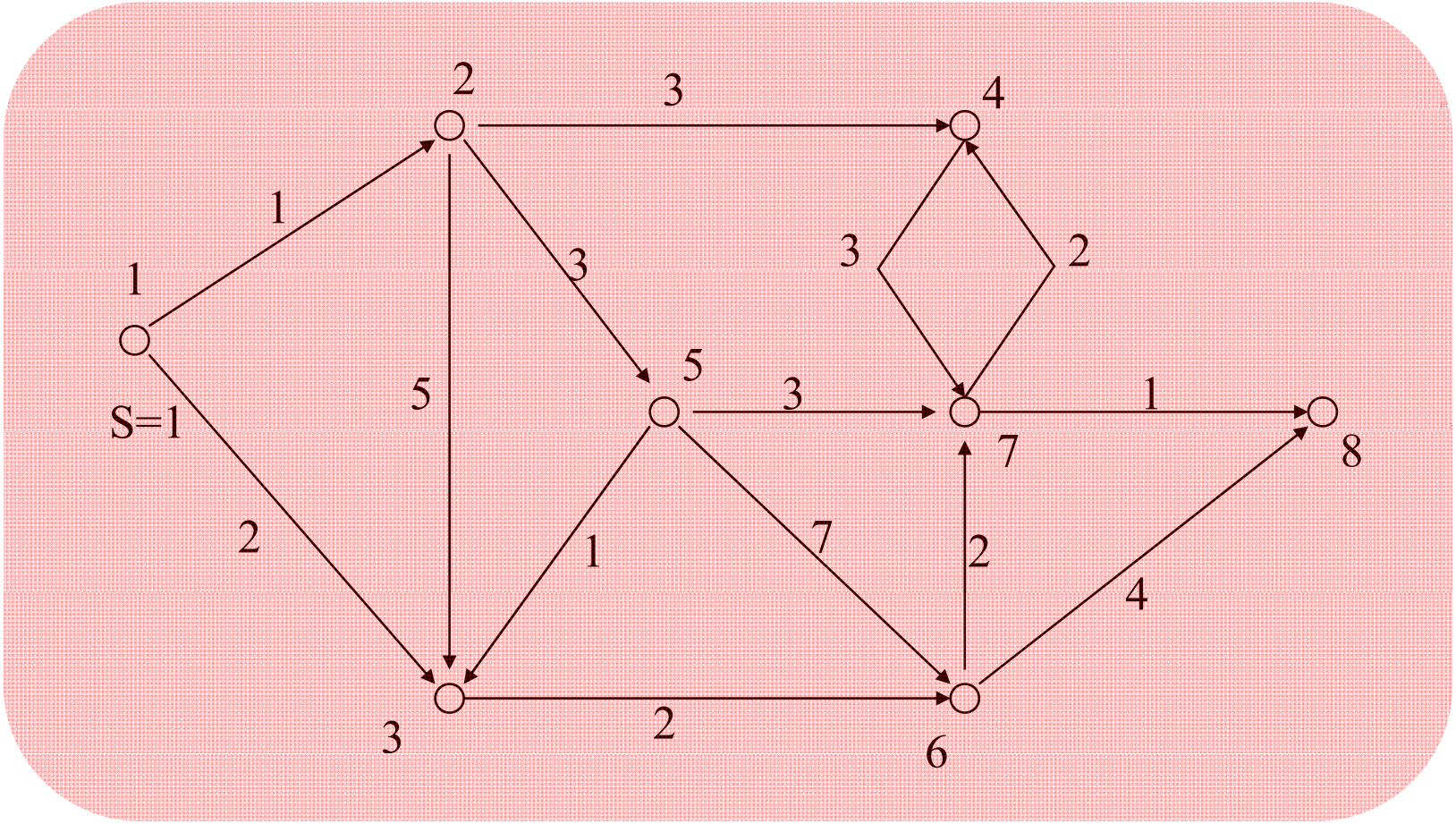
Κατευθυνόμενος γράφος

- Κατευθυνόμενος γράφος με βάρη:

  $G = (X, A, W)$

  - X σύνολο n κόμβων
  - A σύνολο m (κατευθυνόμενων) πλευρών
  - W(i, j) κόστος (κατευθυνόμενης) πλευράς (x_i, y_j)

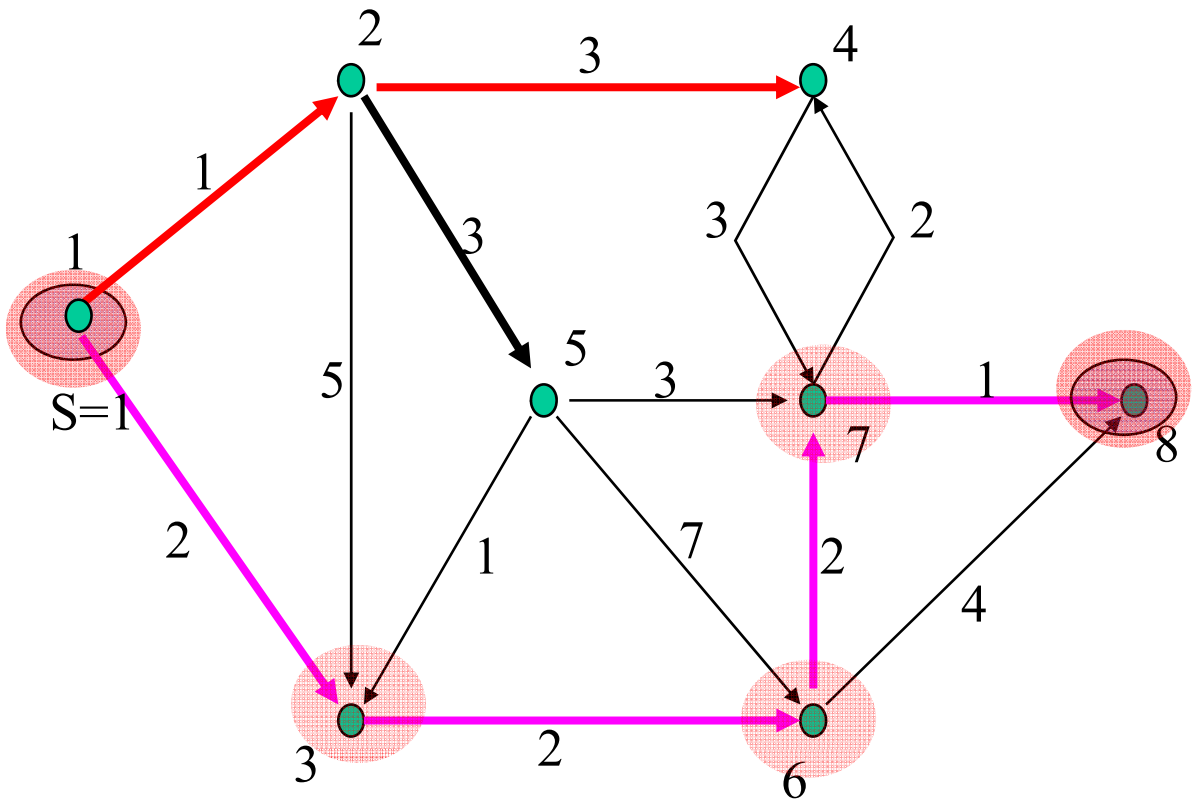
Παράδειγμα: G= (X,A,W)



Μονοπάτια

- Κόστος ενός μονοπατιού που συνδέει δύο κόμβους του είναι το άθροισμα των κοστών των (κατευθυνόμενων) πλευρών του.

Κόστος μονοπατιού



$1 \rightarrow 8$ κόστος 7

$1 \rightarrow 6$ κόστος 4

$1 \rightarrow 5$ κόστος 4

$1 \rightarrow 4$ κόστος 4

.

.

Τύποι Προβλημάτων

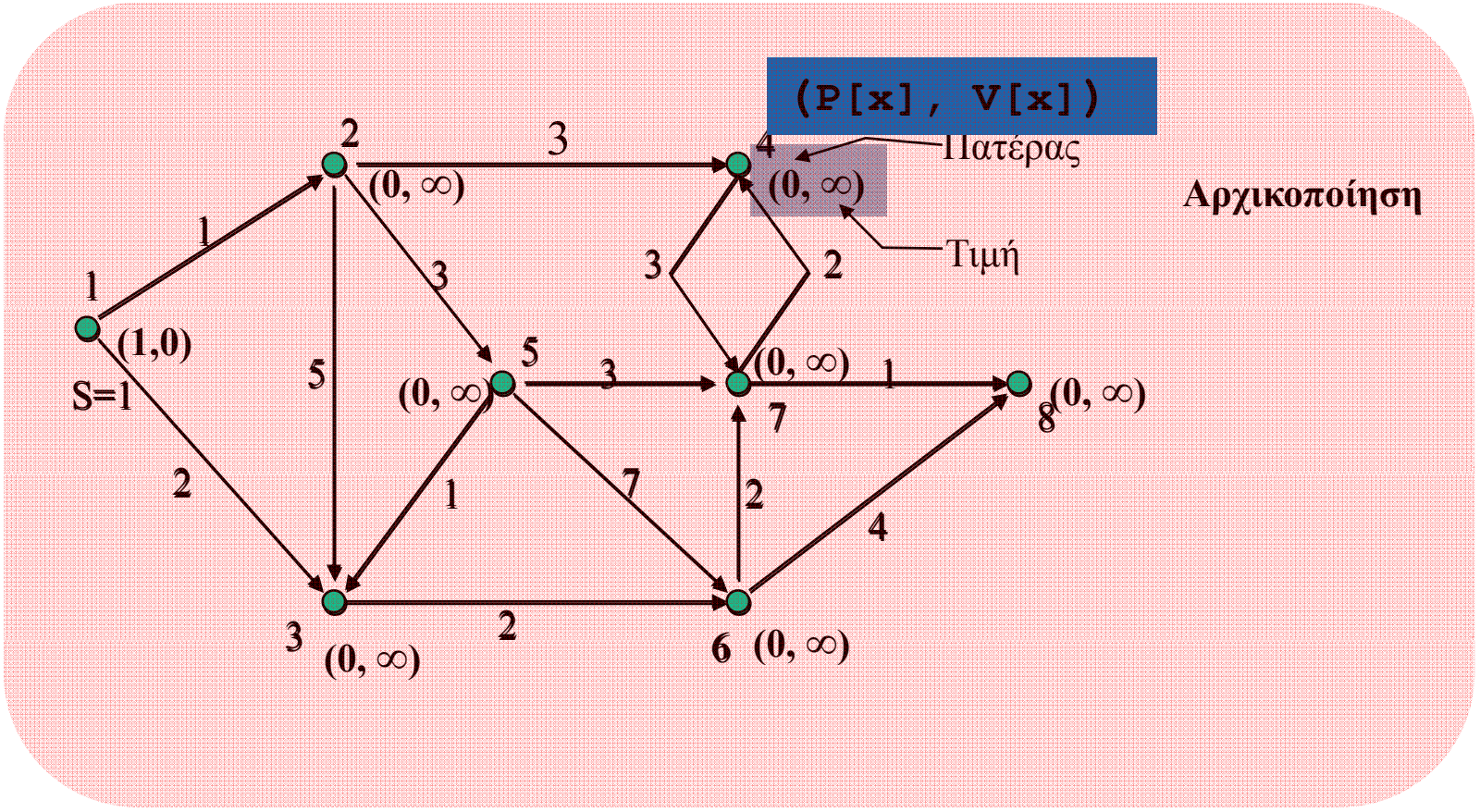
- **Πρόβλημα Α**: Δεδομένων δύο κόμβων s και t, να βρεθεί το συντομότερο μονοπάτι από τον κόμβο s προς τον κόμβο t.
- **Πρόβλημα Β**: Δεδομένου ενός κόμβου αφετηρίας s, να βρεθεί το συντομότερο μονοπάτι από το s προς κάθε άλλο κόμβο.
- **Πρόβλημα C**: Να βρεθεί το συντομότερο μονοπάτι ανάμεσα σε κάθε ζευγάρι κόμβων.

Αλγόριθμοι με οριστικοποίηση ετικετών (Greedy για πρόβλημα Β)

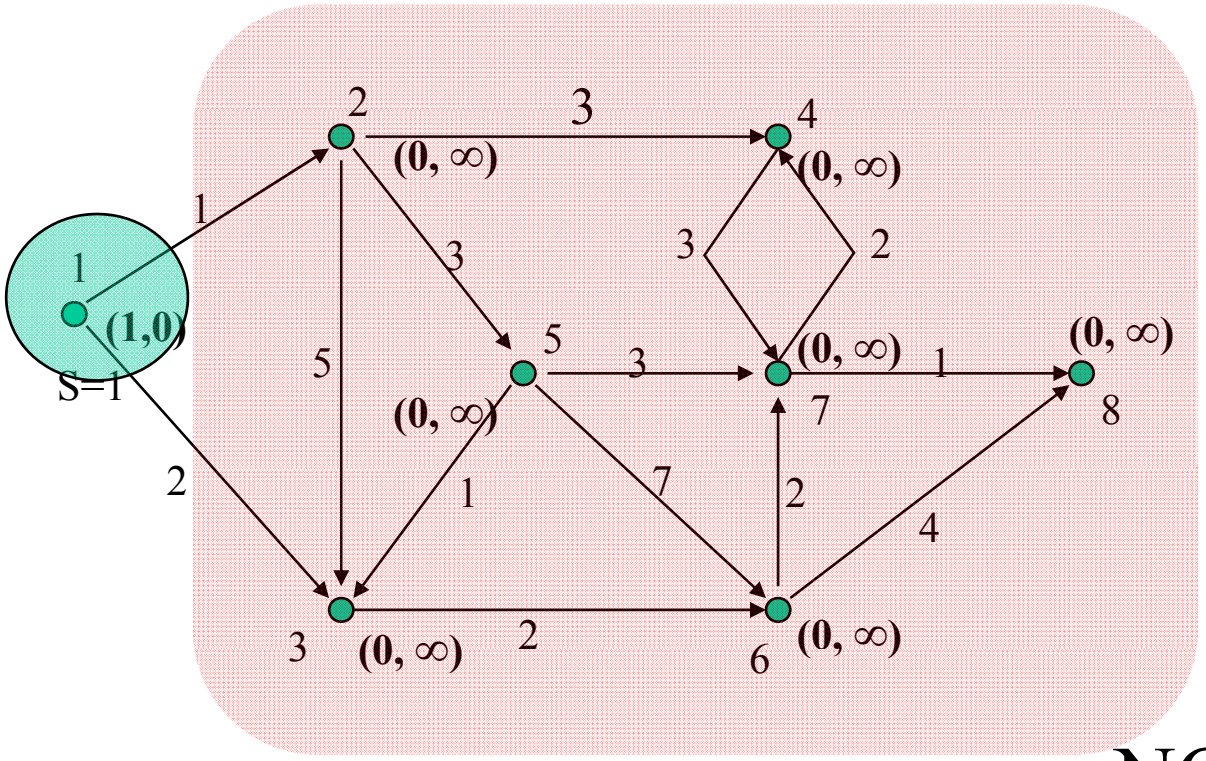
- Γράφος G=(X, A, w), |X| = n, |A| = m και w(i,j)≥0
- Για κάθε κόμβο **x** : V[x], τιμή του συντομότερου μονοπατιού από τον κόμβο αφετηρία **s** προς τον κόμβο **x**.
- Διατήρηση για κάθε κόμβο **x** προσπελάσιμο ξεκινώντας από τον κόμβο **s** : P[x].
- Σε κάθε επανάληψη, ένας κόμβος **x** δέχεται την οριστική του ετικέτα (θα λέμε ότι είναι φιξαρισμένος). Ένας πίνακας **FIXE[ ]** θα δείχνει τους φιξαρισμένους κόμβους.

Αλγόριθμος Dijkstra

παράδειγμα



Αλγόριθμος Dijkstra

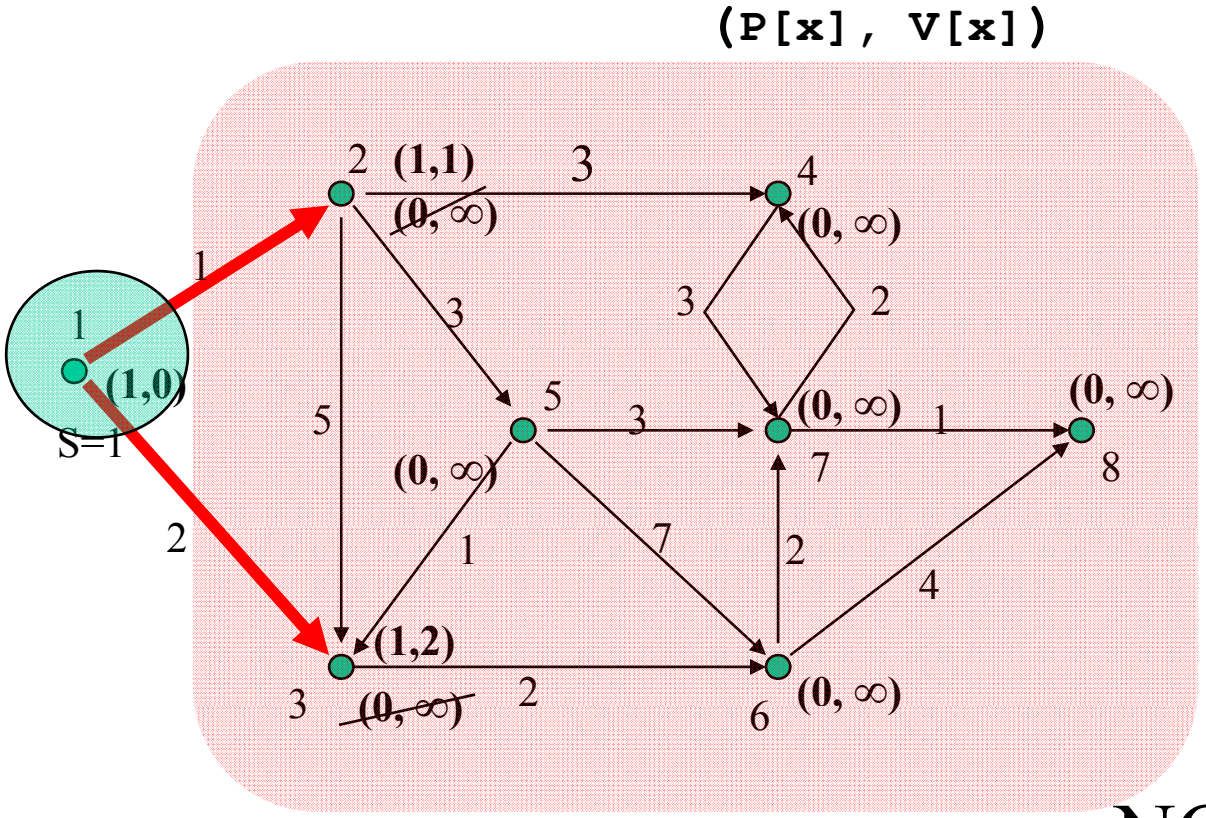


Αρχικοποίηση

Βήμα 1 → επέλεξε κόμβο 1

NOT FIXED

Αλγόριθμος Dijkstra



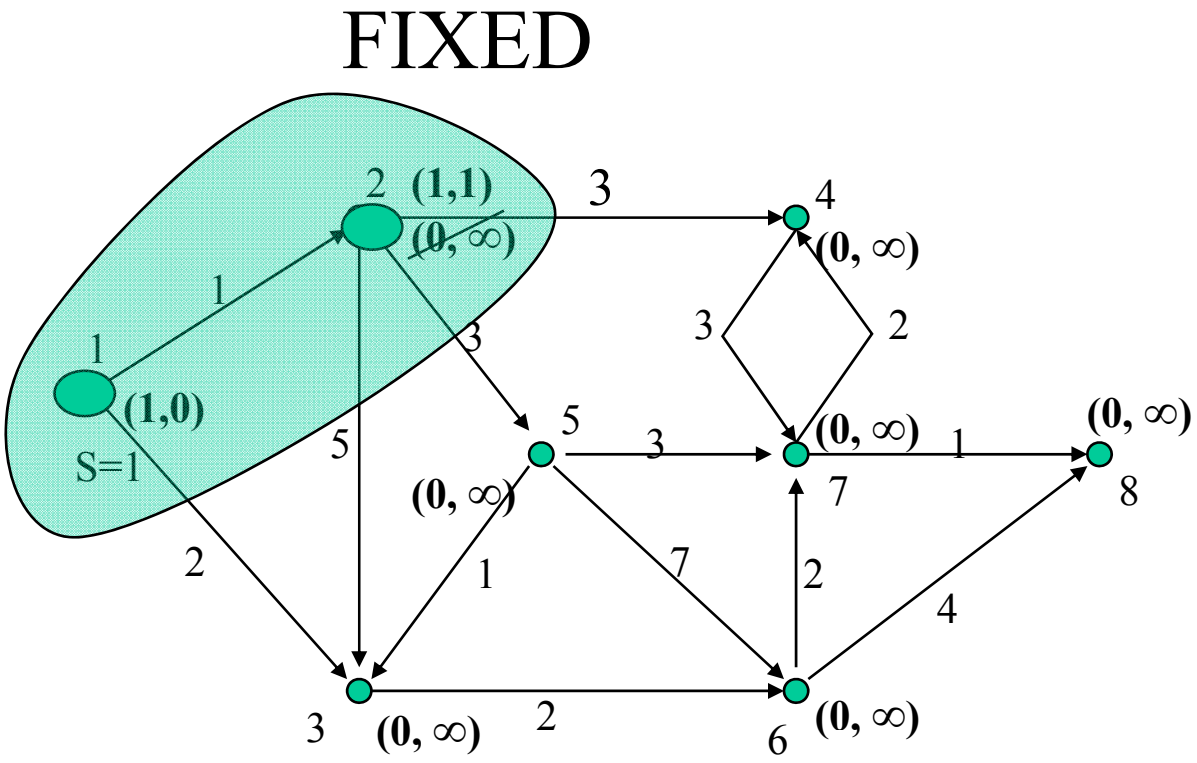
Αρχικοποίηση

Βήμα 1 → επέλεξε κόμβο 1

(ενημέρωση γειτόνων)

