Θεμελιώδη Θέματα Επιστήμης Υπολογιστών

ΣΗΜΜΥ – ΣΕΜΦΕ ΕΜΠ

5η ενότητα:
Αλγόριθμοι γράφων και δικτύων

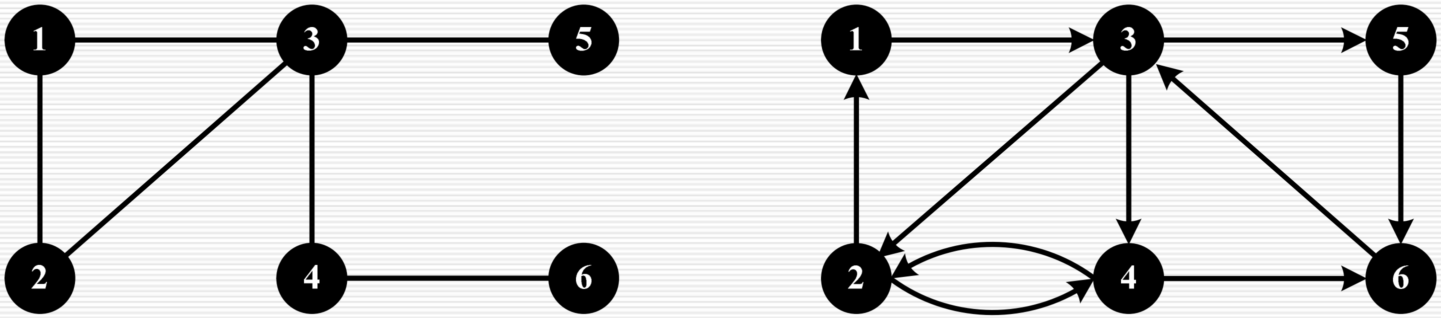
Επιμέλεια διαφανειών:
Στάθης Ζάχος, Άρης Παγουρτζής, Δημήτρης Φωτάκης

Ευχαριστίες: μέρος των διαφανειών προέρχεται από διαφάνειες του συναδέλφου Σταύρου Νικολόπουλου (Παν. Ιωαννίνων)

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Εθνικό Μετσόβιο Πολυτεχνείο

Γράφοι (επανάληψη)

(V_1,E_1)

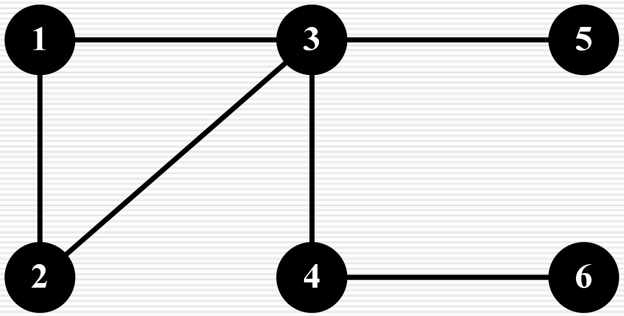


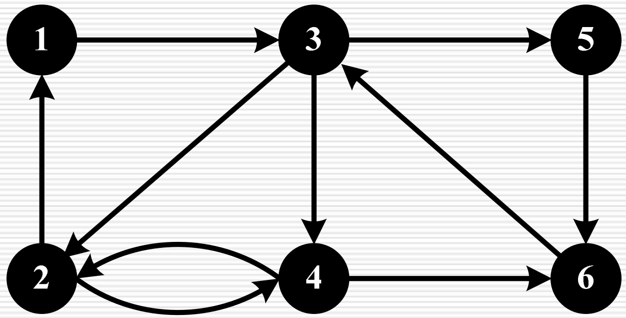
(V_2,E_2)

- **Γράφος (ή γράφημα):** ζεύγος **(V,E)**, **V** ένα μη κενό σύνολο, **E** διμελής σχέση πάνω στο **V**
- **Μη κατευθυνόμενος γράφος:** σχέση **E** συμμετρική
- **V**: **κορυφές** (vertices), **κόμβοι** (nodes)
- **E**: **ακμές** (edges)
  - $E_1 = \{\{1,2\}, \{1,3\}, \{2,3\}, \{3,4\}, \{3,5\}, \{4,6\}\}$
  - $E_2 = \{(1,3), (2,1), (2,4), (3,2), (3,4), (3,5), (4,2), (4,6), (5,6), (6,3)\}$

Γράφοι (επανάληψη): ορολογία

(V_1,E_1)

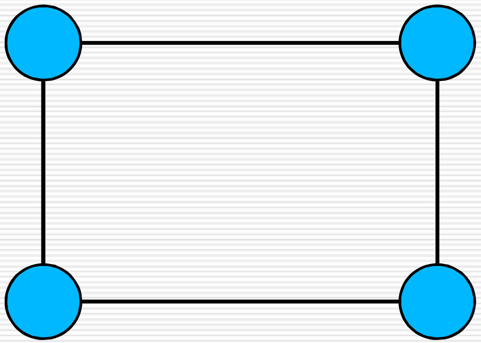




(V_2,E_2)

- **Γειτονικές** (adjacent) **κορυφές:** συνδέονται με ακμή, π.χ. 4 και 6
- **Άκρα** (endpoints) ακμής
- **Προσπίπτουσα** (incident) ακμή (σε κόμβο)
- **Γειτονικές** ακμές

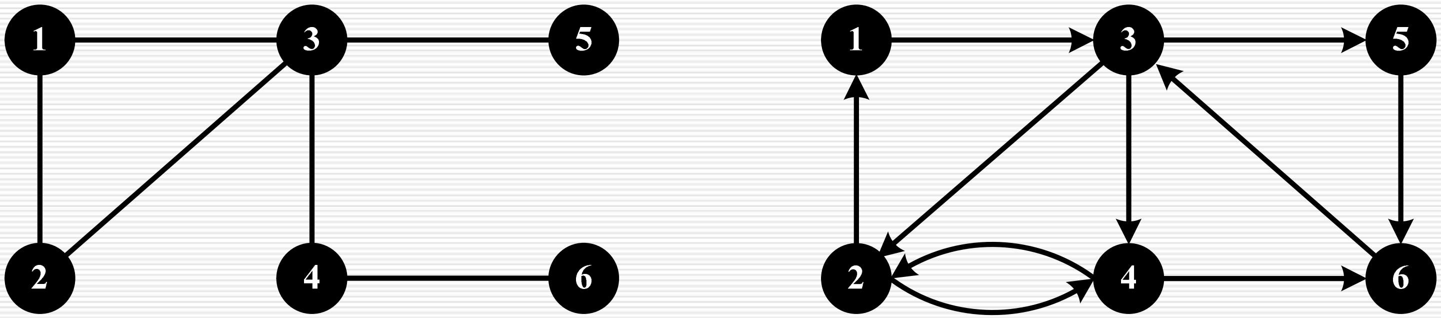
Γράφοι (επανάληψη): ορολογία



2-κανονικός γράφος

- **Βαθμός** (degree, valence) **κορυφής** v**:** ο αριθμός των ακμών που προσπίπτουν στην v, deg(v)
- Ένας (μη κατευθυνόμενος) γράφος όπου deg(v)=*k* για κάθε κορυφή v, λέγεται ***k*-κανονικός** (*k*-regular)
- Σημαντική ιδιότητα: Σ deg(v) = 2|E|
- Σε κατευθυνόμενο γράφο: in-deg(v), out-deg(v)

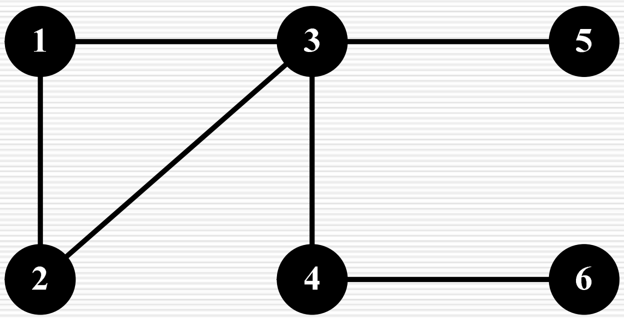
Γράφοι (επανάληψη): διαδρομές



- **Δρόμος:** έγκυρη ακολουθία από κορυφές-ακμές
- **Μονοπάτι:** δρόμος χωρίς επαναλήψεις ακμών
  - **Απλό μονοπάτι:** μονοπάτι χωρίς επαναλήψεις κορυφών
- **Κύκλος:** κλειστό μονοπάτι
  - **Απλός κύκλος:** απλό κλειστό μονοπάτι
- **Μήκος δρόμου:** το πλήθος των ακμών του

Γράφοι (επανάληψη): αναπαράσταση

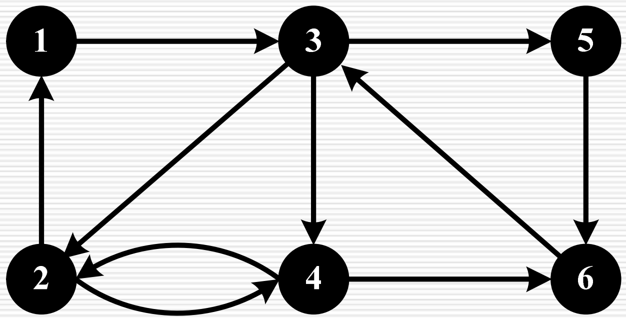
- … με **πίνακα γειτνίασης**: $A[i,j] = \begin{cases} 1 & (v_i, v_j) \in E \\ 0 & (v_i, v_j) \notin E \end{cases}$
  - Αν έχουμε βάρη, $A[i,j] = w(v_i, v_j)$
  - Μη-κατευθυνόμενος: συμμετρικός πίνακας
  - Χώρος: $\Theta(n^2)$
  - Προσπέλαση γειτόνων: $\Theta(n)$
  - Άμεσος έλεγχος ύπαρξης ακμής: $O(1)$



	1	2	3	4	5	6
1	0	1	1	0	0	0
2	1	0	1	0	0	0
3	1	1	0	1	1	0
4	0	0	1	0	0	1
5	0	0	1	0	0	0
6	0	0	0	1	0	0

Γράφοι (επανάληψη): αναπαράσταση

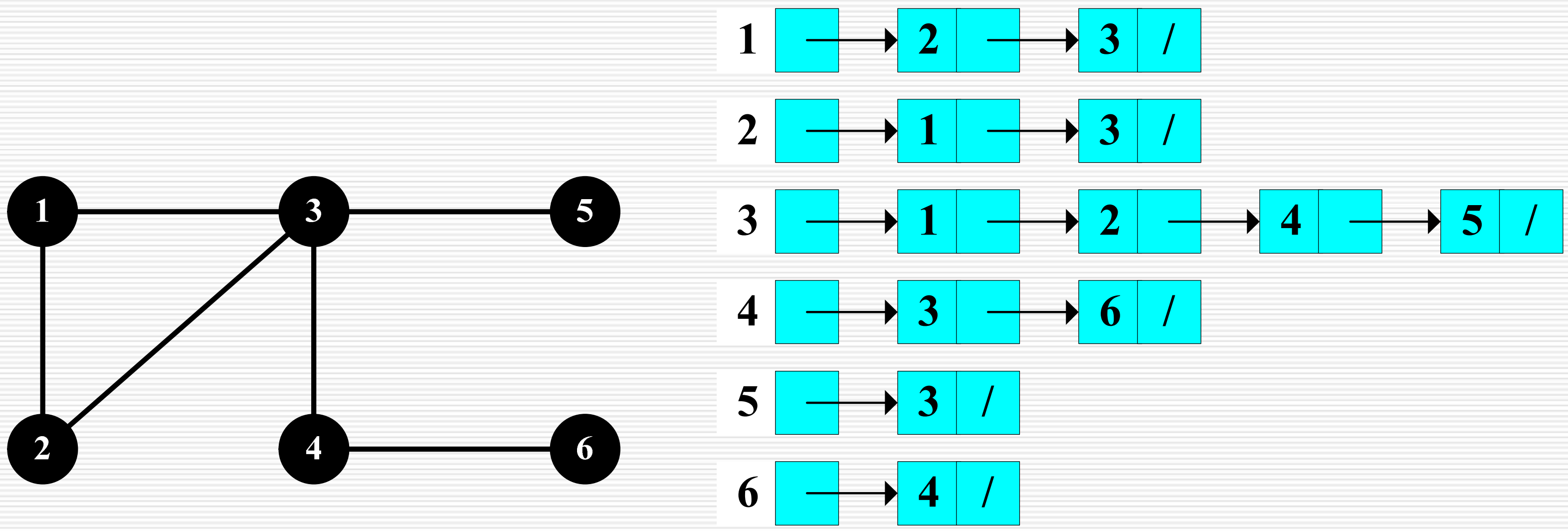
- … με **πίνακα γειτνίασης**: $A[i,j] = \begin{cases} 1 & (v_i, v_j) \in E \\ 0 & (v_i, v_j) \notin E \end{cases}$
  - Αν έχουμε βάρη, $A[i,j] = w(v_i, v_j)$
  - Κατευθυνόμενος: μη-συμμετρικός πίνακας
  - Χώρος: $\Theta(n^2)$
  - Προσπέλαση γειτόνων: $\Theta(n)$
  - Άμεσος έλεγχος ύπαρξης ακμής: $O(1)$



	1	2	3	4	5	6
1	0	0	1	0	0	0
2	1	0	0	1	0	0
3	0	1	0	1	1	0
4	0	1	0	0	0	1
5	0	0	0	0	0	1
6	0	0	1	0	0	0

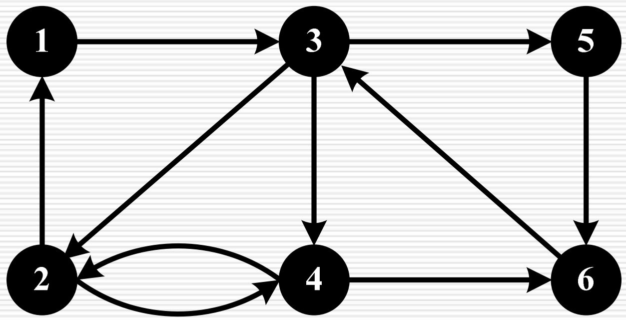
Γράφοι (επανάληψη): αναπαράσταση

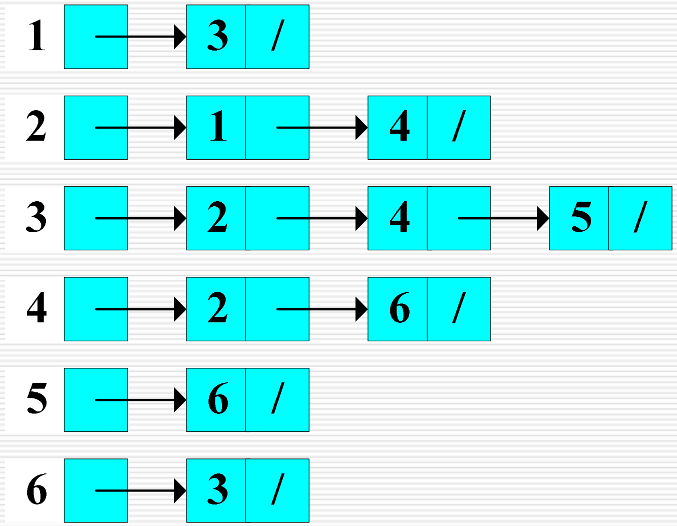
- ... με **λίστες γειτνίασης**: γειτονικές κορυφές σε λίστες
  - Αν έχουμε βάρη, τα αποθηκεύουμε στους κόμβους
  - Χώρος: Θ(m)
  - Προσπέλαση γειτόνων: Θ(deg(u))
  - Έλεγχος ύπαρξης ακμής: Ο(deg(u))



Γράφοι (επανάληψη): αναπαράσταση

- … με **λίστες γειτνίασης**: γειτονικές κορυφές σε λίστες
  - Αν έχουμε βάρη, τα αποθηκεύουμε στους κόμβους
  - Χώρος: Θ(m)
  - Προσπέλαση γειτόνων: Θ(deg(u))
  - Έλεγχος ύπαρξης ακμής: O(deg(u))



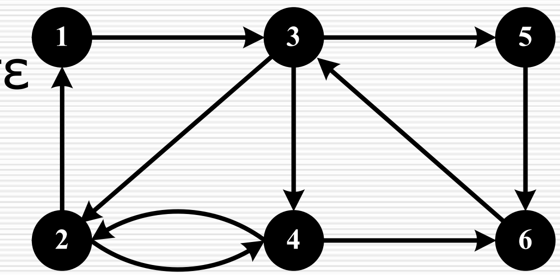


Γράφοι (επανάληψη): συνεκτικότητα

- Ένας μη κατευθυνόμενος γράφος λέγεται συνεκτικός (connected) αν υπάρχει δρόμος μεταξύ οποιωνδήποτε δύο κορυφών του

  Σε συνεκτικό γράφο ισχύει: $n-1 \leq e \leq \frac{n(n-1)}{2}, n = |V|, e = |E|.$

- Ένας κατευθυνόμενος γράφος λέγεται
  - ισχυρά συνεκτικός (strongly connected) αν υπάρχει δρόμος μεταξύ οποιωνδήποτε δύο κορυφών του ακολουθώντας τις κατευθύνσεις των ακμών



  - ασθενώς συνεκτικός (weakly connected) αν υπάρχει δρόμος μεταξύ οποιωνδήποτε δύο κορυφών του αγνοώντας τις κατευθύνσεις των ακμών

Συντομότερες διαδρομές (Dijkstra)

- Σε γράφο με βάρη αναζητούμε τις **διαδρομές ελαχίστου βάρους** από έναν κόμβο-αφετηρία s προς όλους τους άλλους
- Τα βάρη είναι μη αρνητικοί αριθμοί (π.χ. αποστάσεις)
- **Ιδέα Dijkstra**
  - προσωρινές ετικέτες απόστασης από s
  - για κόμβο w με ελάχιστη ετικέτα η διαδρομή ελαχίστου βάρους έχει βρεθεί!,
  - μονιμοποίηση ετικέτας του w
  - ενημέρωση ετικετών γειτόνων του κόμβου w

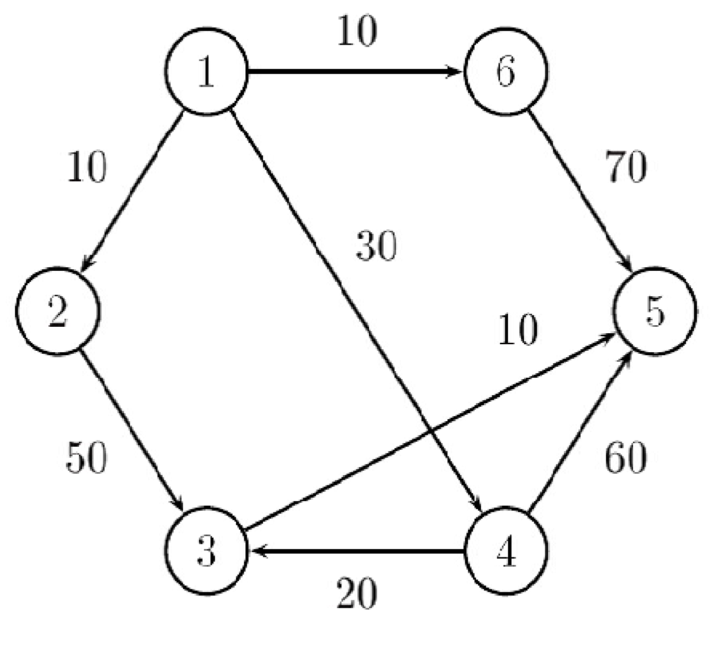
Αλγόριθμος Dijkstra

```
D(s) := 0 ; P(s) := s ; S := {}
for each v<>s do D(v) := ∞

repeat n times
   select w from V\S with minimum D(w)
   S := S U {w}
   for all v in V\S do
      if D(u) + cost(u,v) < D(v) then
         D(v) := D(u) + cost(u,v)
         P(v) := u
```

Πολυπλ/τα $O(|V|^2)$: σε κάθε επανάληψη $O(|V|)$ για εύρεση ελαχίστου, $O(|V|)$ για ενημέρωση αποστάσεων

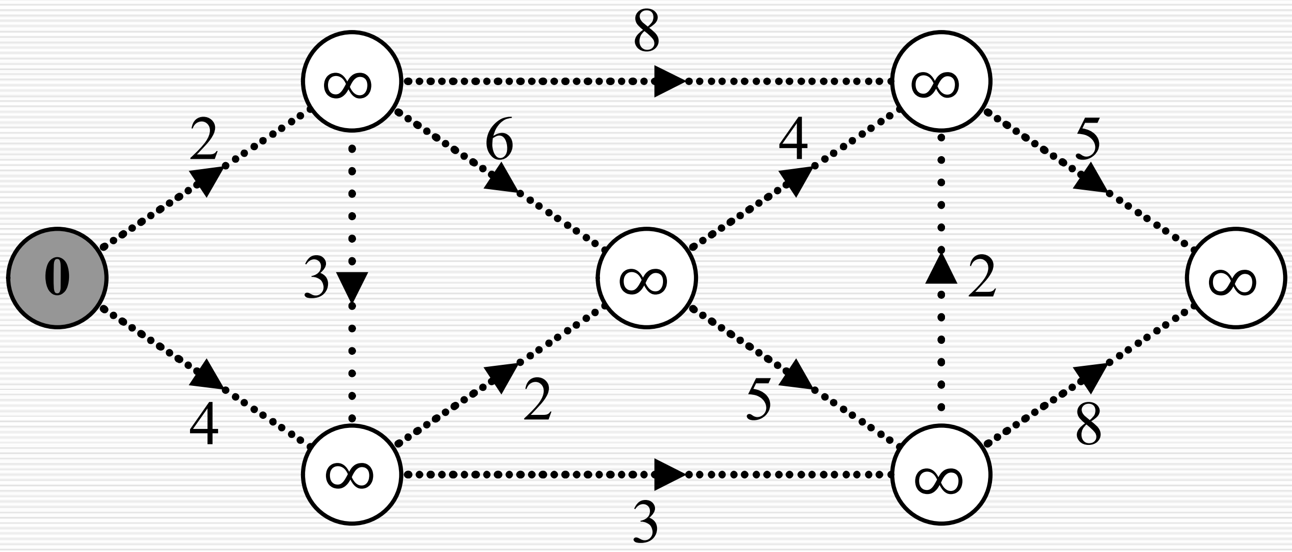
Παράδειγμα Dijkstra

```
D(s) := 0 ; P(s) := s ; S := {}
for each v<>s do D(v) := ∞ ; P(v) := s

repeat n times
    select w from V\S with minimum D(w)
    S := S U {w}
    for all v in V\S do
        if D(u) + cost(u,v) < D(v) then
            D(v) := D(u) + cost(u,v)
            P(v) := u
```



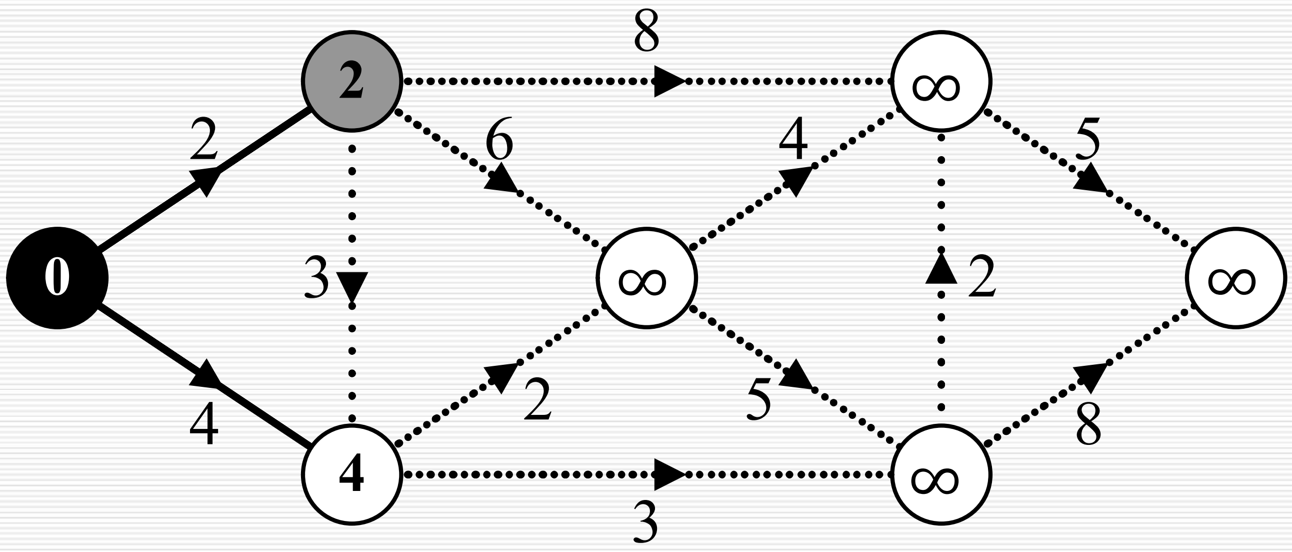
Βήμα	S	w	D					P				
			2	3	4	5	6	2	3	4	5	6
-	{1}	-	10	∞	30	∞	10	1	1	1	1	1
2	{1,2}	2		60	30	∞	10		2			
3	{1,2,6}	6		60	30	80					6	
4	{1,2,6,4}	4		50		80			4			
5	{1,2,6,4,3}	3				60					3	
6	{1,2,6,4,3,5}	5										

2ο παράδειγμα Dijkstra



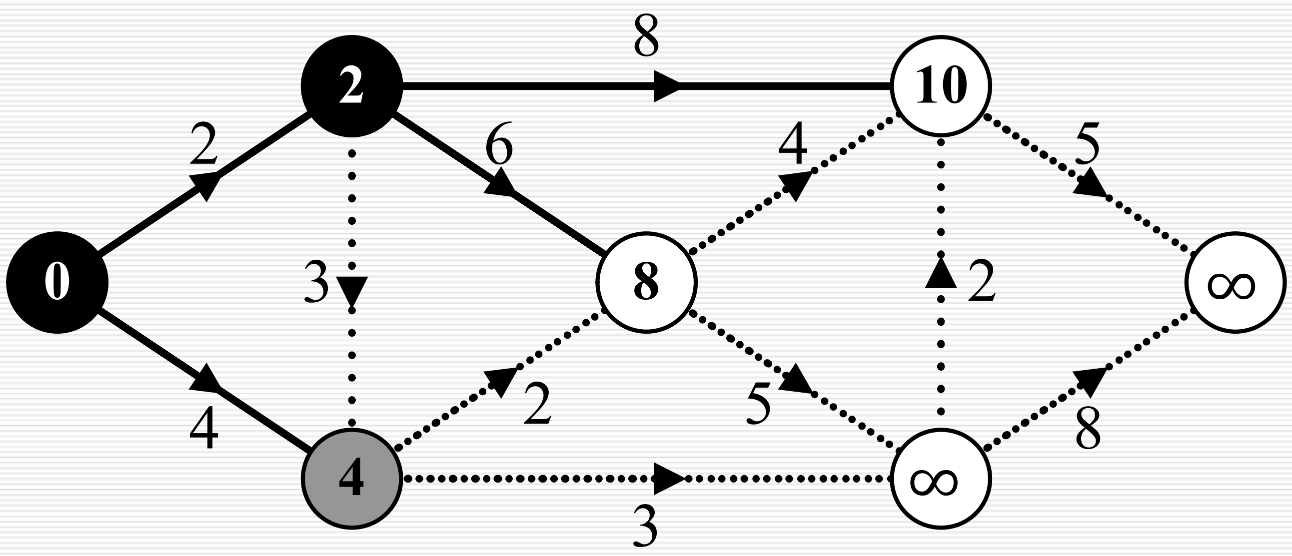
Οι ετικέτες των κόμβων δείχνουν την μέχρι στιγμής ελάχιστη απόσταση από τον αρχικό κόμβο (πίνακας D).

