Θεμελιώδη Θέματα Επιστήμης Υπολογιστών

ΣΗΜΜΥ – ΣΕΜΦΕ ΕΜΠ

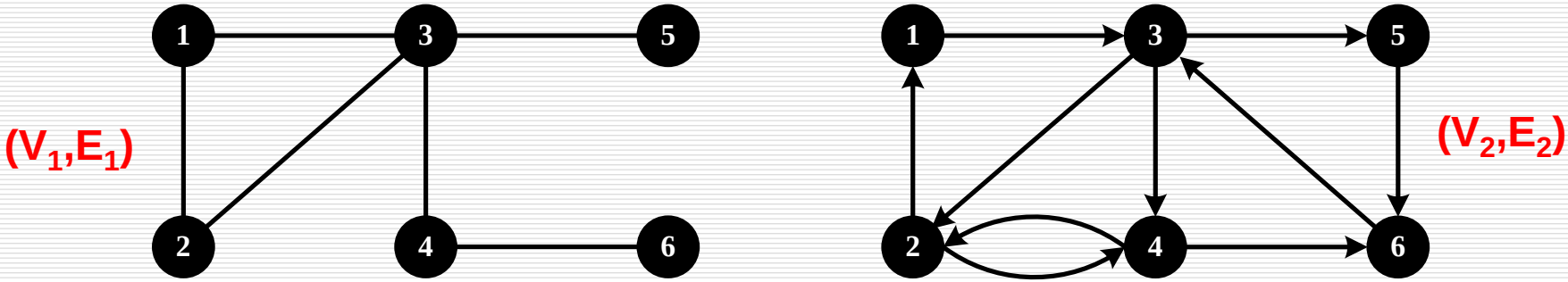
5η ενότητα:
Αλγόριθμοι γράφων και δικτύων

Επιμέλεια διαφανειών:
Στάθης Ζάχος, Άρης Παγουρτζής, Δημήτρης Φωτάκης

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

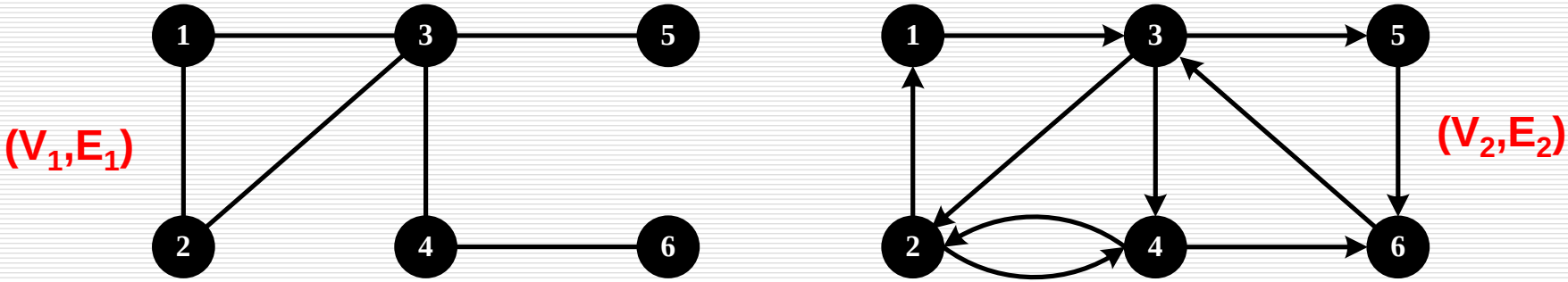
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Εθνικό Μετσόβιο Πολυτεχνείο

Γράφοι (επανάληψη)



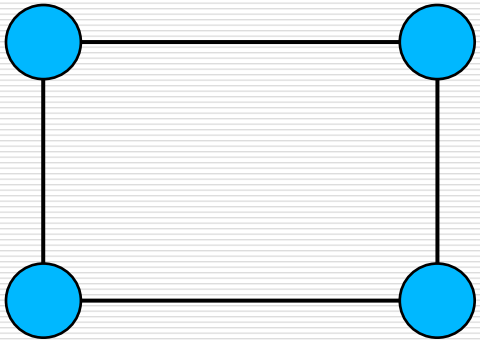
- **Γράφος (ή γράφημα):** ζεύγος **(V,E)**, **V** ένα μη κενό σύνολο, **E** διμελής σχέση πάνω στο **V**
- **Μη κατευθυνόμενος γράφος:** σχέση **E** συμμετρική
- **V**: **κορυφές** (vertices), **κόμβοι** (nodes)
- **E**: **ακμές** (edges)
  - E_1 = {{1,2}, {1,3}, {2,3}, {3,4}, {3,5}, {4,6}}
  - E_2 =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

Γράφοι (επανάληψη): ορολογία



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

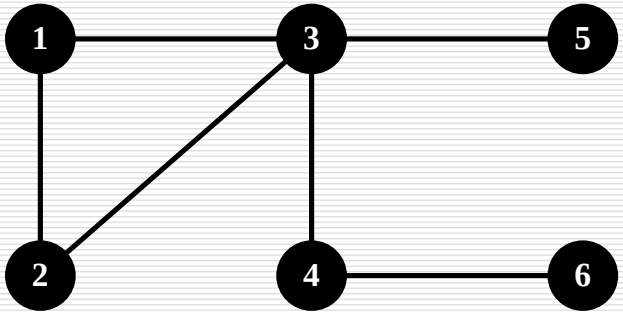
Γράφοι (επανάληψη): ορολογία

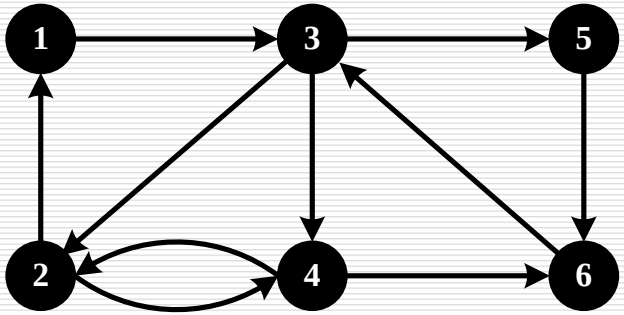


2-κανονικός γράφος

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

Γράφοι (επανάληψη): διαδρομές

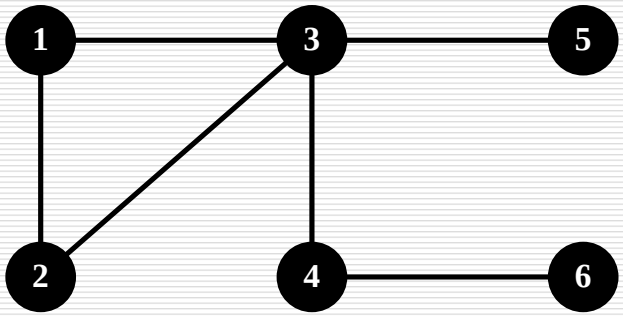




- **Διαδρομή:** ακολουθία από «διαδοχικές» κορυφές-ακμές
- **Μονοκονδυλιά:** δρόμος χωρίς επαναλήψεις ακμών
- **(Απλό) Μονοπάτι:** μονοπάτι χωρίς επαναλήψεις κορυφών
- **Κύκλωμα:** κλειστή μονοκονδυλιά
- **(Απλός) Κύκλος:** κλειστό μονοπάτι
- **Μήκος δρόμου:** το πλήθος των ακμών του

Γράφοι (επανάληψη): αναπαράσταση

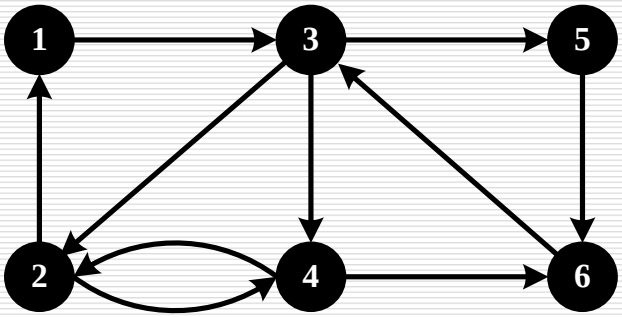
- ... με **πίνακα γειτνίασης**: $A[i,j] = \begin{cases} 1 & (v_i, v_j) \in E \\ 0 & (v_i, v_j) \notin E \end{cases}$
  - Αν έχουμε βάρη, $A[i,j] = w(v_i, v_j)$
  - Μη-κατευθυνόμενος: συμμετρικός πίνακας
  - Χώρος: $\Theta(n^2)$
  - Προσπέλαση γειτόνων: $\Theta(n)$
  - Άμεσος έλεγχος ύπαρξης ακμής: $O(1)$