NOT FIXED

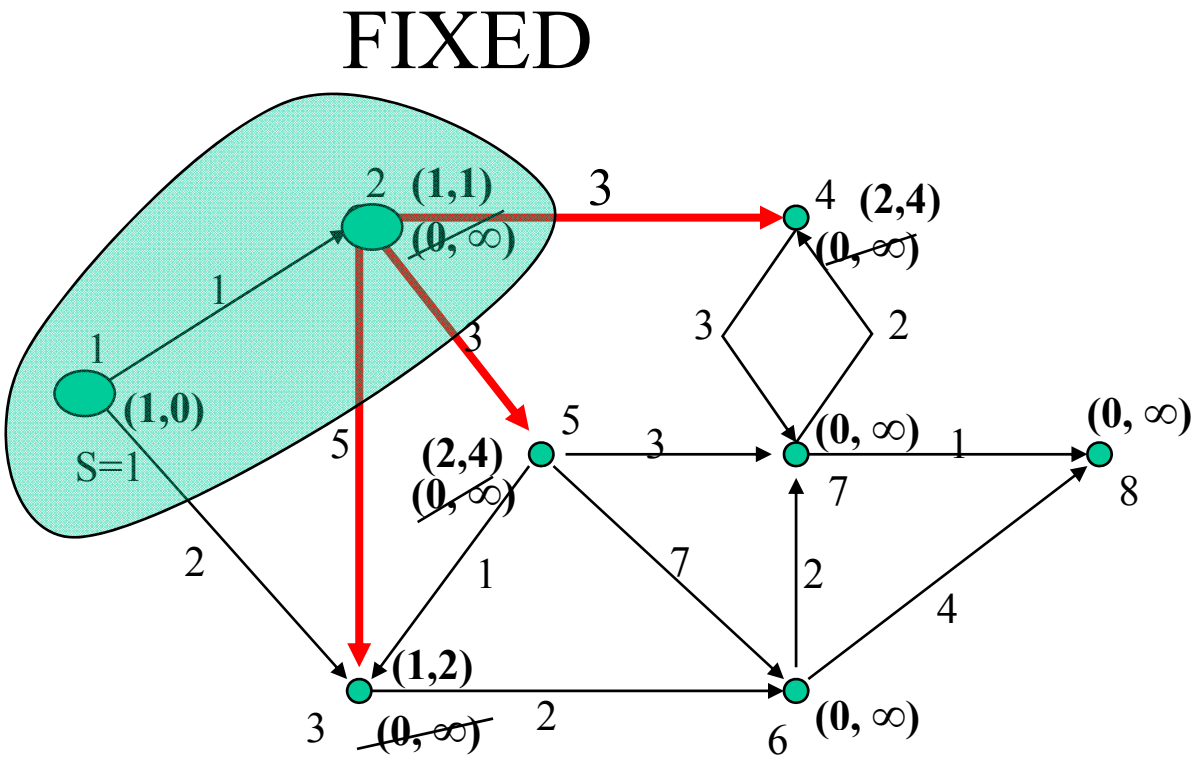
Αλγόριθμος Dijkstra



Αρχικοποίηση

Βήμα 1 → επέλεξε κόμβο **1**
Βήμα 2 → επέλεξε κόμβο **2**

Αλγόριθμος Dijkstra

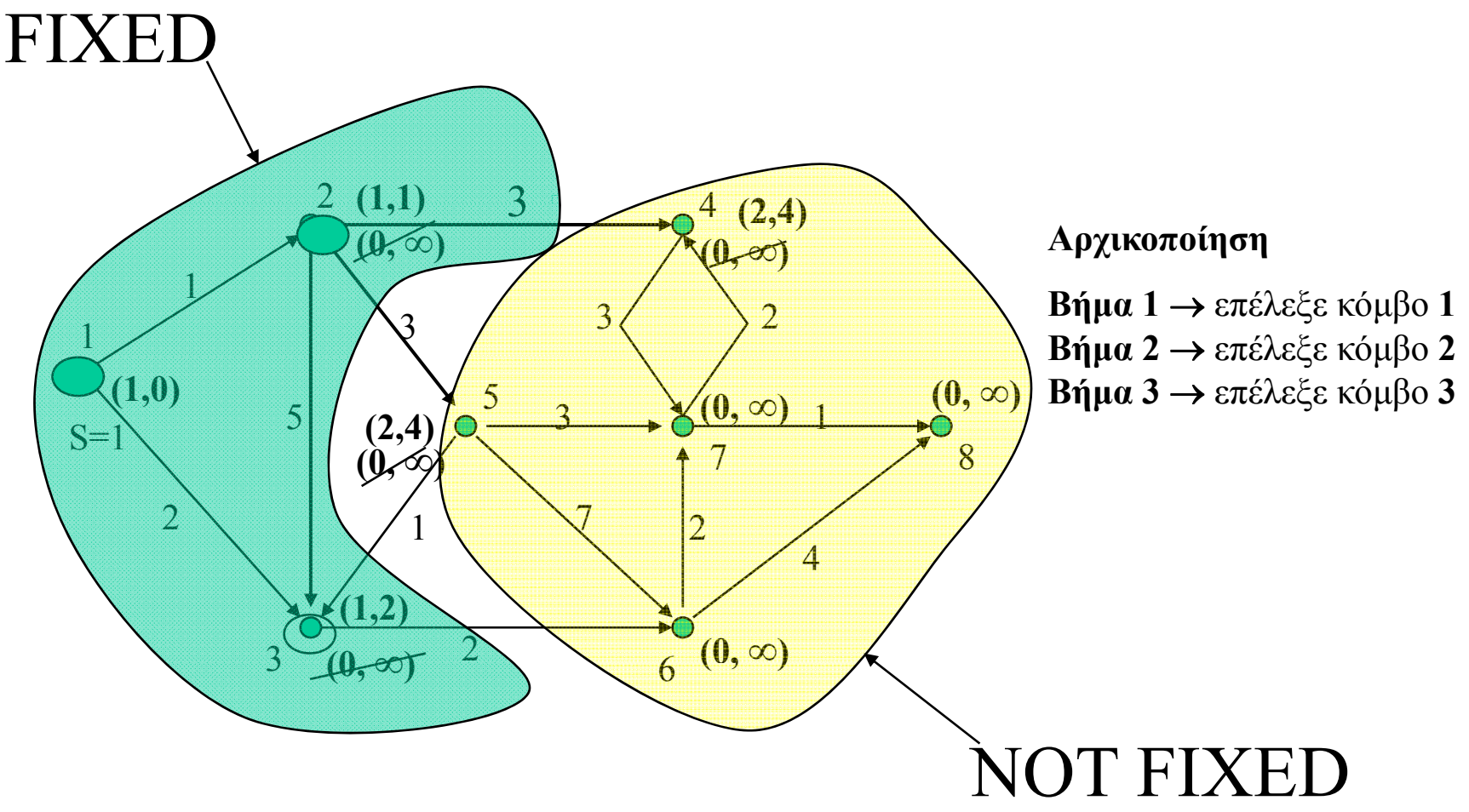


Αρχικοποίηση

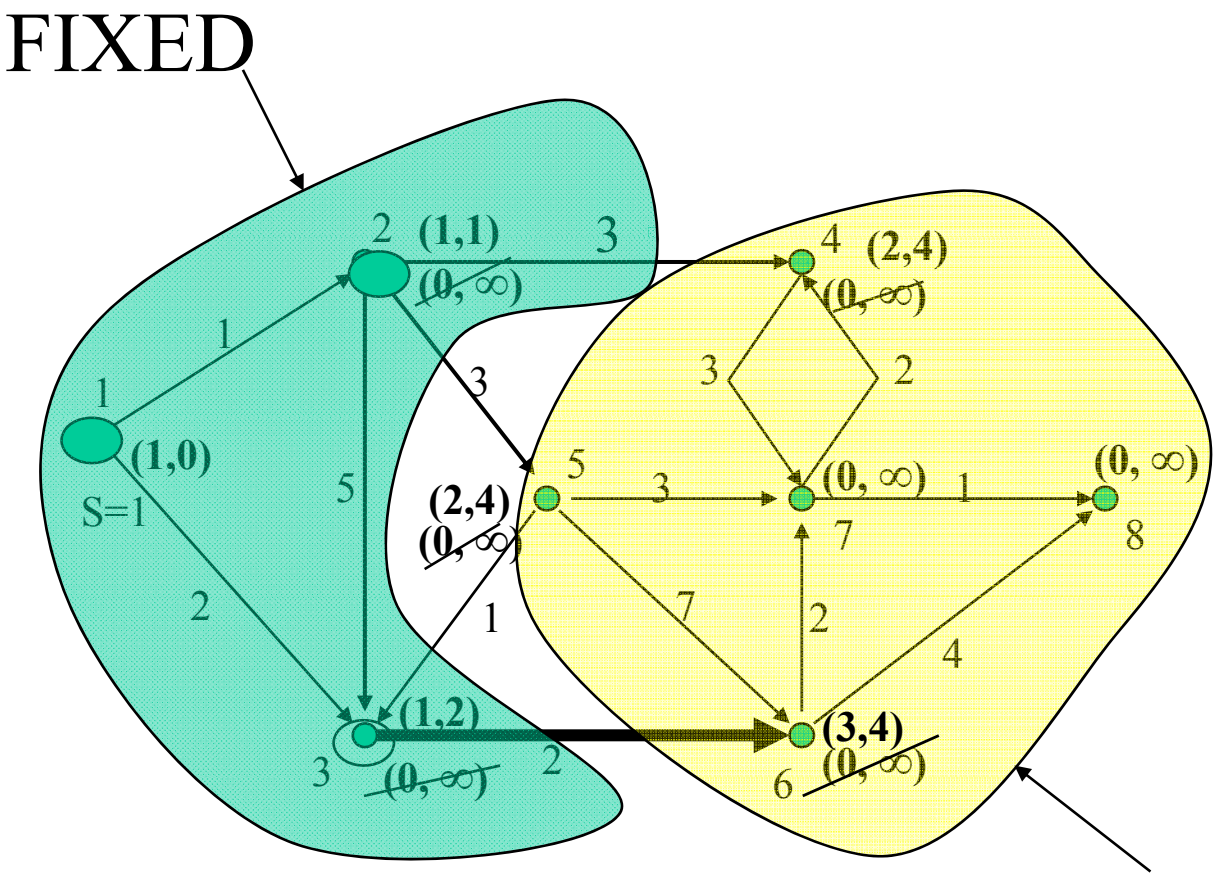
Βήμα 1 → επέλεξε κόμβο **1**
Βήμα 2 → επέλεξε κόμβο **2**

(ενημέρωση γειτόνων του 2)

Αλγόριθμος Dijkstra



Αλγόριθμος Dijkstra

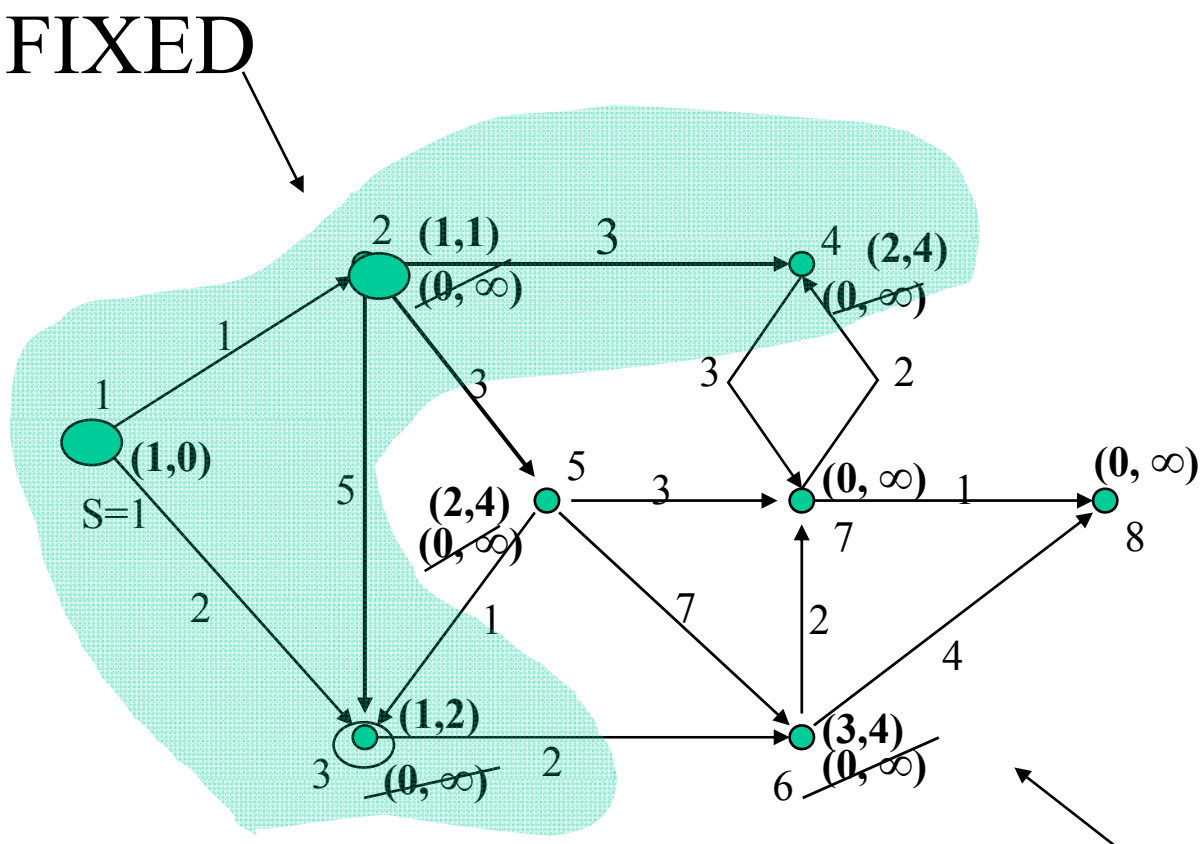


Αρχικοποίηση

Βήμα 1 → επέλεξε κόμβο **1**
Βήμα 2 → επέλεξε κόμβο **2**
Βήμα 3 → επέλεξε κόμβο **3**

(ενημέρωση γειτόνων του 3)

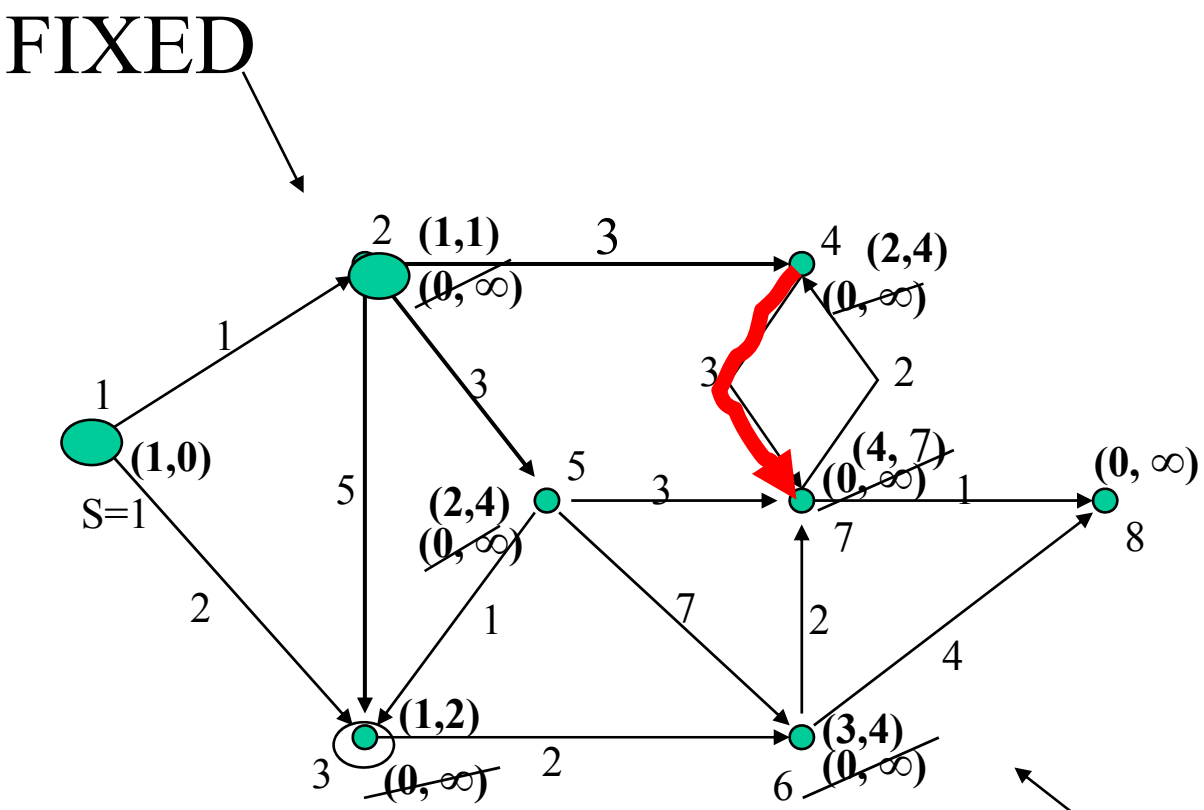
Αλγόριθμος Dijkstra



Αρχικοποίηση

Βήμα 1 → επέλεξε κόμβο **1**
Βήμα 2 → επέλεξε κόμβο **2**
Βήμα 3 → επέλεξε κόμβο **3**
Βήμα 4 → επέλεξε κόμβο **4**

Αλγόριθμος Dijkstra

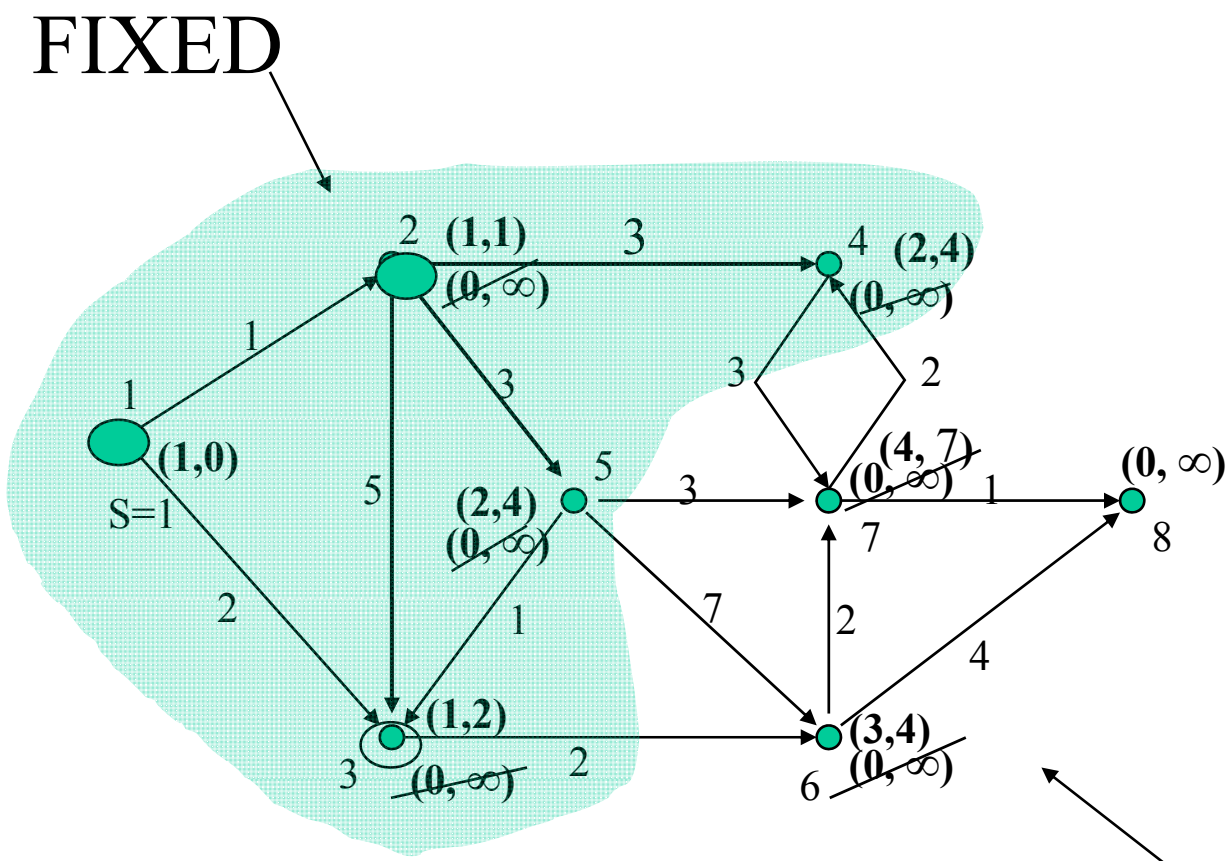


Αρχικοποίηση

Βήμα 1 → επέλεξε κόμβο **1**
Βήμα 2 → επέλεξε κόμβο **2**
Βήμα 3 → επέλεξε κόμβο **3**
Βήμα 4 → επέλεξε κόμβο **4**