2ο παράδειγμα Dijkstra



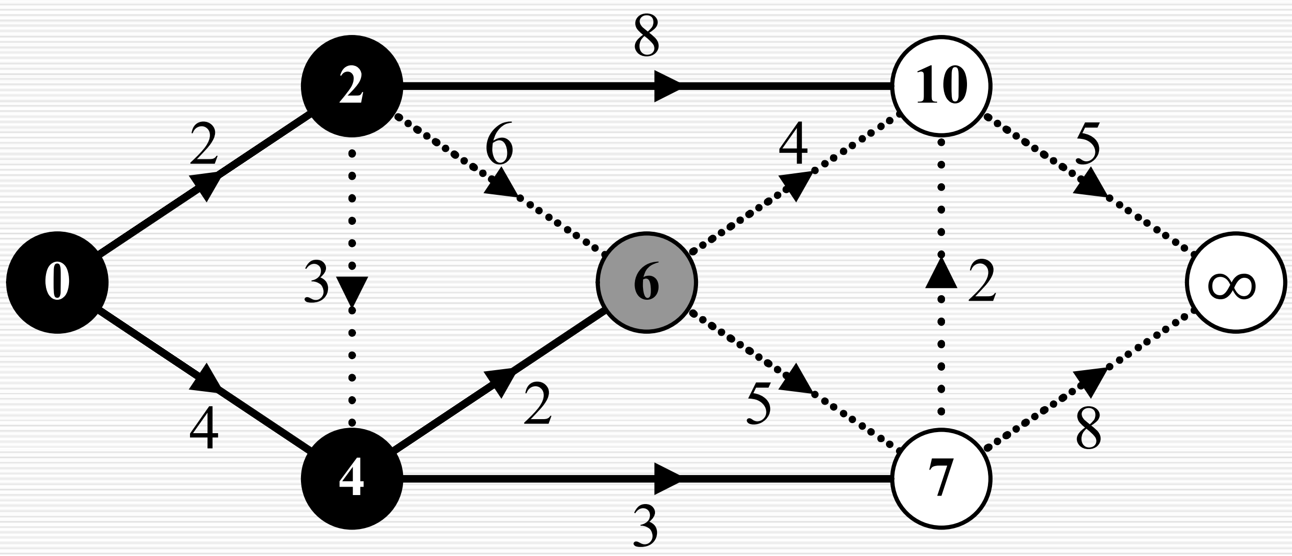
Οι ετικέτες των κόμβων δείχνουν την μέχρι στιγμής ελάχιστη απόσταση από τον αρχικό κόμβο (πίνακας D), οι συνεχείς ακμές δείχνουν ποιος είναι ο αντίστοιχος προηγούμενος κόμβος (πίνακας P).

2o παράδειγμα Dijkstra



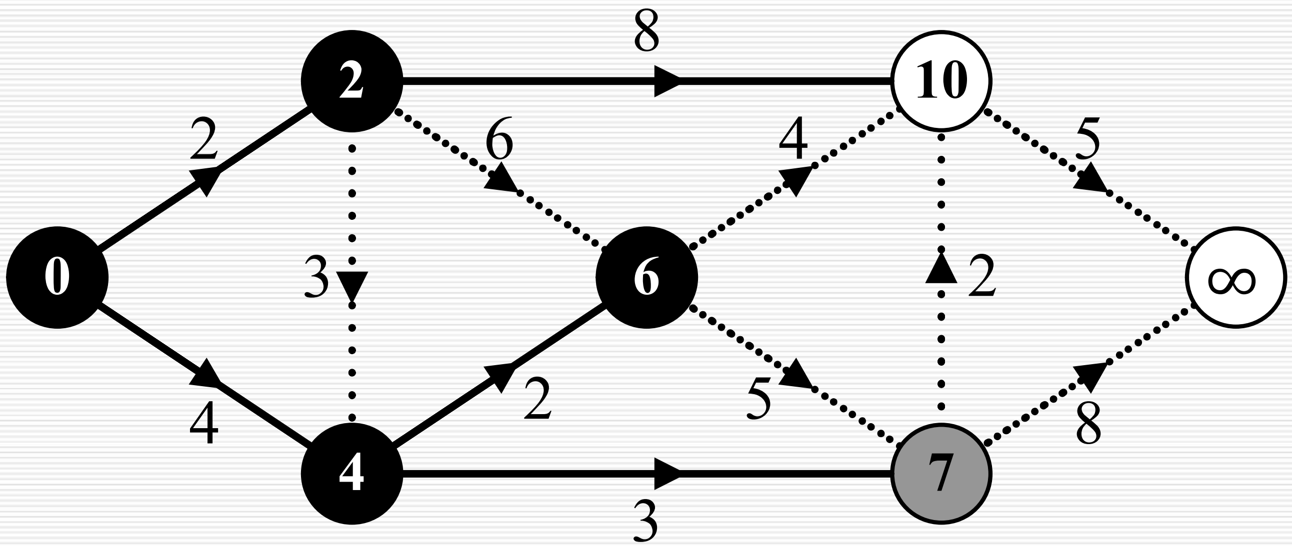
Οι ετικέτες των κόμβων δείχνουν την μέχρι στιγμής ελάχιστη απόσταση από τον αρχικό κόμβο (πίνακας D), οι συνεχείς ακμές δείχνουν ποιος είναι ο αντίστοιχος προηγούμενος κόμβος (πίνακας P).

2ο παράδειγμα Dijkstra



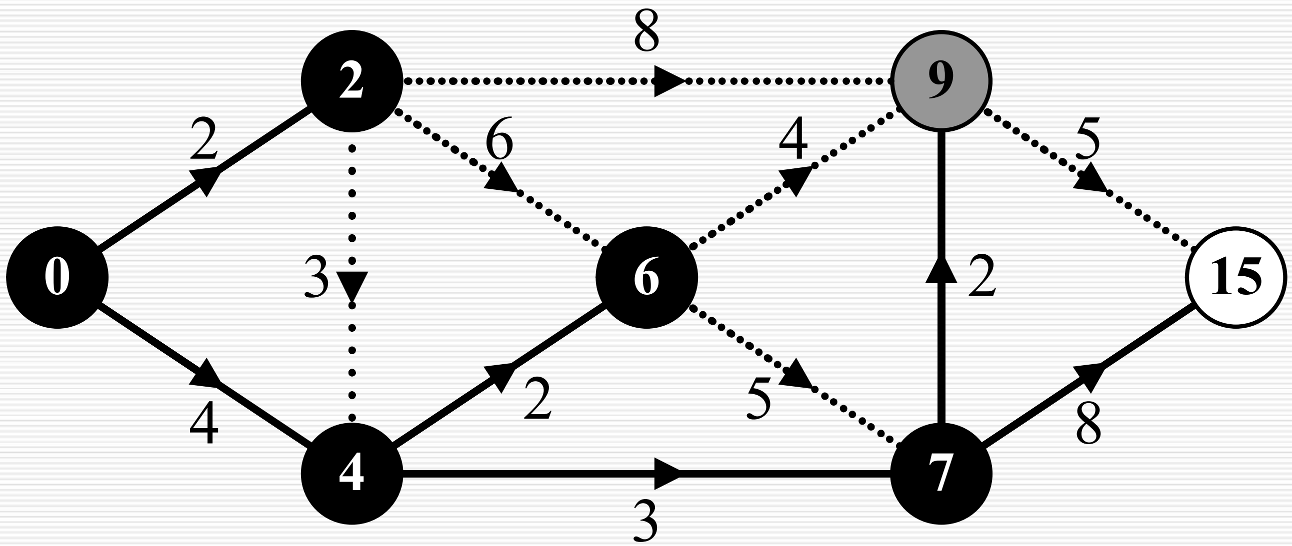
Οι ετικέτες των κόμβων δείχνουν την μέχρι στιγμής ελάχιστη απόσταση από τον αρχικό κόμβο (πίνακας D), οι συνεχείς ακμές δείχνουν ποιος είναι ο αντίστοιχος προηγούμενος κόμβος (πίνακας P).

2ο παράδειγμα Dijkstra



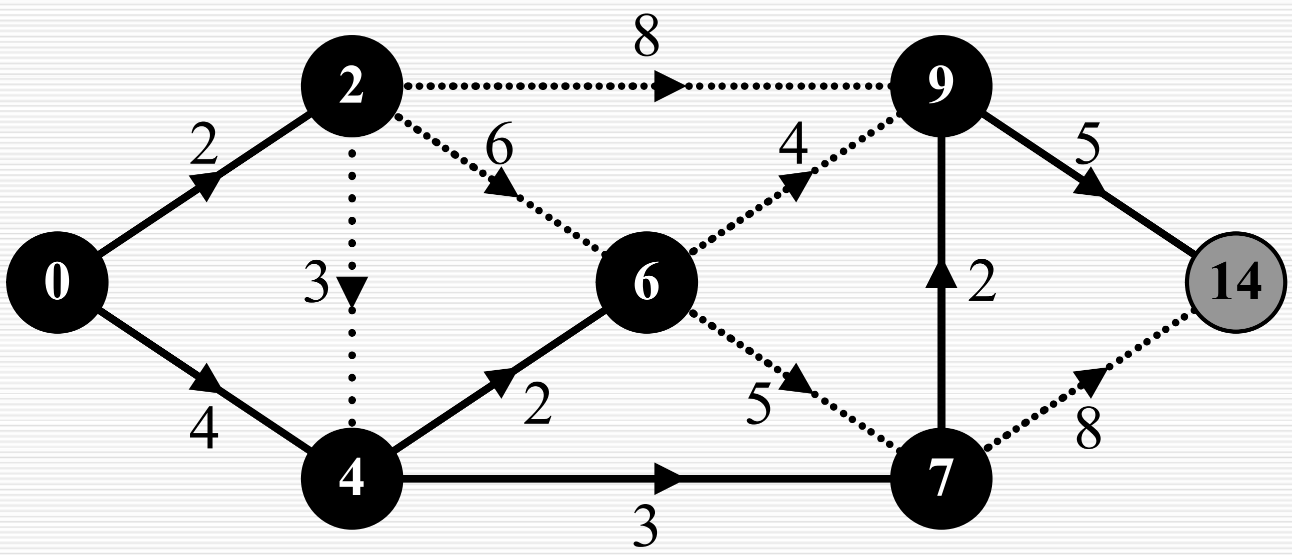
Οι ετικέτες των κόμβων δείχνουν την μέχρι στιγμής ελάχιστη απόσταση από τον αρχικό κόμβο (πίνακας D), οι συνεχείς ακμές δείχνουν ποιος είναι ο αντίστοιχος προηγούμενος κόμβος (πίνακας P).

2ο παράδειγμα Dijkstra



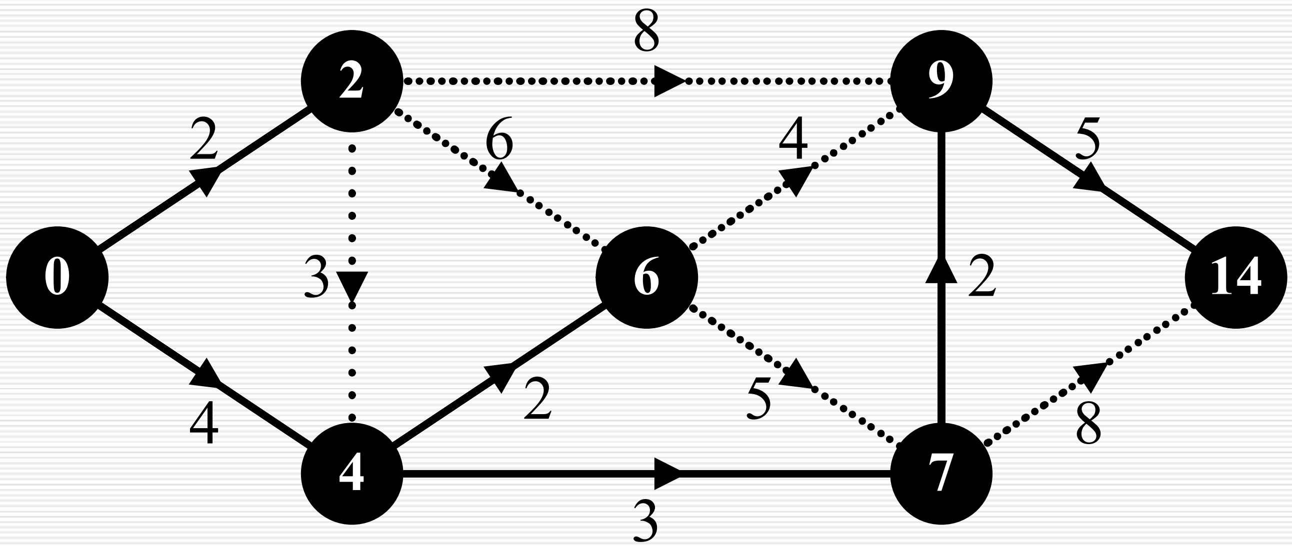
Οι ετικέτες των κόμβων δείχνουν την μέχρι στιγμής ελάχιστη απόσταση από τον αρχικό κόμβο (πίνακας D), οι συνεχείς ακμές δείχνουν ποιος είναι ο αντίστοιχος προηγούμενος κόμβος (πίνακας P).

2ο παράδειγμα Dijkstra



Οι ετικέτες των κόμβων δείχνουν την μέχρι στιγμής ελάχιστη απόσταση από τον αρχικό κόμβο (πίνακας D), οι συνεχείς ακμές δείχνουν ποιος είναι ο αντίστοιχος προηγούμενος κόμβος (πίνακας P).

2ο παράδειγμα Dijkstra

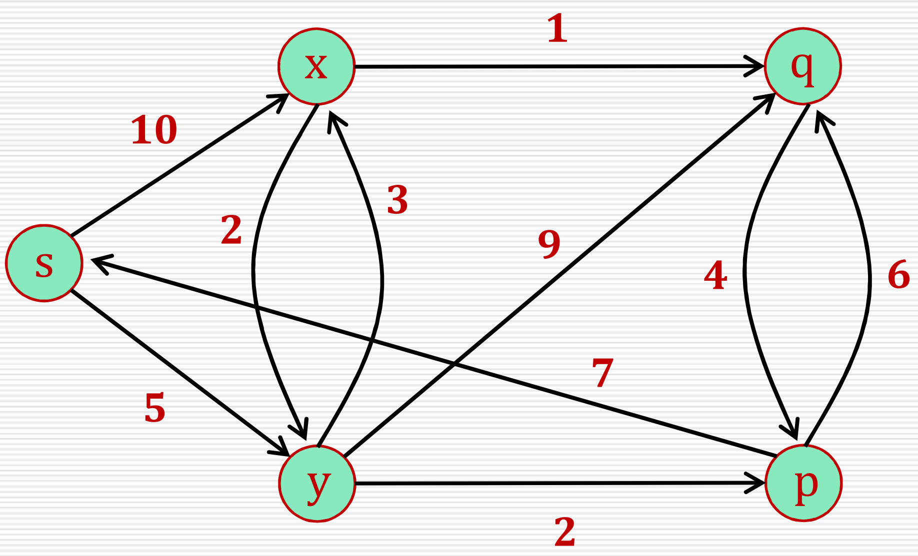


Οι ετικέτες των κόμβων δείχνουν την μέχρι στιγμής ελάχιστη απόσταση από τον αρχικό κόμβο (πίνακας D), οι συνεχείς ακμές δείχνουν ποιος είναι ο αντίστοιχος προηγούμενος κόμβος (πίνακας P).

3ο παράδειγμα Dijkstra

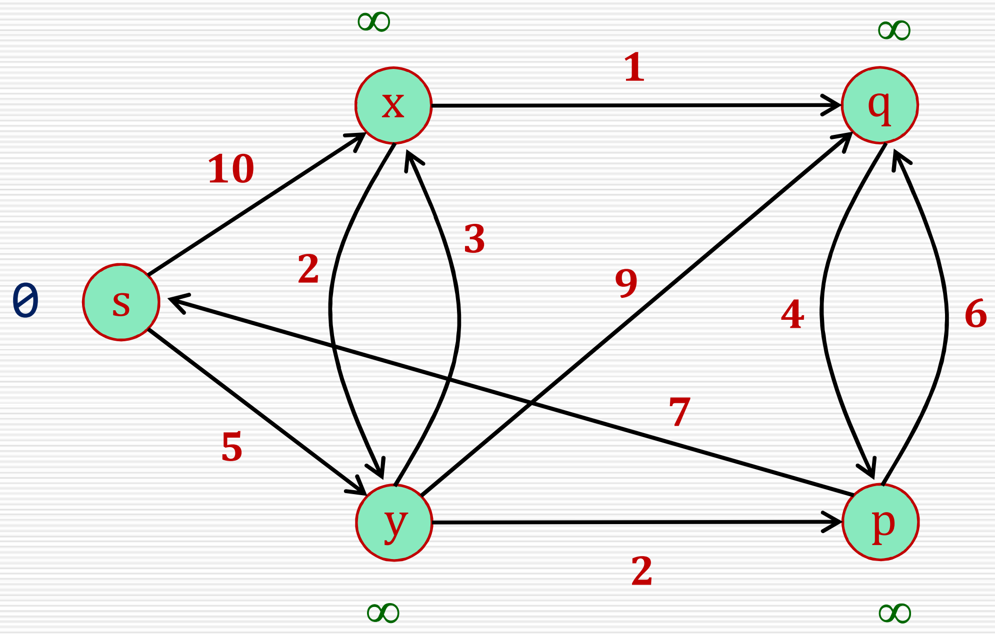
- Παράδειγμα

Είσοδος Αλγόριθμου



3ο παράδειγμα Dijkstra

Παράδειγμα



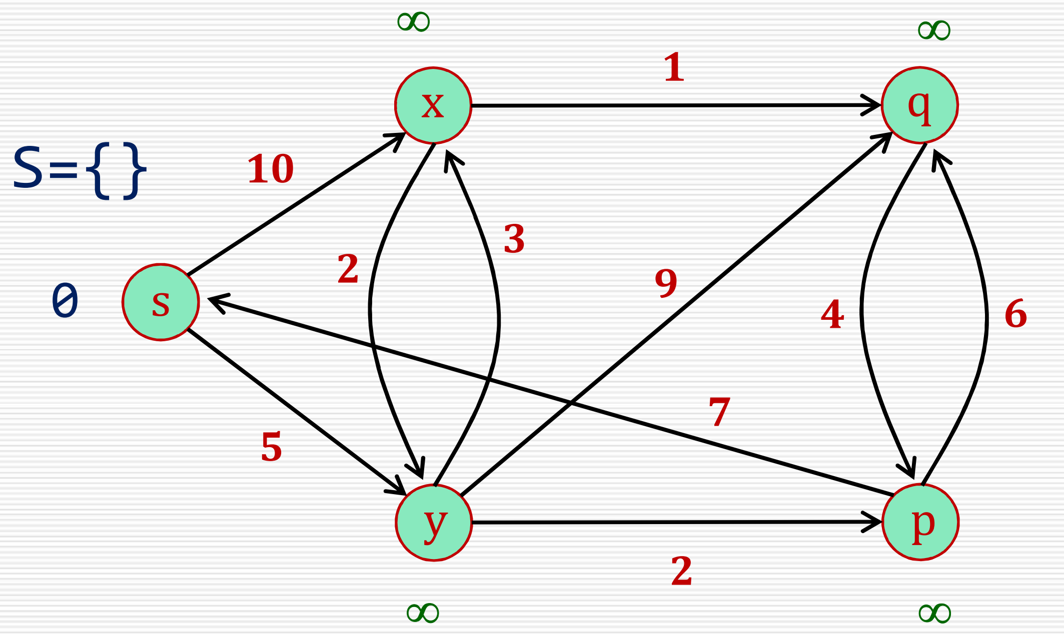

```
Initialize(G,s)
  για κάθε κόμβο v∈V:
      d(v) ← ∞
      prev(v) ← NIL
  d(s) ← 0
```

d(s)=0	d(x)=∞	d(y)=∞	d(p)=∞	d(q)=∞
prev(s)=NIL	prev(x)=NIL	prev(y)=NIL	prev(p)=NIL	prev(q)=NIL

3ο παράδειγμα Dijkstra

- Παράδειγμα

$Q = \{s,x,y,p,q\}$



d(s)=0 d(x)=∞ d(y)=∞ d(p)=∞ d(q)=∞

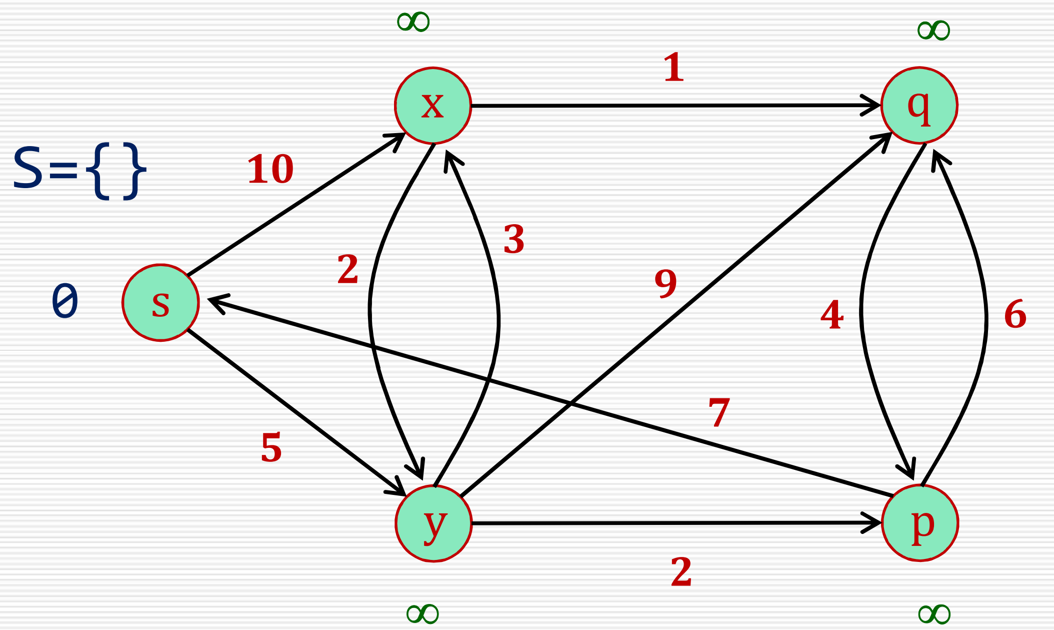
prev(s)=NIL prev(x)=NIL prev(y)=NIL prev(p)=NIL prev(q)=NIL

3ο παράδειγμα Dijkstra

Παράδειγμα

$Q = \{s,x,y,p,q\}$

while $Q \neq \varnothing$:



$d(s)=0$	$d(x)=\infty$	$d(y)=\infty$	$d(p)=\infty$	$d(q)=\infty$
prev(s)=NIL	prev(x)=NIL	prev(y)=NIL	prev(p)=NIL	prev(q)=NIL