	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

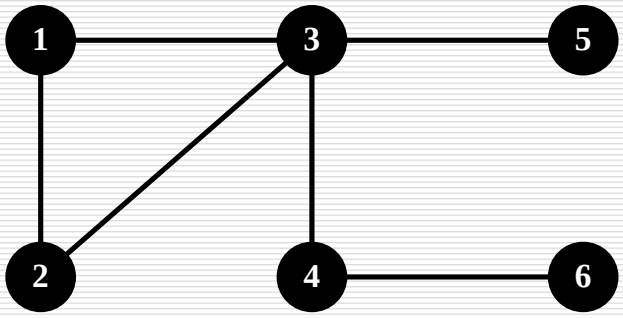
- ... με **πίνακα γειτνίασης**: $A[i,j] = \begin{cases} 1 & (v_i, v_j) \in E \\ 0 & (v_i, v_j) \notin E \end{cases}$
  - Αν έχουμε βάρη, $A[i,j] = w(v_i, v_j)$
  - Κατευθυνόμενος: μη-συμμετρικός πίνακας
  - Χώρος: $\Theta(n^2)$
  - Προσπέλαση γειτόνων: $\Theta(n)$
  - Άμεσος έλεγχος ύπαρξης ακμής: $O(1)$

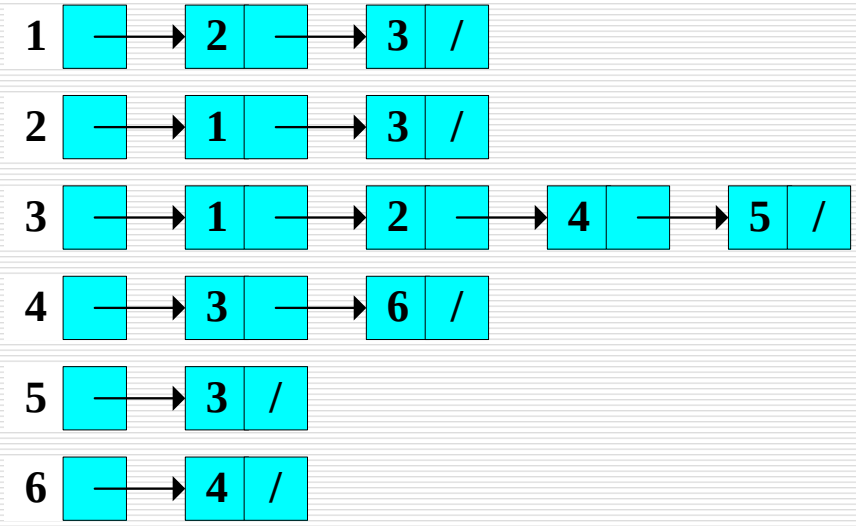


	1	2	3	4	5	6
1	0	0	1	0	0	0
2	1	0	0	1	0	0
3	0	1	0	1	1	0
4	0	1	0	0	0	1
5	0	0	0	0	0	1
6	0	0	1	0	0	0

Γράφοι (επανάληψη): αναπαράσταση

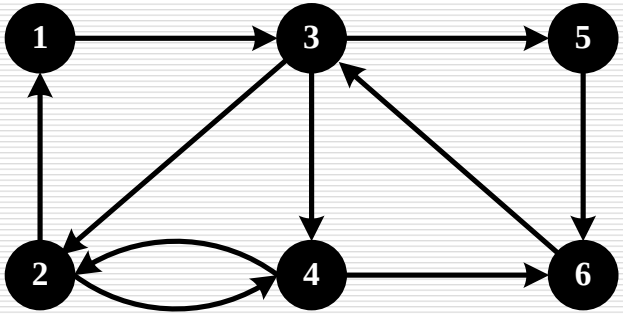
- … με **λίστες γειτνίασης**: γειτονικές κορυφές σε λίστες
  - Αν έχουμε βάρη, τα αποθηκεύουμε στους κόμβους
  - Χώρος: Θ(m)
  - Προσπέλαση γειτόνων: Θ(deg(u))
  - Έλεγχος ύπαρξης ακμής: O(deg(u))

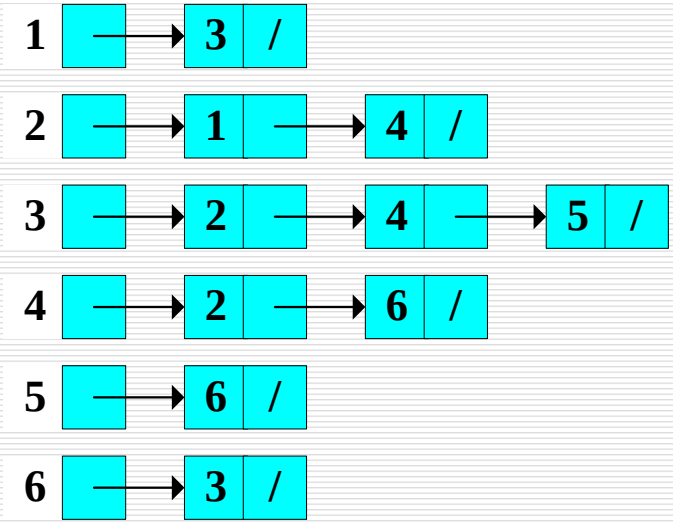




Γράφοι (επανάληψη): αναπαράσταση

- … 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))



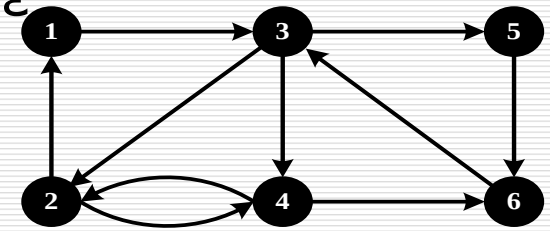


Γράφοι (επανάληψη): συνεκτικότητα

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

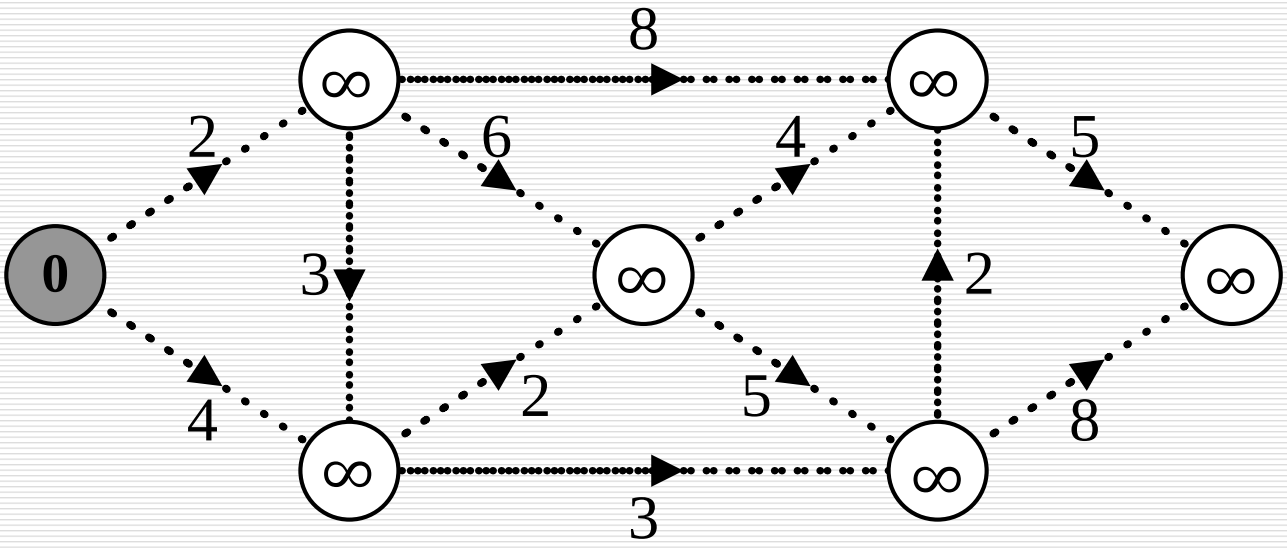
  Σε συνεκτικό γράφο ισχύει: $n-1 \leq e \leq \frac{n(n-1)}{2}, n = |V|, e = |E|.$