(ενημέρωση γειτόνων του 4)

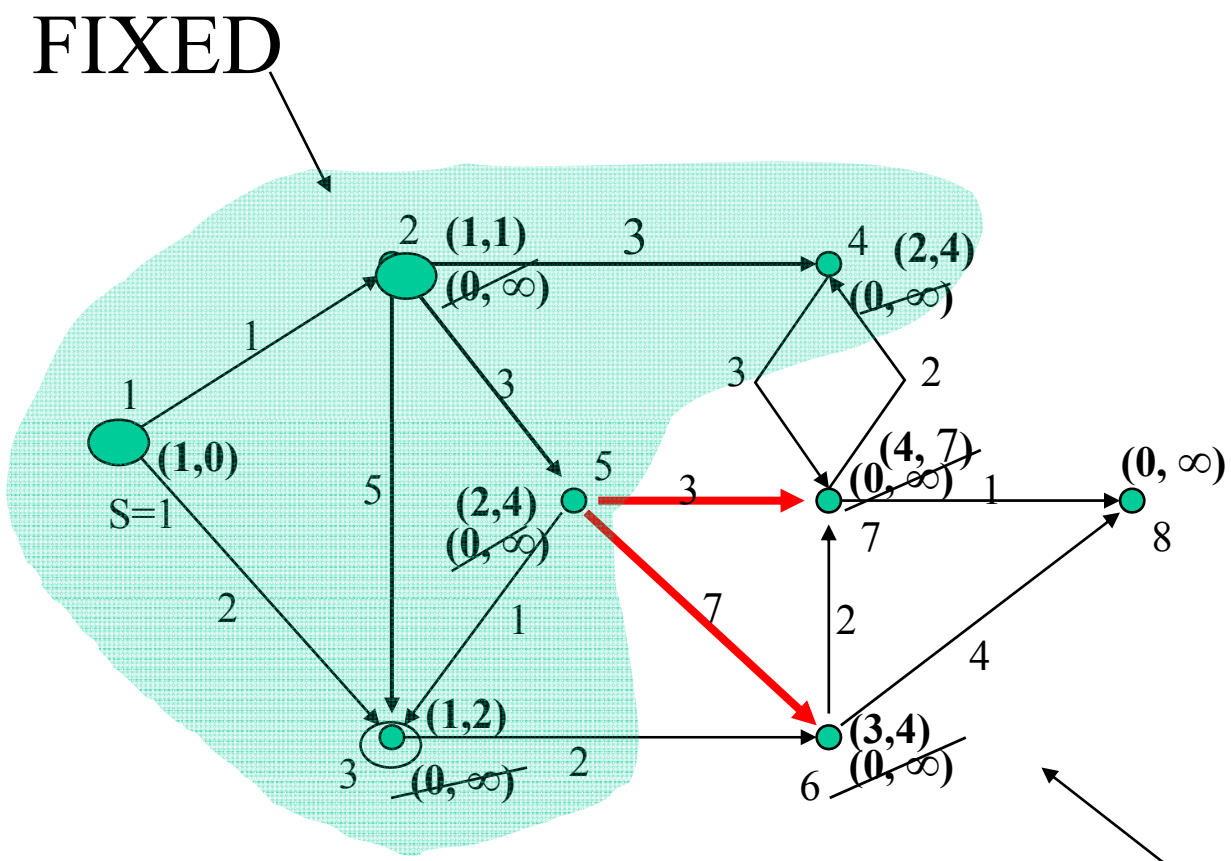
Αλγόριθμος Dijkstra



Αρχικοποίηση

Βήμα 1 → επέλεξε κόμβο **1**
Βήμα 2 → επέλεξε κόμβο **2**
Βήμα 3 → επέλεξε κόμβο **3**
Βήμα 4 →επέλεξε κόμβο **4**
Βήμα 5 → επέλεξε κόμβο **5**

Αλγόριθμος Dijkstra

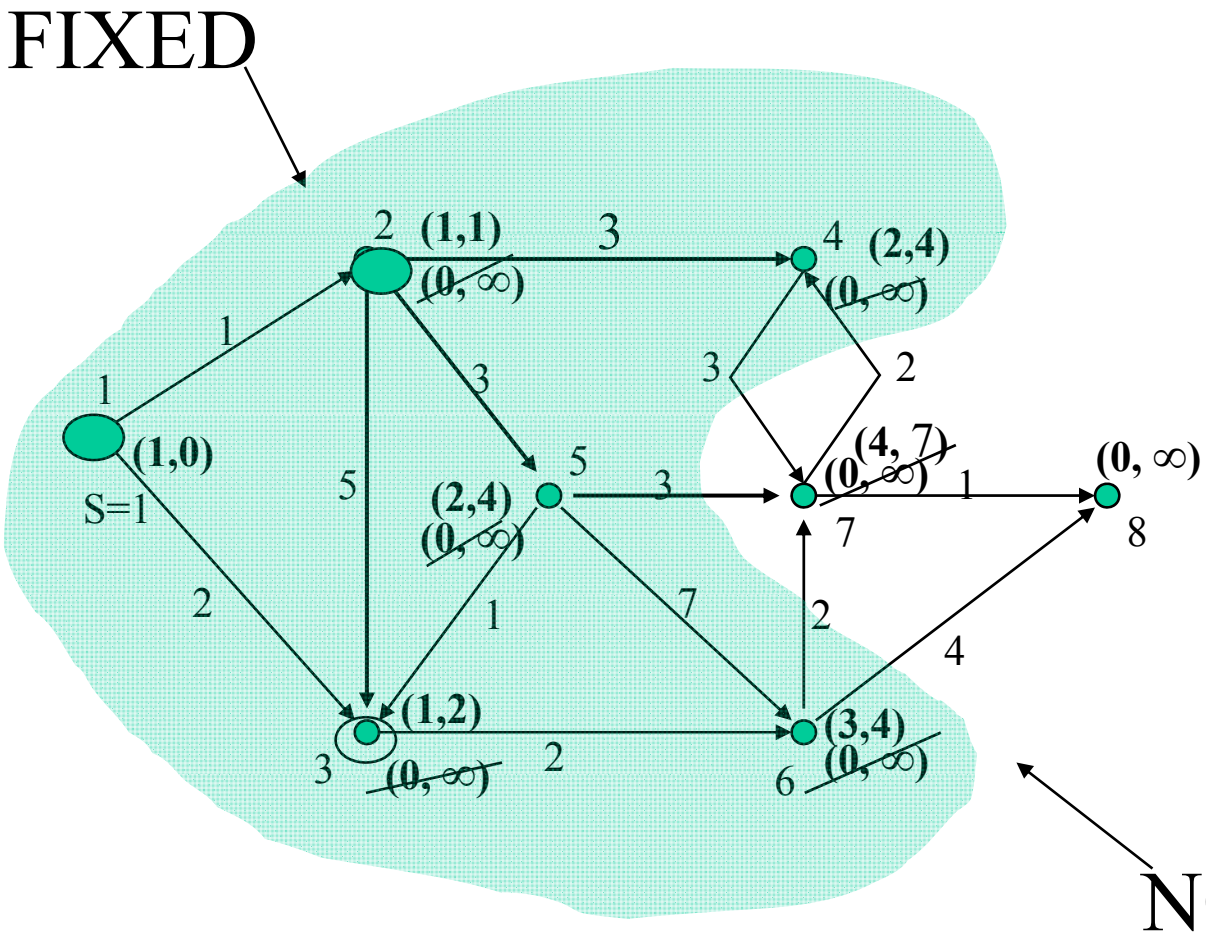


Αρχικοποίηση

Βήμα 1 → επέλεξε κόμβο **1**
Βήμα 2 → επέλεξε κόμβο **2**
Βήμα 3 → επέλεξε κόμβο **3**
Βήμα 4 →επέλεξε κόμβο **4**
Βήμα 5 → επέλεξε κόμβο **5**

(ενημέρωση γειτόνων του 5)

Αλγόριθμος Dijkstra

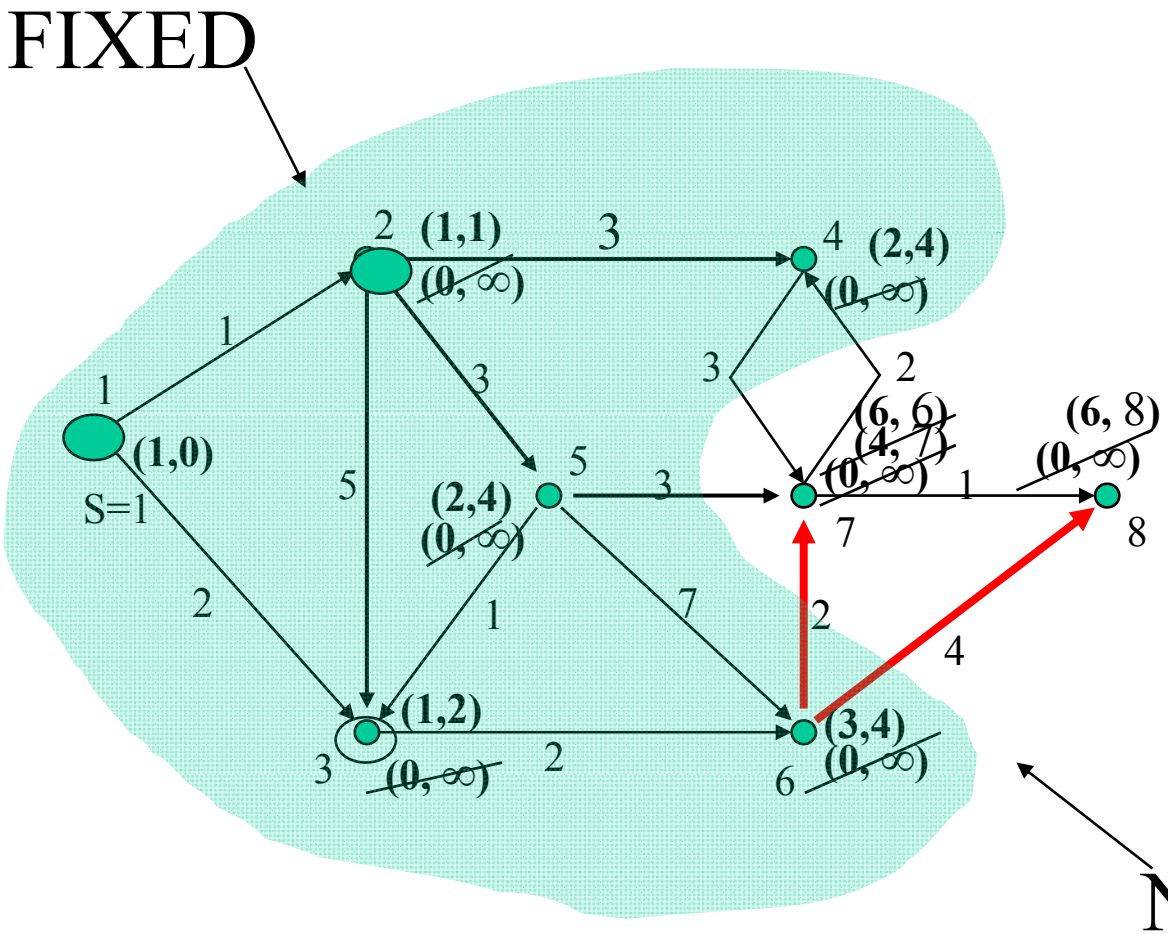


Αρχικοποίηση

Βήμα 1 → επέλεξε κόμβο **1**
Βήμα 2 → επέλεξε κόμβο **2**
Βήμα 3 → επέλεξε κόμβο **3**
Βήμα 4 →επέλεξε κόμβο **4**
Βήμα 5 → επέλεξε κόμβο **5**

Βήμα 6 → επέλεξε κόμβο **6**

Αλγόριθμος Dijkstra



Αρχικοποίηση

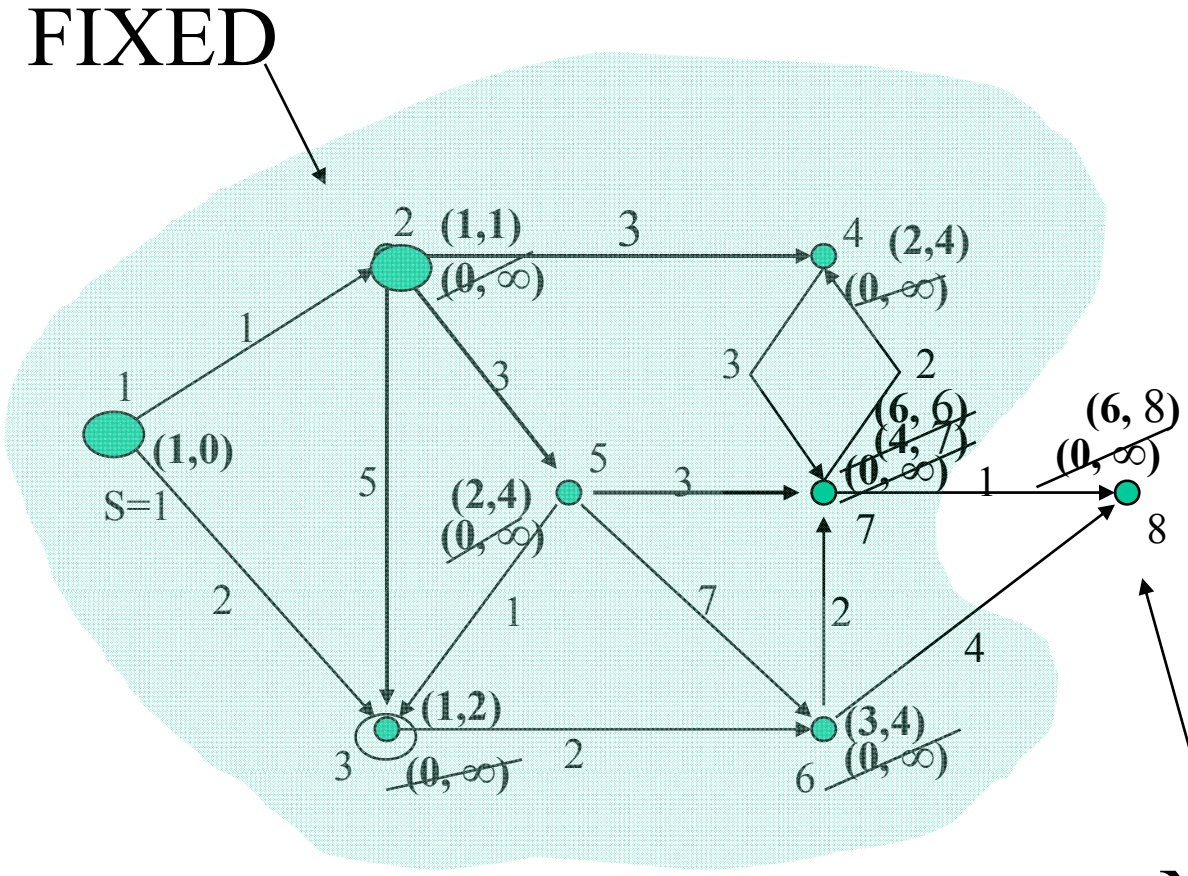
Βήμα 1 → επέλεξε κόμβο **1**
Βήμα 2 → επέλεξε κόμβο **2**
Βήμα 3 → επέλεξε κόμβο **3**
Βήμα 4 →επέλεξε κόμβο **4**
Βήμα 5 → επέλεξε κόμβο **5**

Βήμα 6 → επέλεξε κόμβο **6**

(ενημέρωση γειτόνων του 6)

NOT FIXED

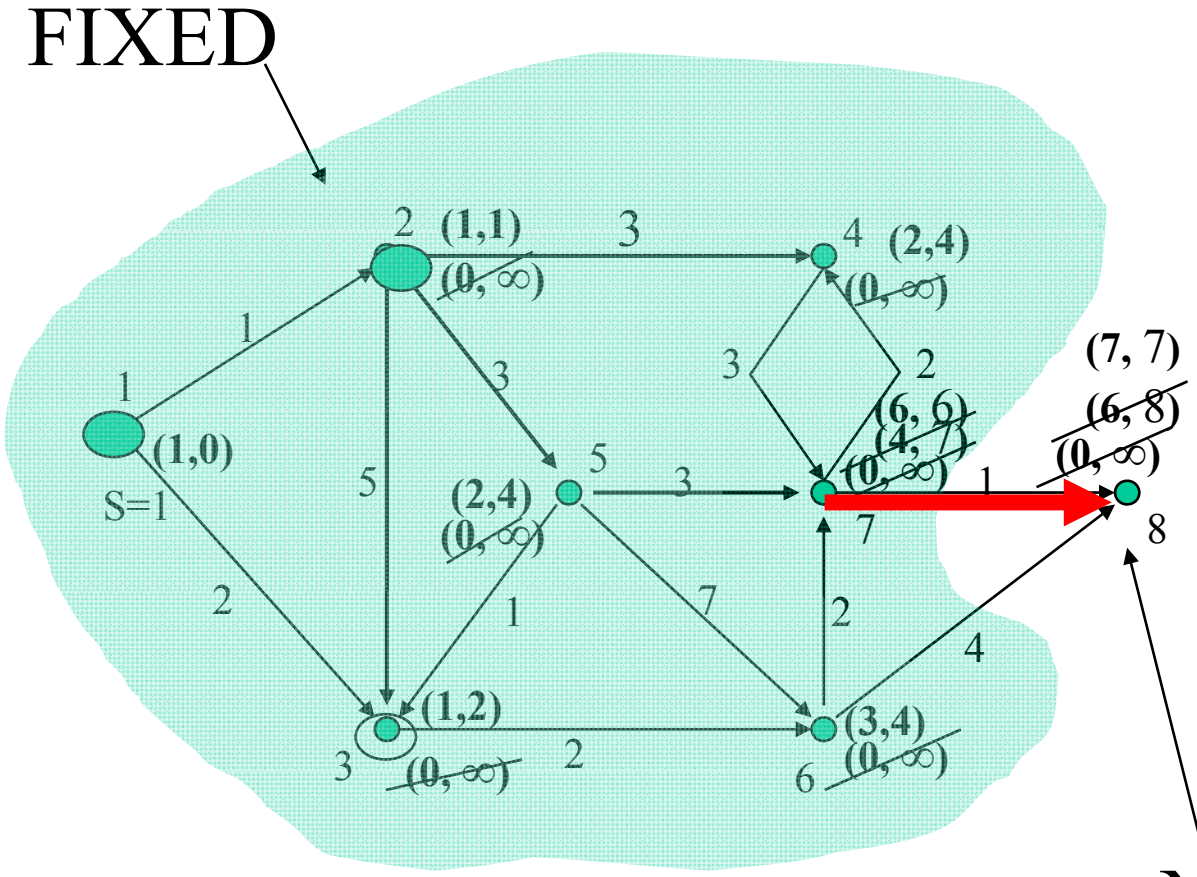
Αλγόριθμος Dijkstra



Αρχικοποίηση

Βήμα 1 → επέλεξε κόμβο **1**
Βήμα 2 → επέλεξε κόμβο **2**
Βήμα 3 → επέλεξε κόμβο **3**
Βήμα 4 →επέλεξε κόμβο **4**
Βήμα 5 → επέλεξε κόμβο **5**
Βήμα 6 → επέλεξε κόμβο **6**
Βήμα 7 → επέλεξε κόμβο **7**

Αλγόριθμος Dijkstra



Αρχικοποίηση

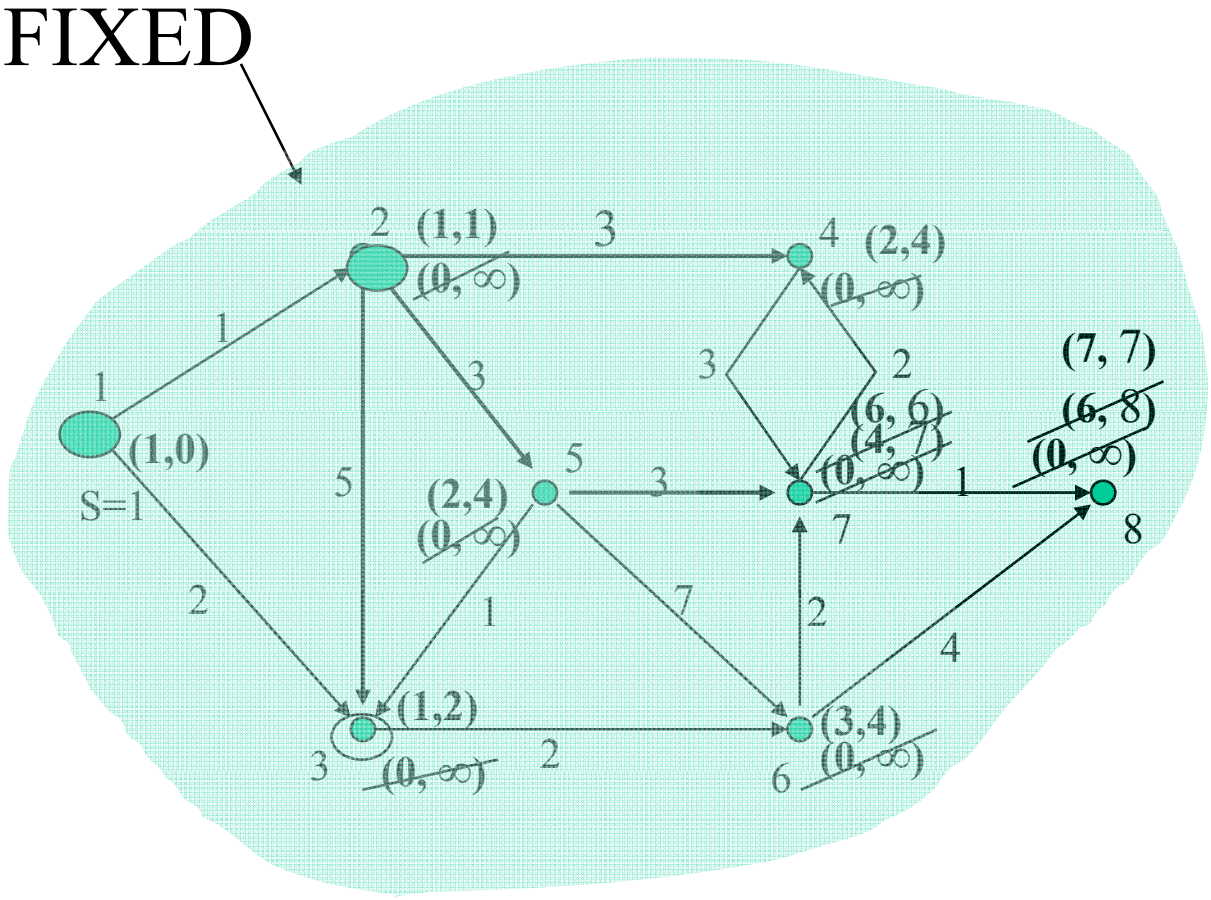
Βήμα 1 → επέλεξε κόμβο **1**
Βήμα 2 → επέλεξε κόμβο **2**
Βήμα 3 → επέλεξε κόμβο **3**
Βήμα 4 →επέλεξε κόμβο **4**
Βήμα 5 → επέλεξε κόμβο **5**