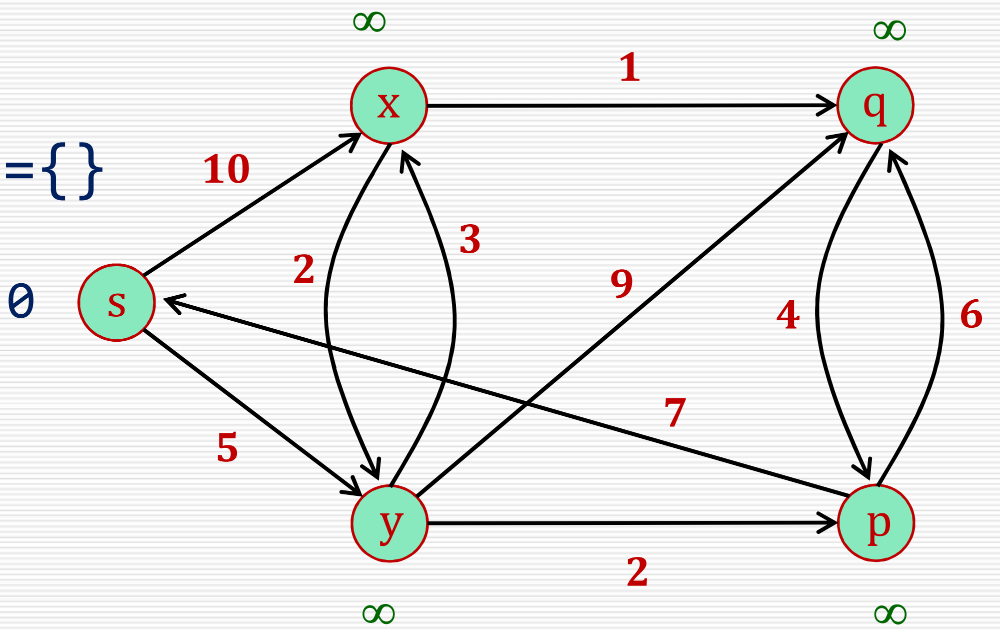
3ο παράδειγμα Dijkstra

Q = {s,x,y,p,q}

- Παράδειγμα

```
while Q ≠ ∅ :
    u ← extract-min(Q);
```

S={}



d(s)=0 d(x)=∞ d(y)=∞ d(p)=∞ d(q)=∞

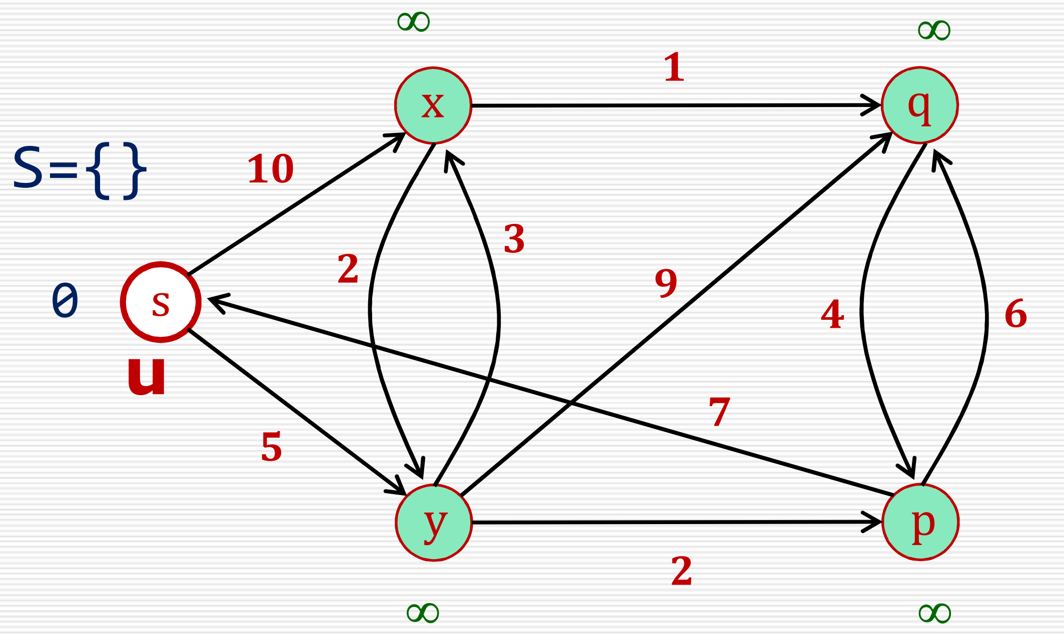
prev(s)=NIL prev(x)=NIL prev(y)=NIL prev(p)=NIL prev(q)=NIL

3ο παράδειγμα Dijkstra

$Q = \{x,y,p,q\}$

- Παράδειγμα

```
while Q ≠ ∅ :
    u ← extract-min(Q);
```



d(s)=0 prev(s)=NIL
d(x)=∞ prev(x)=NIL
d(y)=∞ prev(y)=NIL
d(p)=∞ prev(p)=NIL
d(q)=∞ prev(q)=NIL

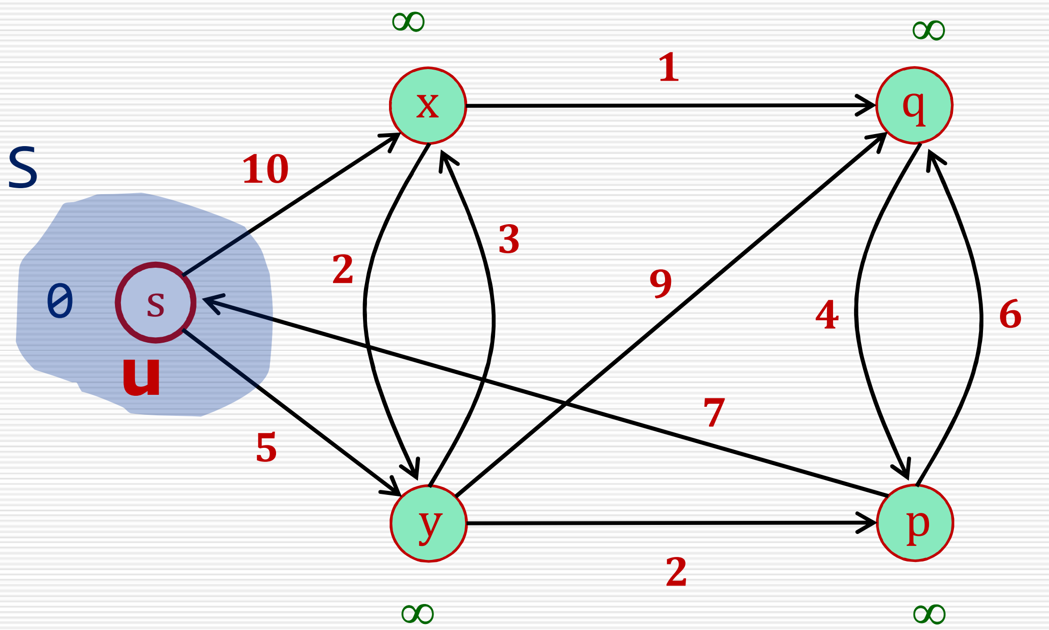
3ο παράδειγμα Dijkstra

Q = {x,y,p,q}

- Παράδειγμα

while Q ≠ ∅:

    u ← extract-min(Q);
    S ← S ∪ {u}



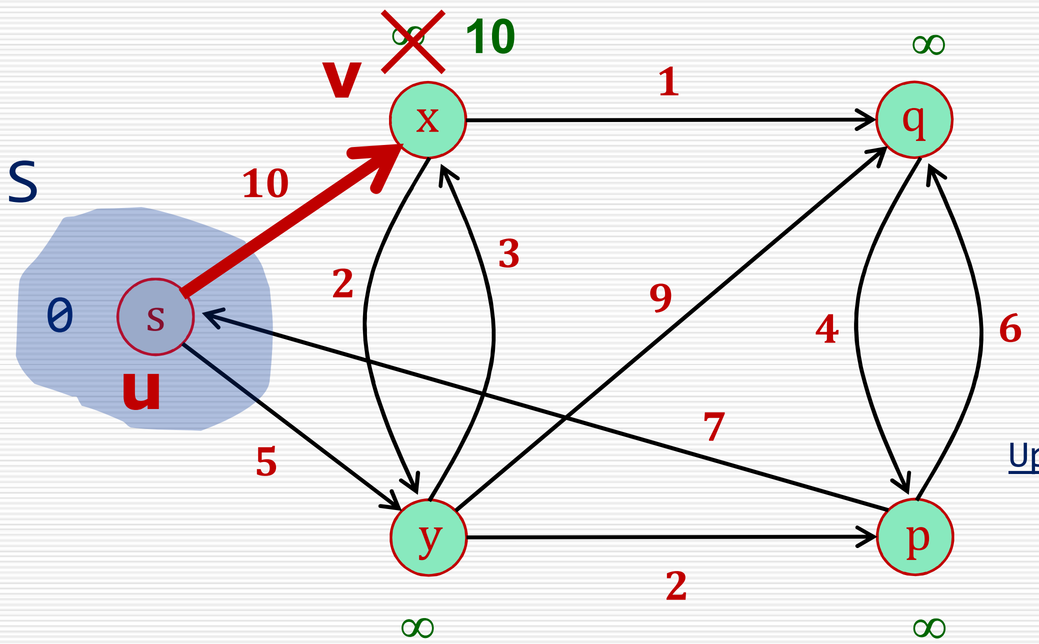
d(s)=0 d(x)=∞ d(y)=∞ d(p)=∞ d(q)=∞

prev(s)=NIL prev(x)=NIL prev(y)=NIL prev(p)=NIL prev(q)=NIL

3ο παράδειγμα Dijkstra

$Q = \{x, y, p, q\}$

- Παράδειγμα



while $Q \neq \emptyset$:

u ← extract-min(Q);

S ← S ∪ {u}

για κάθε κόμβο v∈N(u):

Update(u,v,c)

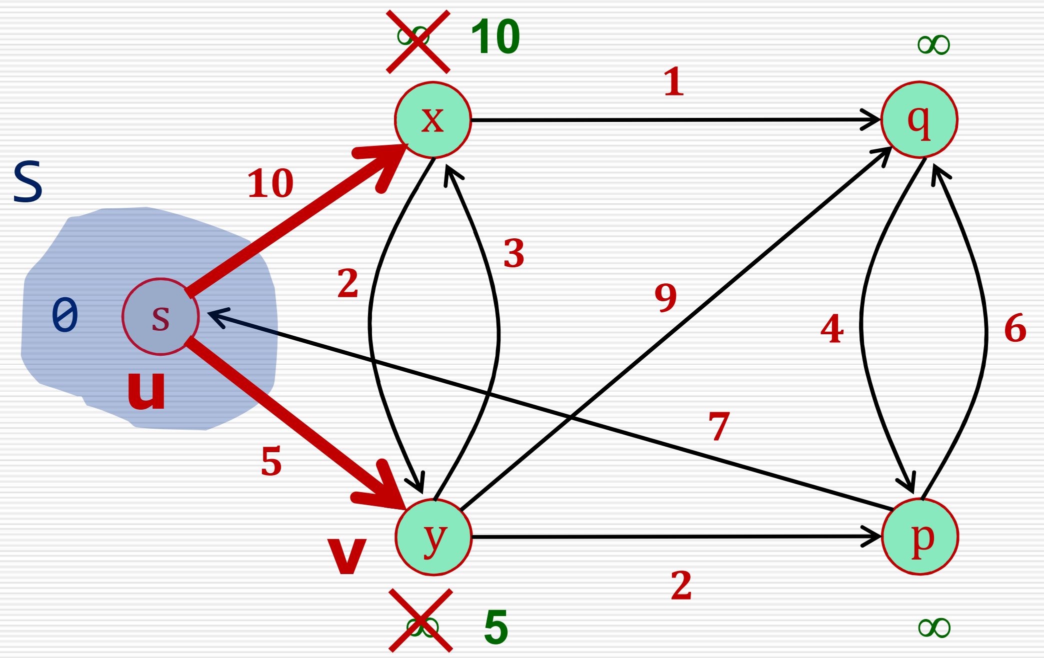
<u>Update(u,v,c)</u>

```
if d(v)>d(u)+c(u,v) then
   d(v) ← d(u)+c(u,v)
   prev(v) ← u
```

d(s)=0	d(x)=10	d(y)=∞	d(p)=∞	d(q)=∞
prev(s)=NIL	prev(x)=s	prev(y)=NIL	prev(p)=NIL	prev(q)=NIL

3ο παράδειγμα Dijkstra

Παράδειγμα



d(s)=0 prev(s)=NIL

d(x)=10 prev(x)=s

d(y)=5 prev(y)=s

d(p)=∞ prev(p)=NIL

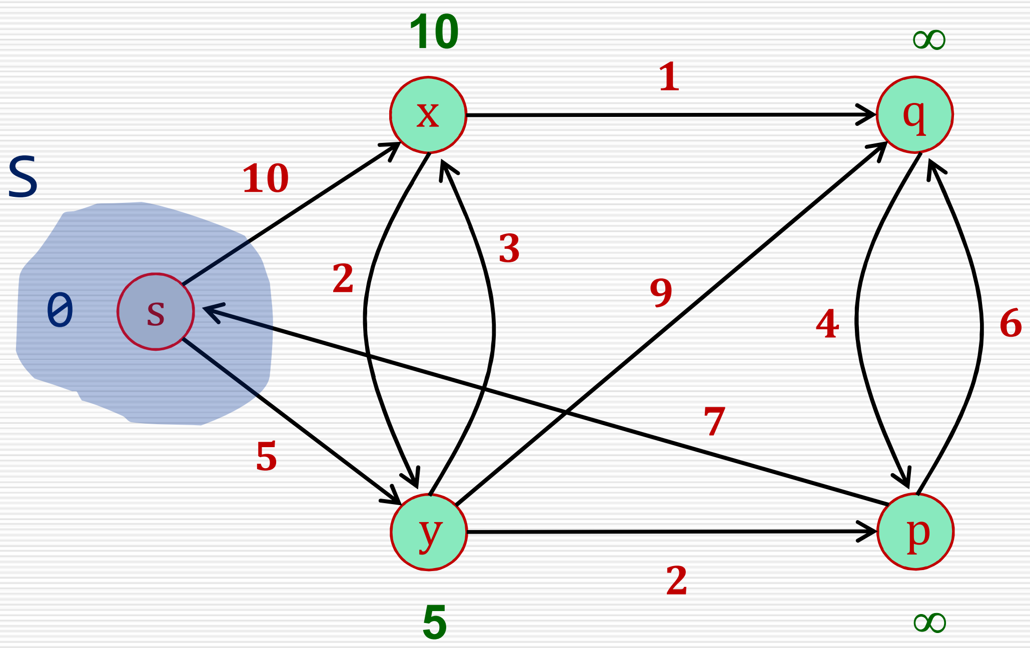
d(q)=∞ prev(q)=NIL

3ο παράδειγμα Dijkstra

$Q = \{x, y, p, q\}$

- Παράδειγμα

```
while Q ≠ ∅ :
    u ← extract-min(Q);
```



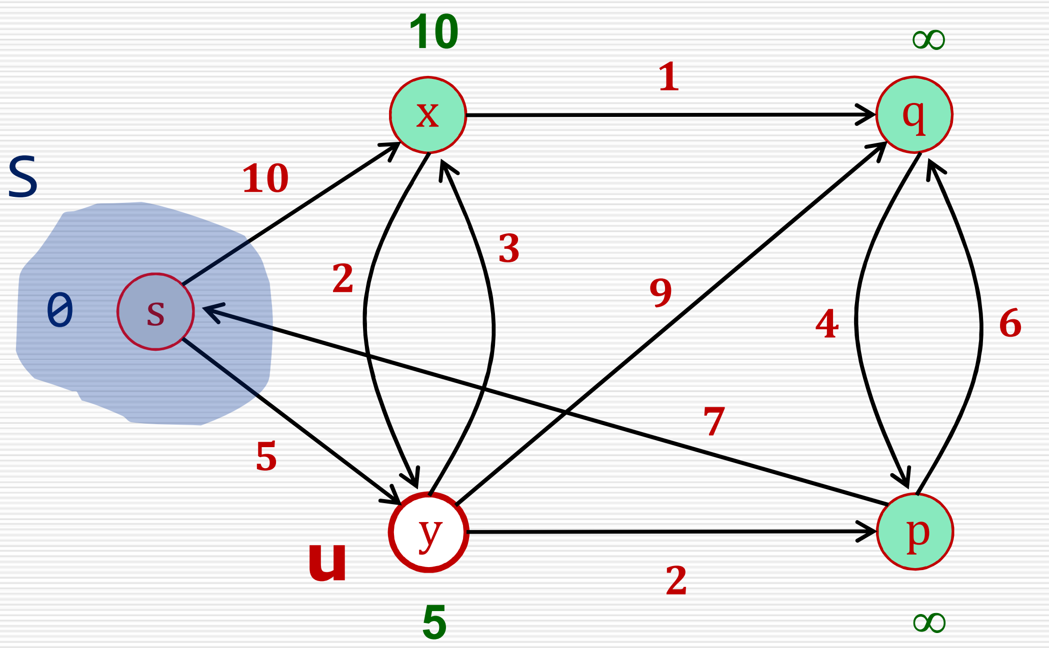
d(s)=0	d(x)=10	d(y)=5	d(p)=∞	d(q)=∞
prev(s)=NIL	prev(x)=s	prev(y)=s	prev(p)=NIL	prev(q)=NIL

3ο παράδειγμα Dijkstra

$Q = \{x,p,q\}$

- Παράδειγμα

```
while Q ≠ ∅:
    u ← extract-min(Q);
```



d(s)=0 d(x)=10 d(y)=5 d(p)=∞ d(q)=∞

prev(s)=NIL prev(x)=s prev(y)=s prev(p)=NIL prev(q)=NIL

3ο παράδειγμα Dijkstra

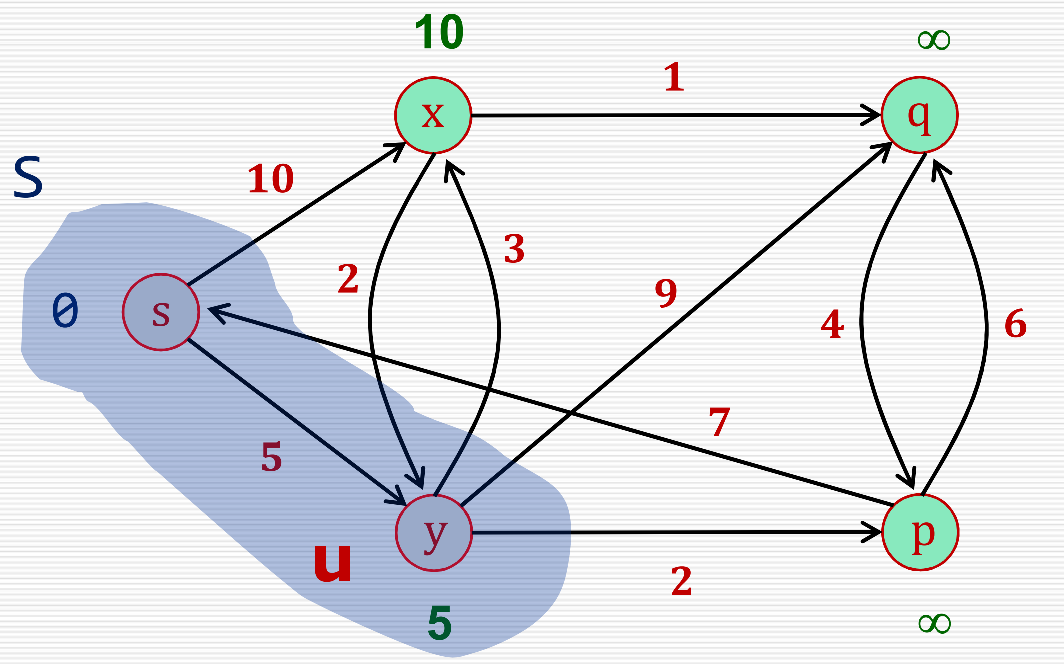
$Q = \{x, p, q\}$

- Παράδειγμα

while $Q \neq \varnothing$:

  u ← extract-min(Q);

  S ← S ∪ {u}



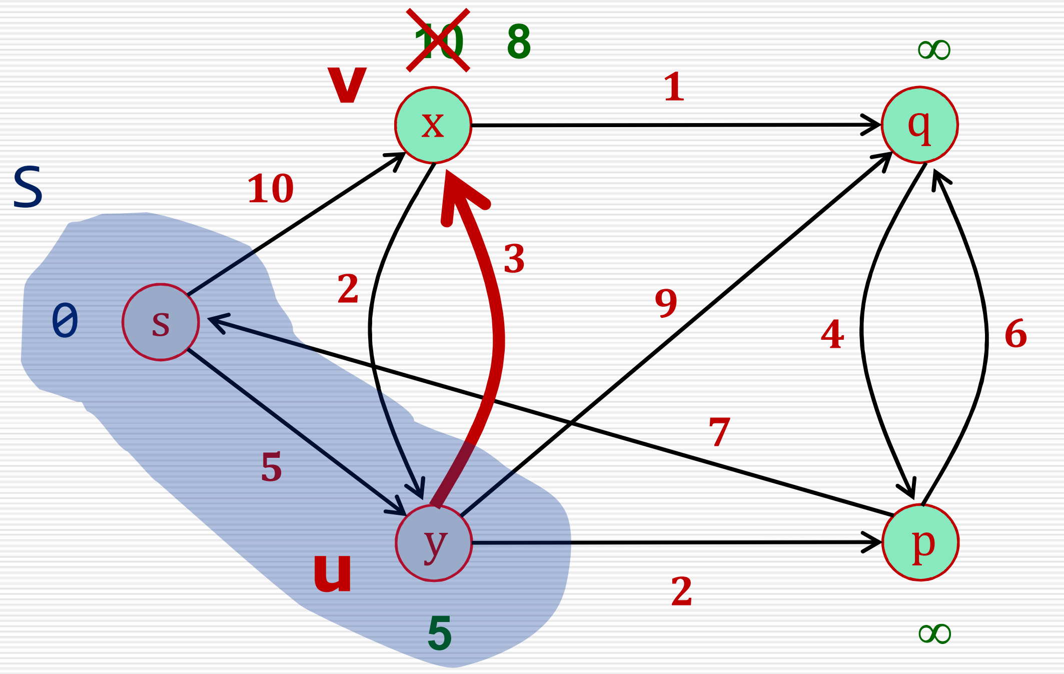
d(s)=0 d(x)=10 d(y)=5 d(p)=∞ d(q)=∞

prev(s)=NIL prev(x)=s prev(y)=s prev(p)=NIL prev(q)=NIL

3ο παράδειγμα Dijkstra

$Q = \{x,p,q\}$

- Παράδειγμα



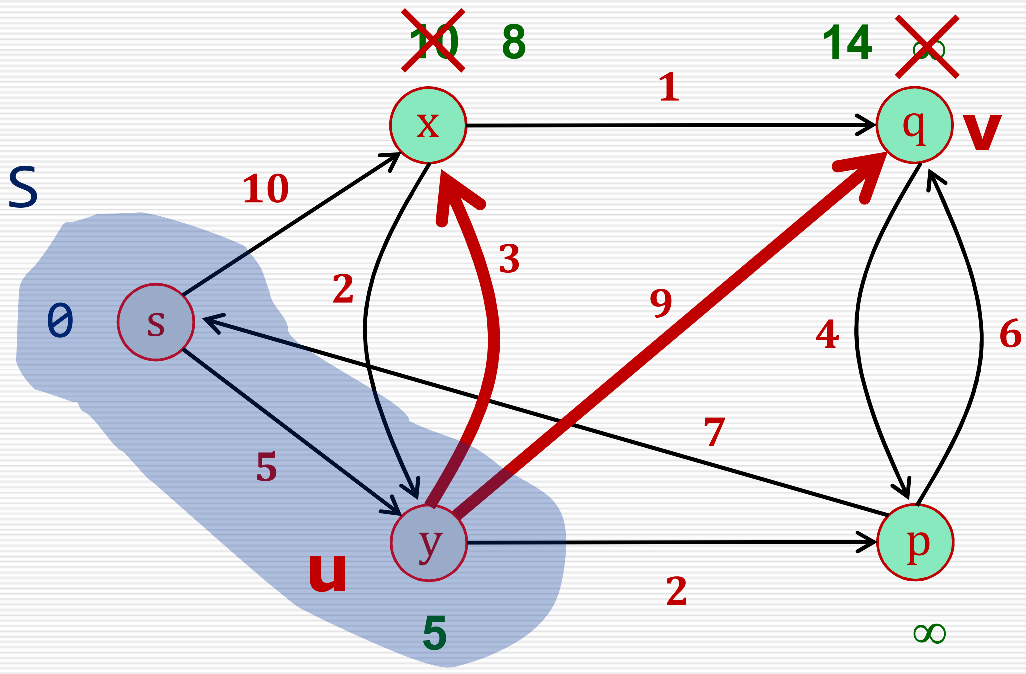

```
while Q ≠ ∅ :
    u ← extract-min(Q);
    S ← S ∪ {u}
    για κάθε κόμβο v∈N(u):
        Update(u,v,c)
```

d(s)=0	d(x)=8	d(y)=5	d(p)=∞	d(q)=∞
prev(s)=NIL	prev(x)=y	prev(y)=s	prev(p)=NIL	prev(q)=NIL

3ο παράδειγμα Dijkstra

$Q = \{x,p,q\}$

Παράδειγμα



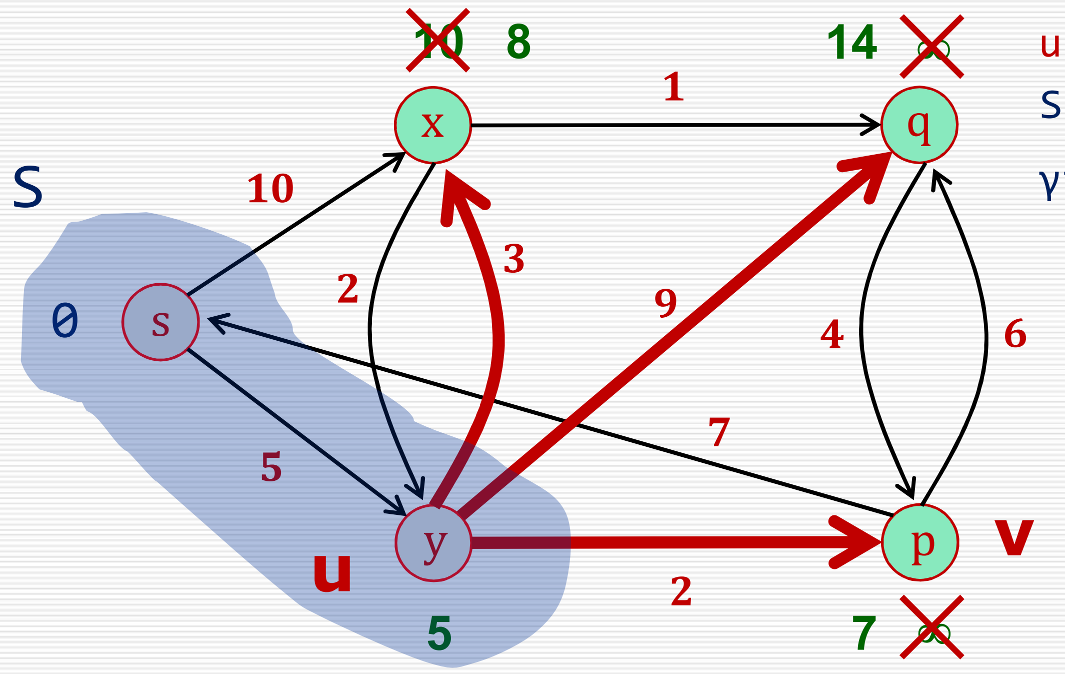

```
while Q ≠ ∅ :
  u ← extract-min(Q);
  S ← S ∪ {u}
  για κάθε κόμβο v∈N(u):
     Update(u,v,c)
```

d(s)=0	d(x)=8	d(y)=5	d(p)=∞	d(q)=14
prev(s)=NIL	prev(x)=y	prev(y)=s	prev(p)=NIL	prev(q)=y

3ο παράδειγμα Dijkstra

$Q = \{x,p,q\}$

- Παράδειγμα




```
while Q ≠ ∅ :
  u ← extract-min(Q);
  S ← S ∪ {u}
  για κάθε κόμβο v∈N(u):
     Update(u,v,c)
```