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



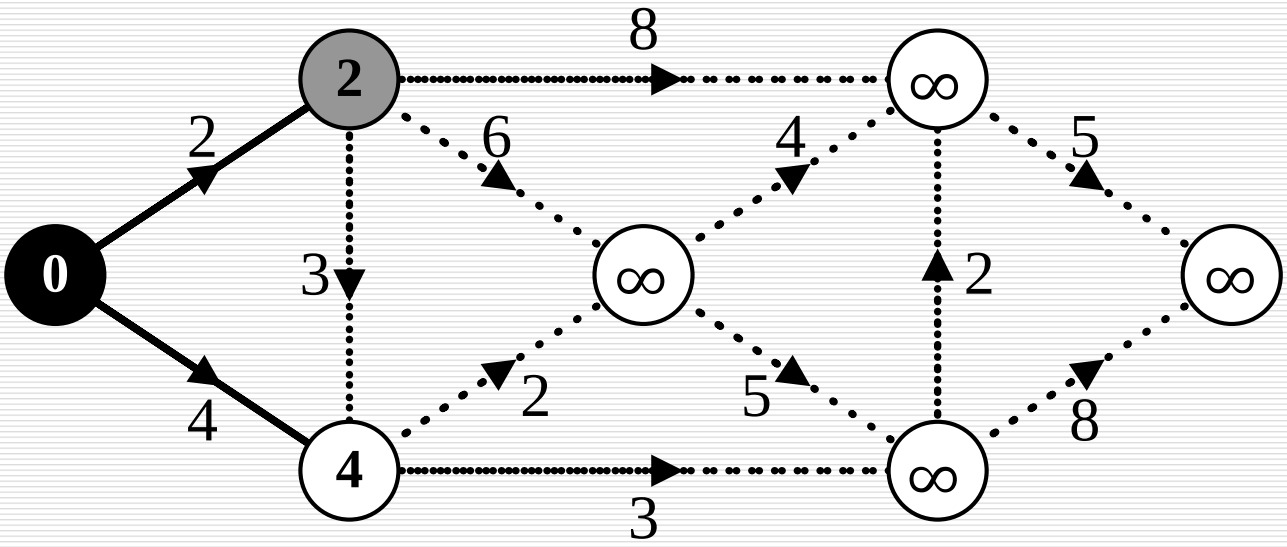
  - **ασθενώς συνεκτικός (weakly connected)** αν υπάρχει διαδρομή μεταξύ οποιωνδήποτε δύο κορυφών του αγνοώντας τις κατευθύνσεις των ακμών

2ο παράδειγμα Dijkstra



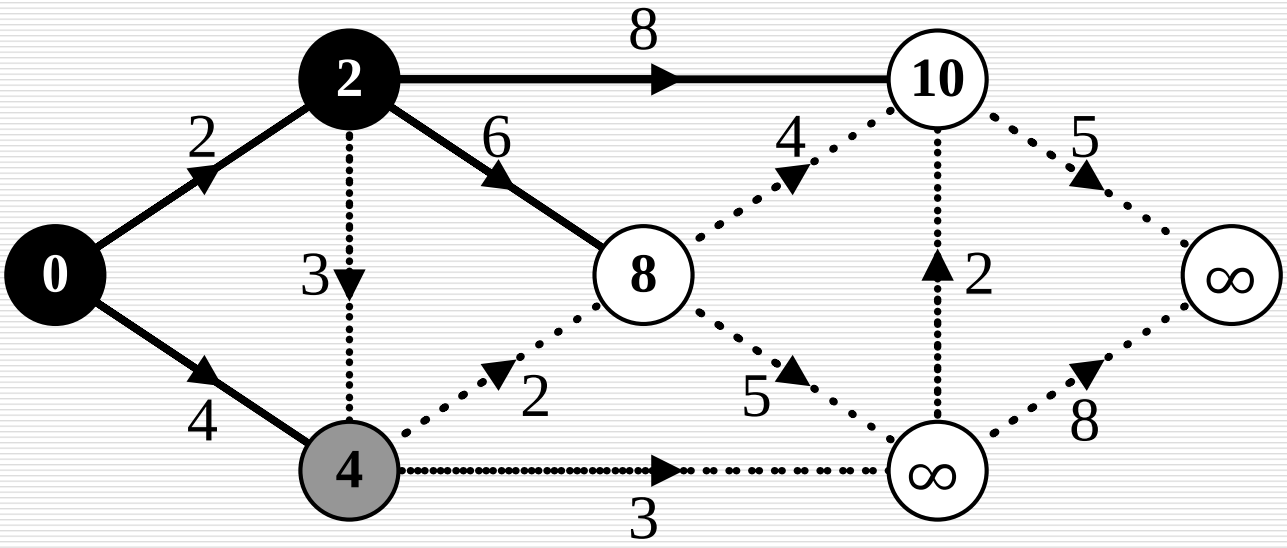
Οι ετικέτες των κόμβων δείχνουν την μέχρι στιγμής ελάχιστη απόσταση από τον αρχικό κόμβο (πίνακας D).

2ο παράδειγμα Dijkstra



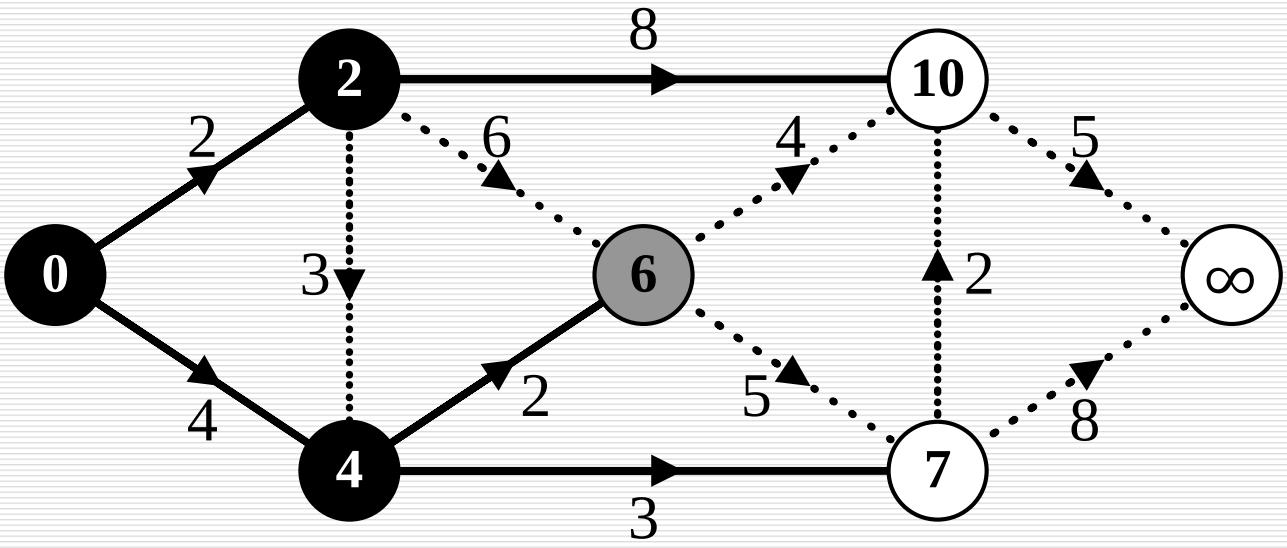
Οι ετικέτες των κόμβων δείχνουν την μέχρι στιγμής ελάχιστη απόσταση από τον αρχικό κόμβο (πίνακας D), οι συνεχείς ακμές δείχνουν ποιος είναι ο αντίστοιχος προηγούμενος κόμβος (πίνακας P).