Βήμα 6 → επέλεξε κόμβο **6**

Βήμα 7 → επέλεξε κόμβο **7**

(ενημέρωση γειτόνων του 7)

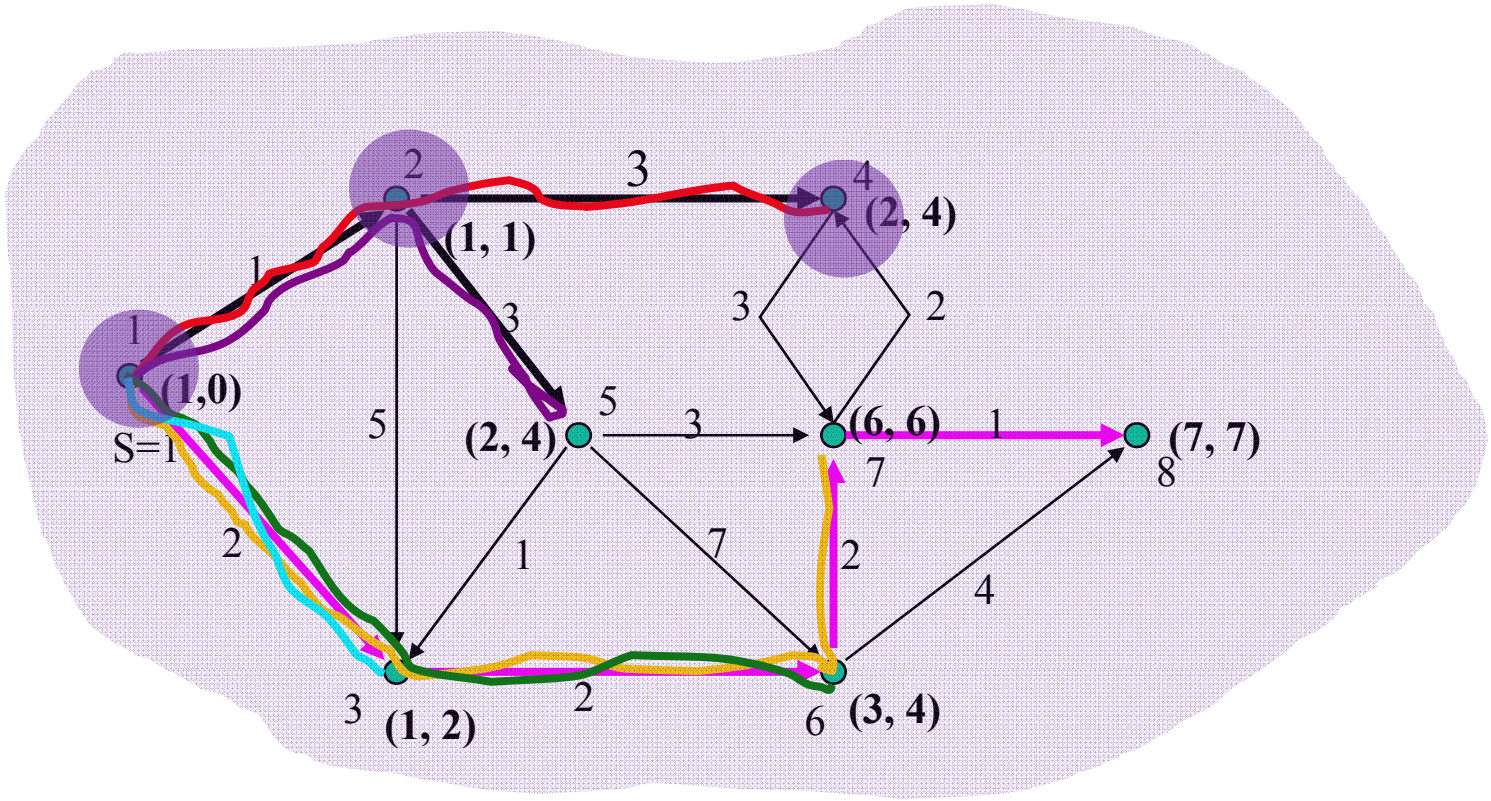
Αλγόριθμος Dijkstra



Αρχικοποίηση

Βήμα 1 → επέλεξε κόμβο **1**
Βήμα 2 → επέλεξε κόμβο **2**
Βήμα 3 → επέλεξε κόμβο **3**
Βήμα 4 →επέλεξε κόμβο **4**
Βήμα 5 → επέλεξε κόμβο **5**
Βήμα 6 → επέλεξε κόμβο **6**
Βήμα 7 → επέλεξε κόμβο **7**
Βήμα 8 → επέλεξε κόμβο **8**

Αλγόριθμος Dijkstra



Κόμβος 4: P[4]=2, P[2]=1, P[1]=1. Μονοπάτι: 1→2→4

Κατευθυνόμενο δέντρο ελάχιστων μονοπατιών

Αλγόριθμος Dijkstra

- Αρχικοποίησε τον πίνακα **V** με ∞
- --#-- **P** με 0
- --#-- **FIXE** σε False
- Θέσε **V[S] = 0** και **P[S] = 1**

Repeat

Until {x|V[x] < ∞ and x ΜΗ φιξαρισμένος} = Ø

Αλγόριθμος Dijkstra

Repeat

Επέλεξε ένα κόμβο x **ΜΗ φιξαρισμένο** και με **ελάχιστο** V[x]

ΑΝ V[x] < ∞ ΤΟΤΕ

Φίξαρε x

Για κάθε y επόμενο του x και ΜΗ **φιξαρισμένο**

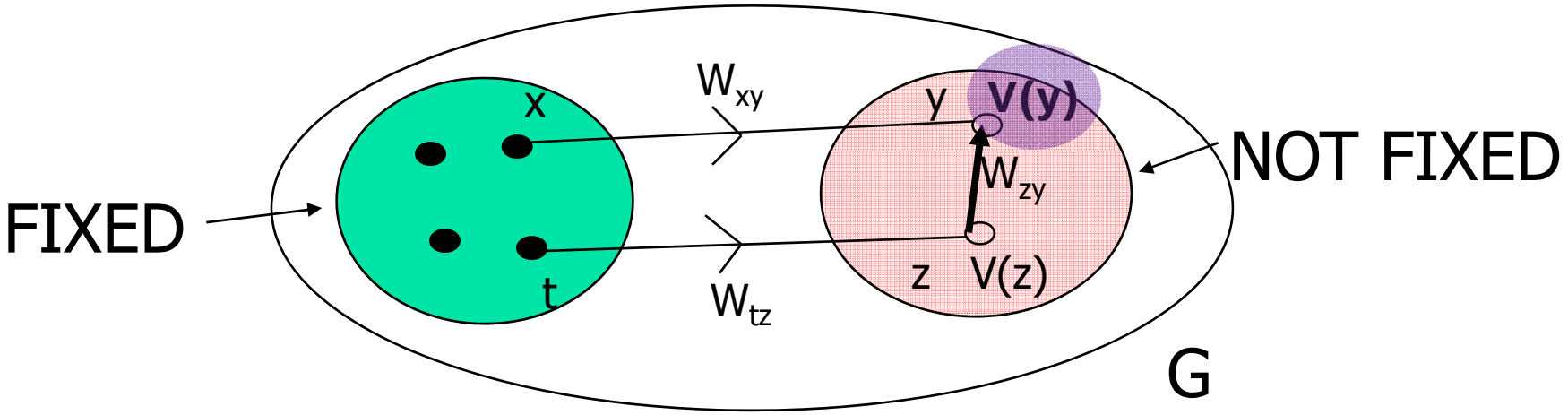
An V[y] **>** V[x] + W(x,y)

ΤΟΤΕ V[y] = V[x] + W(x,y)

Θέσε P[y] = x */*απομνημόνευσε ότι ερχόμαστε στο y απο τον x */*

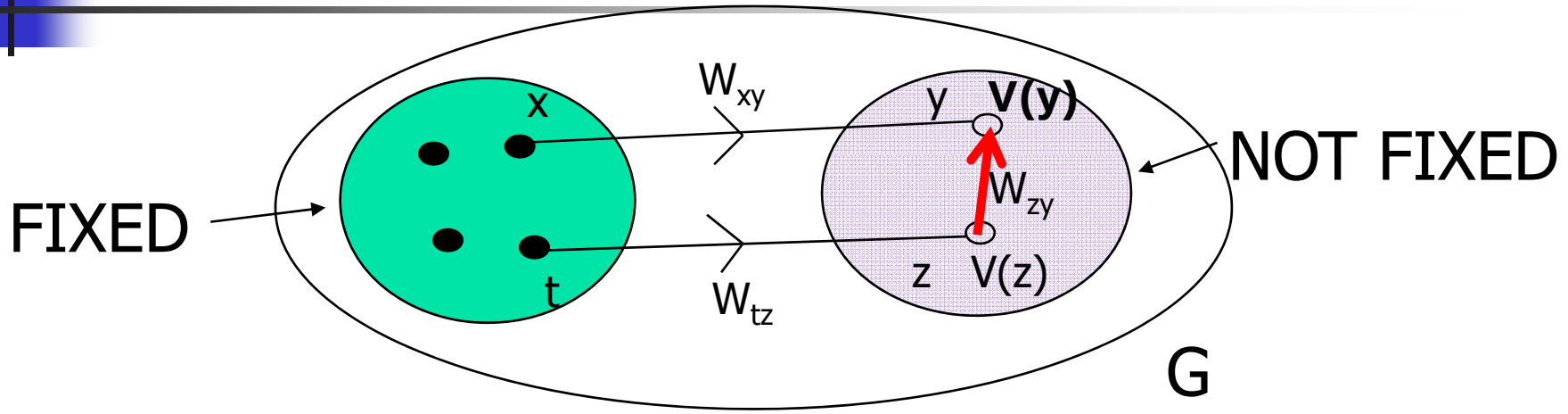
Until {x|V[x] < ∞ **and x ΜΗ φιξαρισμένος**} = ∅

Ορθότητα αλγορίθμου Dijkstra



$V[z] \geq V[y]$

Ορθότητα αλγορίθμου Dijkstra



$$V[y] \le V[z]$$

$$V[z] + W_{zy} < V[y] \le V[z] \Rightarrow$$

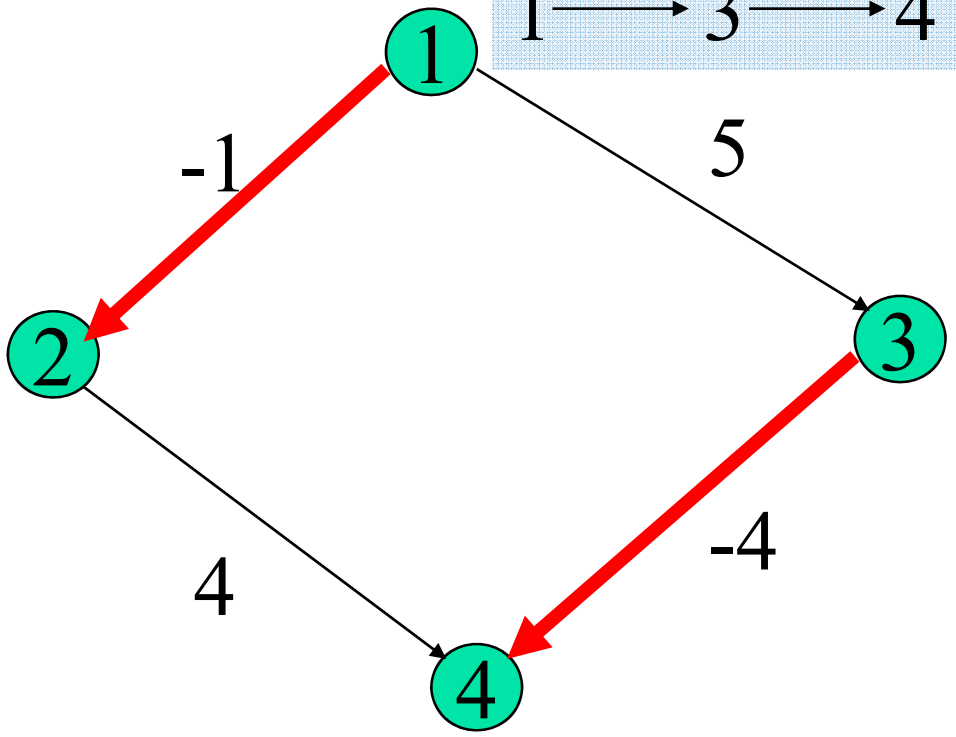
$$W_{zy} \le 0 \Rightarrow$$

ΑΔΥΝΑΤΟ

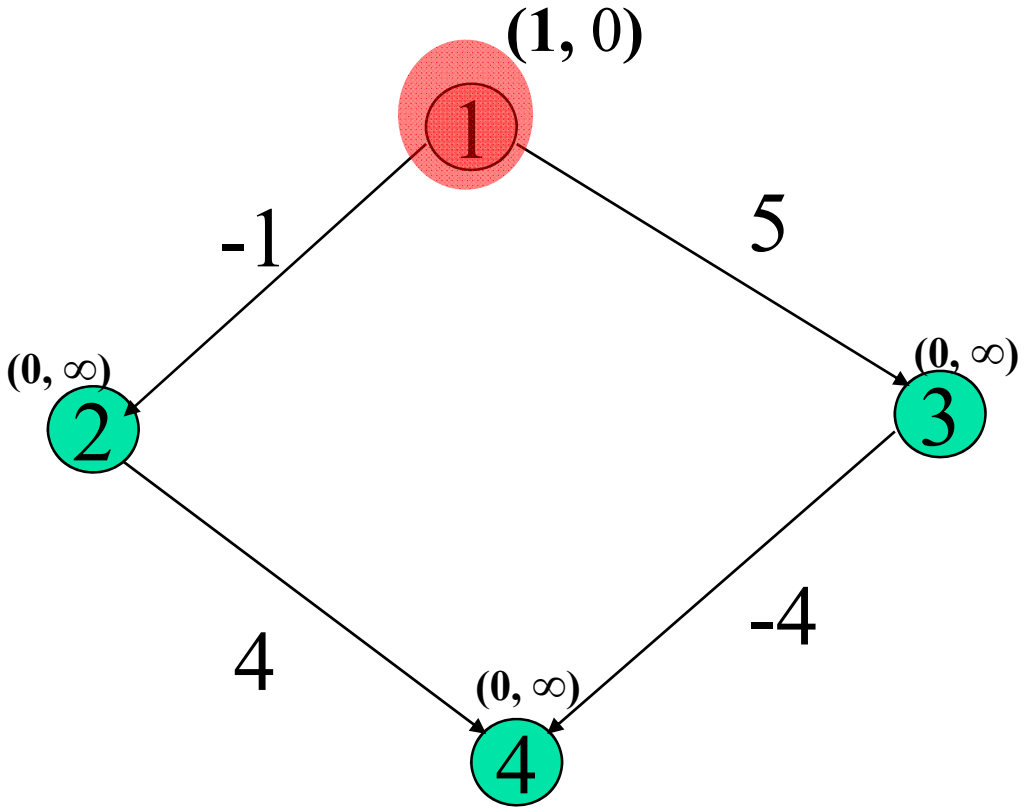
Αρνητικά βάρη

1 ⟶ 2 ⟶ 4 (κόστος = 3)

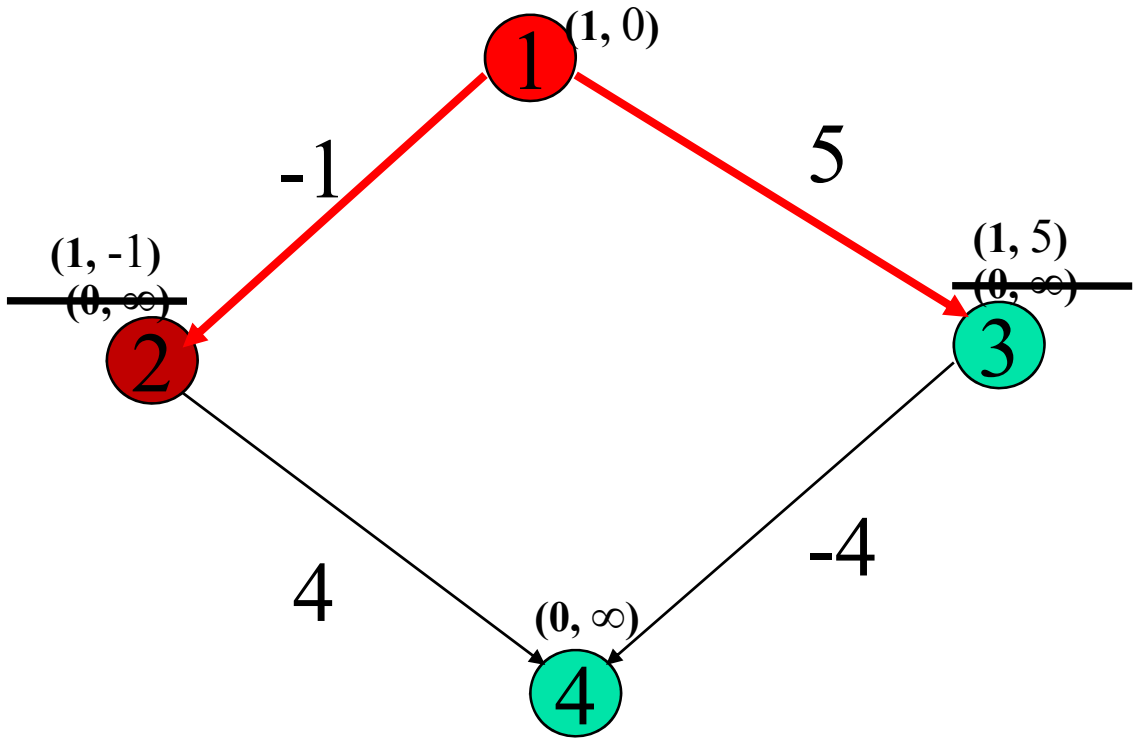
1 ⟶ 3 ⟶ 4 (κόστος = 1)