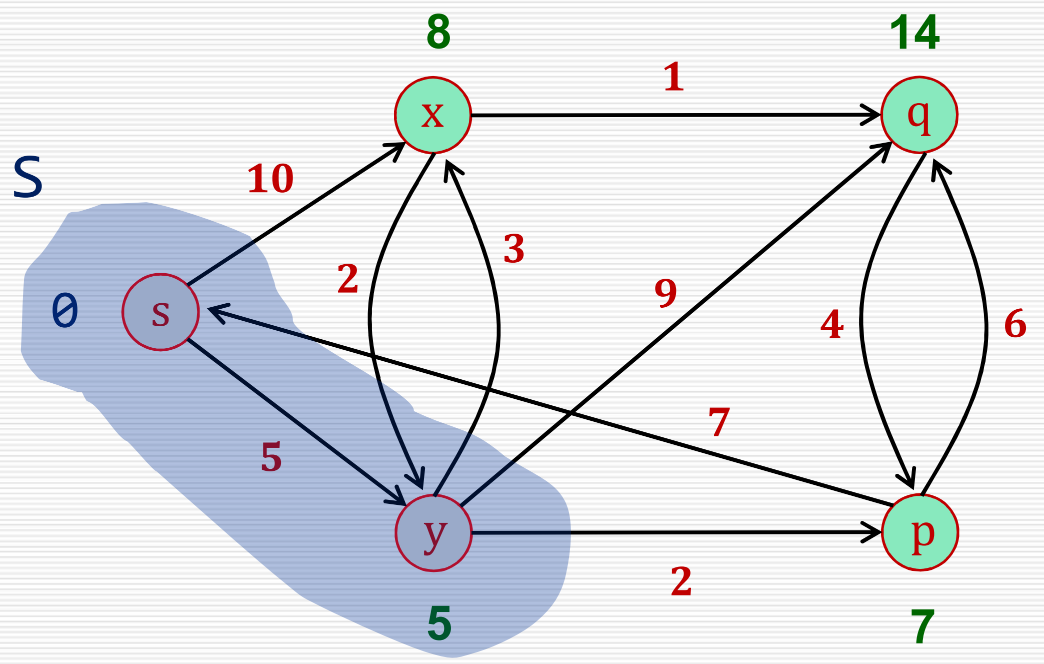
d(s)=0	d(x)=8	d(y)=5	d(p)=7	d(q)=13
prev(s)=NIL	prev(x)=y	prev(y)=s	prev(p)=y	prev(q)=y

3ο παράδειγμα Dijkstra

$Q = \{x,p,q\}$

- Παράδειγμα

```
while Q ≠ ∅ :
    u ← extract-min(Q);
```



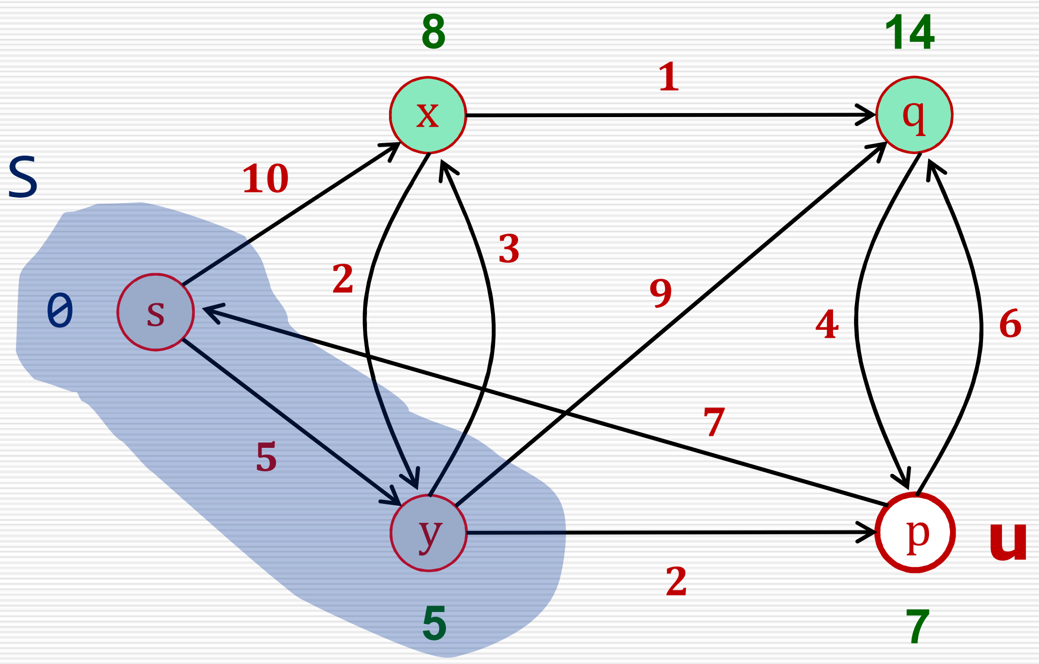
d(s)=0 d(x)=8 d(y)=5 d(p)=7 d(q)=13

prev(s)=NIL prev(x)=y prev(y)=s prev(p)=y prev(q)=y

3ο παράδειγμα Dijkstra

$Q = \{x, q\}$

- Παράδειγμα



while $Q \neq \varnothing$:

u ← extract-min(Q);

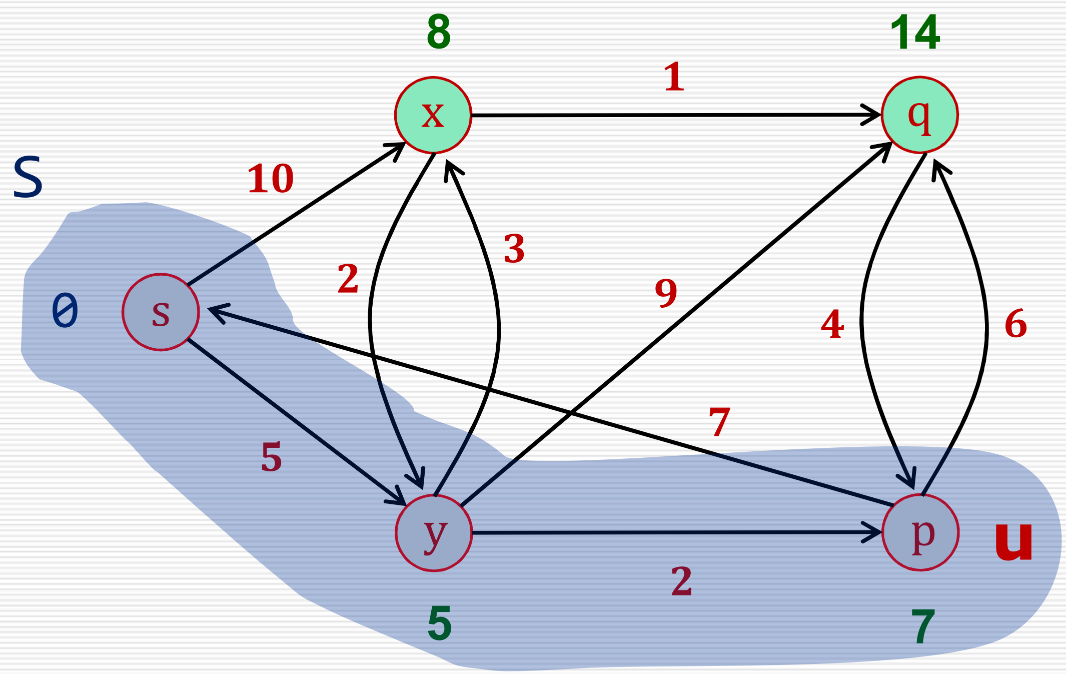
d(s)=0	d(x)=8	d(y)=5	d(p)=7	d(q)=13
prev(s)=NIL	prev(x)=y	prev(y)=s	prev(p)=y	prev(q)=y

3ο παράδειγμα Dijkstra

Q = {x,q}

- Παράδειγμα

```
while Q ≠ ∅:
    u ← extract-min(Q);
    S ← S ∪ {u}
```



d(s)=0 prev(s)=NIL

d(x)=8 prev(x)=y

d(y)=5 prev(y)=s

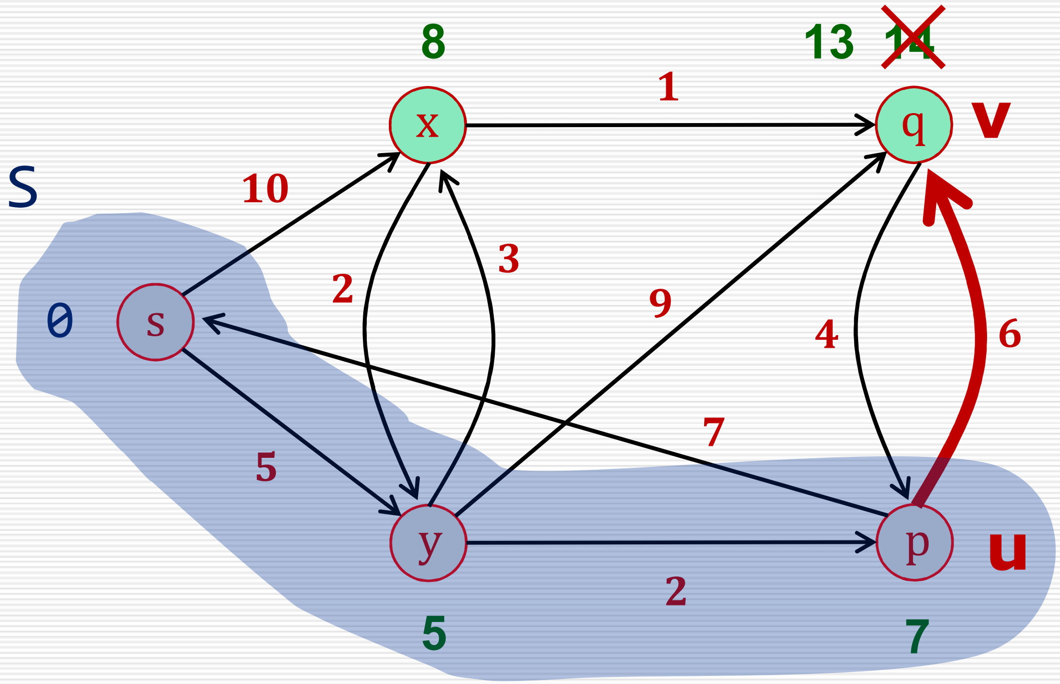
d(p)=7 prev(p)=y

d(q)=13 prev(q)=y

3ο παράδειγμα Dijkstra

$Q = \{x,q\}$

Παράδειγμα




```
while Q ≠ ∅ :
   u ← extract-min(Q);
   S ← S ∪ {u}
   για κάθε κόμβο v∈N(u)
       Update(u,v,c)
```

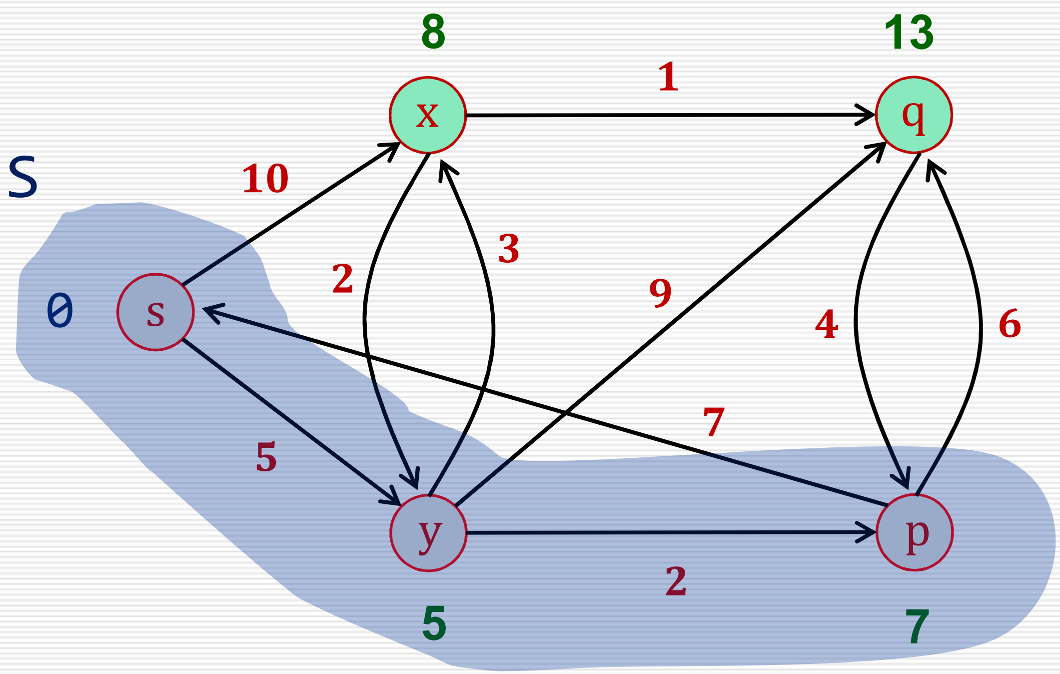
d(s)=0	d(x)=8	d(y)=5	d(p)=7	d(q)=13
prev(s)=NIL	prev(x)=y	prev(y)=s	prev(p)=y	prev(q)=y

3ο παράδειγμα Dijkstra

$Q = \{x,q\}$

- Παράδειγμα

```
while Q ≠ ∅ :
    u ← extract-min(Q);
```



d(s)=0 d(x)=8 d(y)=5 d(p)=7 d(q)=13

prev(s)=NIL prev(x)=y prev(y)=s prev(p)=y prev(q)=y

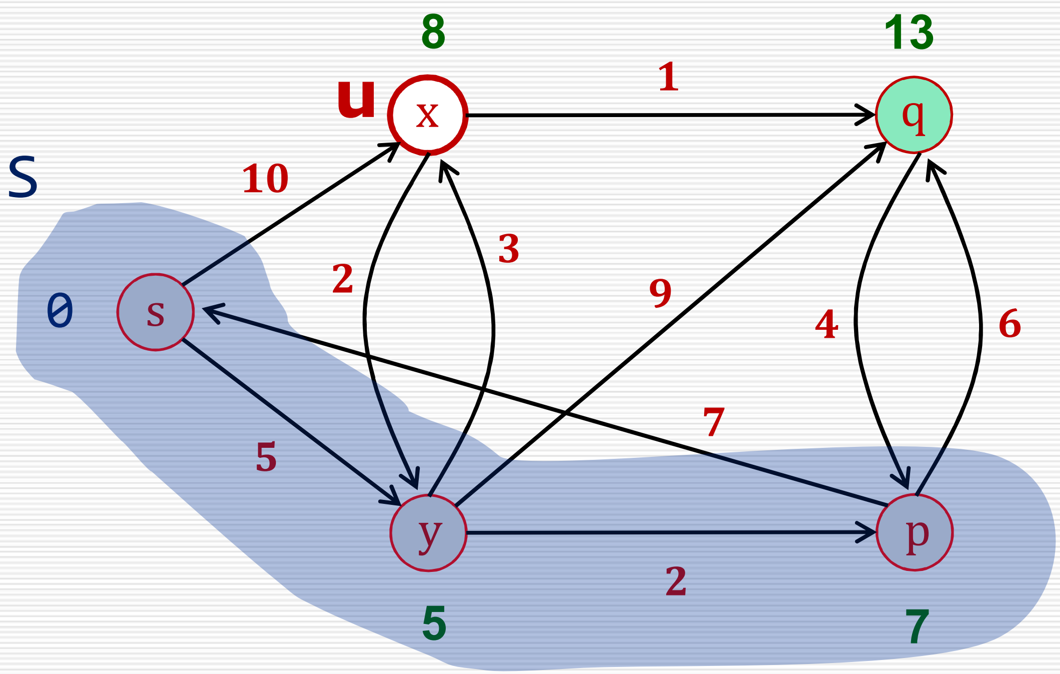
3ο παράδειγμα Dijkstra

Q = {q}

Παράδειγμα

while Q ≠ ∅ :

u ← extract-min(Q);



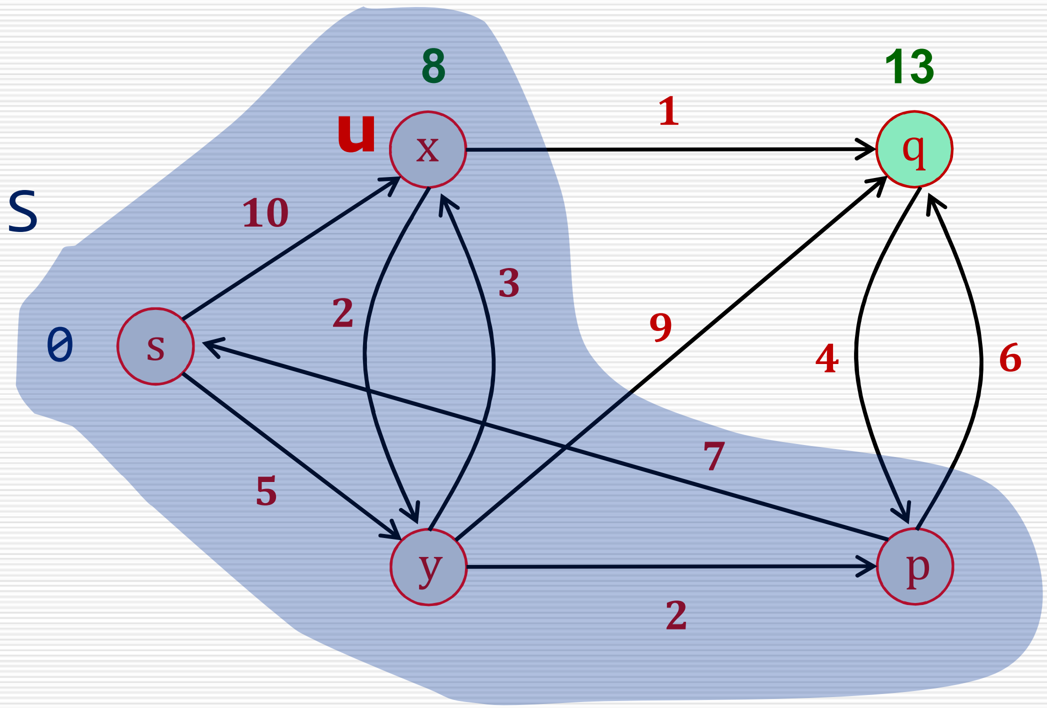
d(s)=0 d(x)=8 d(y)=5 d(p)=7 d(q)=13

prev(s)=NIL prev(x)=y prev(y)=s prev(p)=y prev(q)=y

3ο παράδειγμα Dijkstra

Q = {q}

- Παράδειγμα



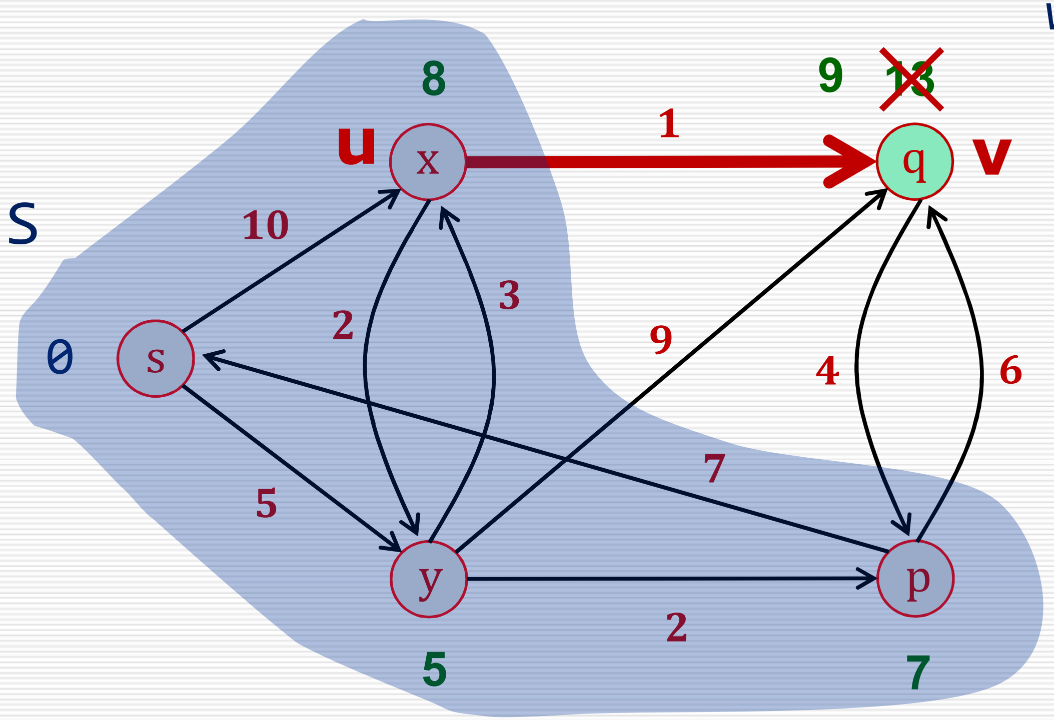

```
while Q ≠ ∅ :
    u ← extract-min(Q);
    S ← S ∪ {u}
```

d(s)=0	d(x)=8	d(y)=5	d(p)=7	d(q)=13
prev(s)=NIL	prev(x)=y	prev(y)=s	prev(p)=y	prev(q)=y

3ο παράδειγμα Dijkstra

Q = {q}

- Παράδειγμα




```
while Q ≠ ∅ :
   u ← extract-min(Q);
   S ←S ∪ {u}
   για κάθε κόμβο v∈N(u)
      Update(u,v,c)
```

d(s)=0	d(x)=8	d(y)=5	d(p)=7	d(q)=9
prev(s)=NIL	prev(x)=y	prev(y)=s	prev(p)=y	prev(q)=x

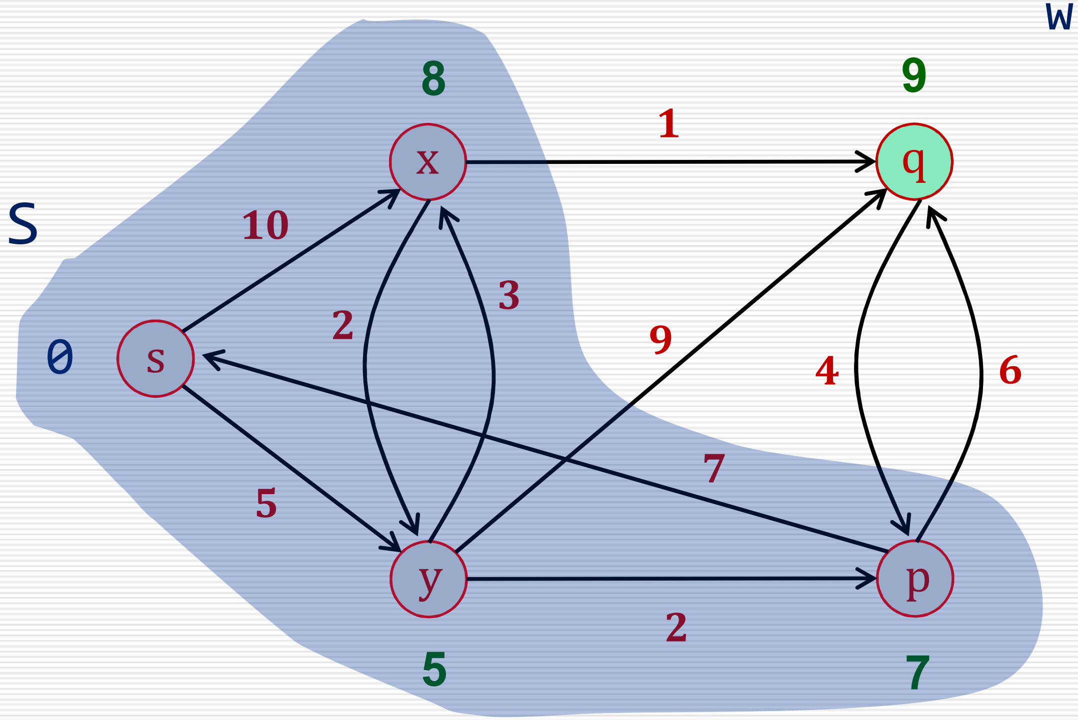
3ο παράδειγμα Dijkstra

Q = {q}

- Παράδειγμα

while Q ≠ ∅ :

u ← extract-min(Q);



d(s)=0 d(x)=8 d(y)=5 d(p)=7 d(q)=9

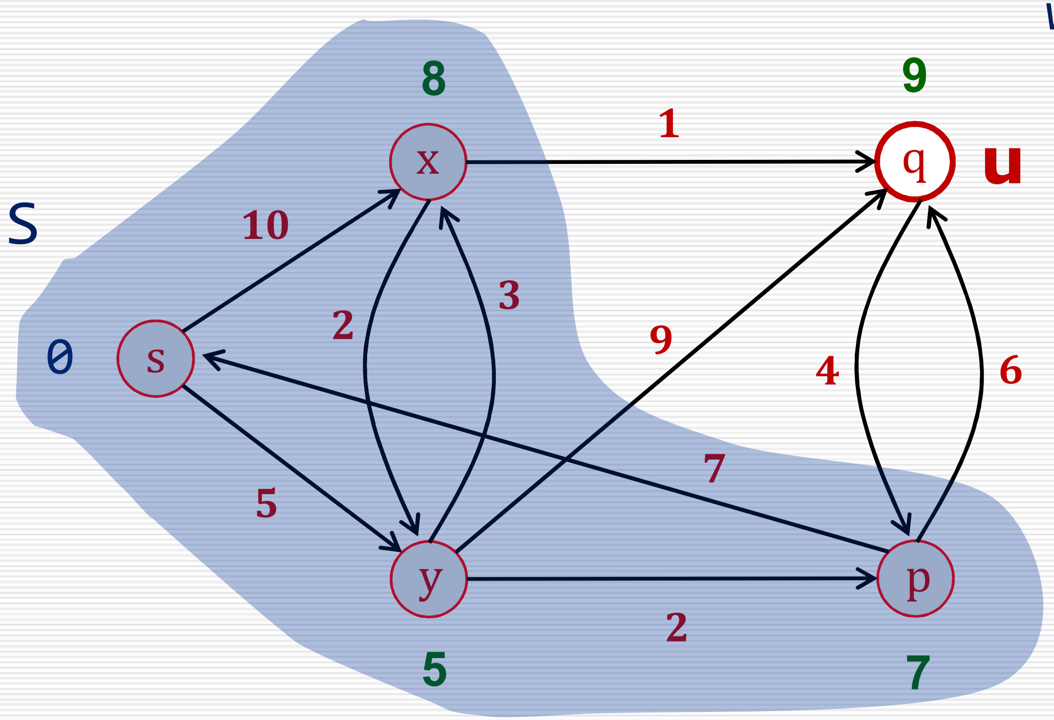
prev(s)=NIL prev(x)=y prev(y)=s prev(p)=y prev(q)=x