2ο παράδειγμα Dijkstra



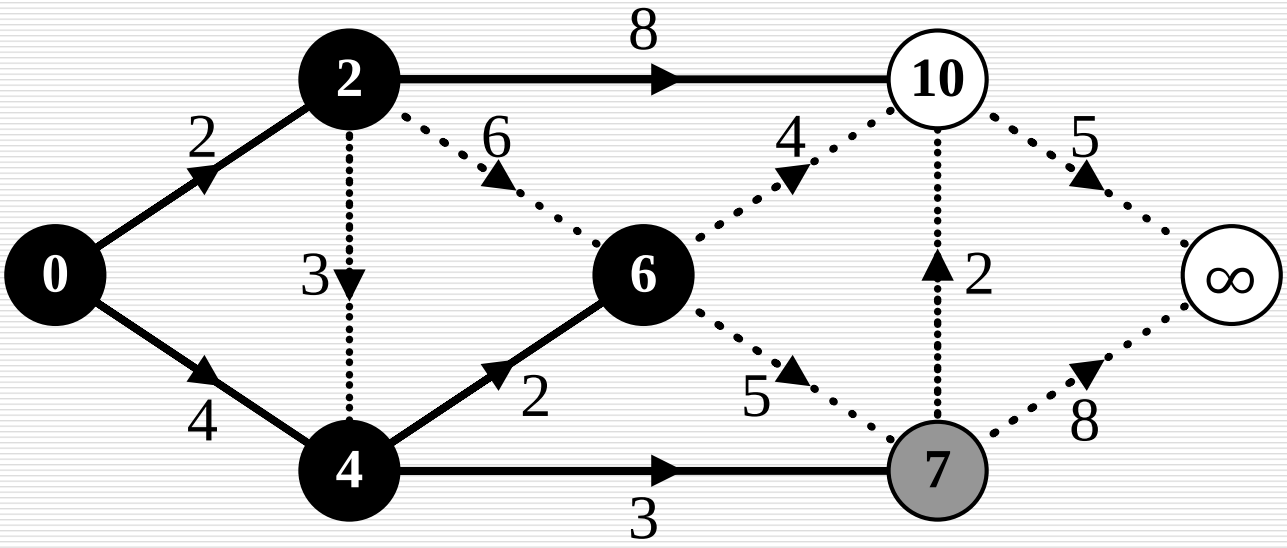
Οι ετικέτες των κόμβων δείχνουν την μέχρι στιγμής ελάχιστη απόσταση από τον αρχικό κόμβο (πίνακας D), οι συνεχείς ακμές δείχνουν ποιος είναι ο αντίστοιχος προηγούμενος κόμβος (πίνακας P).

2ο παράδειγμα Dijkstra



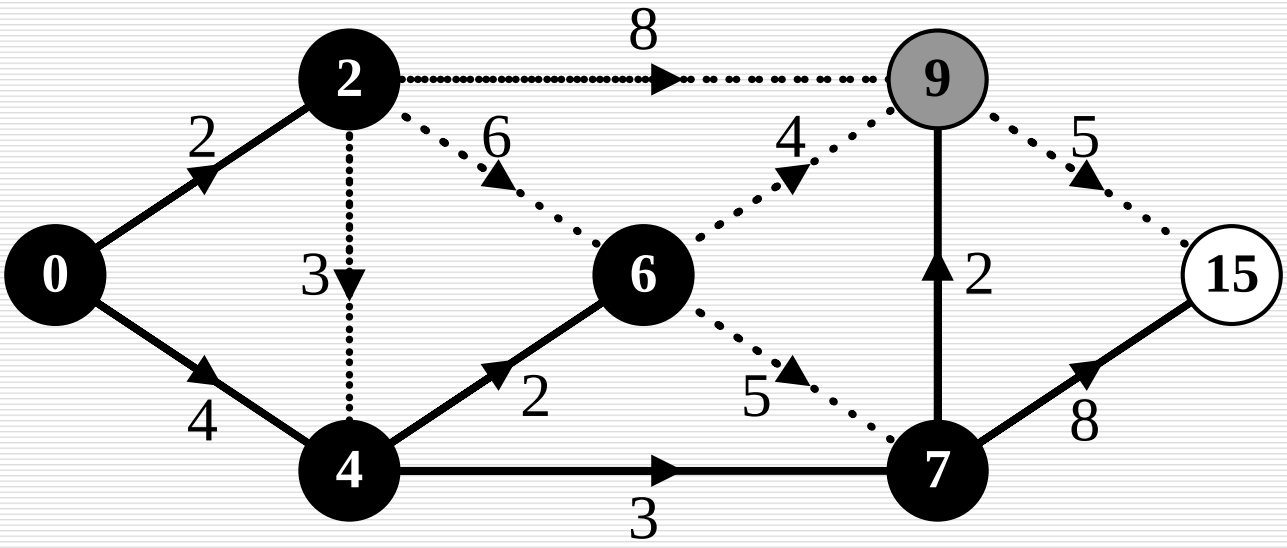
Οι ετικέτες των κόμβων δείχνουν την μέχρι στιγμής ελάχιστη απόσταση από τον αρχικό κόμβο (πίνακας D), οι συνεχείς ακμές δείχνουν ποιος είναι ο αντίστοιχος προηγούμενος κόμβος (πίνακας P).

2ο παράδειγμα Dijkstra



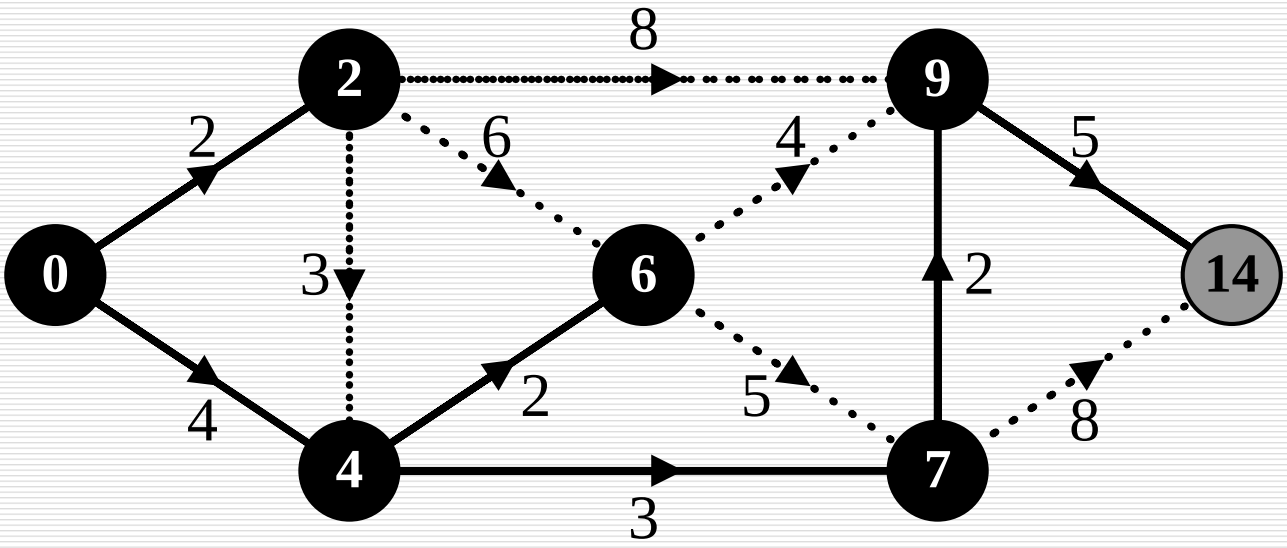
Οι ετικέτες των κόμβων δείχνουν την μέχρι στιγμής ελάχιστη απόσταση από τον αρχικό κόμβο (πίνακας D), οι συνεχείς ακμές δείχνουν ποιος είναι ο αντίστοιχος προηγούμενος κόμβος (πίνακας P).