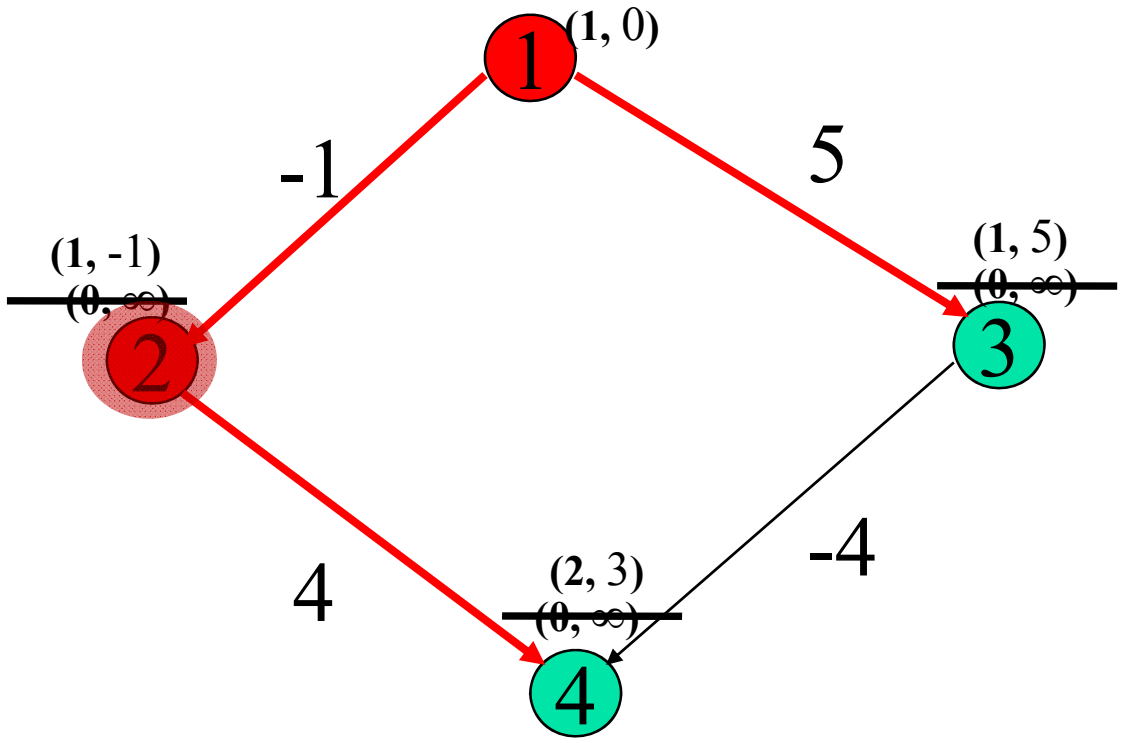
Αλγόριθμος Dijkstra



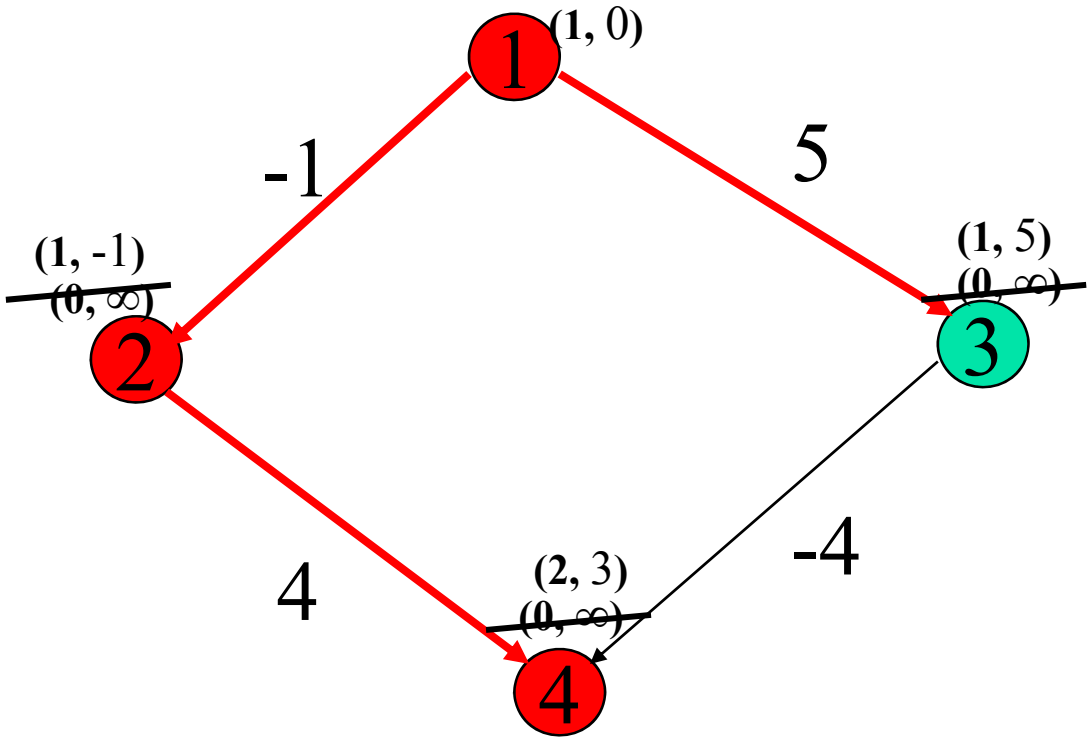
Αλγόριθμος Dijkstra



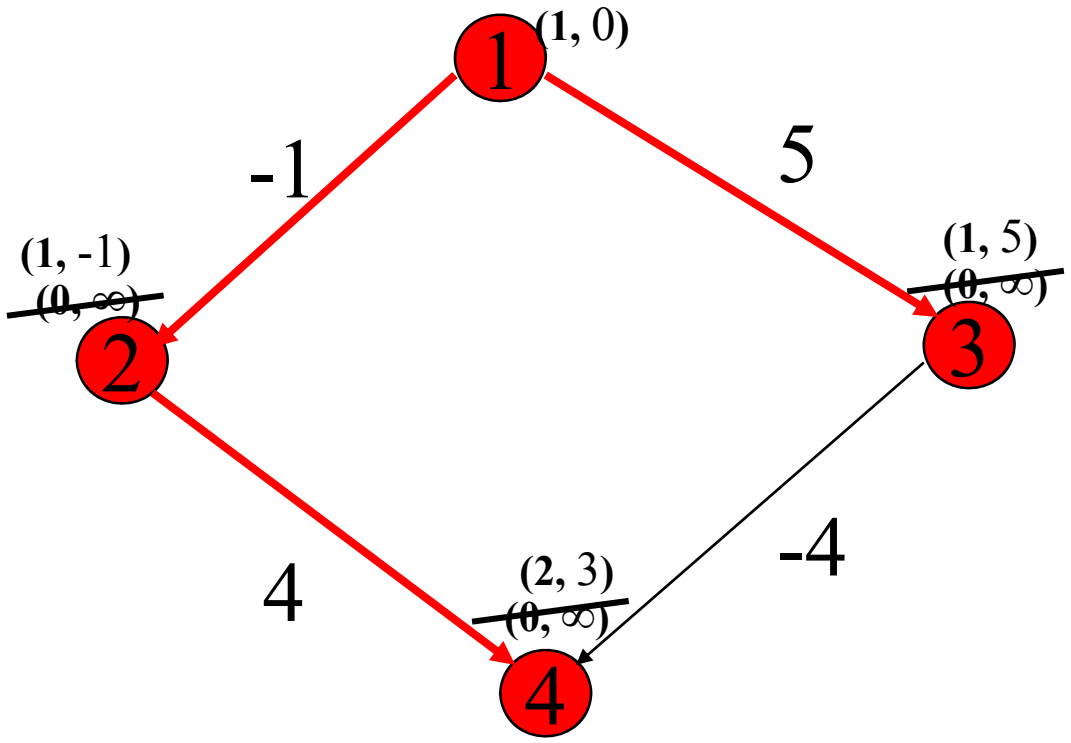
Αλγόριθμος Dijkstra



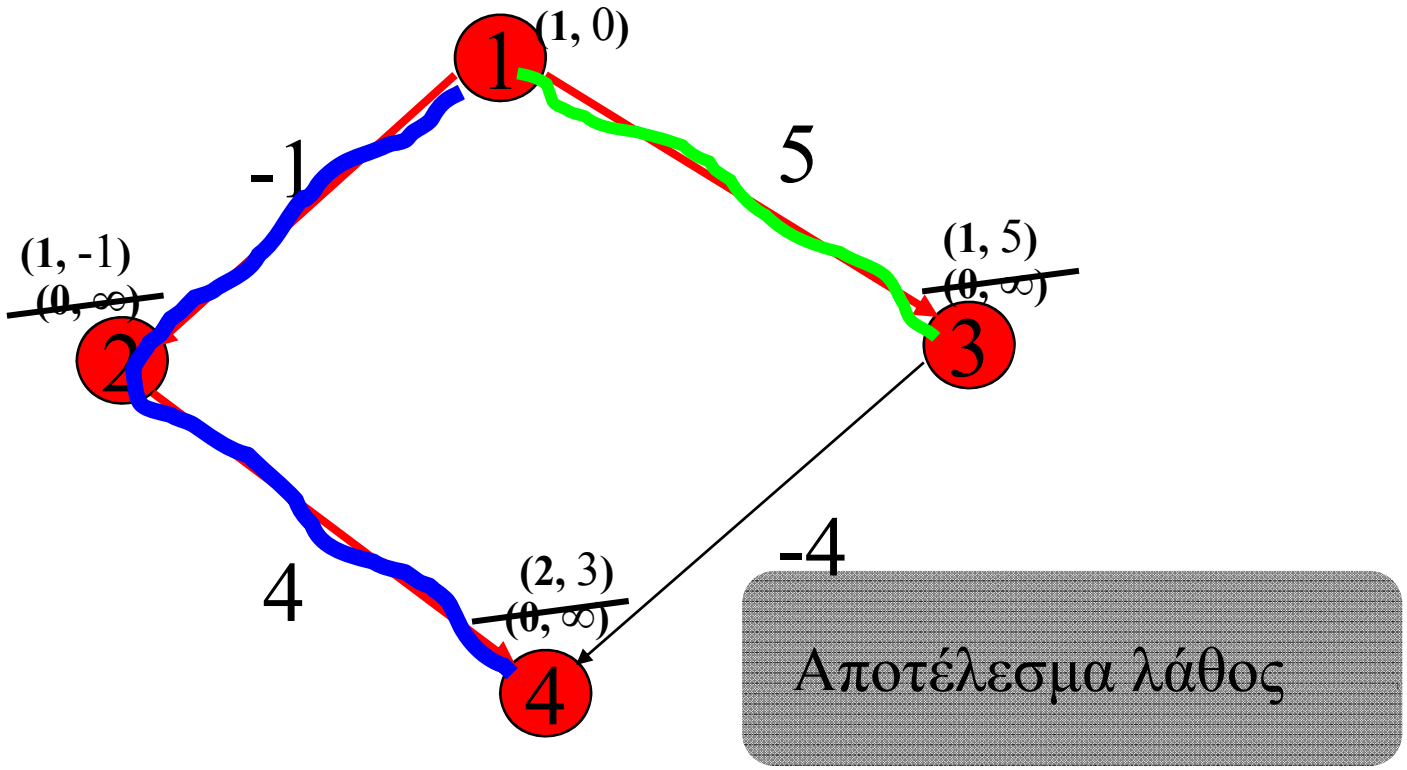
Αλγόριθμος Dijkstra



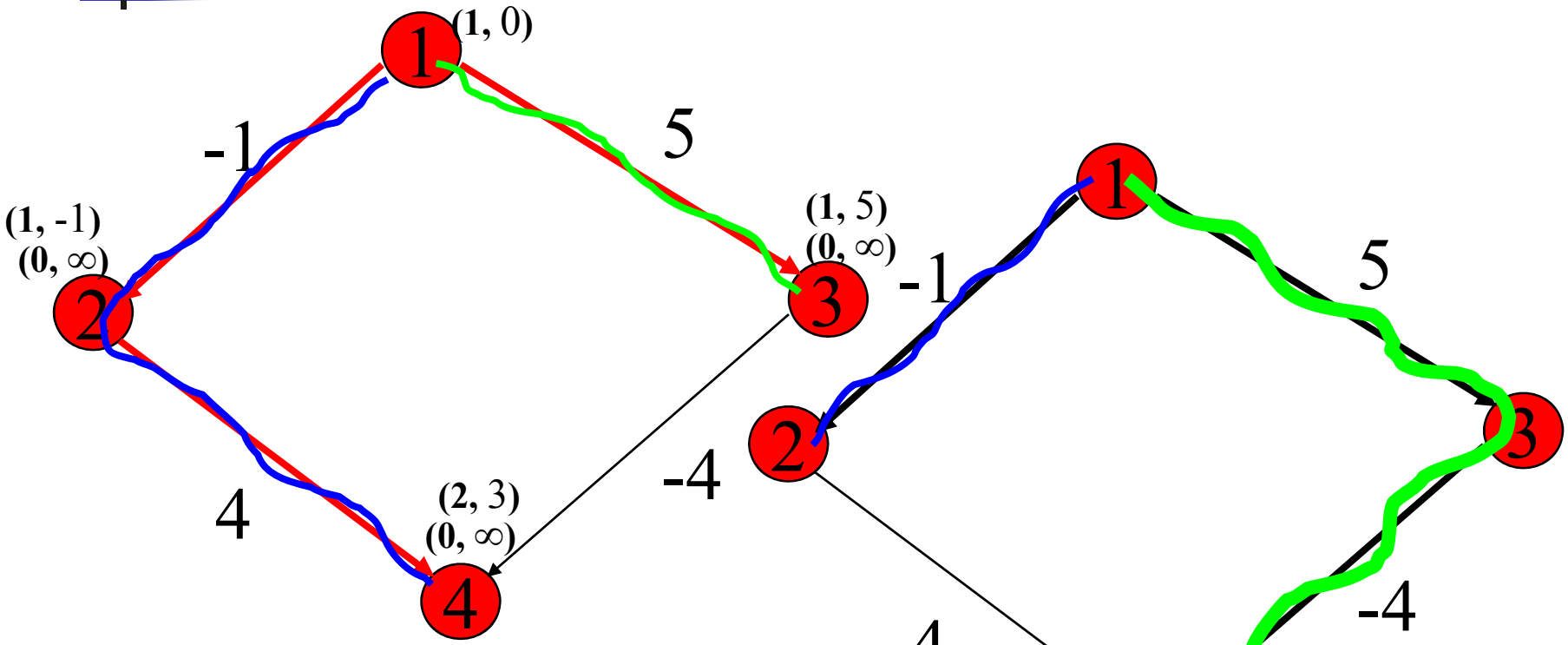
Αλγόριθμος Dijkstra



Αλγόριθμος Dijkstra



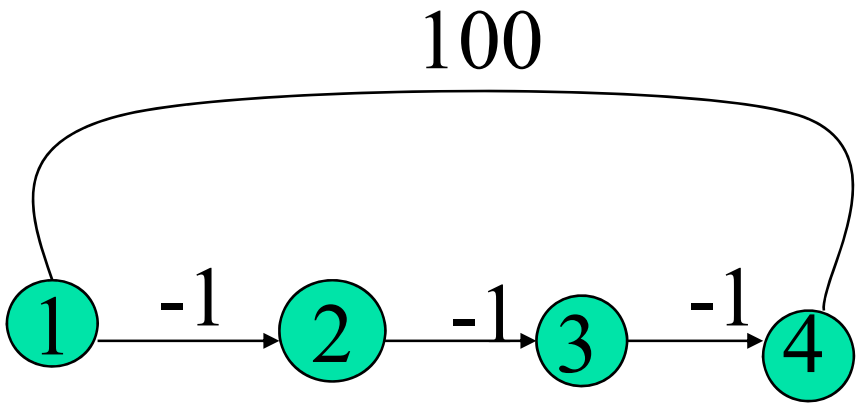
Αρνητικά βάρη (Dijkstra)



αποτέλεσμα λάθος

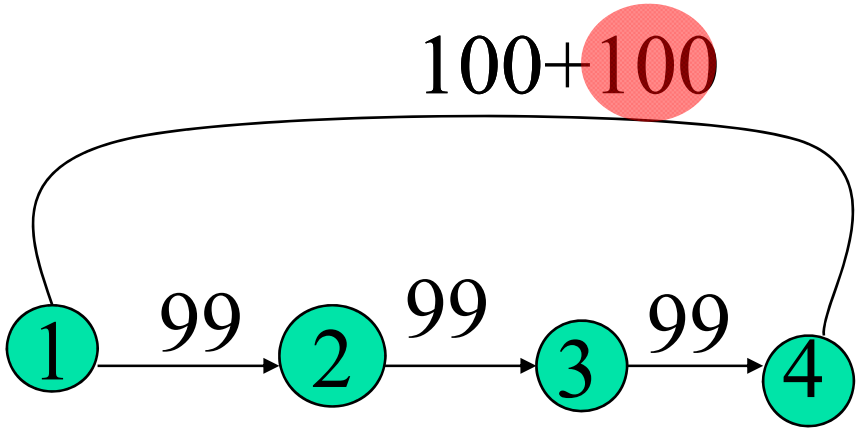
αποτέλεσμα ορθό

Αρνητικά βάρη



Επιδιόρθωση είναι εφικτή?

Αρνητικά βάρη



Πρόσθεση ενός μεγάλου αριθμού δεν επιδιορθώνει το πρόβλημα!

Αλγόριθμος Prim
Αλγόριθμος Kruskal

Αλγόριθμος Prim

- Ξεκινάμε από ένα δένδρο Τ αποτελούμενο από **ένα μόνο** κόμβο.

- Στη συνέχεια, σε κάθε επανάληψη, αυξάνουμε το δένδρο Τ συνδέοντάς το στο **πλησιέστερο ελεύθερο κόμβο** ως προς την έννοια του βάρους (κόστους).

Αλγόριθμος Prim

G=(V,E)

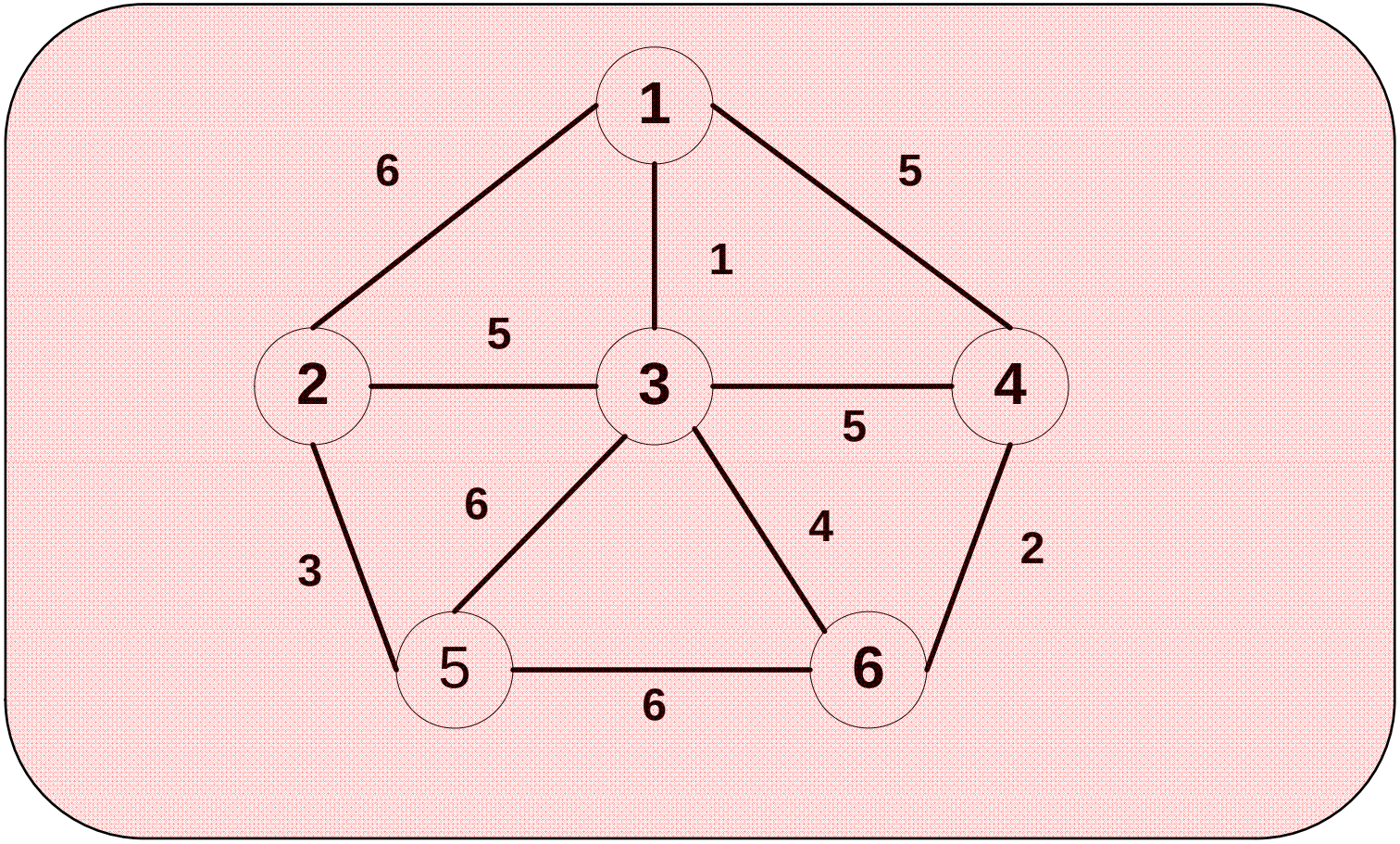
T={u, οποιοσδήποτε κόμβος}

Repeat

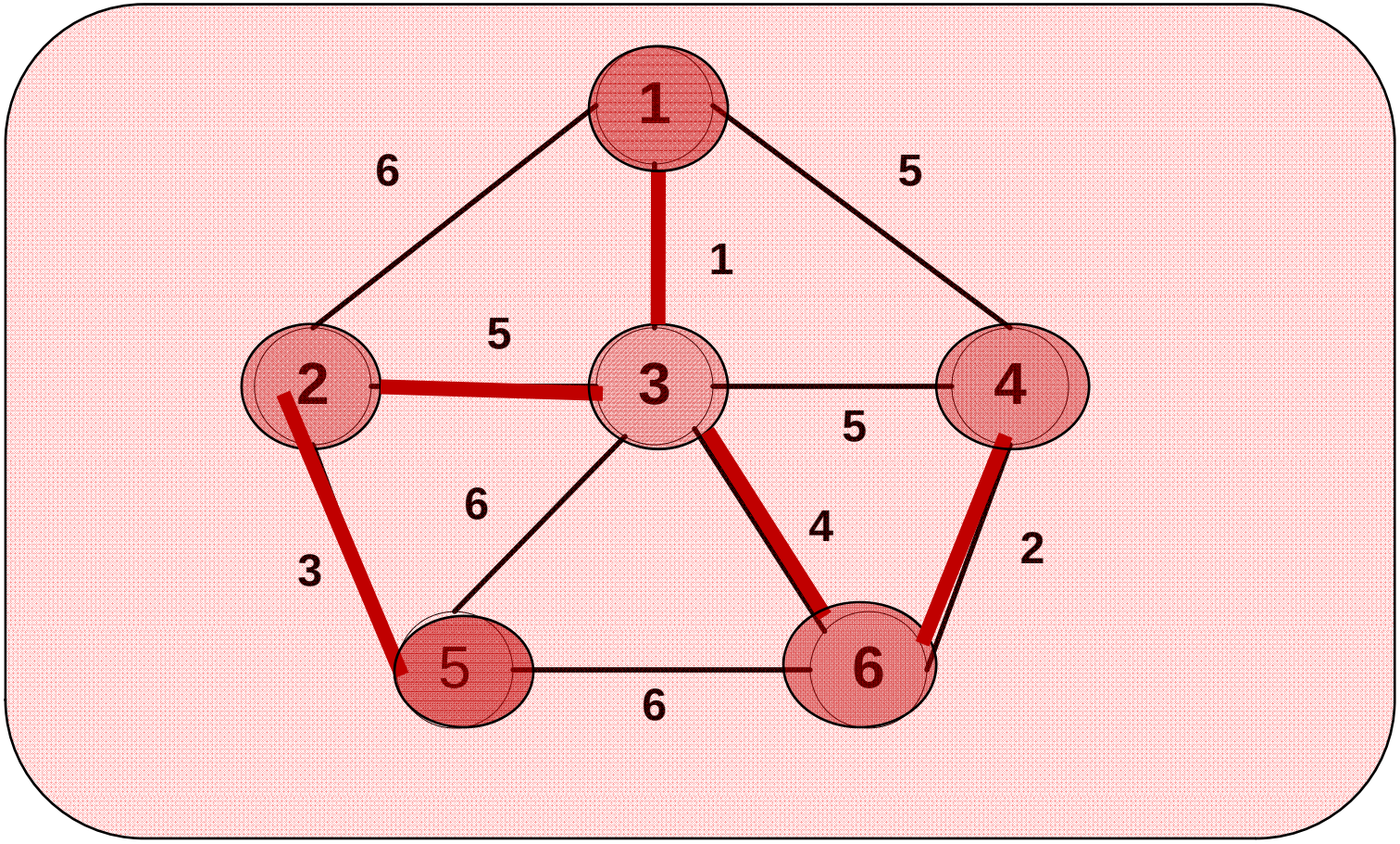
T←T ∪ {v}, v πλησιέστερος ελεύθερος κόμβος στο T