3ο παράδειγμα Dijkstra

$Q = \{\}$

- Παράδειγμα

```
while Q ≠ ∅ :
    u ← extract-min(Q);
```



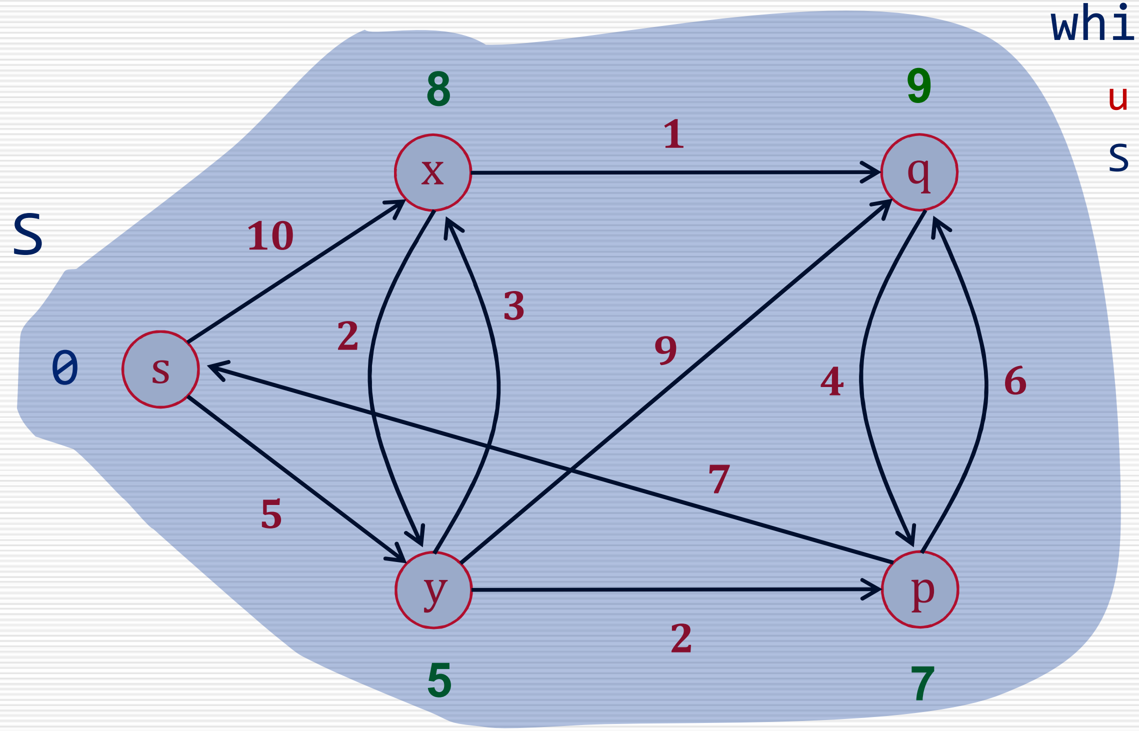
d(s)=0 d(x)=8 d(y)=5 d(p)=7 d(q)=9

prev(s)=NIL prev(x)=y prev(y)=s prev(p)=y prev(q)=x

3ο παράδειγμα Dijkstra

$Q = \{\}$

- Παράδειγμα



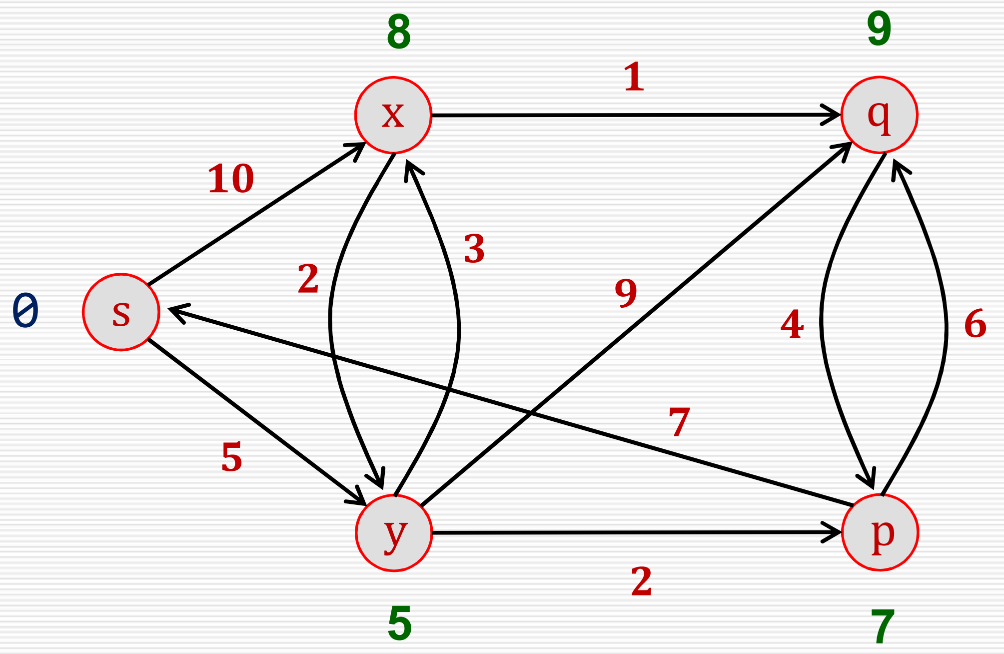

```
while Q ≠ ∅ :
   u ← extract-min(Q);
   S ← S ∪ {u}
```

d(s)=0	d(x)=8	d(y)=5	d(p)=7	d(q)=9
prev(s)=NIL	prev(x)=y	prev(y)=s	prev(p)=y	prev(q)=x

3ο παράδειγμα Dijkstra

Q = {}

- Παράδειγμα



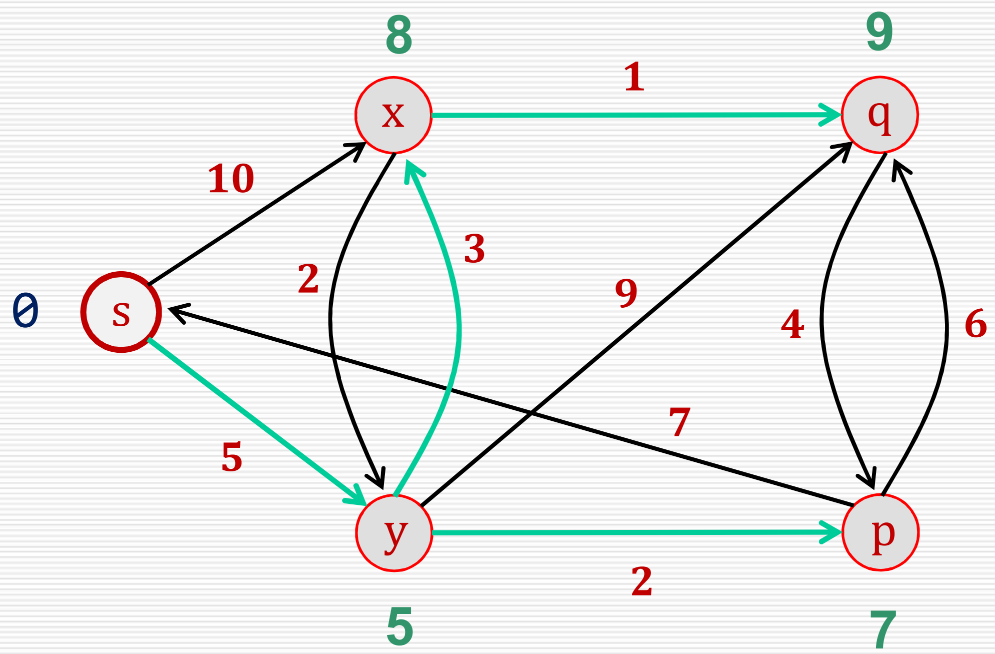
return d(), prev()

d(s)=0	d(x)=8	d(y)=5	d(p)=7	d(q)=9
prev(s)=NIL	prev(x)=y	prev(y)=s	prev(p)=y	prev(q)=x

3ο παράδειγμα Dijkstra

- Παράδειγμα

Έξοδος Αλγόριθμου



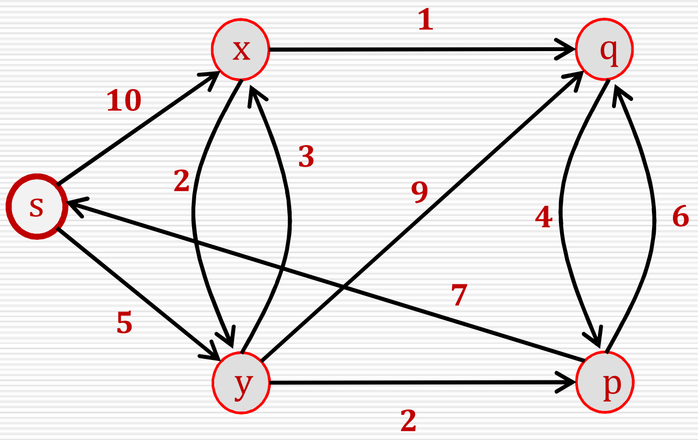
Κόστος συντομότερων διαδρομών

Προηγούμενοι κόμβοι στις ελάχιστες διαδρομές

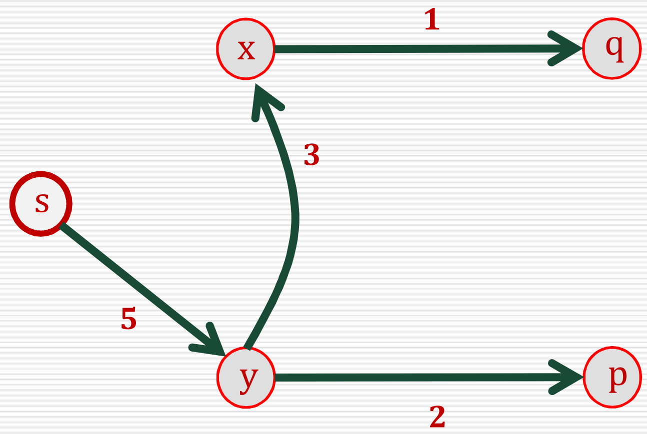
d(s)=0	d(x)=8	d(y)=5	d(p)=7	d(q)=9
prev(s)=NIL	prev(x)=y	prev(y)=s	prev(p)=y	prev(q)=x

3ο παράδειγμα Dijkstra

- Παράδειγμα

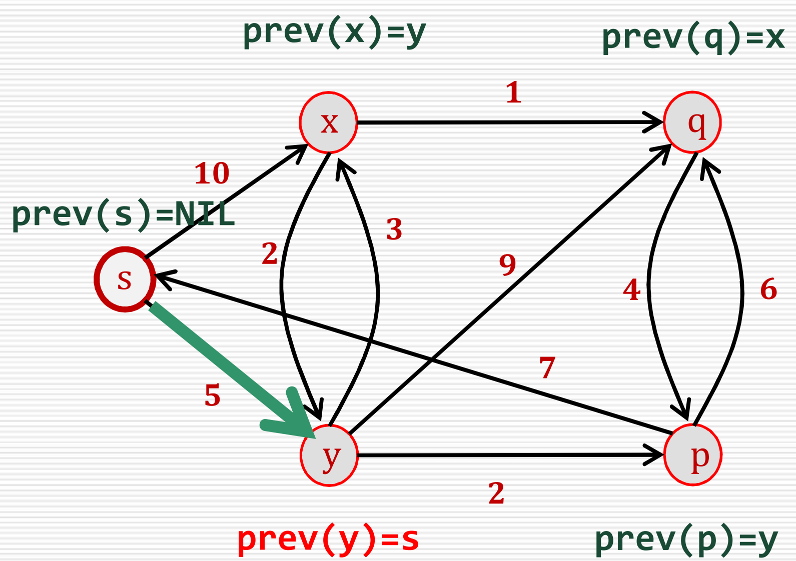


Δένδρο Ελάχιστων Διαδρομών



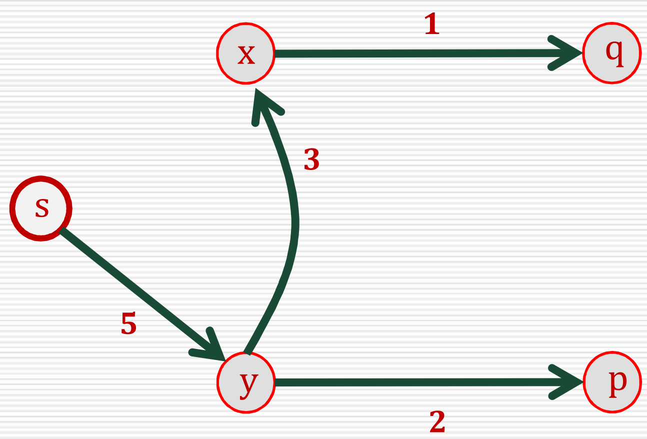
3ο παράδειγμα Dijkstra

Παράδειγμα



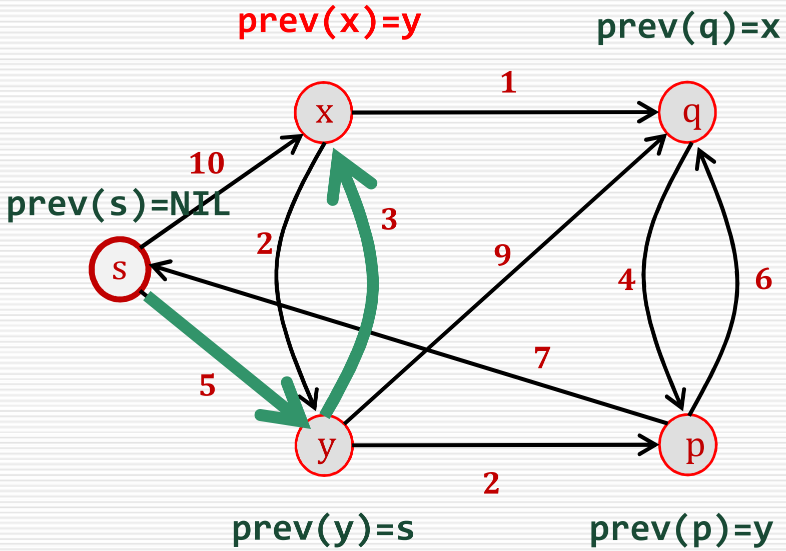
Δένδρο Ελάχιστων Διαδρομών

ΚΑΤΑΣΚΕΥΗ



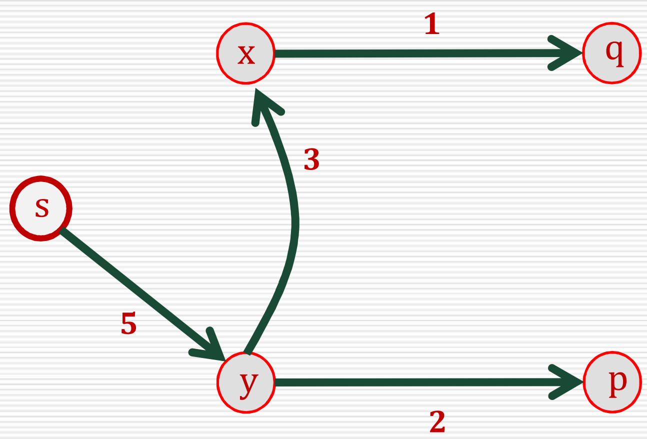
3ο παράδειγμα Dijkstra

- Παράδειγμα



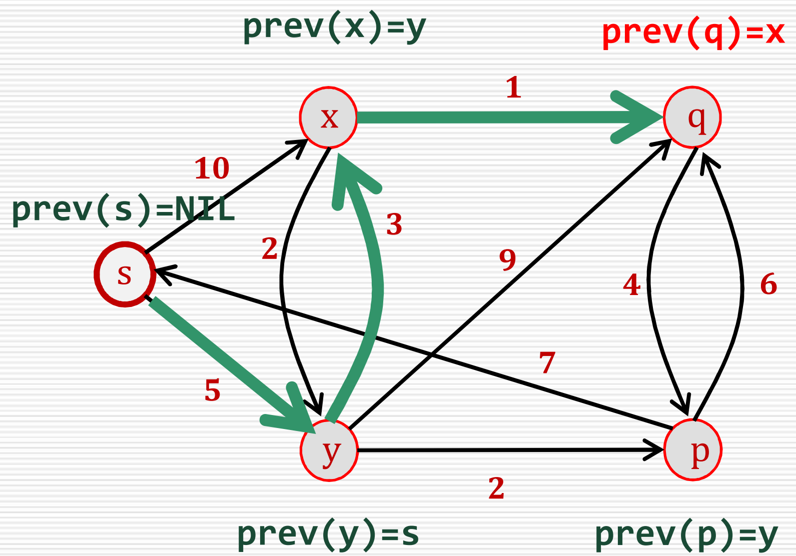
Δένδρο Ελάχιστων Διαδρομών

ΚΑΤΑΣΚΕΥΗ



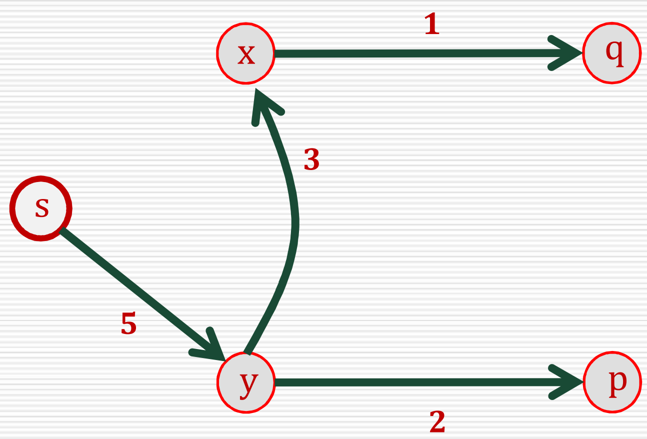
3ο παράδειγμα Dijkstra

Παράδειγμα



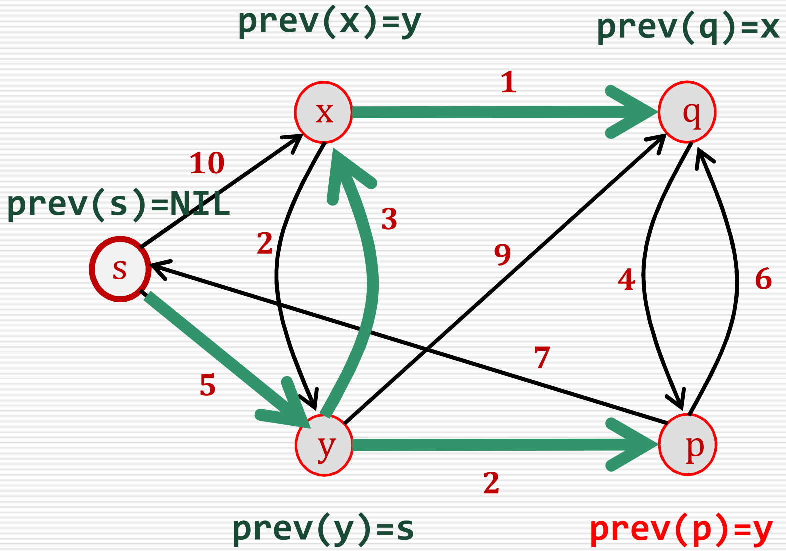
Δένδρο Ελάχιστων Διαδρομών

ΚΑΤΑΣΚΕΥΗ



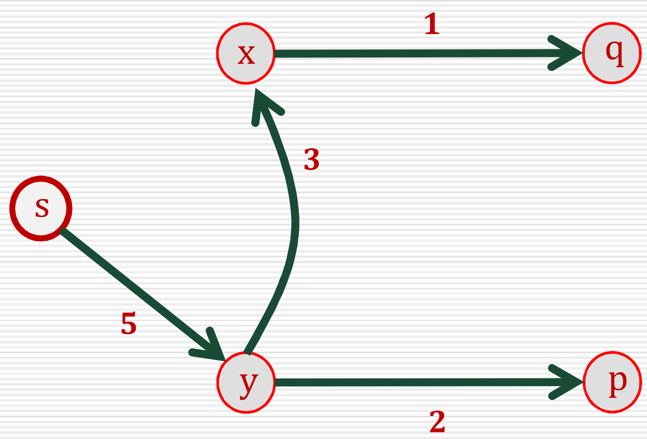
3ο παράδειγμα Dijkstra

- Παράδειγμα



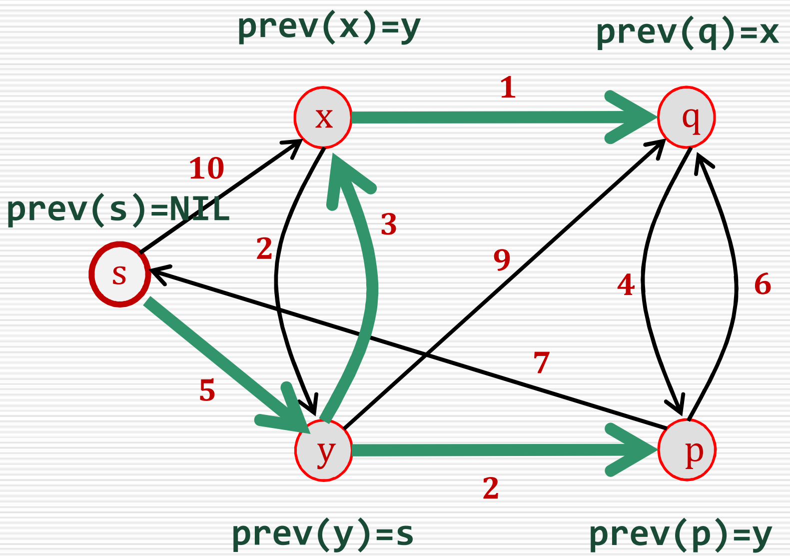
Δένδρο Ελάχιστων Διαδρομών

ΚΑΤΑΣΚΕΥΗ

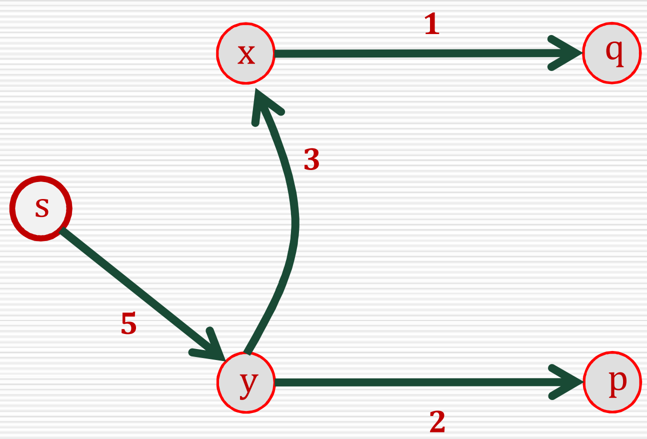


3ο παράδειγμα Dijkstra

Παράδειγμα

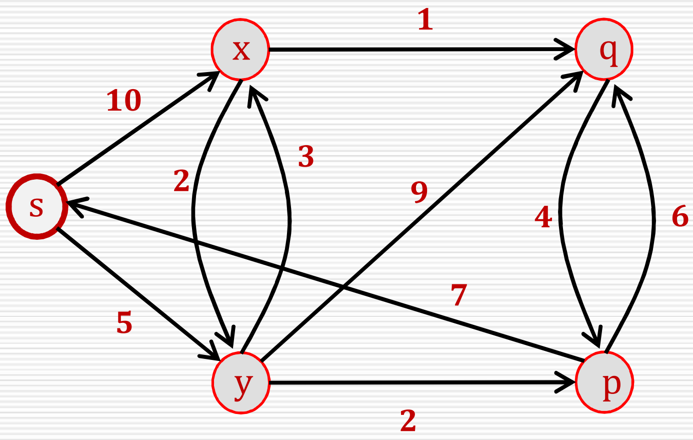


Δένδρο Ελάχιστων Διαδρομών
ΚΑΤΑΣΚΕΥΗ



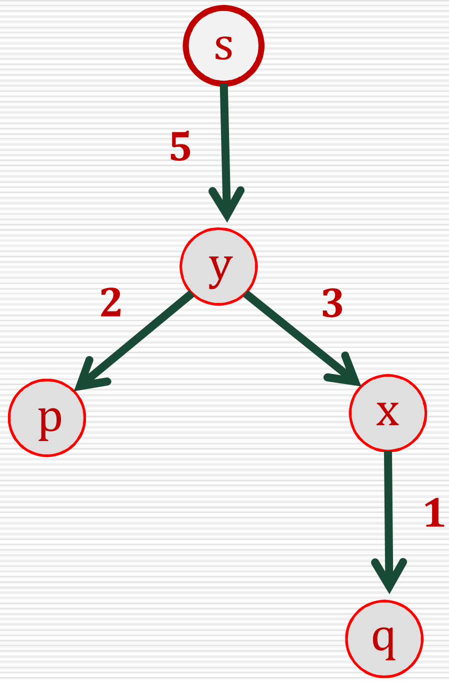
3ο παράδειγμα Dijkstra

Παράδειγμα



Δένδρο Ελάχιστων Διαδρομών

ΚΑΤΑΣΚΕΥΗ



Πολυπλοκότητα Dijkstra

- Ομοιότητες με αλγόριθμο BFS;
- Πιο αργός από BFS (ουρά προτεραιότητας vs απλή ουρά)
- Απαιτεί **n=|V|** λειτουργίες insert στην ουρά, και **m=|E|** λειτουργίες update:

  |V| insert / extract-min

  |E| update
- Υλοποίηση ουράς με πίνακα $\Rightarrow$ **$O(|V|^2)$**
- Υλοποίηση ουράς με δυαδικό σωρό $\Rightarrow$ **$O(|E| \log|V|)$**
- Υλοποίηση ουράς με σωρό Fibonacci $\Rightarrow$ **$O(|E|+|V|\log|V|)$**

Ορθότητα αλγορίθμου Dijkstra

- Ο αλγόριθμος δημιουργεί σταδιακά ένα δένδρο συντομότερων διαδρομών. Το δένδρο αρχικοποιείται με τον αρχικό κόμβο *s*.
- Σε κάθε επανάληψη επιλέγεται ο κόμβος ***w*** με την ελάχιστη προσωρινή ετικέτα (τρέχουσα απόσταση) από τον *s*.
- Η απόδειξη στηρίζεται σε δύο αναλλοίωτες συνθήκες βρόχου.