2ο παράδειγμα Dijkstra



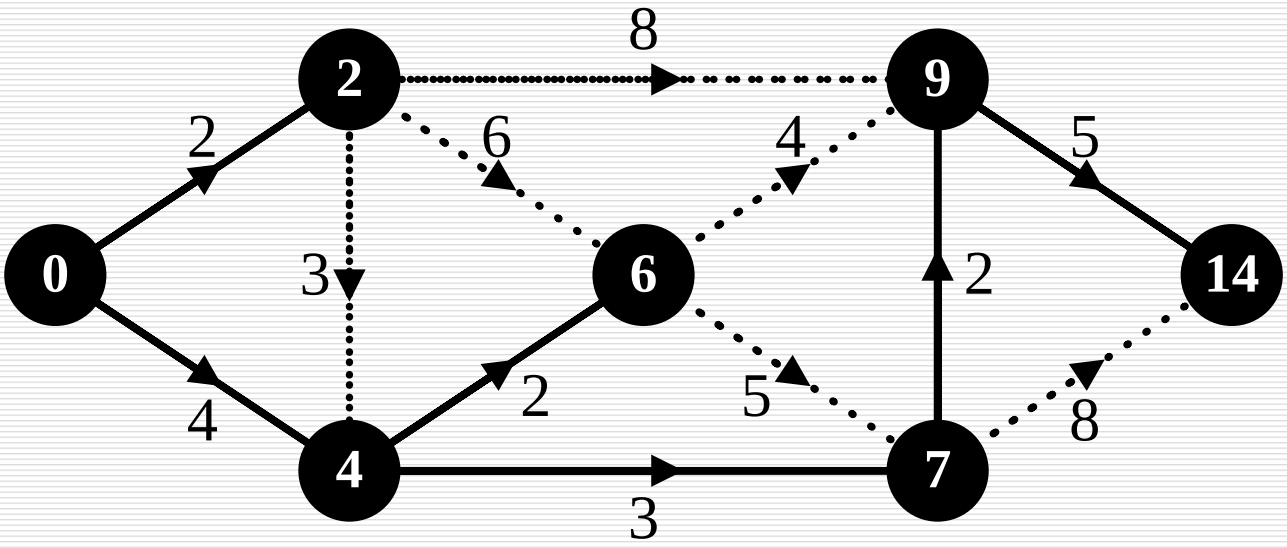
Οι ετικέτες των κόμβων δείχνουν την μέχρι στιγμής ελάχιστη απόσταση από τον αρχικό κόμβο (πίνακας D), οι συνεχείς ακμές δείχνουν ποιος είναι ο αντίστοιχος προηγούμενος κόμβος (πίνακας P).

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).

2ο παράδειγμα Dijkstra

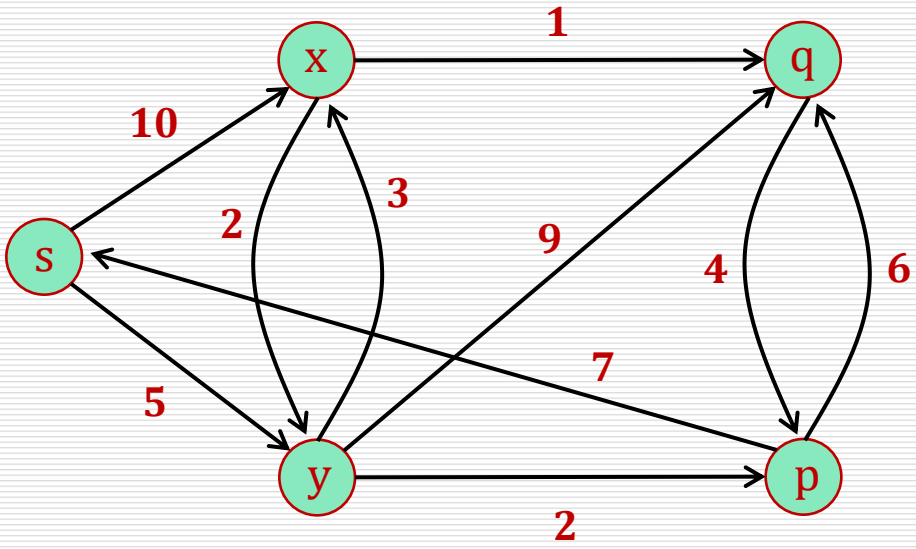


Οι ετικέτες των κόμβων δείχνουν την μέχρι στιγμής ελάχιστη απόσταση από τον αρχικό κόμβο (πίνακας D), οι συνεχείς ακμές δείχνουν ποιος είναι ο αντίστοιχος προηγούμενος κόμβος (πίνακας P).