(****)

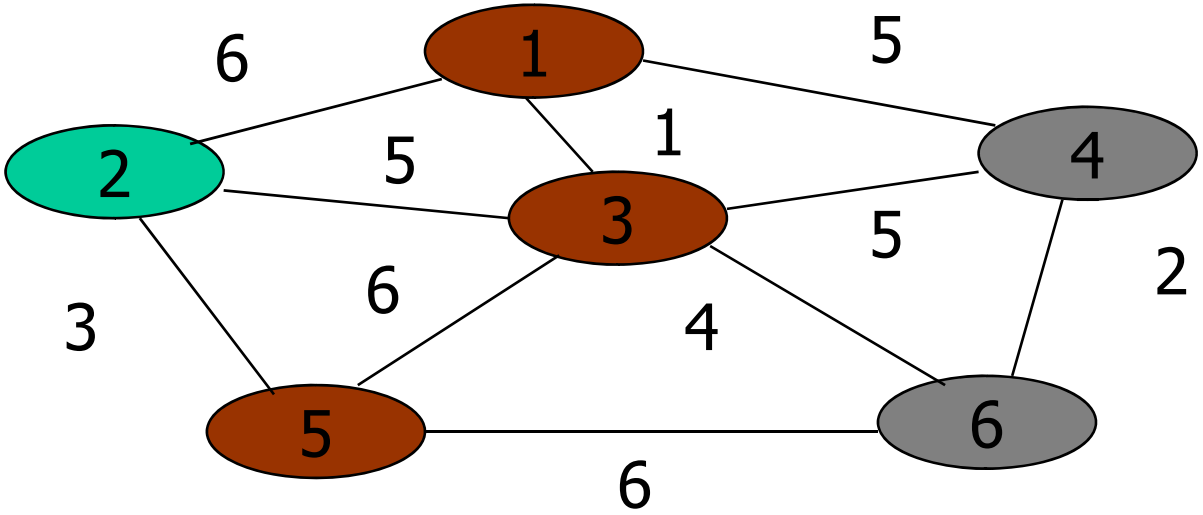
Παράδειγμα



Παράδειγμα



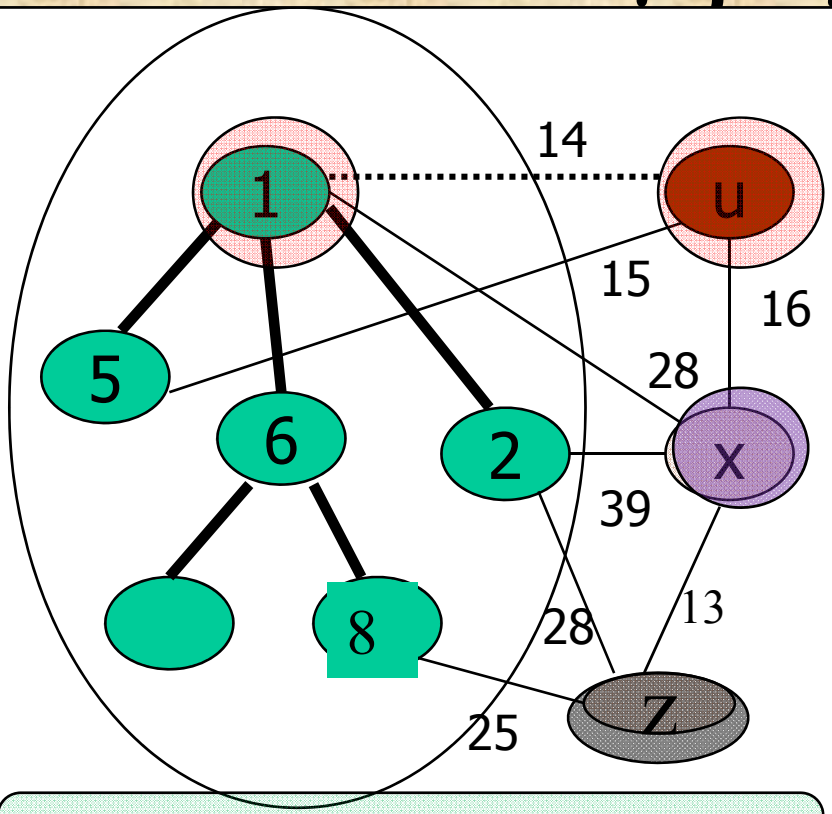
Αλγόριθμος Prim: Δομές



near[x]= {
- x, αν x ανήκει στο T
- *0 αν [x,y] δεν ανήκει στο E, για όλα τα y του T*
- **y, αν [x,y] ανήκει** στο E, **y** ο πλησιέστερος κόμβος

dist[x]= {
- **$W_{x,y}$ αν near[x]=y**
- *$+\infty$ αν near[x]=0*
- 0 αν near[x]=x

Αλγόριθμος Prim: Δομές

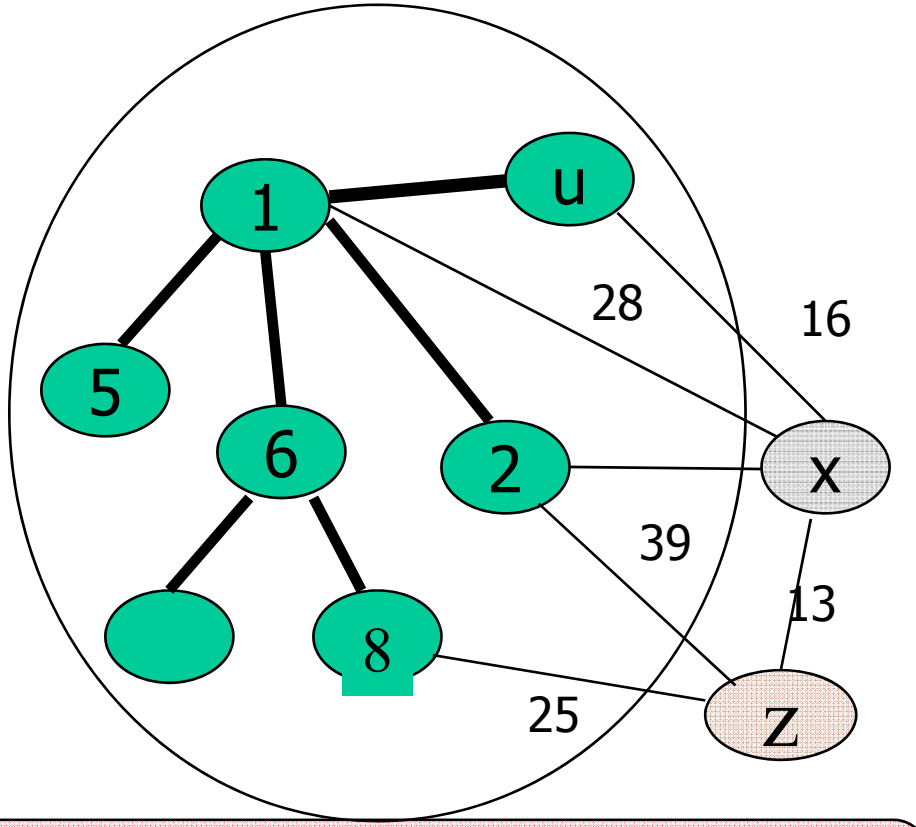


near[u]=1, dist[u]=14

near[x]=1, dist[x]=28

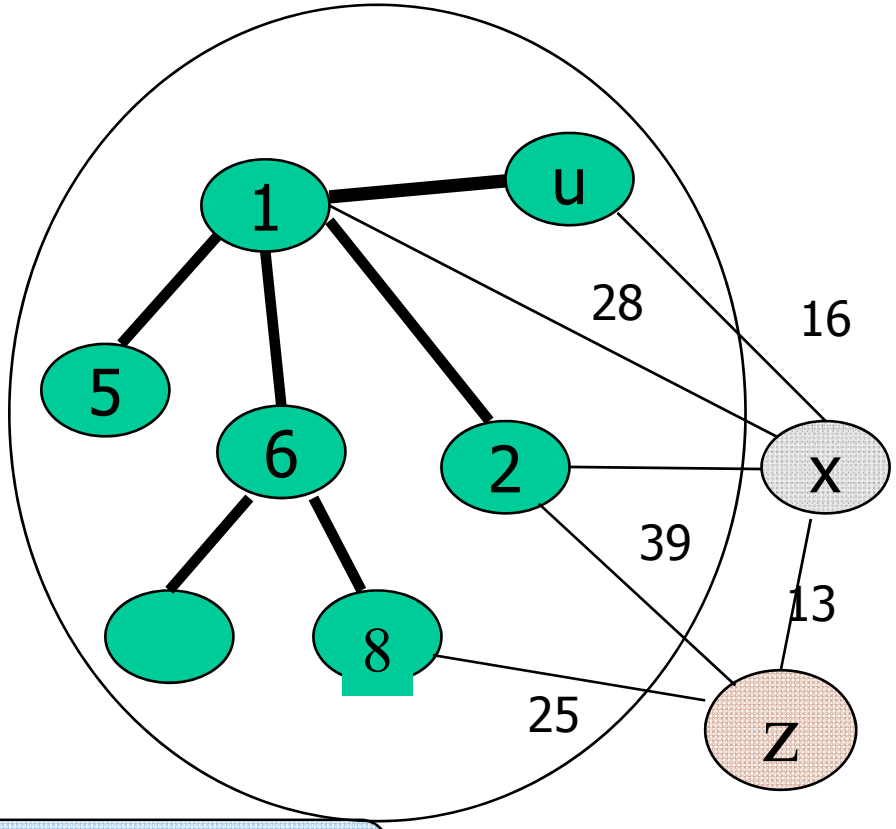
near[z]=8, dist[z]=25

Αλγόριθμος Prim: Δομές



near[u]=u, dist[u]=0

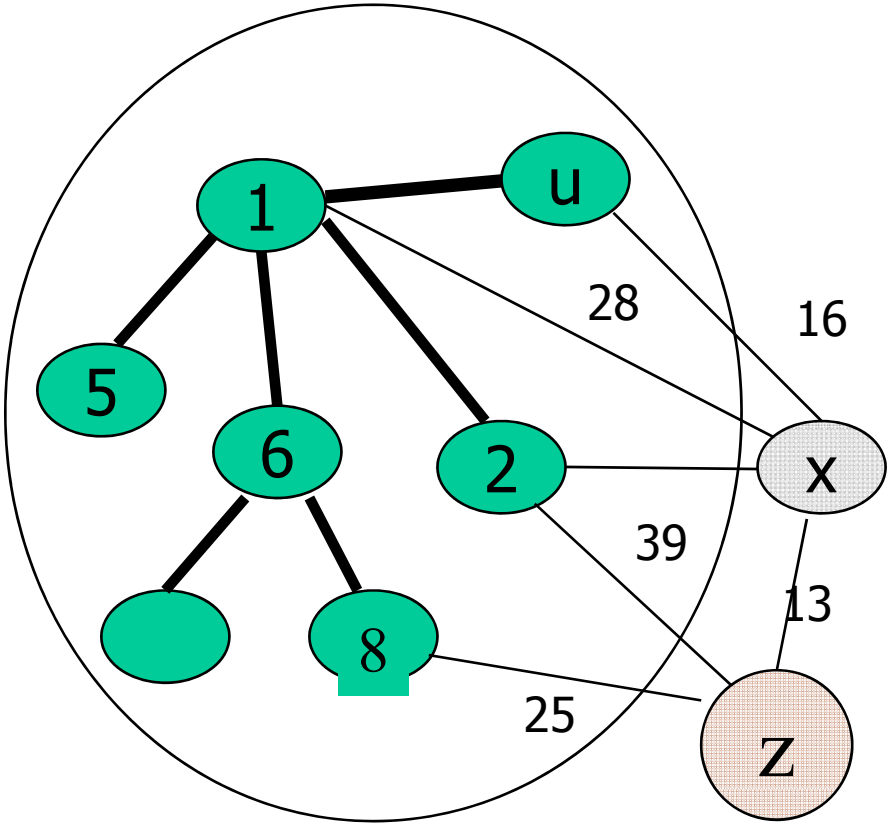
Αλγόριθμος Prim: Δομές



near[u]=u, dist[u]=0

near[x]=u, dist[x]=16 (ενημέρωση μόνο των γειτόνων του εισερχόμενου κόμβου)

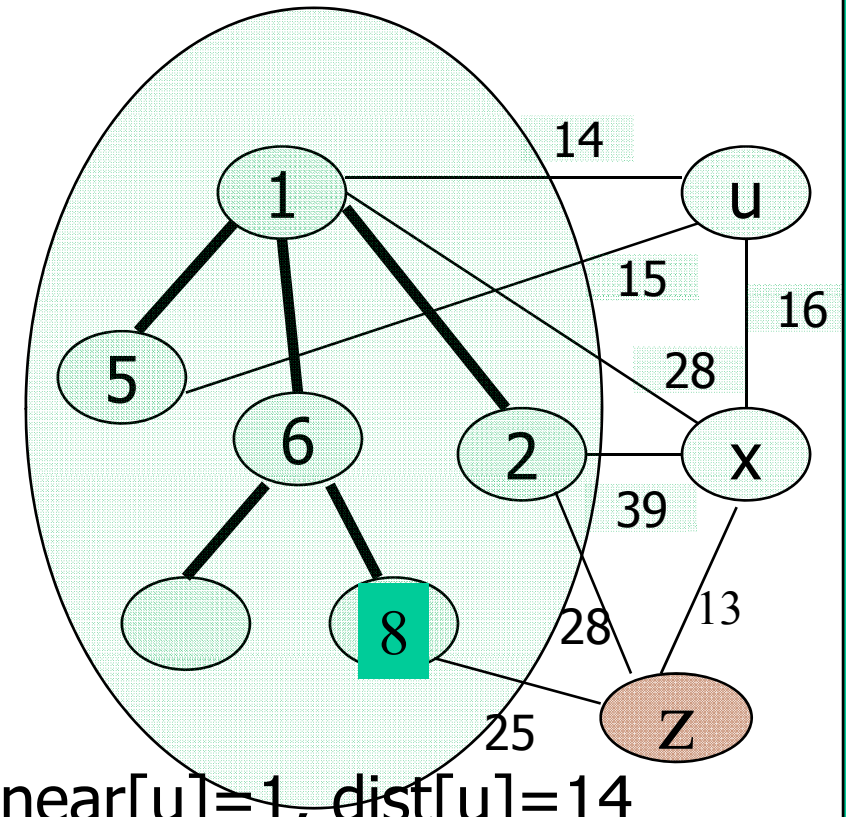
Αλγόριθμος Prim: Δομές



near[u]=u, dist[u]=0

near[x]=u, dist[x]=16 (ενημέρωση μόνο των γειτόνων του εισερχόμενου κόμβου)

Αλγόριθμος Prim: Δομές



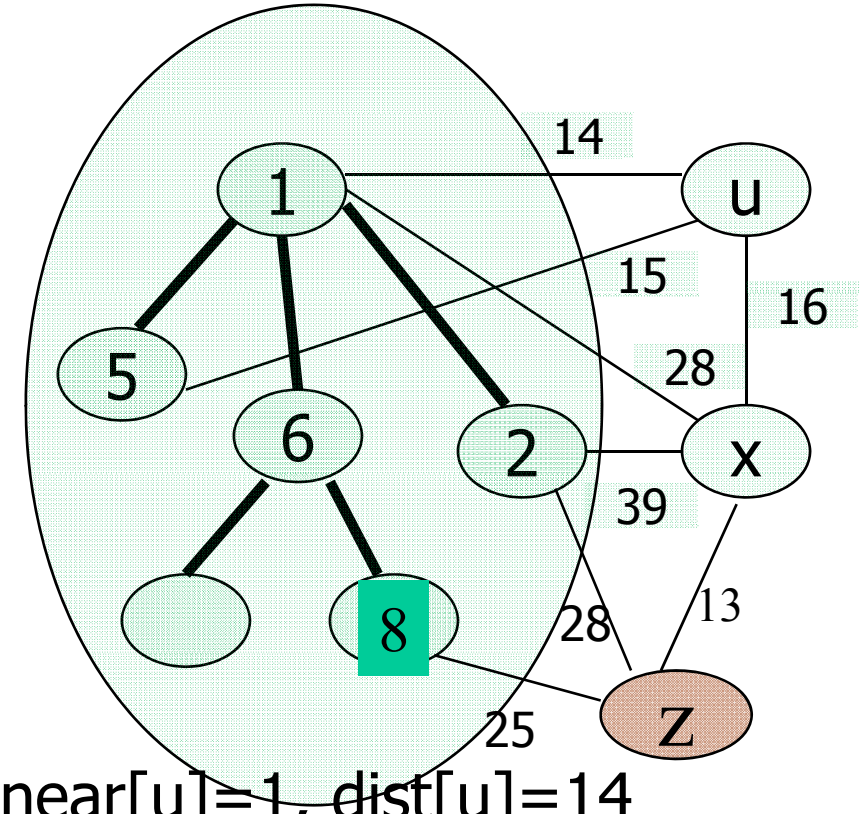
near[u]=1, dist[u]=14

near[x]=1, dist[x]=28

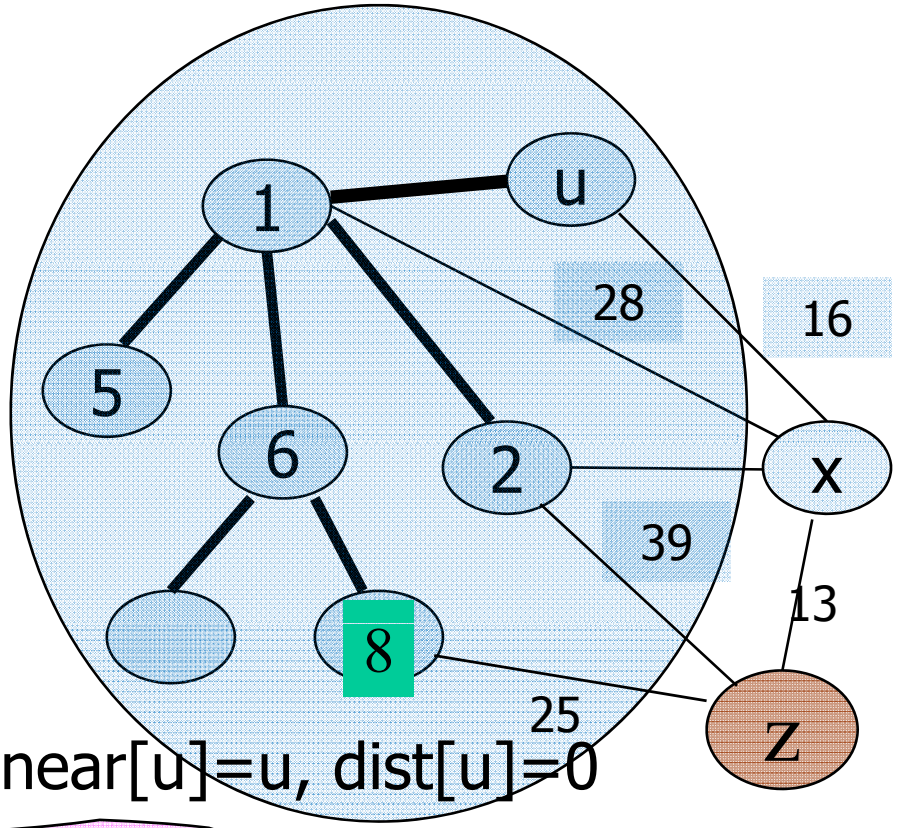
near[z]=8, dist[z]=25

Οι κόμβοι που δεν είναι γείτονες του εισερχόμενου κόμβου δεν επηρεάζονται

Αλγόριθμος Prim: Δομές



near[u]=1, dist[u]=14



near[u]=u, dist[u]=0

near[x]=1, dist[x]=28

near[x]=u, dist[x]=16

near[z]=8, dist[z]=25

near[z]=8, dist[z]=25

Οι κόμβοι που δεν είναι γείτονες του εισερχόμενου κόμβου δεν επηρεάζονται

Αλγόριθμος Prim (nearest neighbour)

initialization;

choose s arbitrary; near[s] ← s; dist(s)=0;