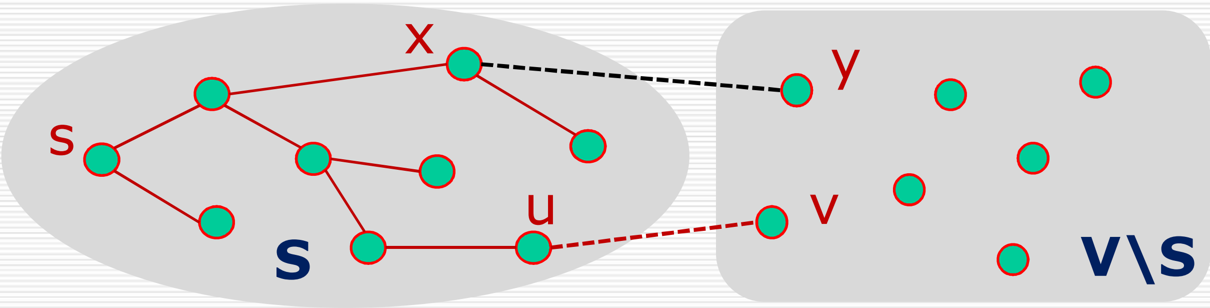
Ορθότητα αλγορίθμου Dijkstra

- 1η αναλλοίωτη βρόχου: *μετά από κάθε επανάληψη του εξωτερικού βρόχου, η ετικέτα D(u) κάθε κόμβου u του S ισούται με το κόστος της συντομότερης διαδρομής από τον s προς τον u, και η ετικέτα D(v) κάθε κόμβου v του V\S ισούται με το κόστος της συντομότερης διαδρομής από τον s στον v, μεταξύ όλων των διαδρομών που περνούν μόνο από κόμβους του συνόλου S.*
- 2η αναλλοίωτη βρόχου: *μετά από κάθε επανάληψη του εξωτερικού βρόχου, για τον κόμβο w που ανήκει στο V\S και έχει ελάχιστη (μεταξύ κόμβων του V\S) ετικέτα D(w), αυτή ισούται με το κόστος της συντομότερης διαδρομής από τον s στον w.*
- Απόδειξη: με επαγωγή.

Ορθότητα αλγορίθμου Dijkstra

Λήμμα: 1η αναλλοίωτη => 2η αναλλοίωτη

Έστω G = (V, E) γράφος με μη-αρνητικά βάρη, $S \in V$, και (S, V\S) μια διαμέριση του V τ.ώ. $s \in S$ και $\forall u \in S$ η ετικέτα του u ισούται με την ελάχιστη απόσταση d(s, u).



Έστω $v \in V \backslash S$ ο κόμβος με την ελάχιστη ετικέτα στο V\S. Τότε υπάρχει ακμή (u, v) που ελαχιστοποιεί την ποσότητα

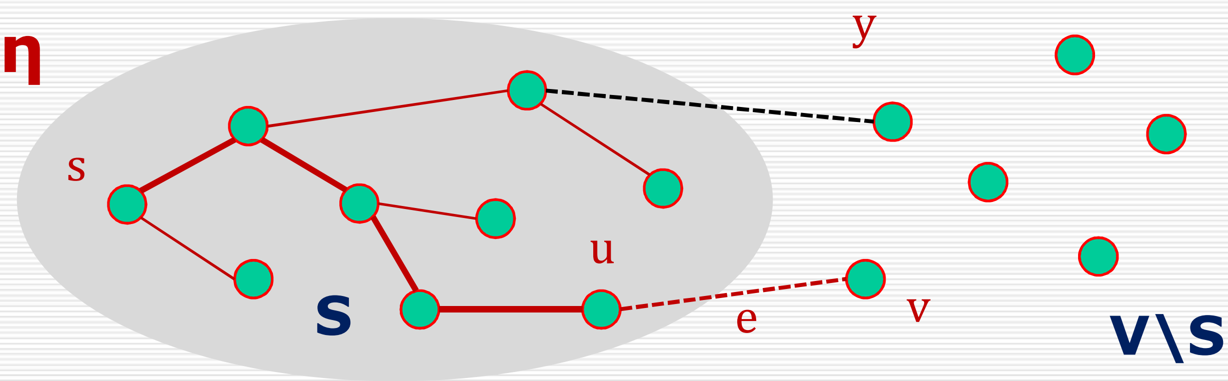
$$\mathbf{d(s, u) + c(u, v)}$$

μεταξύ όλων των ακμών (x, y) με $x \in S$ και $y \in V \backslash S$. Θ.δ.ό.

P = (s, …, u, v) είναι σ.δ. από s σε v.

Ορθότητα αλγορίθμου Dijkstra

Απόδειξη



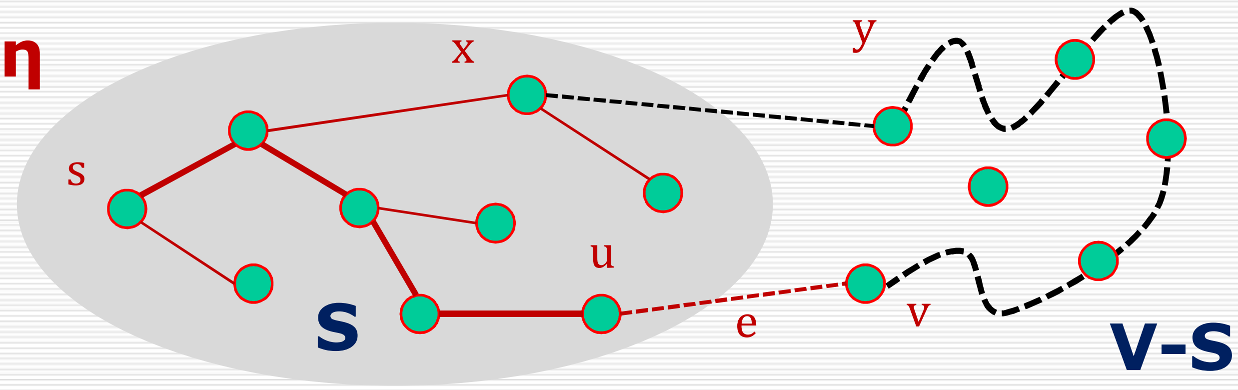
Έστω e = (u, v) και έστω (s, …, u) η ελάχιστη διαδρομή από τον s στον u.

Για την διαδρομή P = (s, …, u, v) από τον s στον v ισχύει :

$$c(P) = d(s,u) + c(u,v) \qquad (1)$$

Ορθότητα αλγορίθμου Dijkstra

Απόδειξη



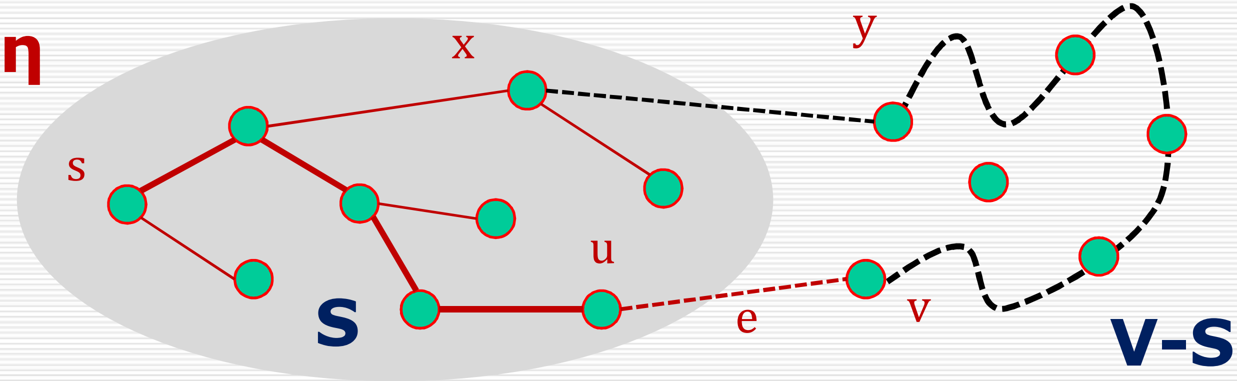
Έστω Q = (s, …, x, y, …, v) μια ελάχιστη διαδρομή από τον s στον v, και

έστω y ο πρώτος κόμβος της διαδρομής Q : $y \in V \setminus S$

Θα δείξουμε ότι **c(P)** $\leq$ **c(Q)**

Ορθότητα αλγορίθμου Dijkstra

Απόδειξη

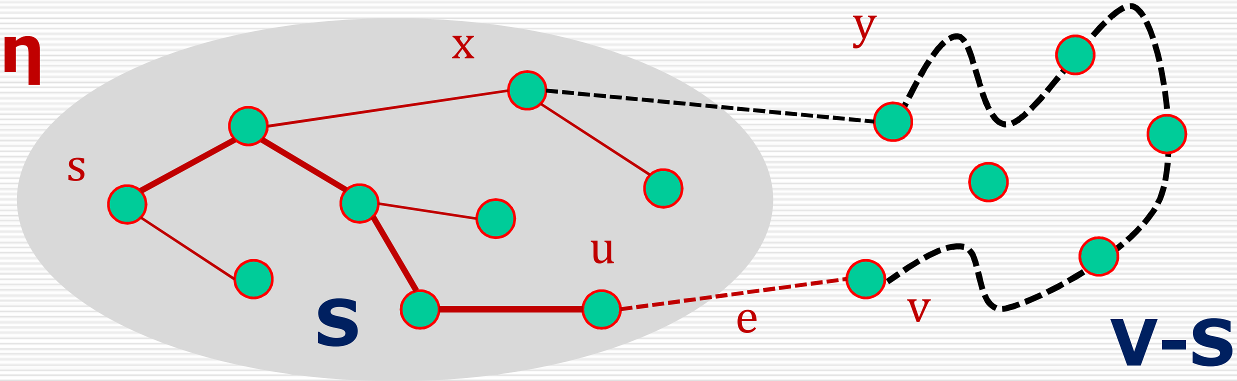


Από (1) και από επιλογή της ακμής e = (u, v), έχουμε:

c(P) = d(s,u) + c(u,v) ≤ d(s, x) + c(x, y) ≤ **c(Q)**

Ορθότητα αλγορίθμου Dijkstra

Απόδειξη



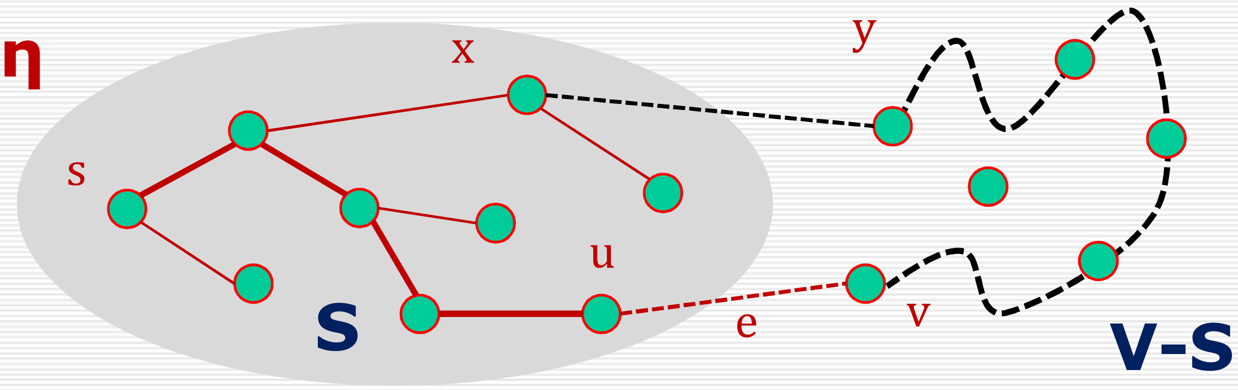
Από (1) και από επιλογή της ακμής e = (u, v), έχουμε:

$$\mathbf{c(P)} = d(s,u) + c(u,v) \leq d(s, x) + c(x, y) \leq \mathbf{c(Q)}$$

Ερώτηση: πότε δεν ισχύει το παραπάνω;

Ορθότητα αλγορίθμου Dijkstra

Απόδειξη



Από (1) και από επιλογή της ακμής e = (u, v), έχουμε:

$$\mathbf{c(P)} = d(s,u) + c(u,v) \leq d(s,x) + c(x,y) \leq \mathbf{c(Q)}$$

Ερώτηση: πότε δεν ισχύει το παραπάνω;

Άσκηση: συμπληρώστε την απόδειξη

Ορθότητα αλγορίθμου Dijkstra

- Η ορθότητα ισχύει **μόνο** σε γράφους **χωρίς αρνητικά βάρη**

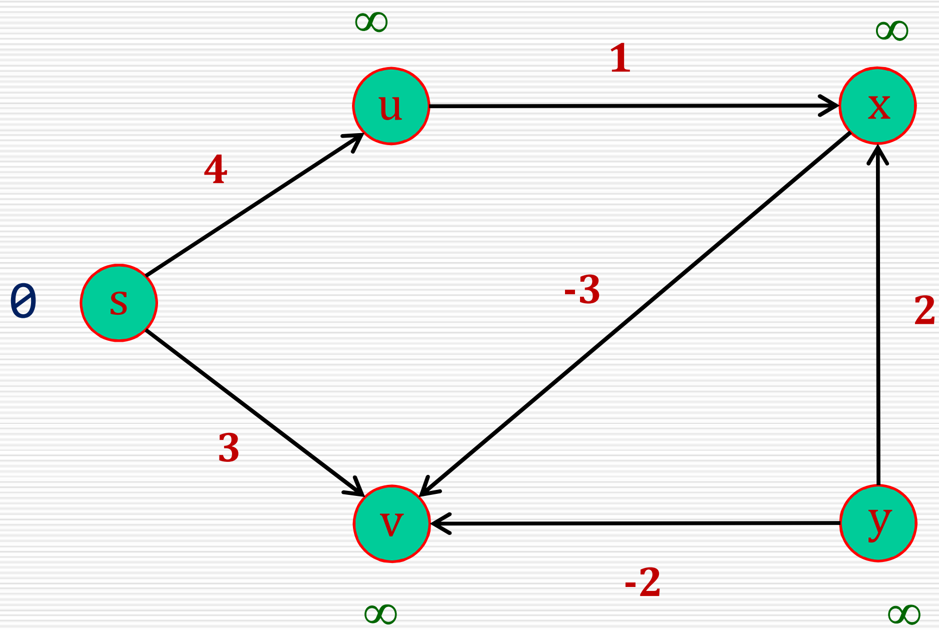
- *Άσκηση*: βρείτε αντιπαράδειγμα.

Αλγόριθμος Bellman-Ford

```
dist(s) := 0 ;
for each v<>s do dist(v) := ∞

repeat n-1 times
   for each edge e = (u,v) do
      if dist(u) + cost(u,v) < dist(v) then
         dist(v) := dist(u) + cost(u,v)
         prev(v) := u
```

Αλγόριθμος Bellman-Ford

Παραδείγματα



n = 5
m = 6

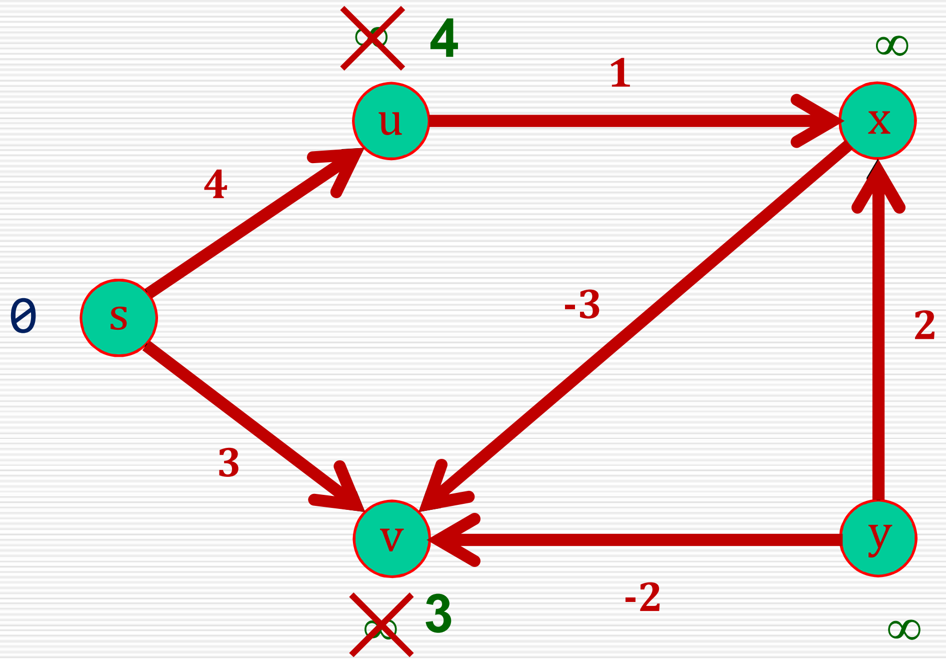
Update στις
6 ακμές
4 φορές

V = {s, u, v, x, y}

E = {(y,x),(u,x),(y,v),(s,u),(x,v),(s,v)}

Αλγόριθμος Bellman-Ford

Παράδειγμα 1



n = 5
m = 6

Update στις
6 ακμές
4 φορές

1η Επανάληψη

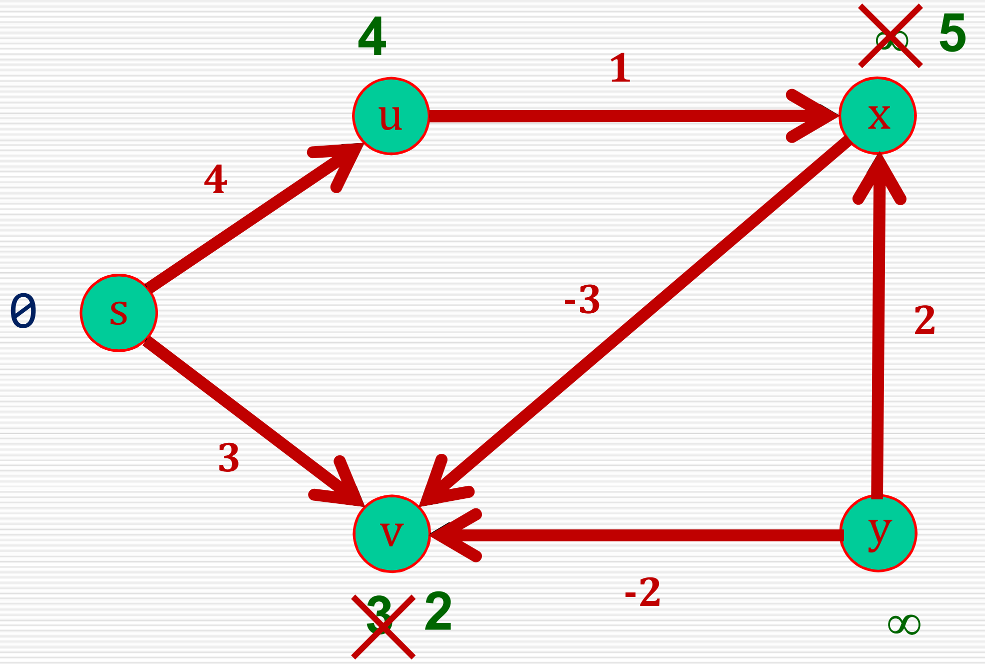
S1: (y,x), (u,x), (y,v), (s,u), (x,v), (s,v)
✓ ✓ ✓ ✓ ✓ ✓

Αλγόριθμος Bellman-Ford

Παράδειγμα 1

n = 5
m = 6

Update στις
6 ακμές
4 φορές



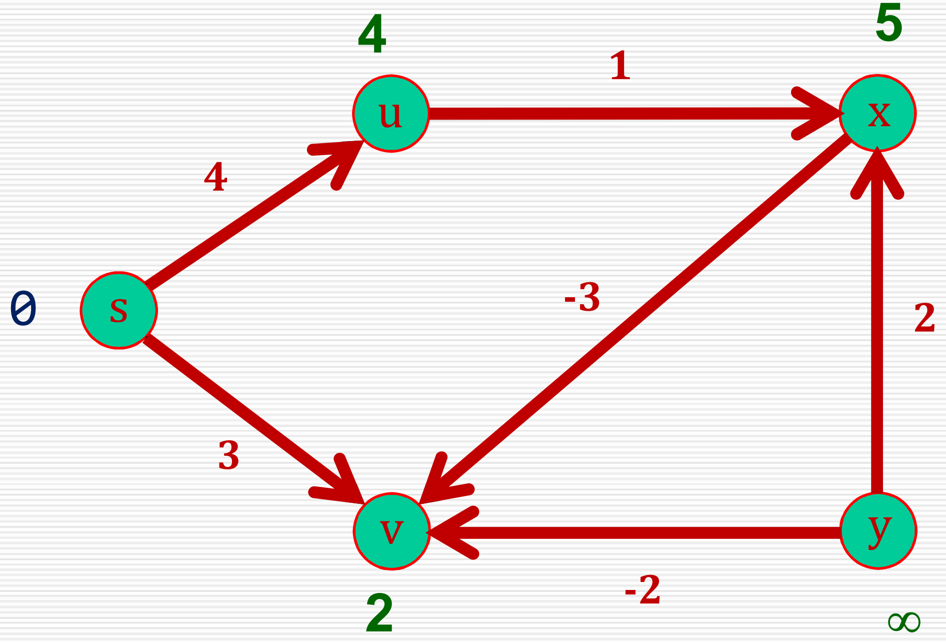
2η Επανάληψη

S1: (y,x), (u,x), (y,v), (s,u), (x,v), (s,v)

✓ ✓ ✓ ✓ ✓ ✓

Αλγόριθμος Bellman-Ford

Παράδειγμα 1



n = 5
m = 6

Update στις
6 ακμές
4 φορές

3η Επανάληψη

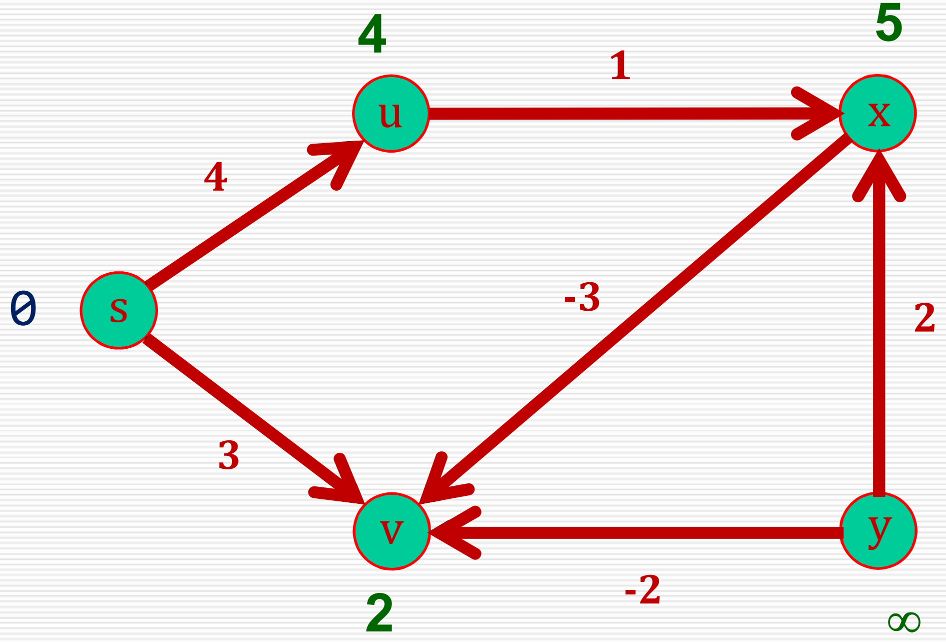
Καμία Αλλαγή
Γιατί ?