3ο παράδειγμα Dijkstra

- Παράδειγμα

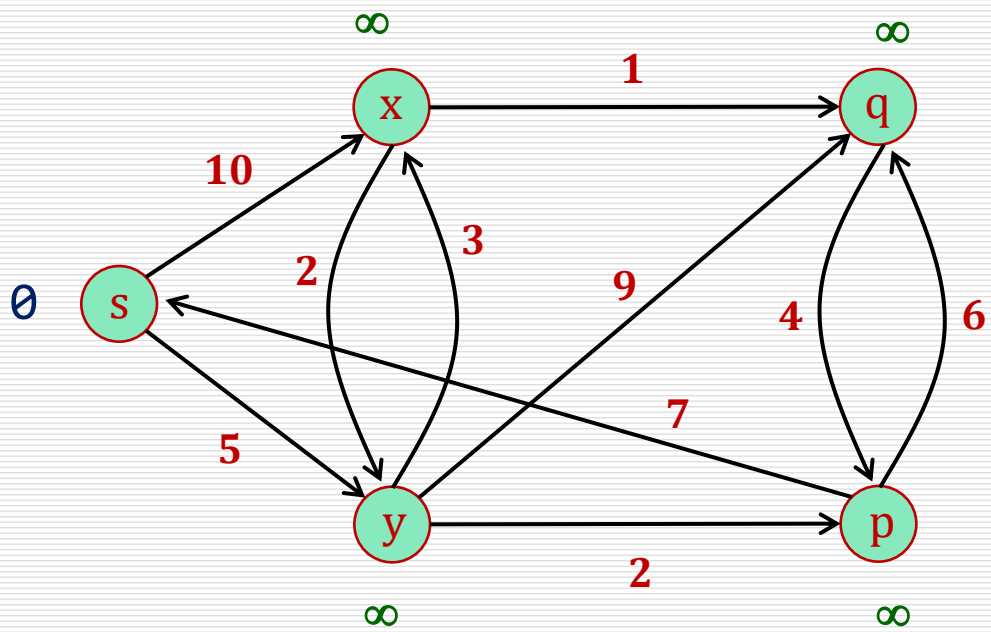
Είσοδος Αλγόριθμου



3ο παράδειγμα Dijkstra

Παράδειγμα

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

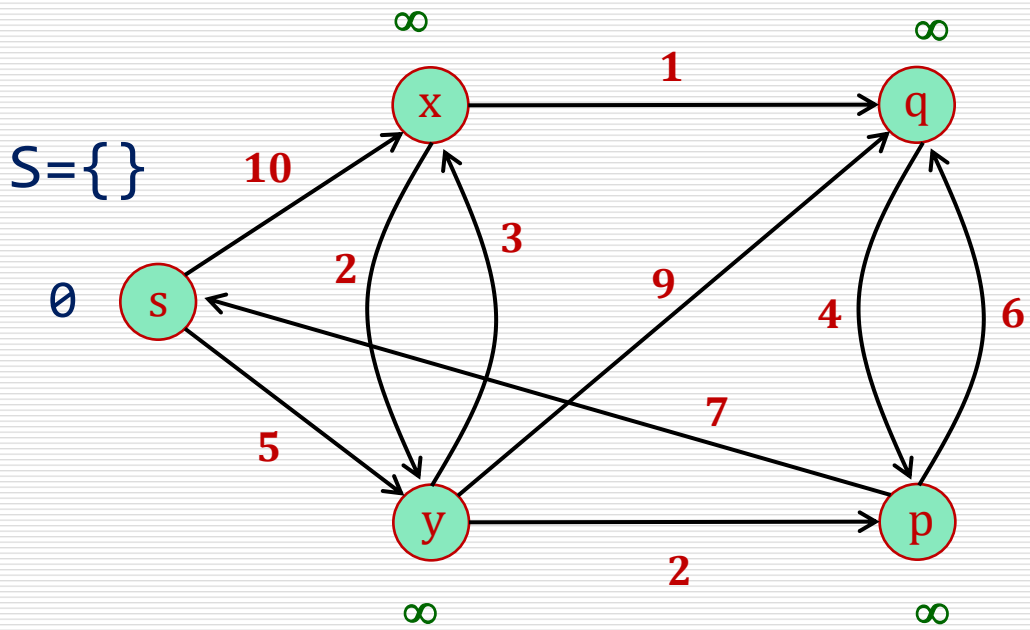


d(s)=0	d(x)=∞	d(y)=∞	d(p)=∞	d(q)=∞
prev(s)=NIL	prev(x)=NIL	prev(y)=NIL	prev(p)=NIL	prev(q)=NIL

3ο παράδειγμα Dijkstra

Παράδειγμα

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



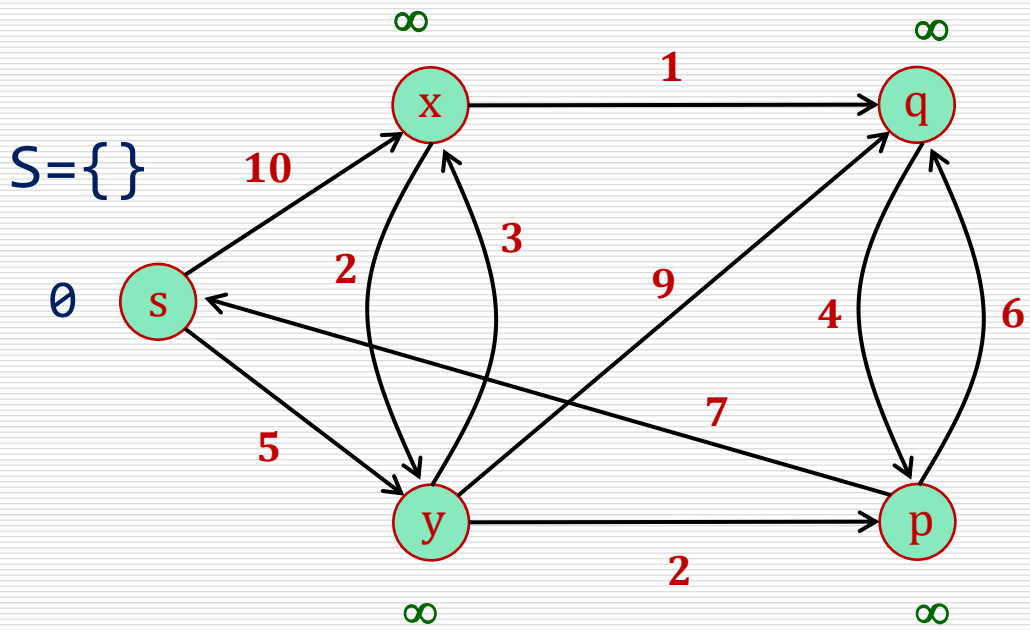
d(s)=0	d(x)=∞	d(y)=∞	d(p)=∞	d(q)=∞
prev(s)=NIL	prev(x)=NIL	prev(y)=NIL	prev(p)=NIL	prev(q)=NIL

3ο παράδειγμα Dijkstra

Παράδειγμα

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

while Q ≠ ∅ :



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

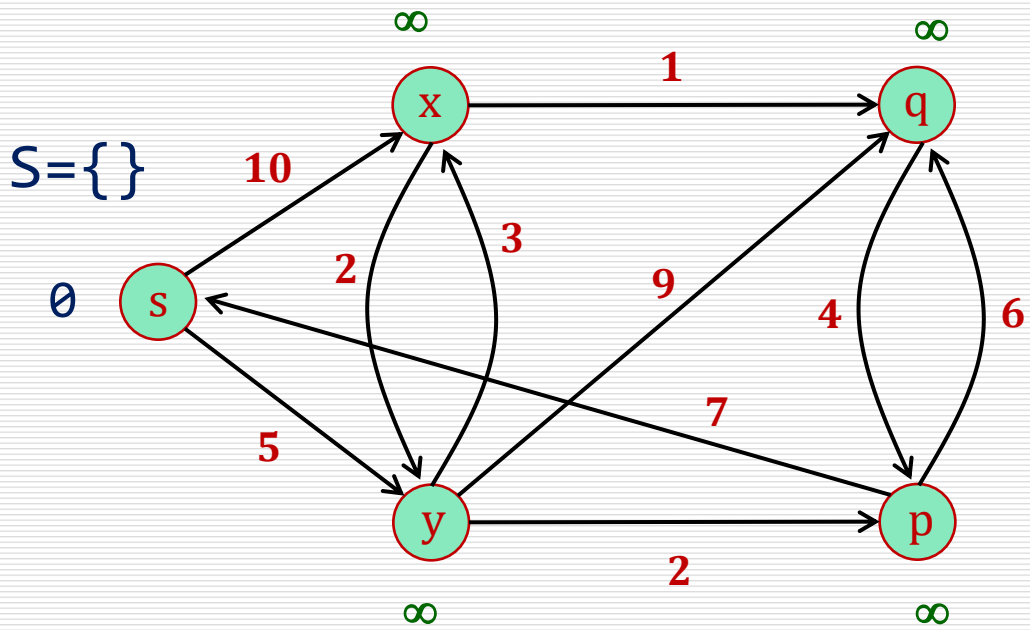
3ο παράδειγμα Dijkstra

Παράδειγμα

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

while Q ≠ ∅ :

u ← extract-min(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

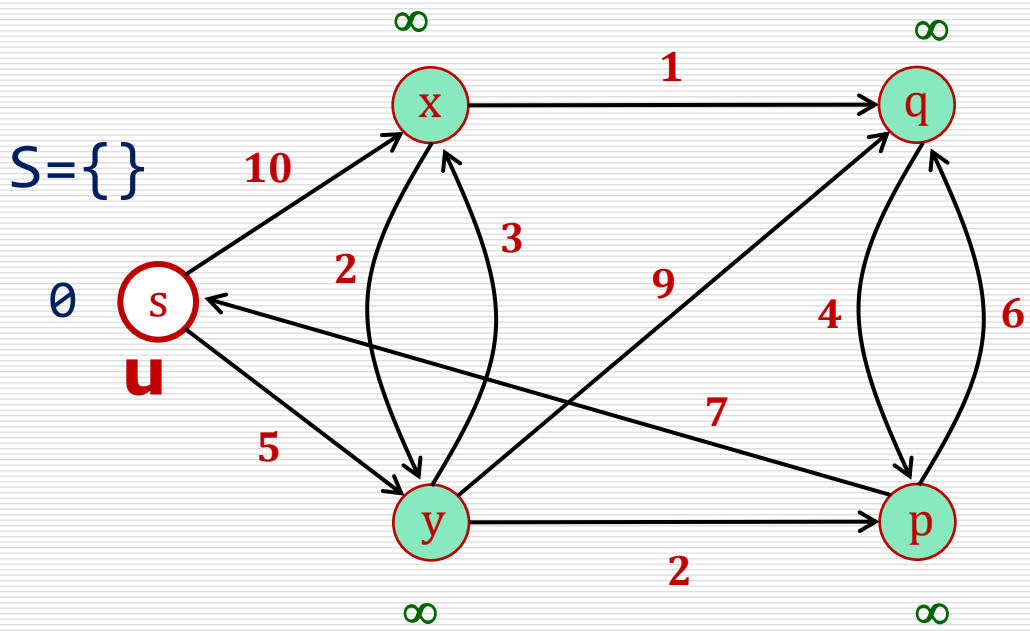
3ο παράδειγμα Dijkstra

Παράδειγμα

Q = {x,y,p,q}

while Q ≠ ∅ :

u ← extract-min(Q);