For i in Γ(s) do

near(i) ← s;

dist(i) ← $w_{s,i}$;

for every node i other than s and not in Γ(s) do

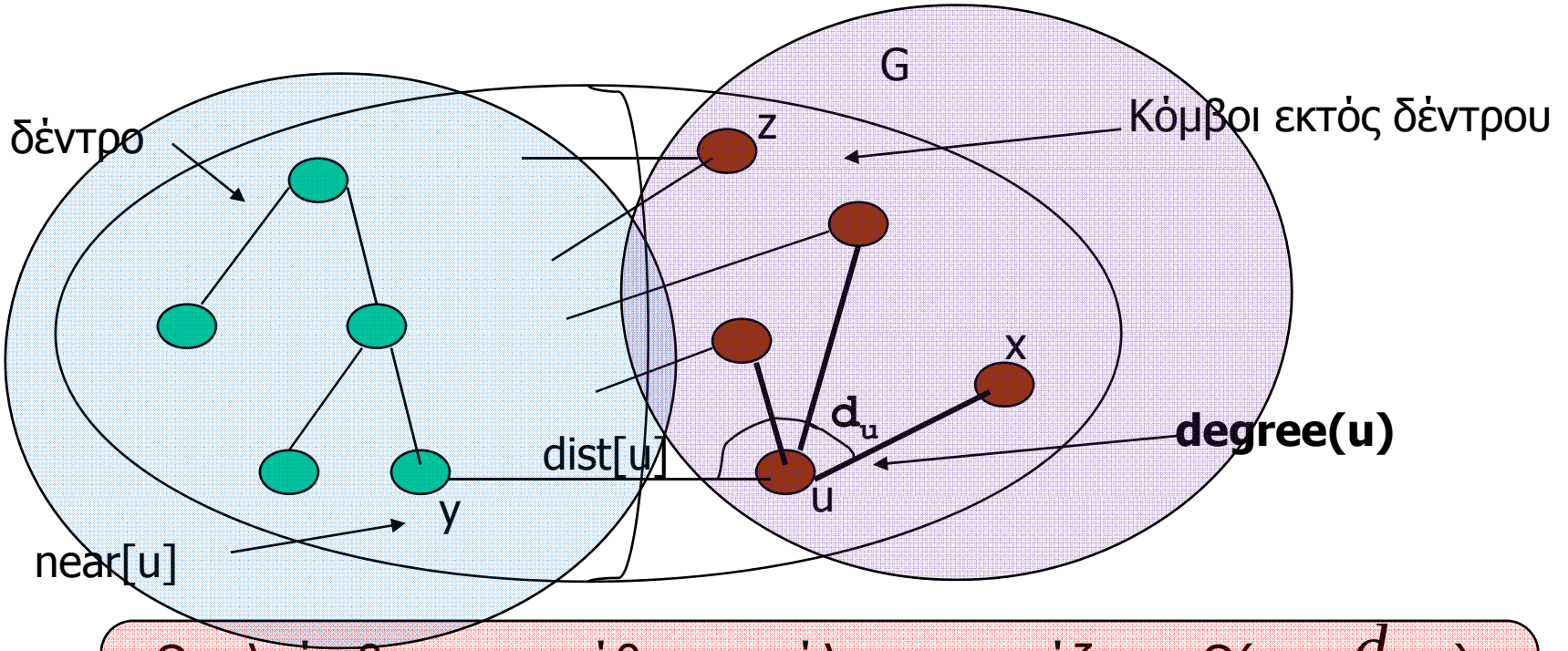
near(i) ← 0; dist(i) ← ∞;

V_T ← {s}; E_T ← ∅;

Αλγόριθμος Prim (nearest neighbour)

```
while |V_T| < n do
    u ← vertex in(V −V_T) with smallest dist(u);
    if (dist(u) >= ∞ then
       "graph disconnected"; exit;
    E_T ← E_T ∪ {(u, near(u))}; V_T ← V_T ∪ {u},
                          dist(u)=0; near(u)=u;
    for x in Γ(u) and x in (V - V_T) do
      if w_ux < dist(x) then
          dist(x) ← w_ux; near(x) ← u
end;   (****)
```

Πολυπλοκότητα Prim



Ο αλγόριθμος σε κάθε επανάληψη χρειάζεται O(n + d_i) πράξεις, όπου d_i ο βαθμός του εισερχόμενου στο δέντρο κόμβου στην i επανάληψη.

Αφού ο αλγόριθμος απαιτεί n-1 επαναλήψεις, η πολυπλοκότητα είναι O(n^2)

Συνεκτικότητα

$\Rightarrow$ Αν G συνεκτικός $\longrightarrow$ εύρεση του ΔΕΕΚ σε n-1 επαναλήψεις.

$\Rightarrow$ Αν στην i επανάληψη (i < n) δεν ευρίσκουμε πλευρά [x,y] με x στο δένδρο και y έξω από το δένδρο, τότε ο γράφος δεν είναι συνδεδεμένος

(αλλά ήδη γνωρίζουμε έλεγχο συνεκτικότητας σε O(m) με την BFS)

Δάσος επικάλυψης

Αν ο γράφος δεν είναι συνεκτικός, τότε ...
Ένα δάσος επικάλυψης ελαχίστου κόστους

Ο αλγόριθμος δίνει τη βέλτιστη λύση (από το θεώρημα βέλτιστου ΔΕΕΚ παρατηρώντας ότι το δάσος εκκίνησης συνίσταται από n μεμονωμένους κόμβους).

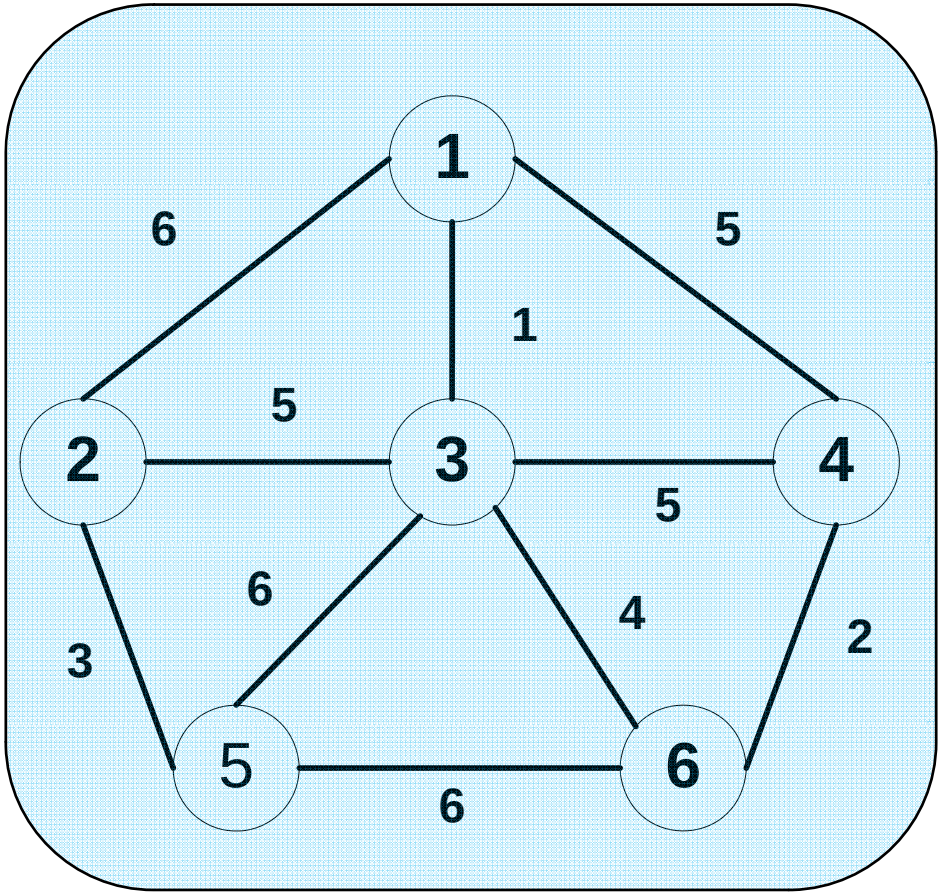
Δένδρα επικάλυψης ελάχιστου κόστους

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)

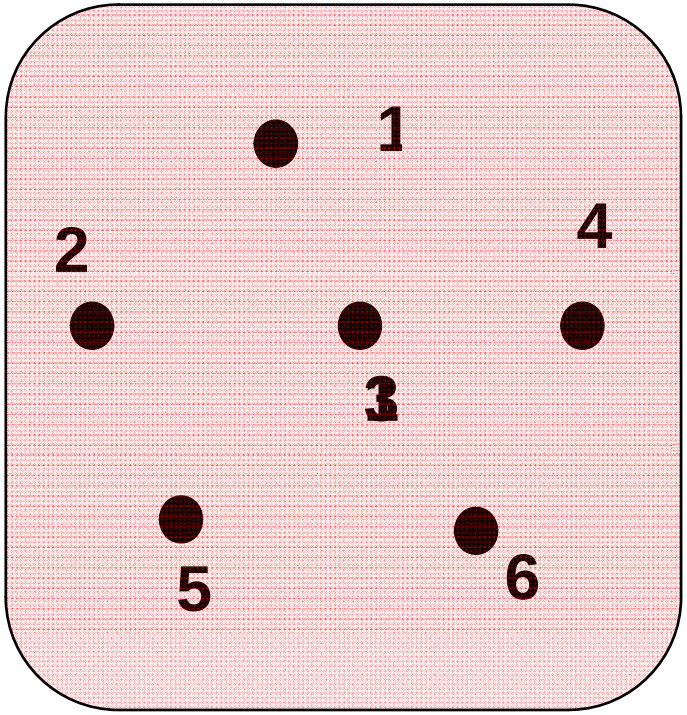
Αλγόριθμος Kruskal - Παράδειγμα



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

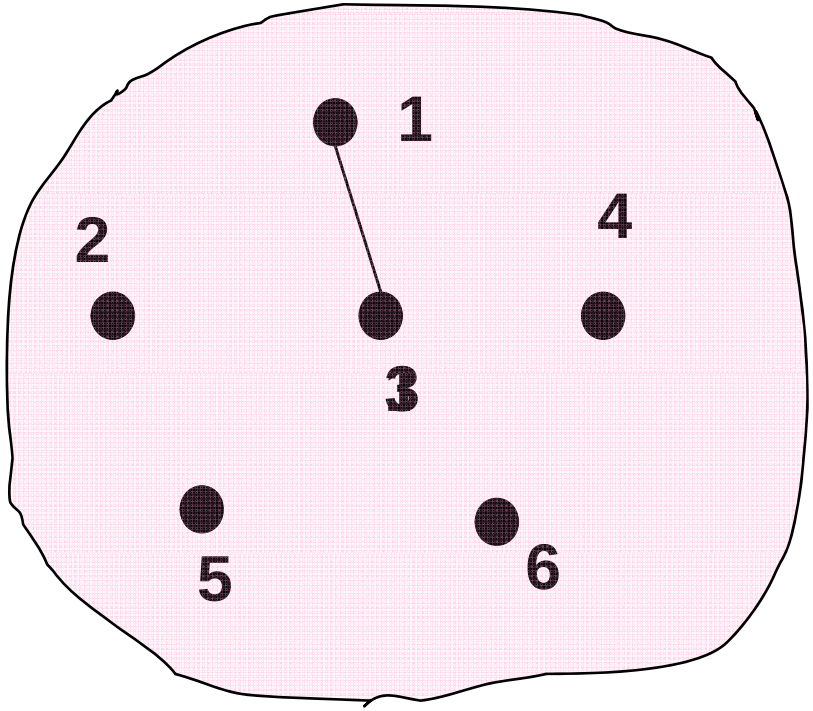
Αριθμός συνιστωσών n = αριθμός μεμονομένων κόμβων



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

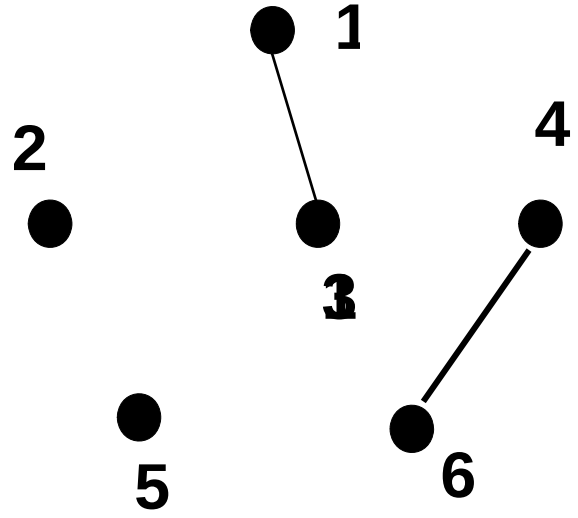
Αλγόριθμος Kruskal: βήμα 1



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

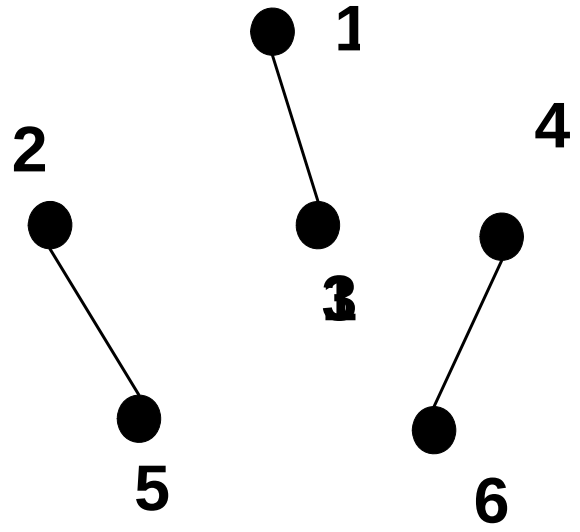
[3,5] : *κόστος* 6

[5,6] : *κόστος* 6

Αλγόριθμος Kruskal: βήμα 3

Διάταξη πλευρών
κατά αύξουσα τάξη

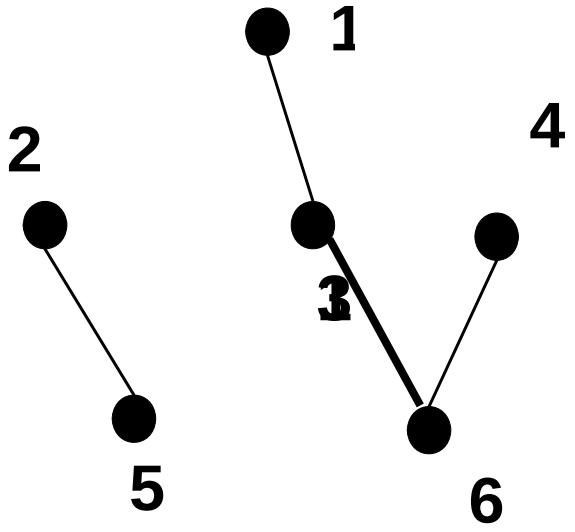
[1,3] : *κόστος* 1
[4,6] : *κόστος* 2
[2,5] : *κόστος* 3
[3,6] : *κόστος* 4
[1,4] : *κόστος* 5
[3,4] : *κόστος* 5
[2,3] : *κόστος* 5
[1,2] : *κόστος* 6
[3,5] : *κόστος* 6
[5,6] : *κόστος* 6



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

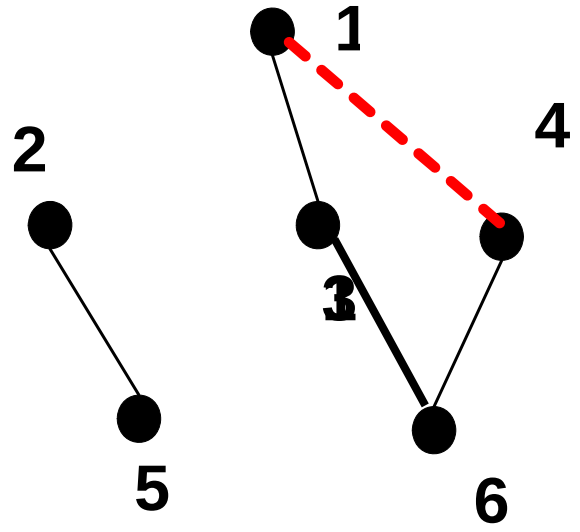
[3,4] : *κόστος* 5

[2,3] : *κόστος* 5

[1,2] : *κόστος* 6

[3,5] : *κόστος* 6

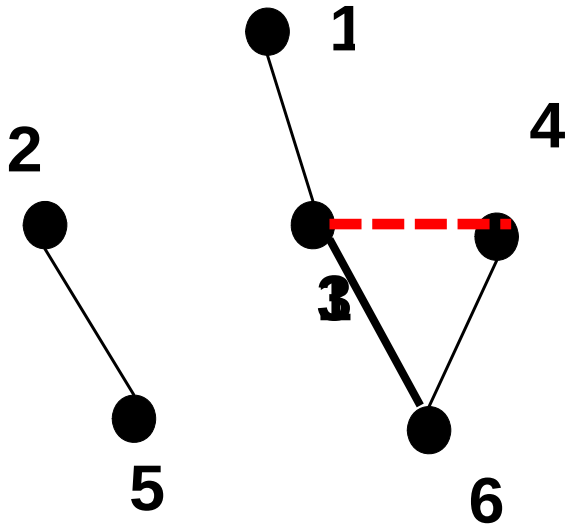
[5,6] : *κόστος* 6



Αλγόριθμος Kruskal: βήμα 6

Διάταξη πλευρών
κατά αύξουσα τάξη

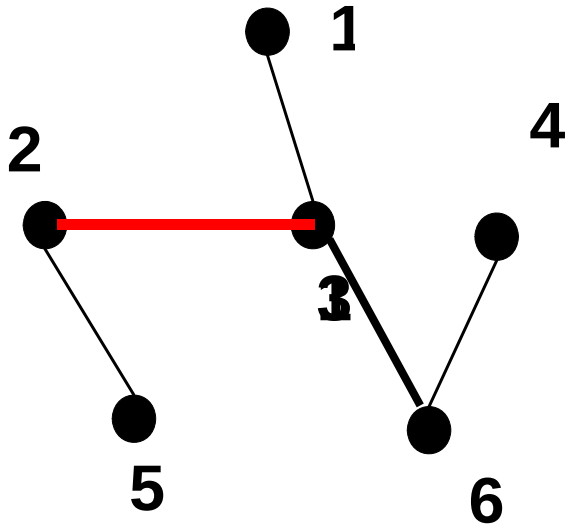
[1,3] : *κόστος* 1
[4,6] : *κόστος* 2
[2,5] : *κόστος* 3
[3,6] : *κόστος* 4
[1,4] : *κόστος* 5
[3,4] : *κόστος* 5
[2,3] : *κόστος* 5
[1,2] : *κόστος* 6
[3,5] : *κόστος* 6
[5,6] : *κόστος* 6



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

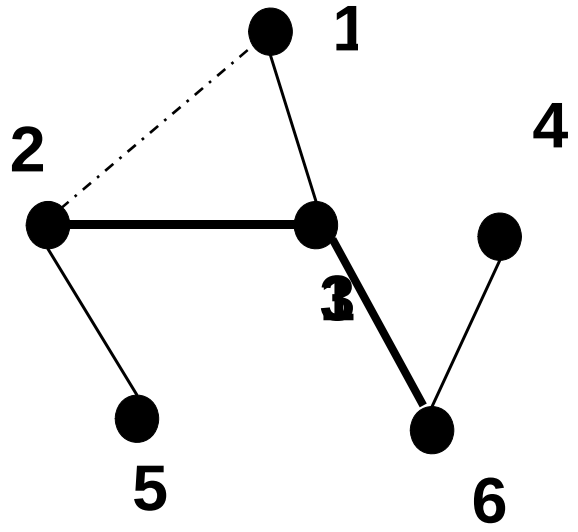


δέντρο επικάλυψης: τέλος

Αλγόριθμος Kruskal: βήμα 8

Διάταξη πλευρών
κατά αύξουσα τάξη

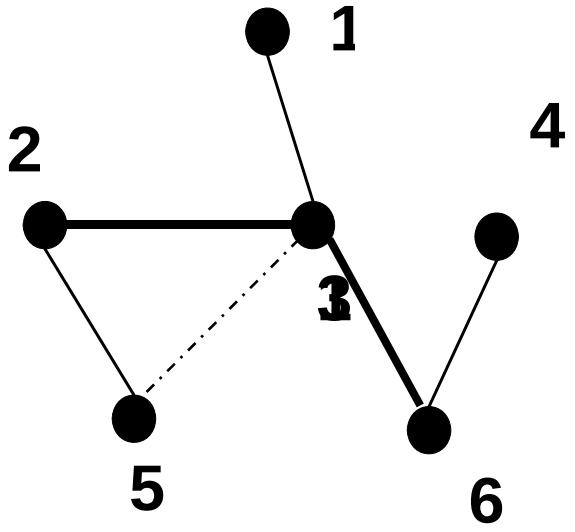
[1,3] : *κόστος* 1
[4,6] : *κόστος* 2
[2,5] : *κόστος* 3
[3,6] : *κόστος* 4
[1,4] : *κόστος* 5
[3,4] : *κόστος* 5
[2,3] : *κόστος* 5
[1,2] : *κόστος* 6
[3,5] : *κόστος* 6
[5,6] : *κόστος* 6