S1: (y,x), (u,x), (y,v), (s,u), (x,v), (s,v)

✓ ✓ ✓ ✓ ✓ ✓

Αλγόριθμος Bellman-Ford

Παράδειγμα 1



n = 5
m = 6

Update στις
6 ακμές
4 φορές

4η Επανάληψη

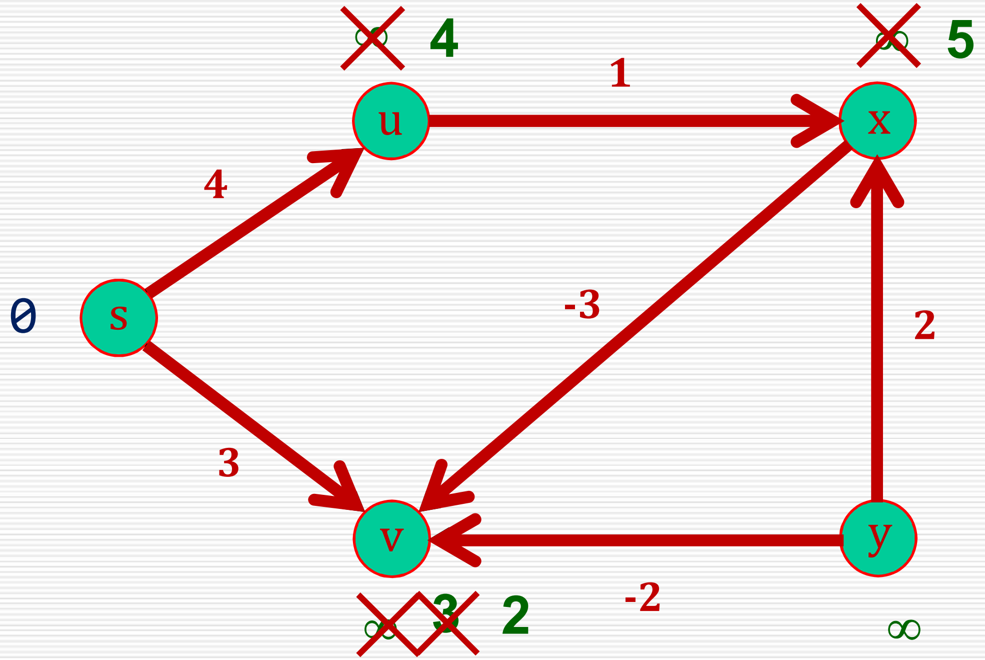
Καμία Αλλαγή

S1: (y,x), (u,x), (y,v), (s,u), (x,v), (s,v)

✓ ✓ ✓ ✓ ✓ ✓

Αλγόριθμος Bellman-Ford

Παράδειγμα 2



n = 5
m = 6

Update στις
6 ακμές
4 φορές

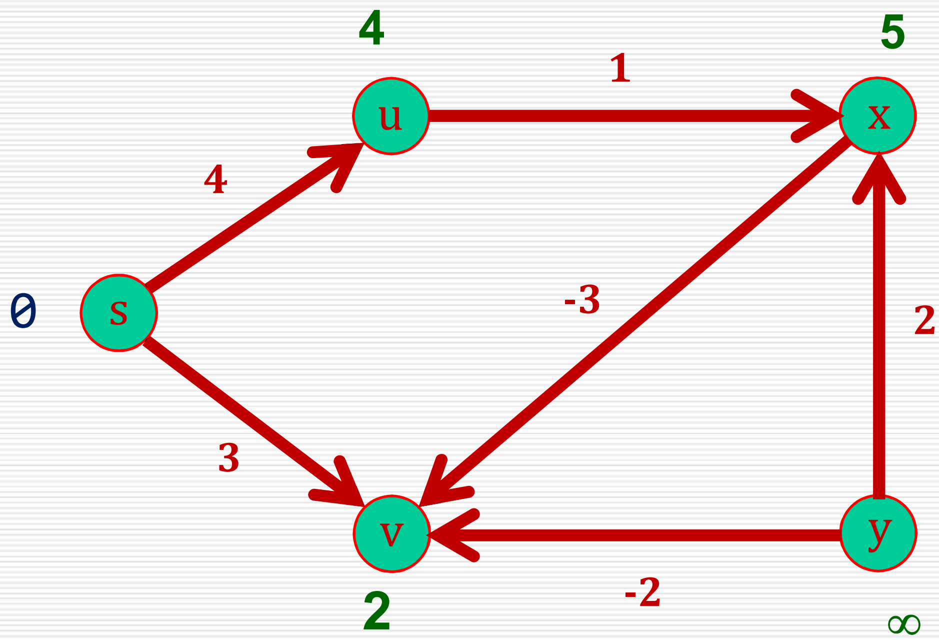
1η Επανάληψη

S2: (s,u), (u,x), (y,x), (s,v), (x,v), (y,v)

✓ ✓ ✓ ✓ ✓ ✓

Αλγόριθμος Bellman-Ford

Παράδειγμα 2



n = 5
m = 6

Update στις
6 ακμές
4 φορές

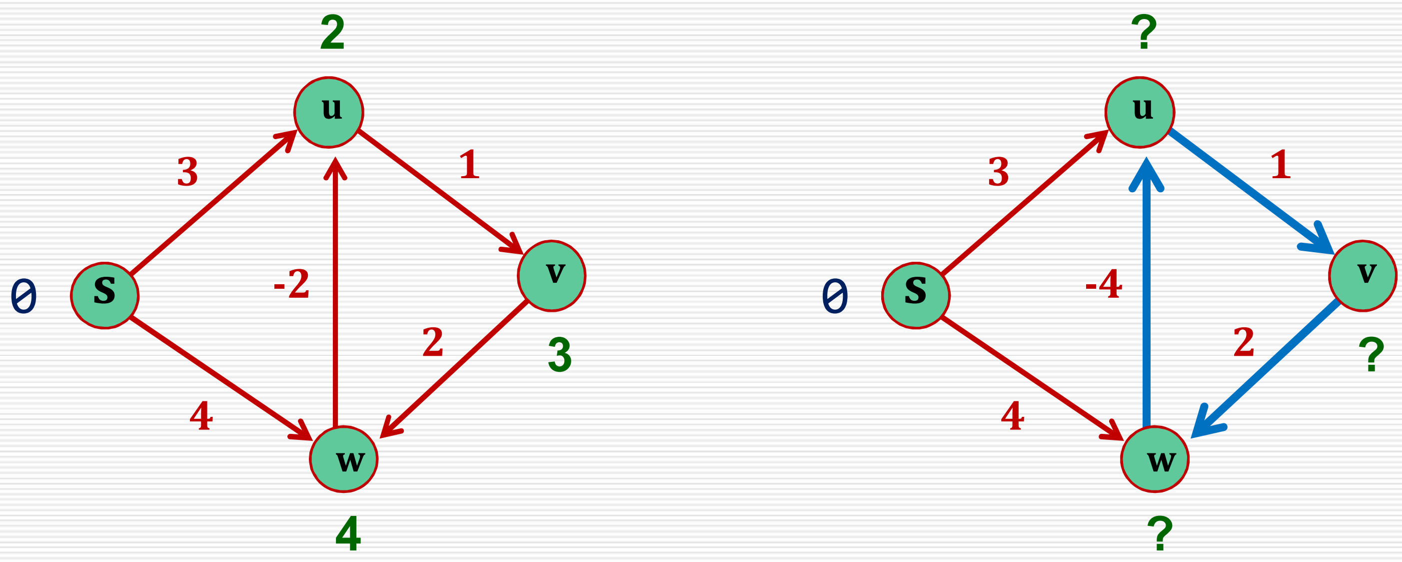
2η Επανάληψη

Καμία Αλλαγή
Γιατί ?

S2: (s,u) ✓, (u,x) ✓, (y,x) ✓, (s,v) ✓, (x,v) ✓, (y,v) ✓

Αλγόριθμος Bellman-Ford

- **Αρνητικοί Κύκλοι**

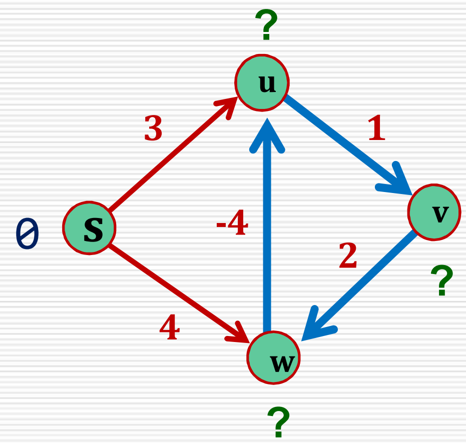


Εάν υπάρχει αρνητικός κύκλος δεν έχει νόημα να αναζητούμε ελάχιστες διαδρομές !!!

Αλγόριθμος Bellman-Ford

Εντοπισμός αρνητικών κύκλων

```
Bellman-Ford-detection(G,w,s)

1.  initialize(G,s)
2.  επανάλαβε n-1 φορές:
        για κάθε ακμή (u,v)∈E:
            Update(u,v,c)
3.  για κάθε ακμή (u,v)∈E:
        if d(v) > d(u) + c(u,v) then
            return FALSE
4.  return d(.)
```



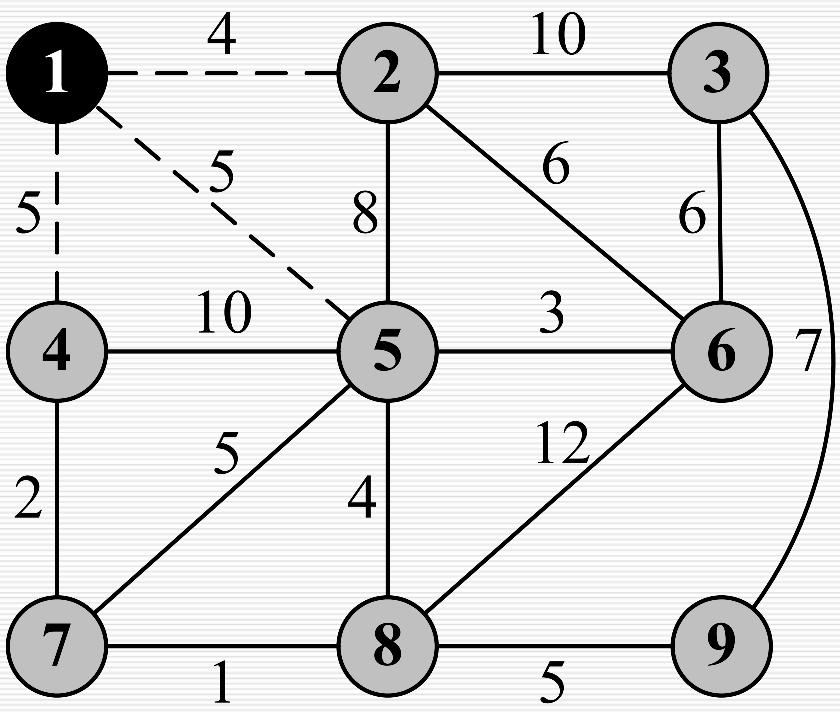
Αλγόριθμος Bellman-Ford

- **Ορθότητα:** στο τέλος της *k*-οστής επανάληψης έχουν υπολογιστεί σωστά οι συντομότερες διαδρομές που αποτελούνται από το πολύ *k* ακμές (*άσκηση*: αποδείξτε το).
- **Πολυπλοκότητα**: **O(|V||E|)**

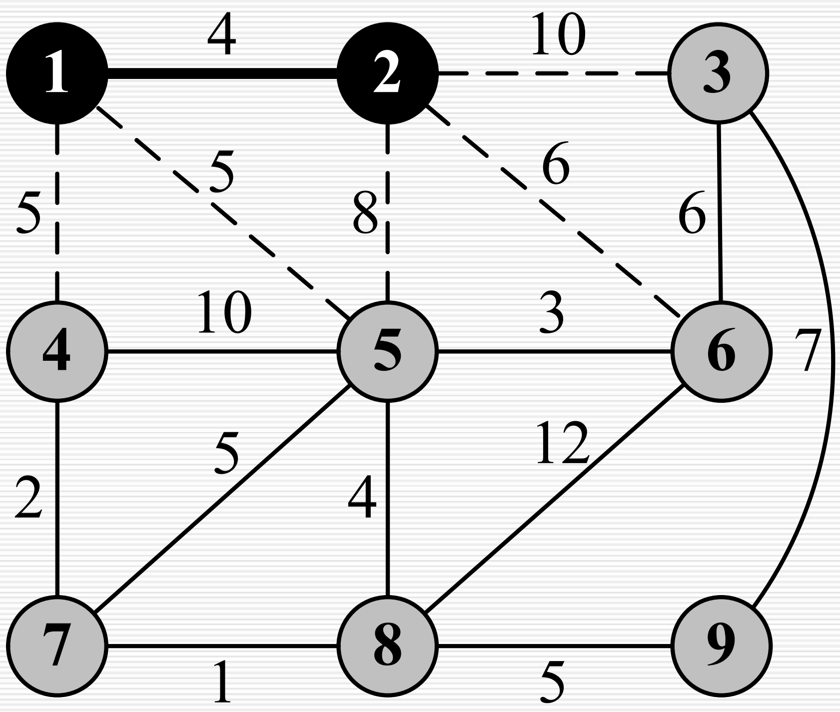
Ελάχιστο Συνδετικό Δένδρο (MST)

- **Κριτήριο Prim:** Διαλέγουμε κάθε φορά την ακμή ελαχίστου κόστους έτσι ώστε ο νέος υπογράφος να παραμένει δένδρο (έναρξη από οποιονδήποτε κόμβο)
- **Κριτήριο Kruskal:** Διαλέγουμε κάθε φορά την ακμή ελαχίστου κόστους έτσι ώστε ο νέος υπογράφος να μην έχει κύκλους

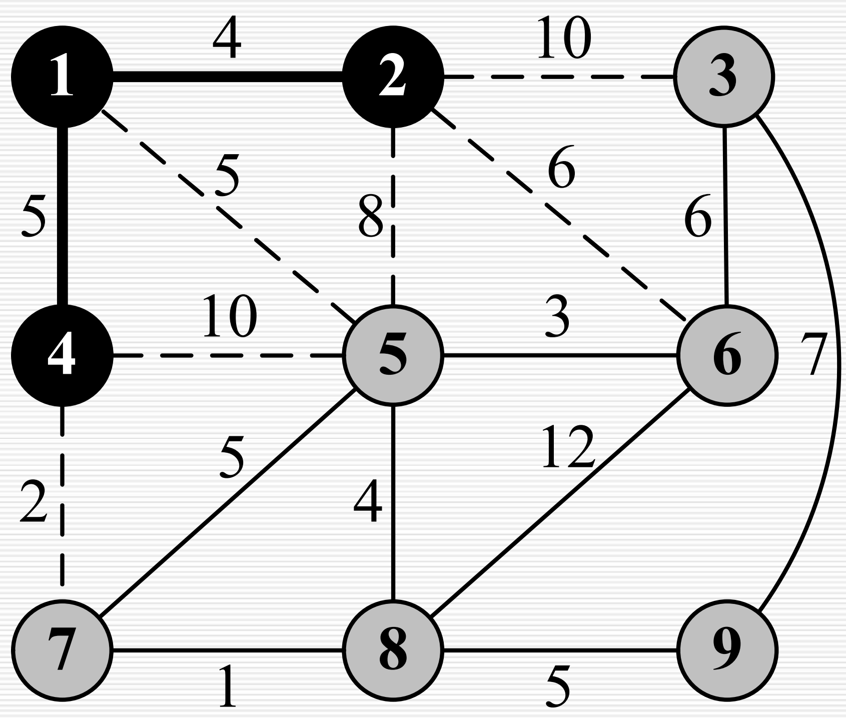
Αλγόριθμος Prim: παράδειγμα



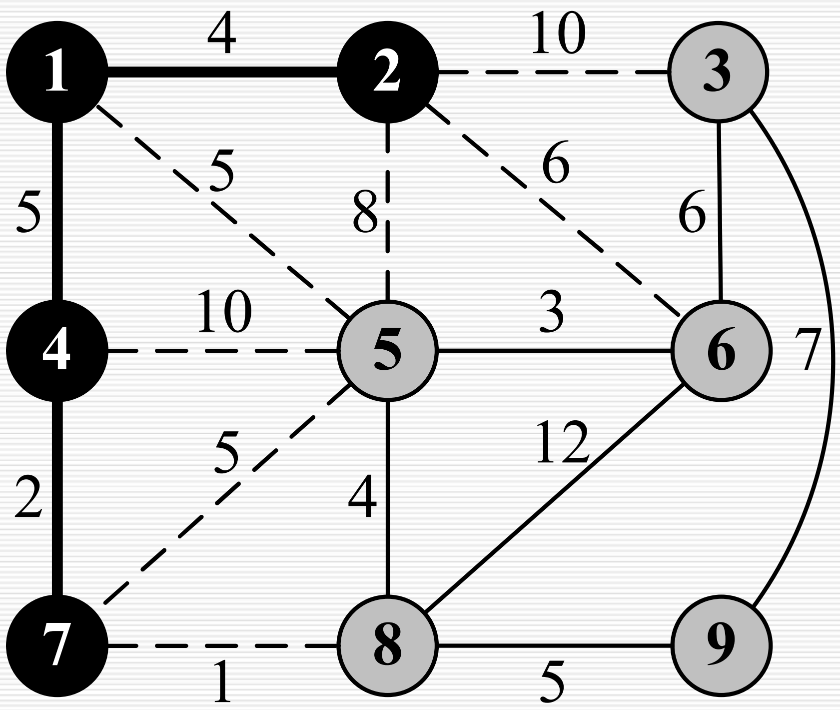
Αλγόριθμος Prim: παράδειγμα



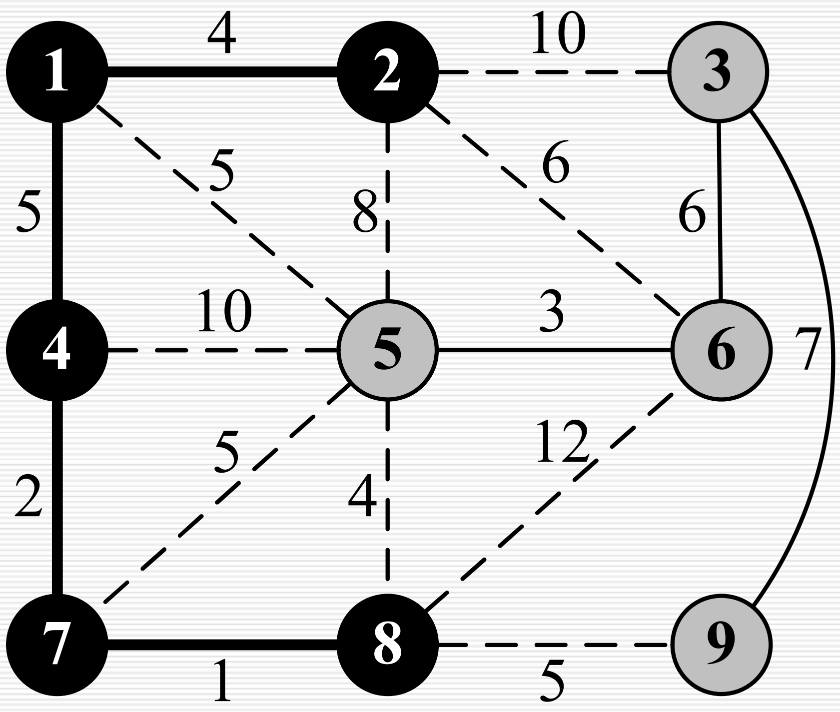
Αλγόριθμος Prim: παράδειγμα



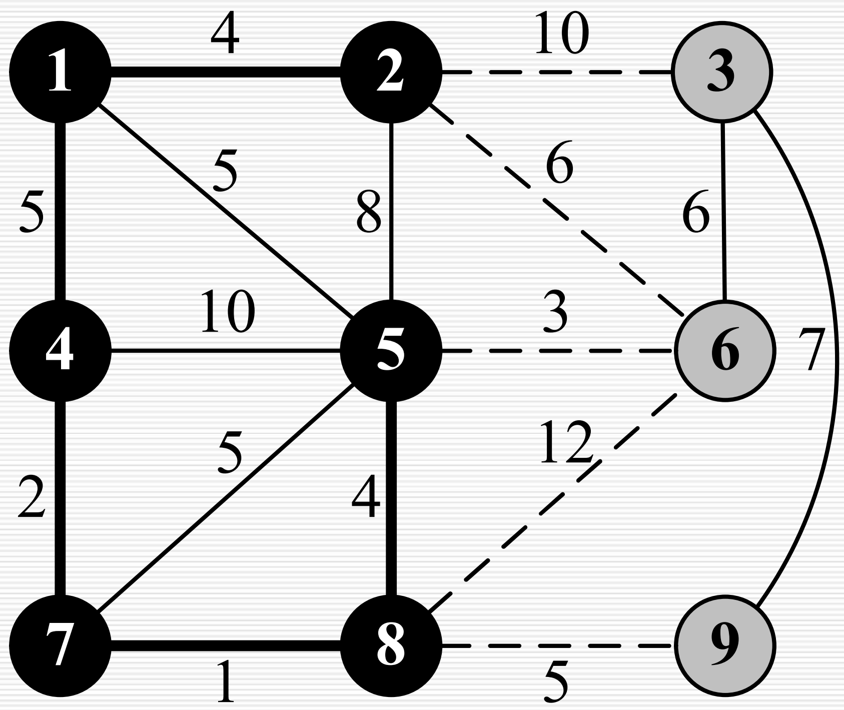
Αλγόριθμος Prim: παράδειγμα



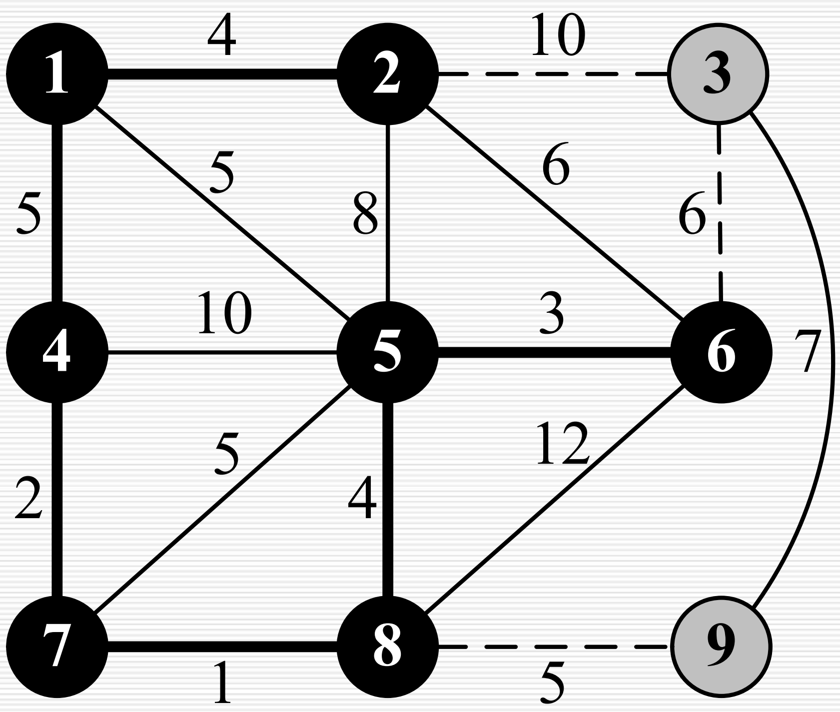
Αλγόριθμος Prim: παράδειγμα



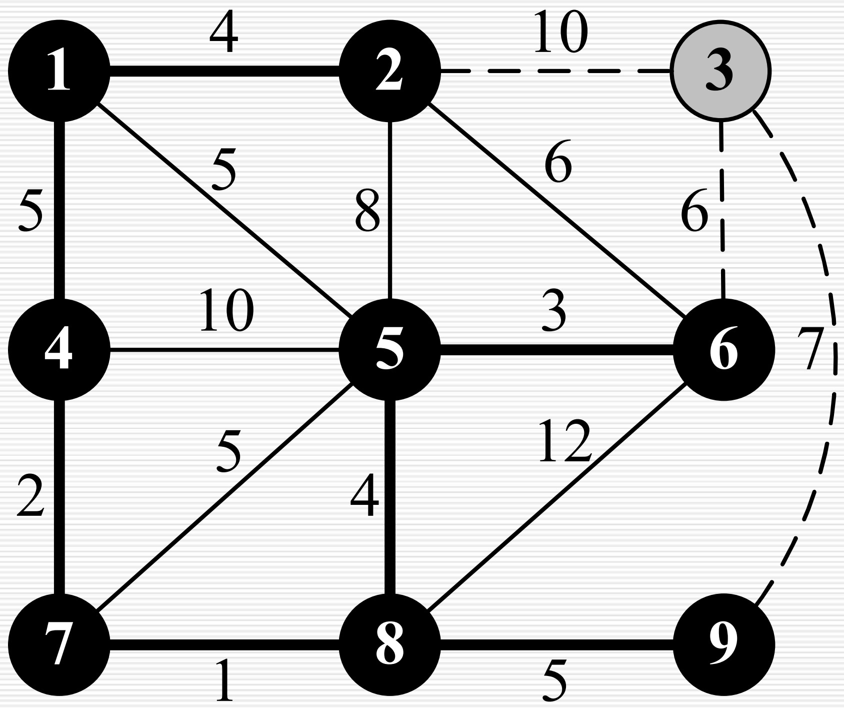
Αλγόριθμος Prim: παράδειγμα



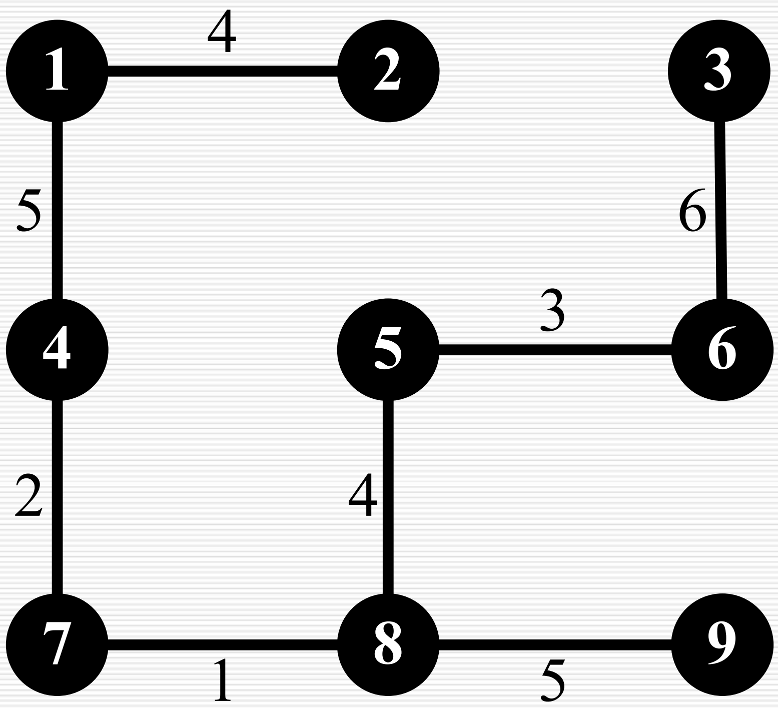
Αλγόριθμος Prim: παράδειγμα



Αλγόριθμος Prim: παράδειγμα



Αλγόριθμος Prim: παράδειγμα



Αλγόριθμος Prim

- Επιλέγεται αρχικός κόμβος, έστω s. Αρχικοποίηση:

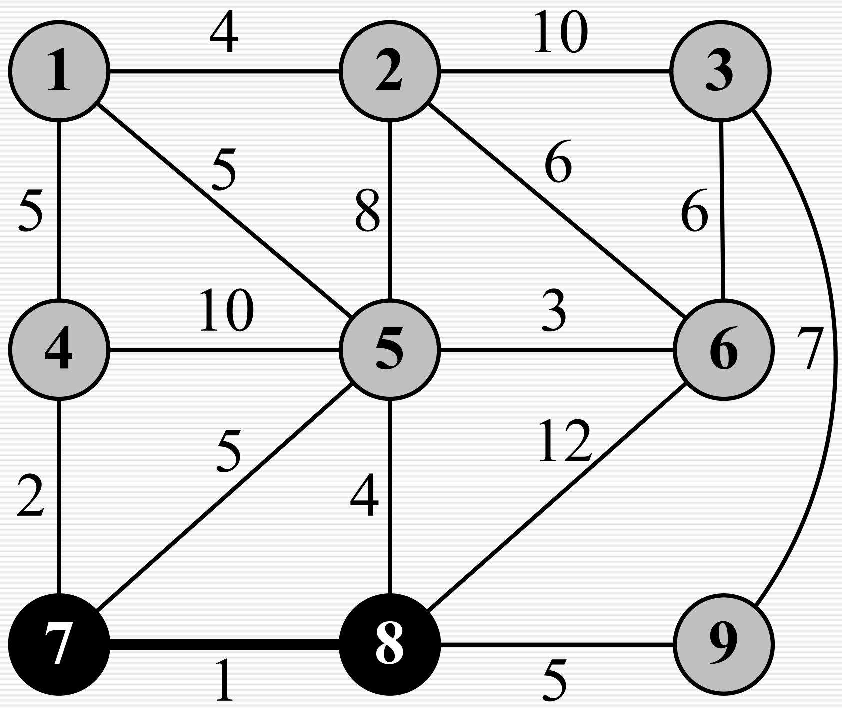
```
dist(s):= 0; for each v<>s do dist(v) := ∞
```