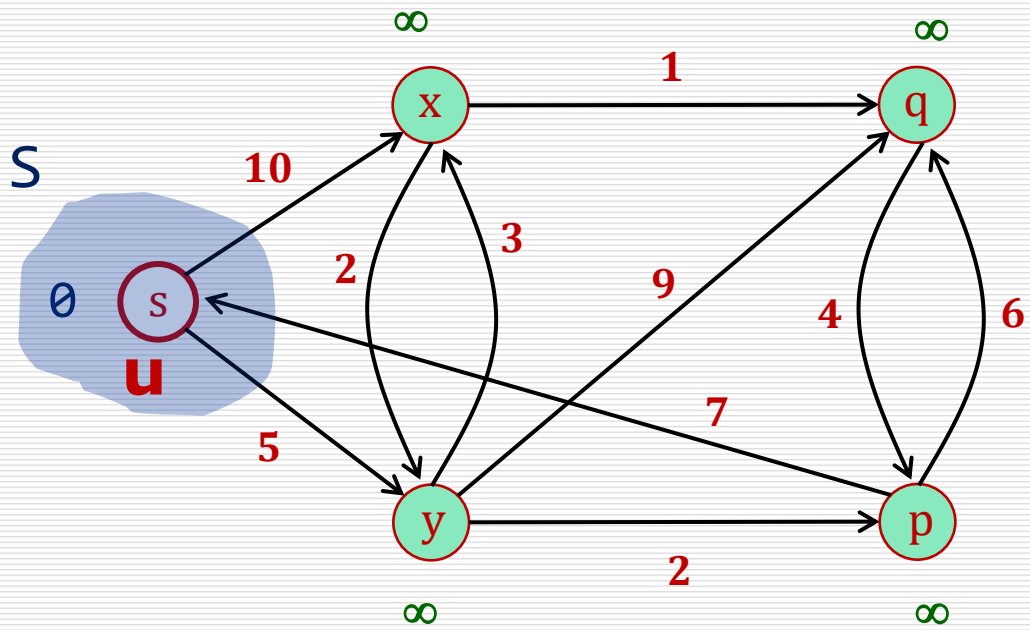
d(s)=0 d(x)=∞ d(y)=∞ d(p)=∞ d(q)=∞

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

3ο παράδειγμα Dijkstra

- Παράδειγμα

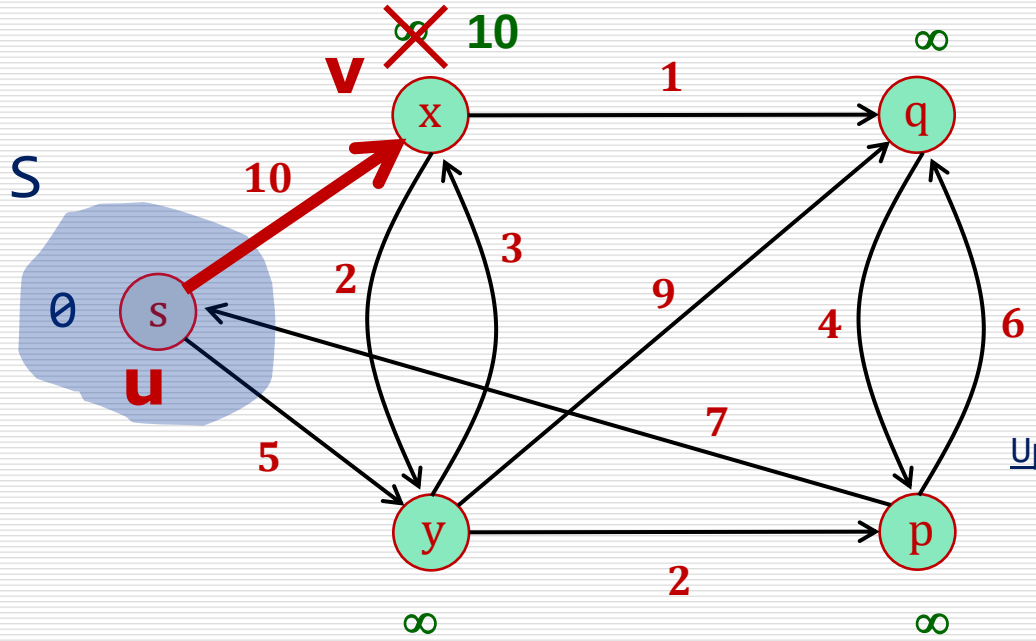
```
Q = {x,y,p,q}
while Q ≠ ∅ :
    u ← extract-min(Q);
    S ← S ∪ {u}
```