Αλγόριθμος Kruskal: βήμα 9

Διάταξη πλευρών
κατά αύξουσα τάξη

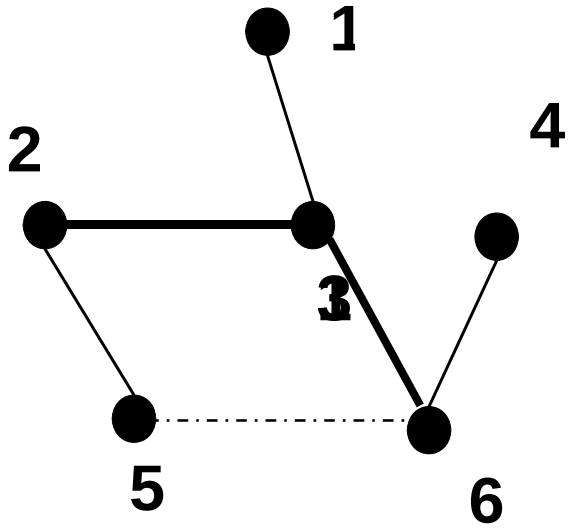
[1,3] : *κόστος* 1
[4,6] : *κόστος* 2
[2,5] : *κόστος* 3
[3,6] : *κόστος* 4
[1,4] : *κόστος* 5
[3,4] : *κόστος* 5
[2,3] : *κόστος* 5
[1,2] : *κόστος* 6
[3,5] : *κόστος* 6
[5,6] : *κόστος* 6



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

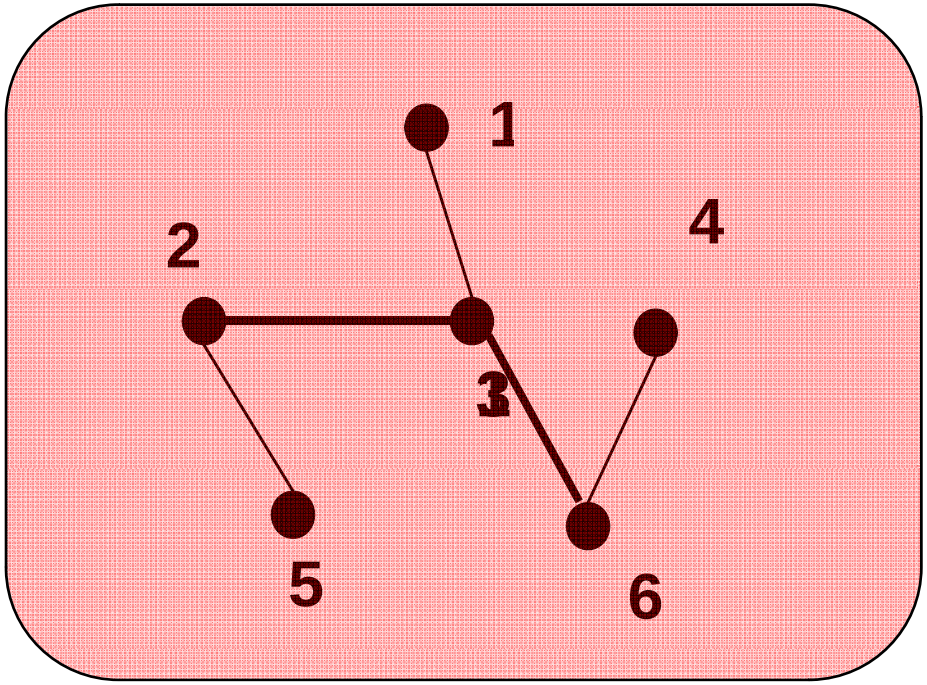
[3,4] : *κόστος* 5

[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, $m < n^2$)

Εξακρίβωση σχηματισμού κύκλου: O(logn)
Επαναλήψεις: m

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

Σύγκριση των 2 αλγορίθμων

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

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

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

$$\Downarrow$$

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

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

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

Θεώρημα βέλτιστου για το ΔΕΕΚ

$G = (V, E)$ **G συνεκτικός**

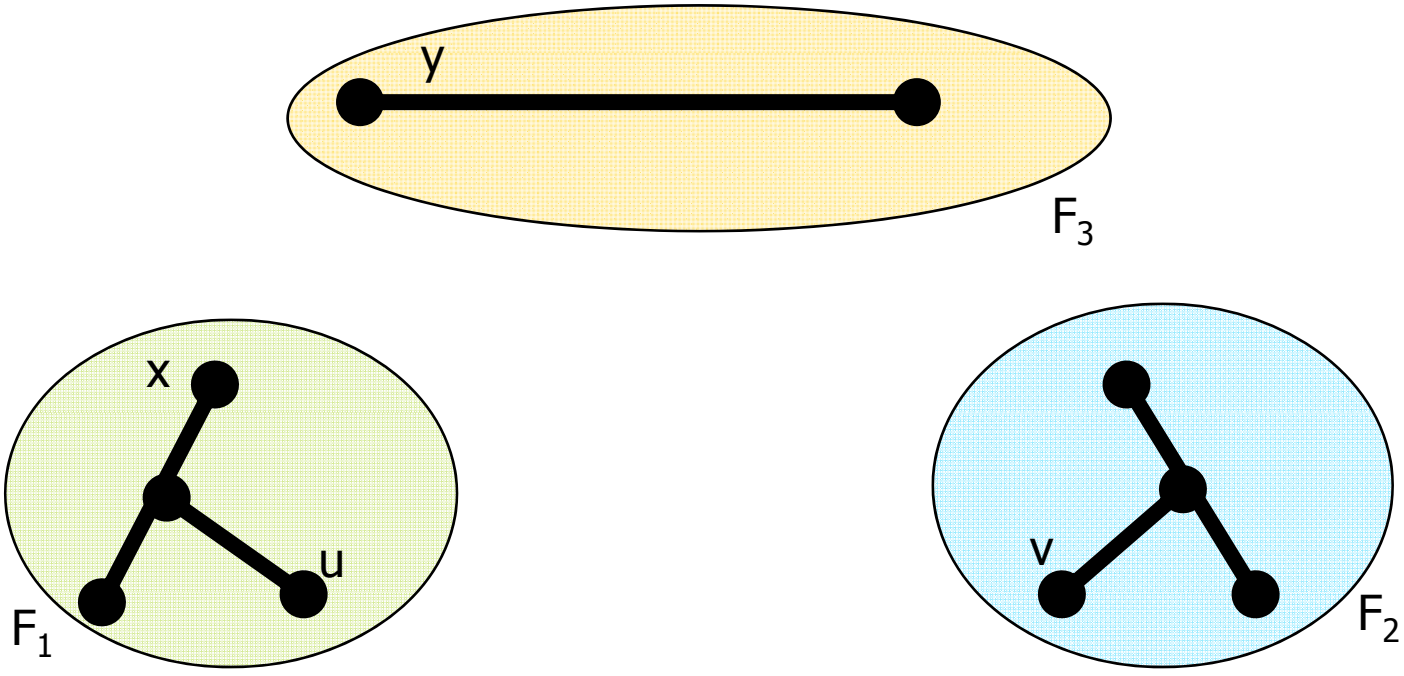
Έστω $F=(F_1, F_2, \ldots F_k)$ $\mu\varepsilon$ $F_i = (V_i, E_i)$ ένα δάσος του γράφου και

$[u, v]$ 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.

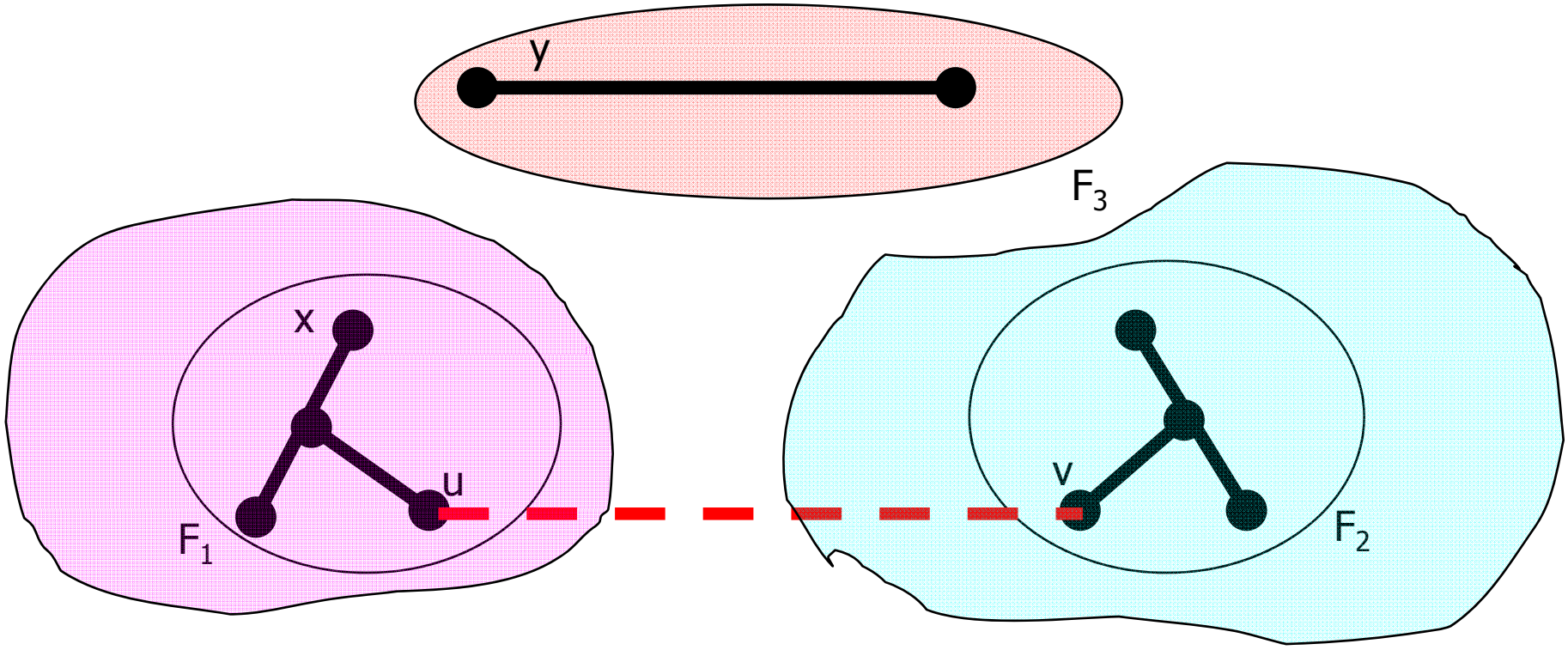
Θεώρημα βέλτιστου για το ΔΕΕΚ - Απόδειξη

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

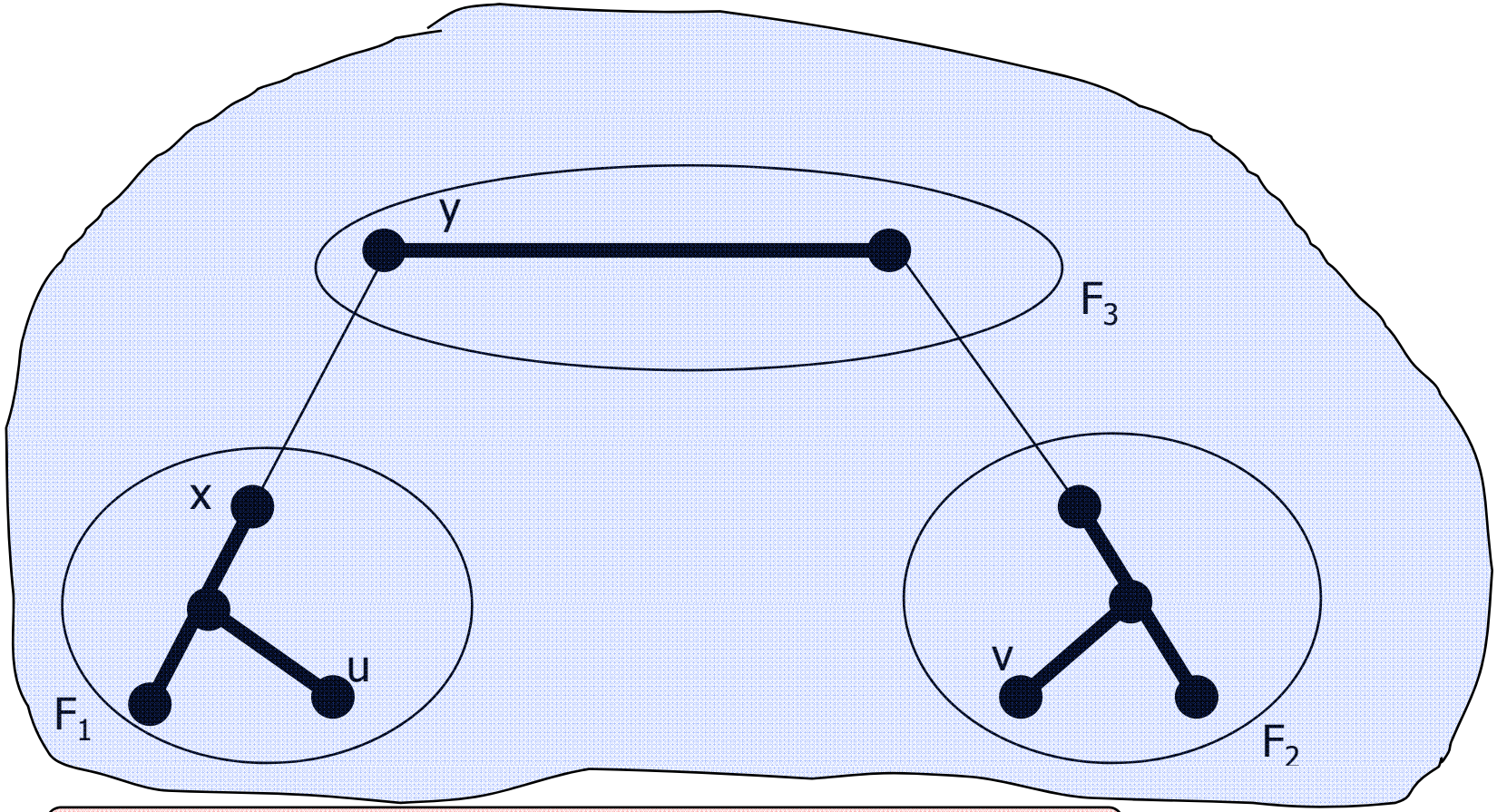


Θεώρημα βέλτιστου για το ΔΕΕΚ - Απόδειξη

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



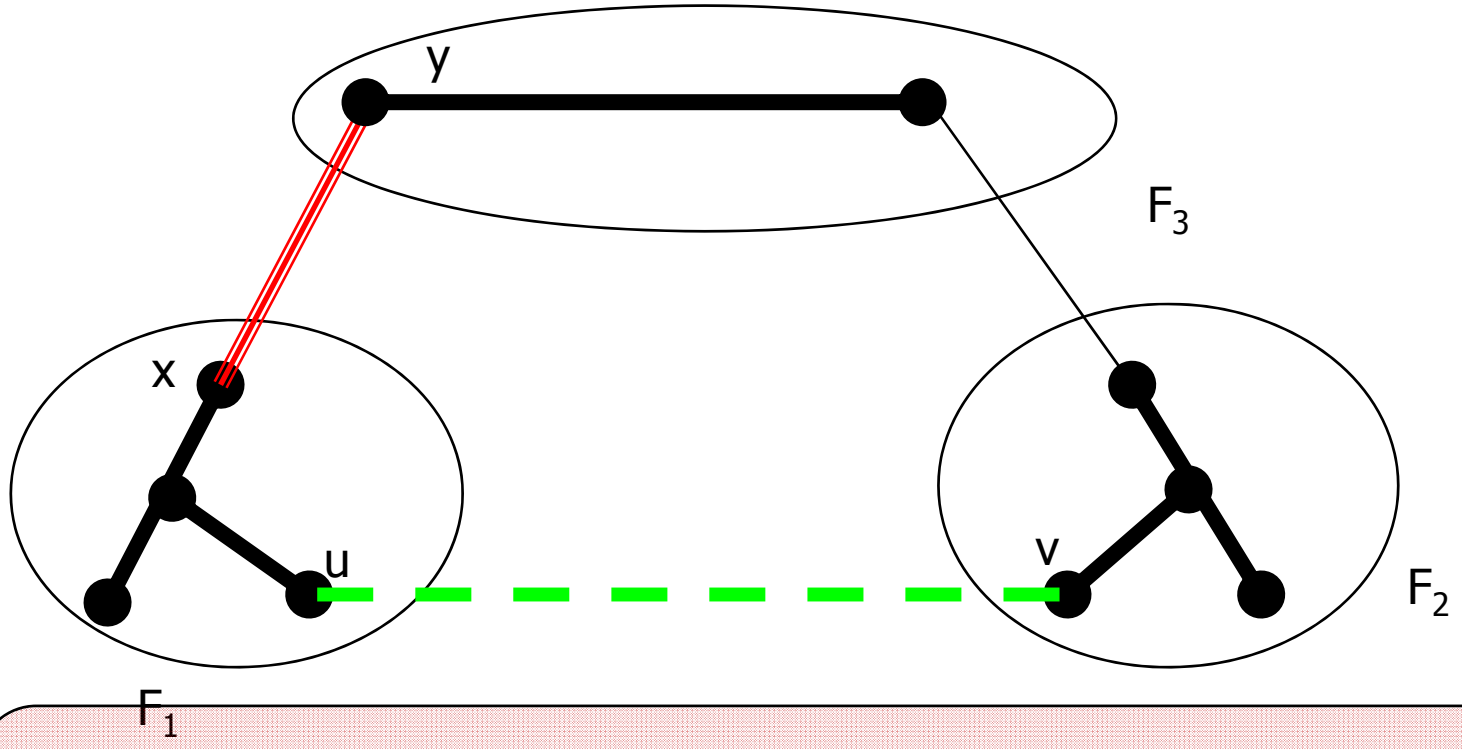
Θεώρημα βέλτιστου για το ΔΕΕΚ - Απόδειξη



Ai δέντρο επικάλυψης που περιέχει το T

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

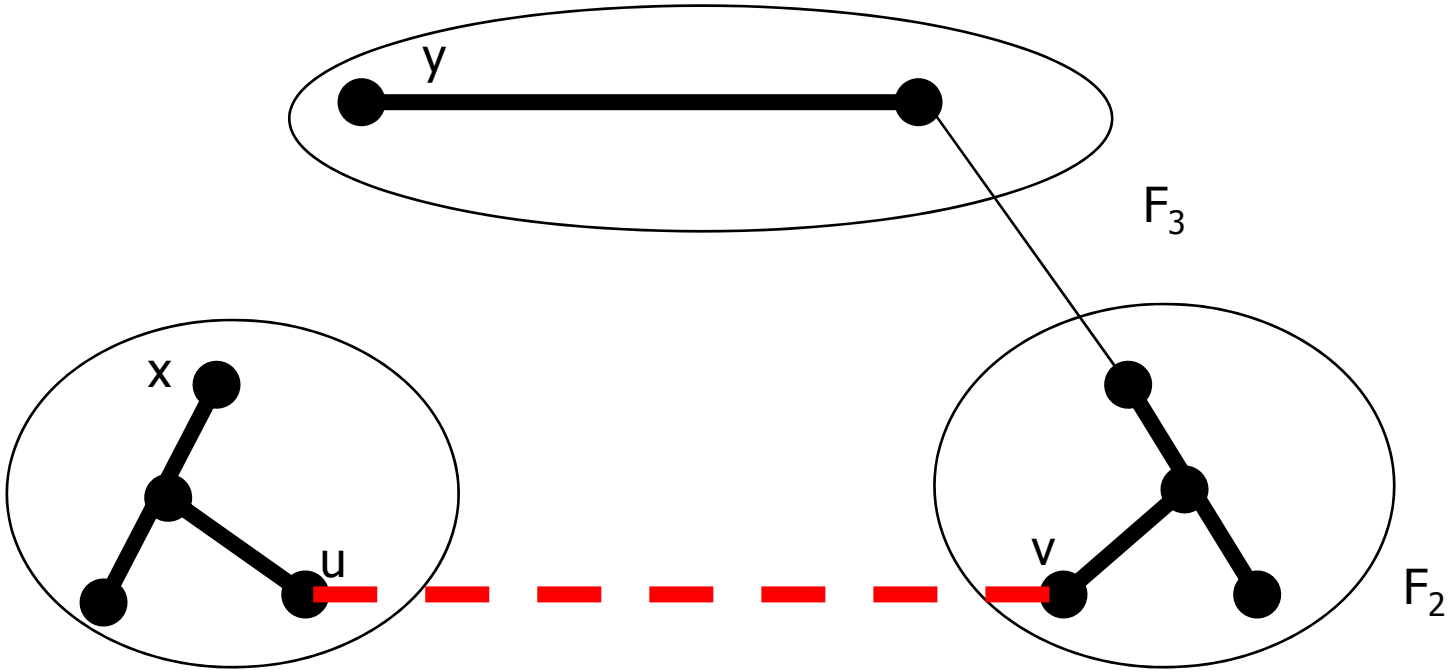
Θεώρημα βέλτιστου για το ΔΕΕΚ - Απόδειξη



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**