- Κάθε φορά επιλέγεται ο κόμβος, έστω w, με την ελάχιστη απόσταση από το μέχρι στιγμής κατασκευασμένο δένδρο, και προστίθεται στο δένδρο. Ενημερώνονται οι αποστάσεις των γειτόνων του w από το δένδρο με βάση το κόστος των ακμών (w,u_i):

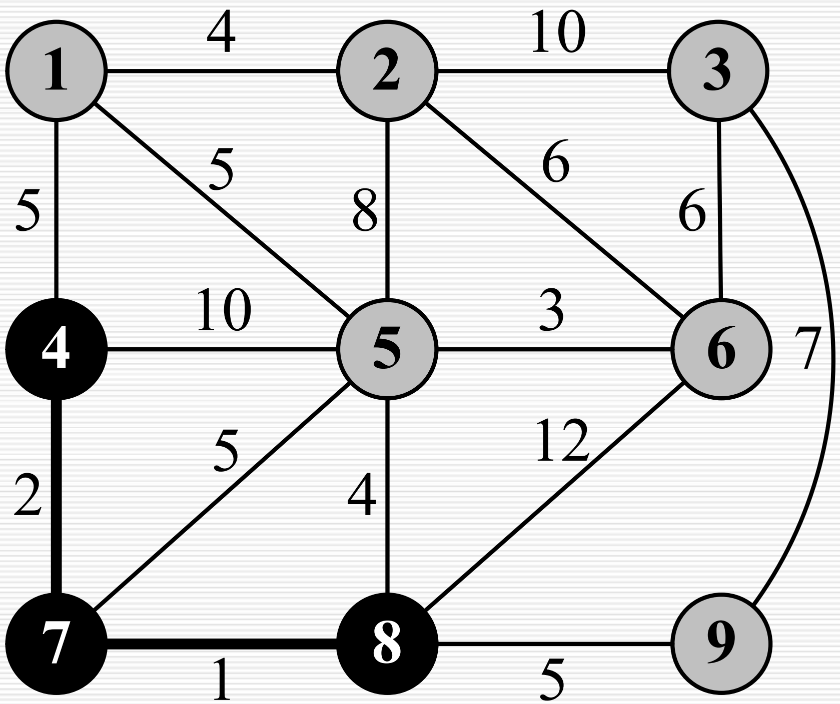
```
if cost(w,u_i) < dist(u_i) then
    dist(u_i) := cost(w,u_i)
```

- Μεγάλη ομοιότητα με Dijkstra (πού διαφέρουν;)
- **Πολυπλοκότητα**: **$O(|V|^2)$, $O(|E|\log|V|)$, $O(|E|+|V|\log|V|)$**

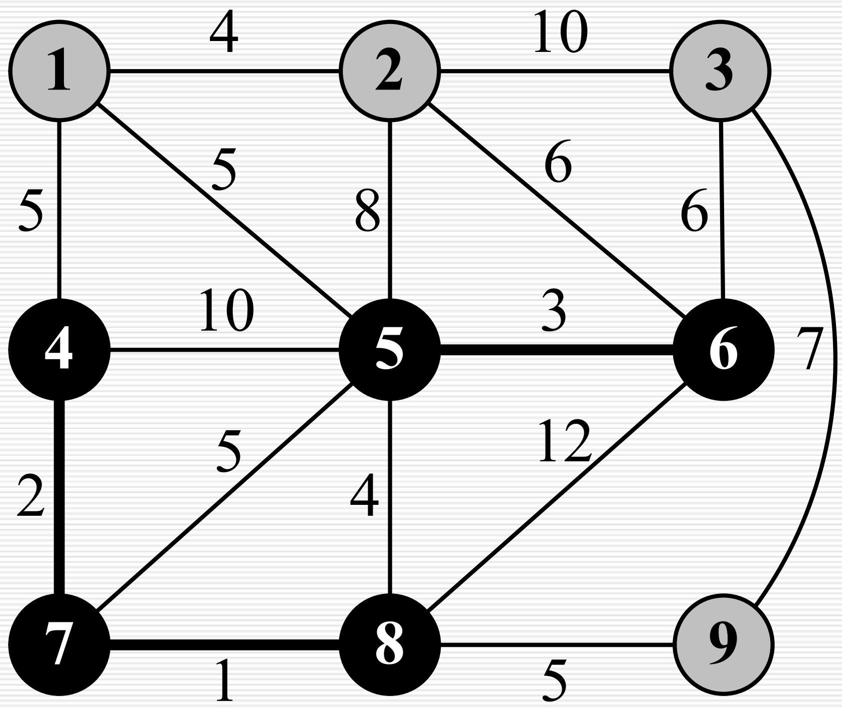
Αλγόριθμος Kruskal: παράδειγμα



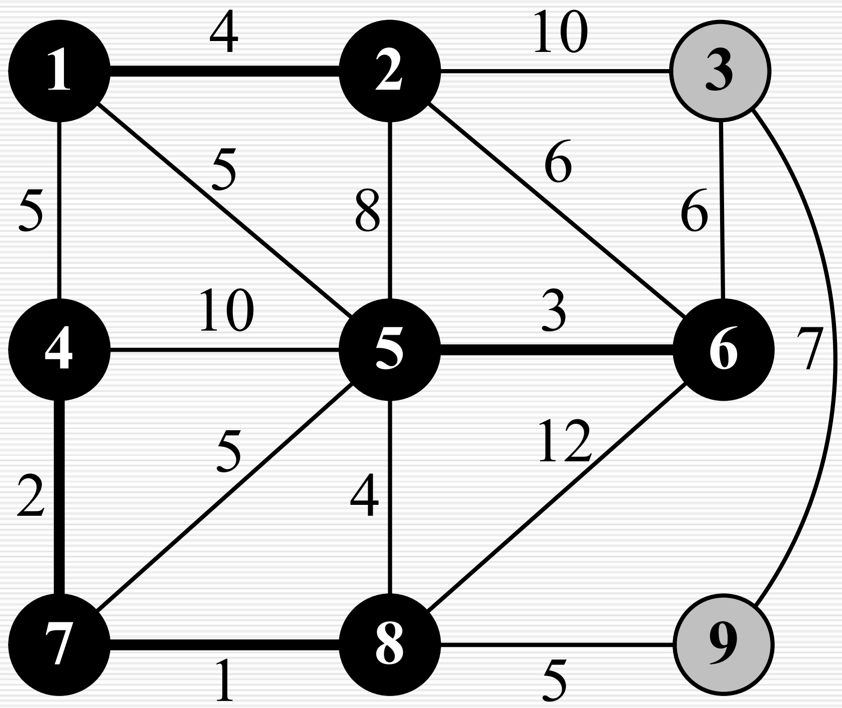
Αλγόριθμος Kruskal: παράδειγμα



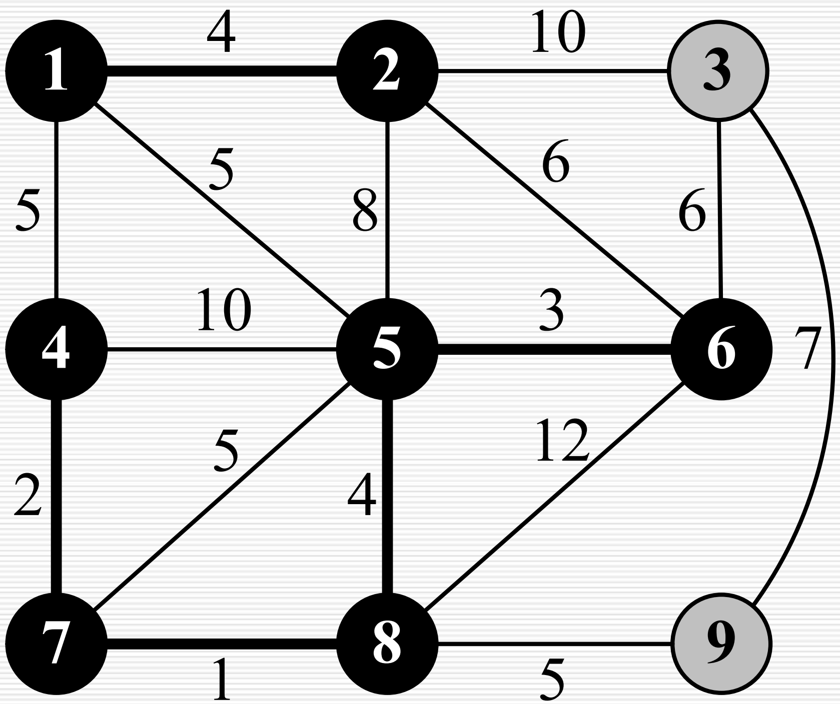
Αλγόριθμος Kruskal: παράδειγμα



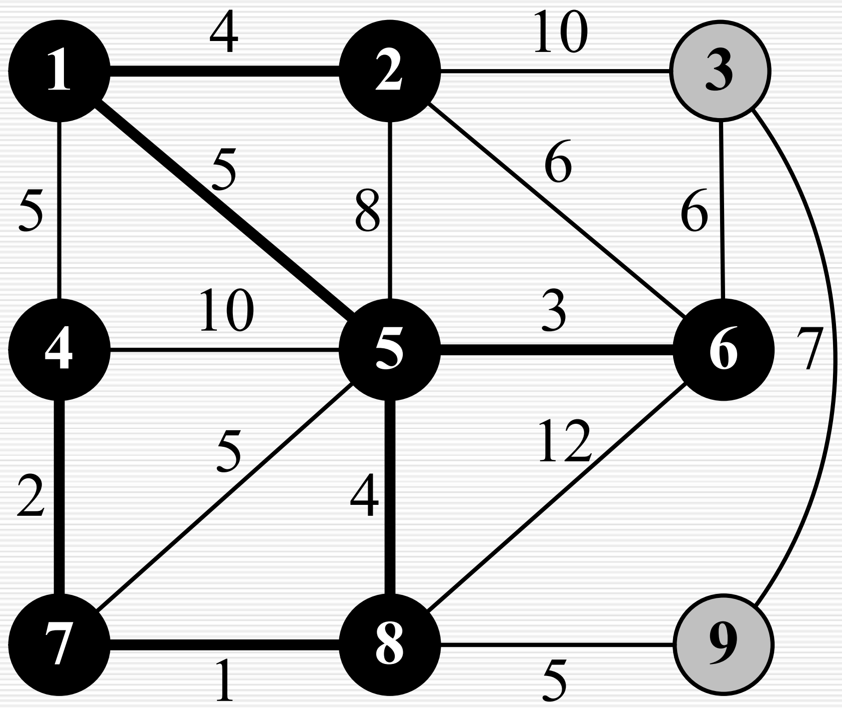
Αλγόριθμος Kruskal: παράδειγμα



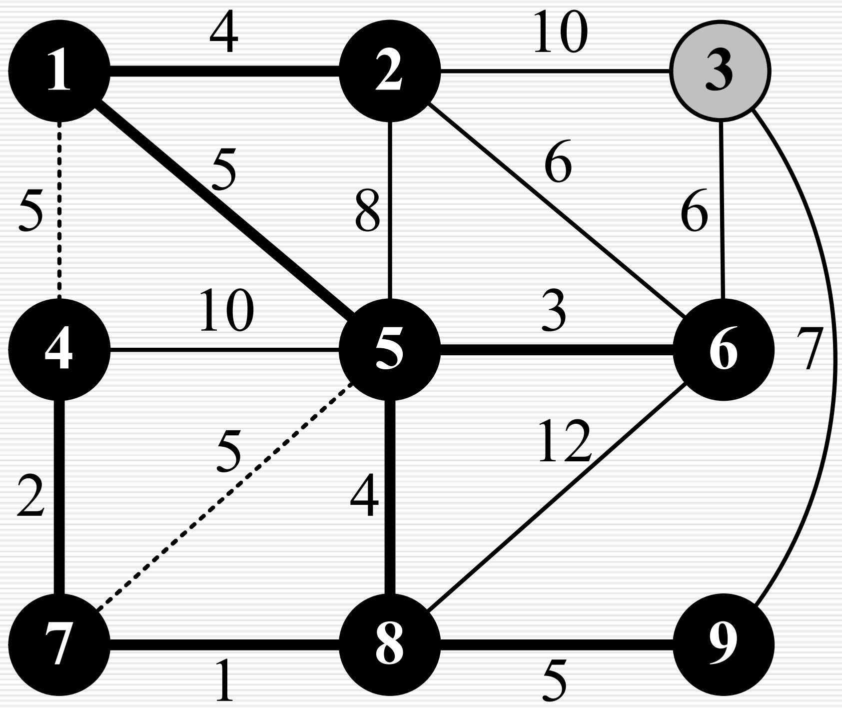
Αλγόριθμος Kruskal: παράδειγμα



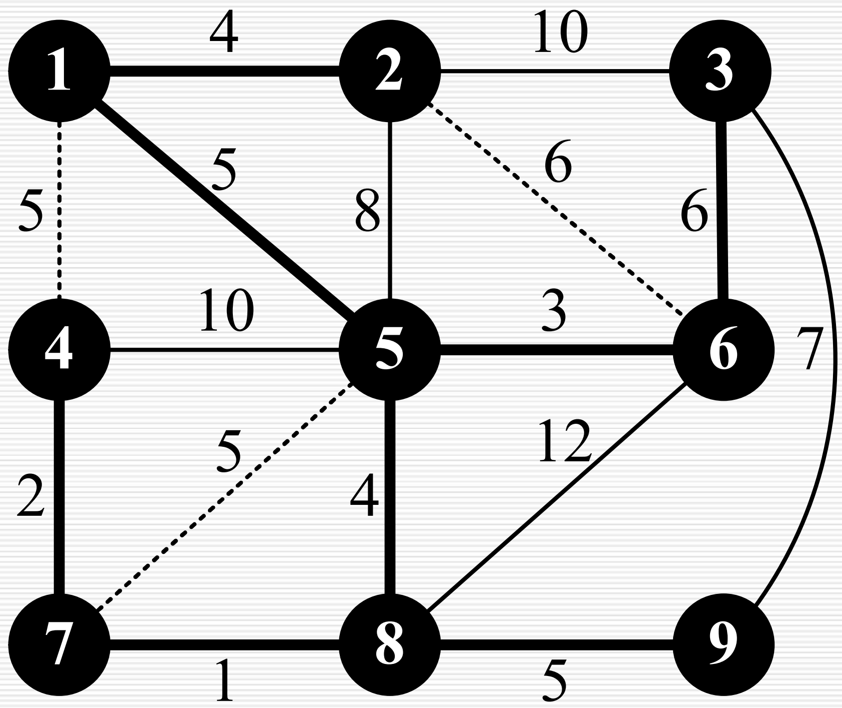
Αλγόριθμος Kruskal: παράδειγμα



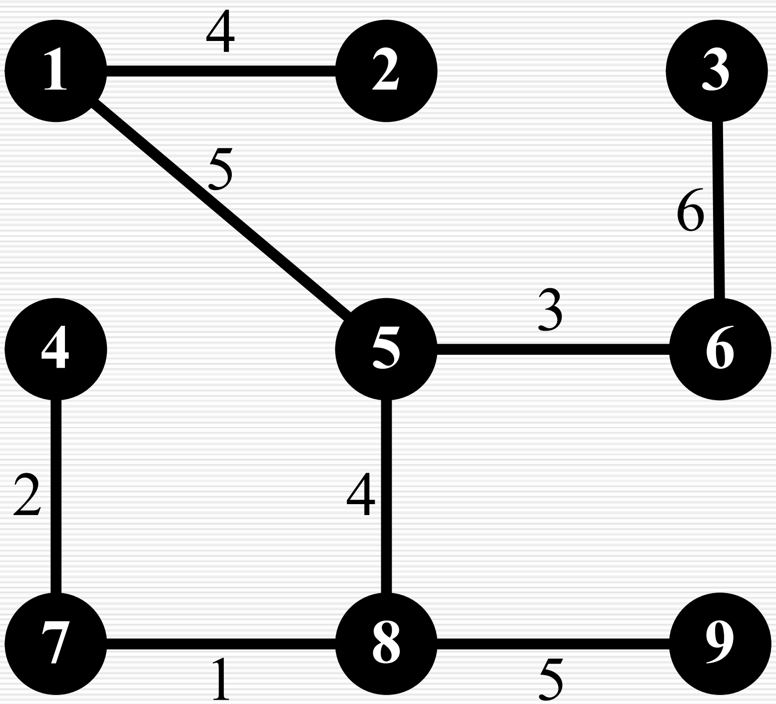
Αλγόριθμος Kruskal: παράδειγμα



Αλγόριθμος Kruskal: παράδειγμα



Αλγόριθμος Kruskal: παράδειγμα



Αλγόριθμος Kruskal

- Οι ακμές ταξινομούνται σε αύξουσα σειρά κόστους. Κάθε φορά επιλέγεται η ακμή ελαχίστου κόστους και αν δε δημιουργεί κύκλο στο μέχρι στιγμής δάσος προστίθεται σε αυτό, αλλιώς απορρίπτεται.
- Για αποδοτική υλοποίηση, η ύπαρξη κύκλου ελέγχεται με χρήση πράξεων συνόλων (UNION-FIND: αρκεί χρήση Union by Size/Rank).
- **Πολυπλοκότητα**: **$O(|E|\log|V|)$** (γιατί;)

Κοινή ιδέα Prim-Kruskal

- έστω ο αρχικός γράφος G=(V, E)
- ξεκινώντας από τον γράφο G'=(V, Ø) που περιέχει όλους τους κόμβους του G αλλά καθόλου ακμές και
- ενώνοντας επαναληπτικά δύο οποιαδήποτε συμπληρωματικά υποσύνολα κόμβων S και V\S που ακόμη δεν έχουν ακμή μεταξύ τους με την ελαφρύτερη δυνατή ακμή από το E
- καταλήγουμε σε ελάχιστο συνδετικό δένδρο

Γιατί δουλεύει η ιδέα;

Θεώρημα

Ένα σύνολο ακμών Α που είναι *υποσχόμενο* (δηλ. υποσύνολο ενός MST) παραμένει υποσχόμενο αν του προσθέσουμε την ελαφρύτερη ακμή e=(u,v) που συνδέει μια συνεκτική συνιστώσα (connected component) V_i του τρέχοντος υπογράφου (που ορίζεται από τους κόμβους του V και τις ακμές του Α) με τον υπόλοιπο υπογράφο $V-V_i$.

Απόδειξη. Θεωρούμε ένα MST *T* που είναι υπερσύνολο του Α. Έστω ότι η e δεν ανήκει στο *T*, τότε υπάρχει μονοπάτι p που συνδέει u,v στο *T*. Έστω e' στο p που διασχίζει τομή (V_i,$V-V_i$). Ισχύει cost(e) <= cost (e'), επομένως:

ανταλλαγή e , e' => MST *T'* που περιέχει την e

Bonus: αλγόριθμος Boruvka

- Λειτουργεί σε γύρους. Αρχικά κάθε κόμβος είναι συνιστώσα μόνος του.
- Σε κάθε γύρο, *κάθε* συνεκτική συνιστώσα συνδέεται με την **ελαφρύτερη** δυνατή ακμή με κάποια από τις υπόλοιπες συνιστώσες. Χρειάζεται τρόπος επίλυσης 'ισοπαλιών'.

Πολυπλοκότητα: **O(|E| log|V|)** (σε κάθε γύρο το πλήθος συνιστωσών μειώνεται στο μισό).

Προσφέρεται για παράλληλη / κατανεμημένη υλοποίηση.

Δύο σημαντικές τεχνικές

- **Άπληστοι** (greedy) αλγόριθμοι: «χτίσιμο» λύσης σταδιακά, από μικρότερα προς μεγαλύτερα υποπροβλήματα. Σε κάθε στάδιο αμετάκλητη επιλογή, δίνει βέλτιστη λύση για αντίστοιχο υποπρόβλημα.
  - Dijkstra, Prim, Kruskal, Boruvka
- **Δυναμικός προγραμματισμός**: «χτίσιμο» λύσης σταδιακά, συνδυάζοντας βέλτιστες λύσεις μικρότερων υποπροβλημάτων ώστε να προκύψει βέλτιστη λύση μεγαλύτερων (αρχή βελτιστότητας υπο-λύσεων).
  - Bellman-Ford
- Σύγκριση με «**διαίρει και κυρίευε**»: στη ΔκΚ τα υποπροβλήματα είναι ανεξάρτητα, στον ΔΠ και στους άπληστους τα υποπροβλήματα έχουν επικάλυψη.

Προβλήματα Γράφων στην Κλάση **P**

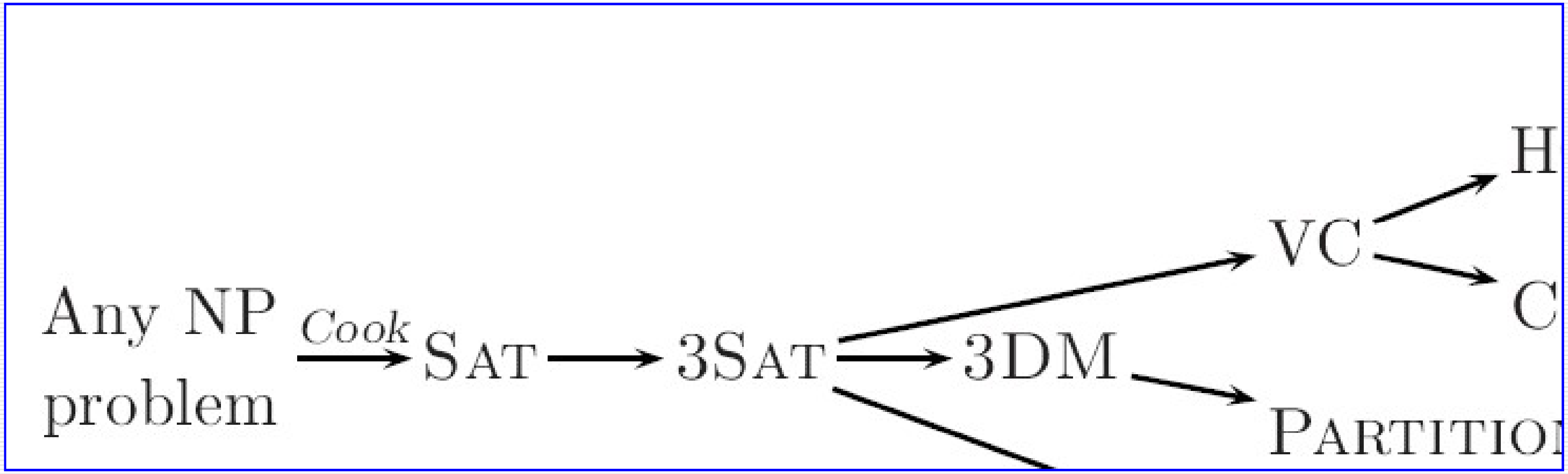
- Κύκλος Euler
- Προσβασιμότητα (reachability) + Διάσχιση (traversal): DFS, BFS, ...
- Συνεκτικές συνιστώσες (connected components)
- Συντομότερα μονοπάτια (shortest paths)
- Ελάχιστο συνδετικό δένδρο (minimum spanning tree)
- Μέγιστη ροή (maximum flow) [7^{o} εξ.]
- Τέλειο ταίριασμα (perfect matching) [7^{o} εξ.]

NP-πλήρη Προβλήματα Γράφων

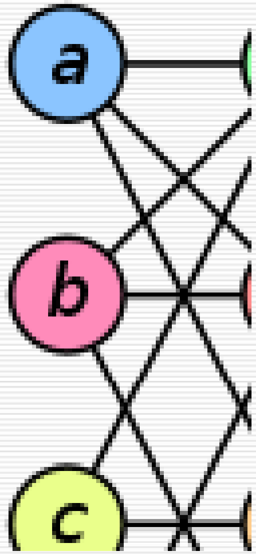
- VERTEX COVER (VC)
- CLIQUE
- HAMILTON CIRCUIT (HC)
- TRAVELING SALESMAN (TSP)
- 3-COLORABILITY
- SUBGRAPH ISOMORPHISM
- 3-DIMENSIONAL MATCHING (3DM)

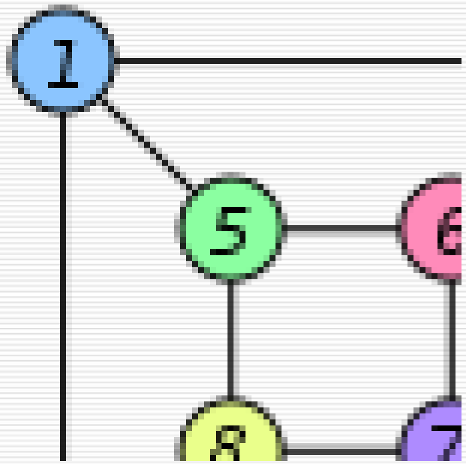
NP-πλήρη Προβλήματα Γράφων

Απόδειξη **NP**-πληρότητας: *αναγωγές*



«Ενδιάμεση» Πολυπλοκότητα;





Ισομορφισμός γράφων: δεν είναι NP-πλήρες πρόβλημα (κάτω από γενικά παραδεκτές υποθέσεις)

Συμπεράσματα

- Αρκετά προβλήματα γράφων λύνονται γρήγορα: διάσχιση (προσβασιμότητα), συνεκτικές συνιστώσες, ελάχιστες διαδρομές, ελάχιστο συνδετικό δένδρο, κύκλος Euler, τέλειο ταίριασμα, μέγιστη ροή, ...
- Πολλά προβλήματα φαίνεται να μην λύνονται γρήγορα: VERTEX COVER, CLIQUE, HAMILTON CIRCUIT, TRAVELING SALESMAN, 3-COLORABILITY, SUBGRAPH ISOMORPHISM, 3-DIMENSIONAL MATCHING, ...
- Κάποια από αυτά λύνονται γρήγορα σε ειδικές περιπτώσεις, ή προσεγγιστικά. Εντατική έρευνα, πολλά ανοιχτά ερωτήματα.