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

Παράδειγμα




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

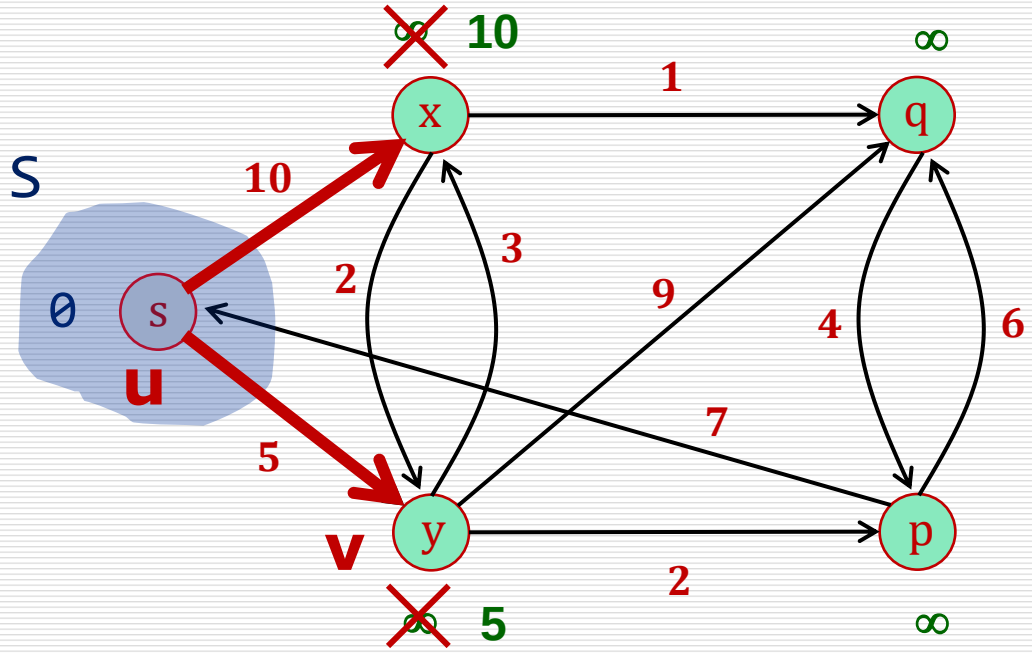
Update(u,v,c)

```
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




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

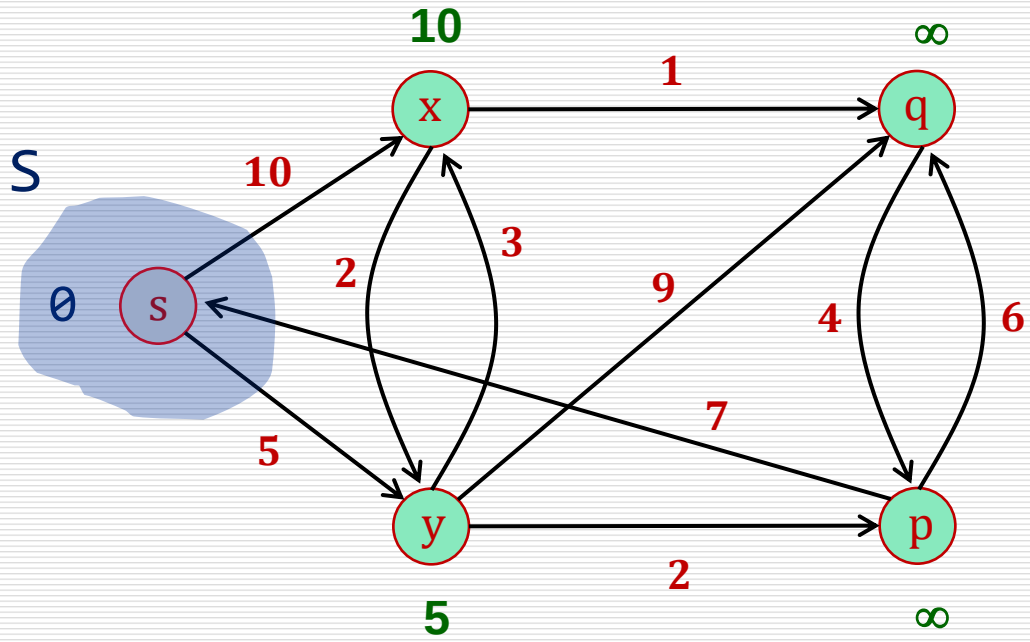
d(s)=0 d(x)=10 d(y)=5 d(p)=∞ d(q)=∞

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

3ο παράδειγμα Dijkstra

Παράδειγμα

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

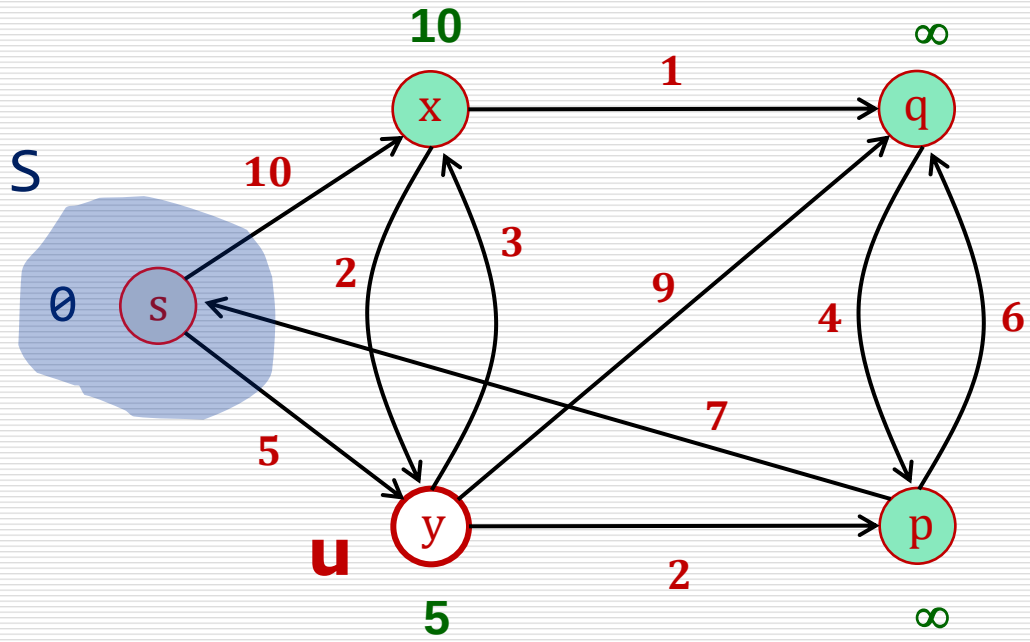
d(s)=0 d(y)=5
prev(s)=NIL prev(y)=s

Παράδειγμα

Q = {x,p,q}

while Q ≠ ∅ :

u ← extract-min(Q);



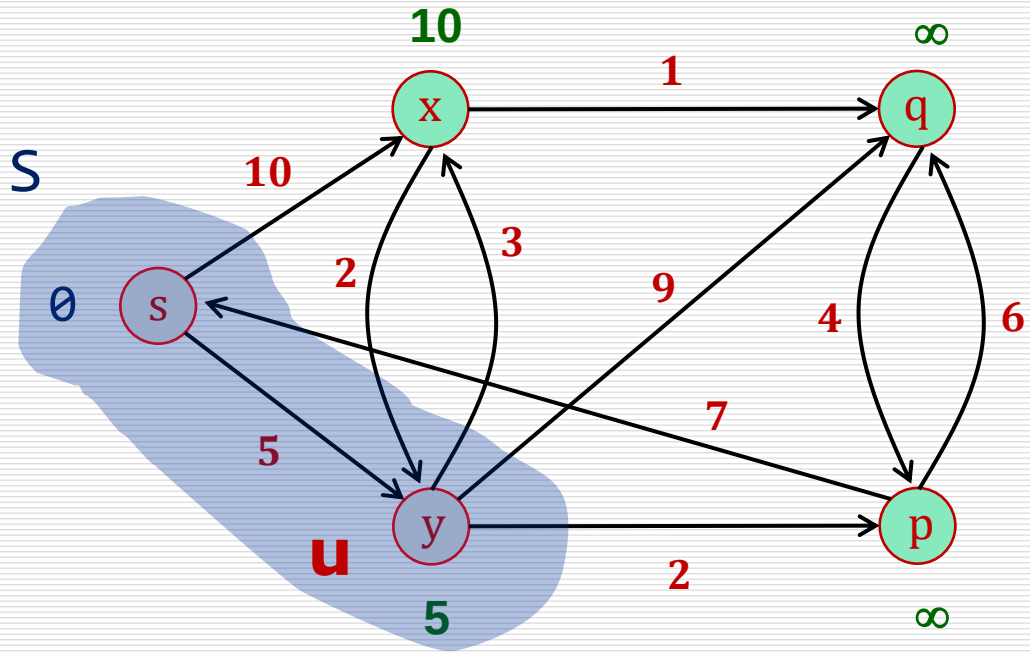
d(s)=0 d(x)=10 d(y)=5 d(p)=∞ d(q)=∞

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

3ο παράδειγμα Dijkstra

- Παράδειγμα

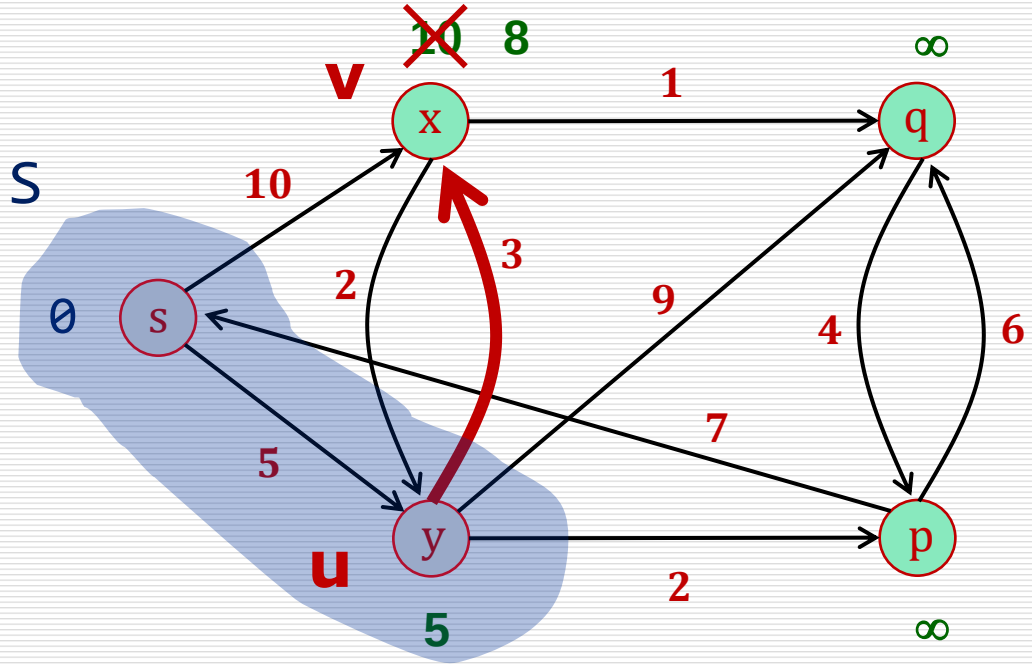
```
Q = {x,p,q}
while Q ≠ ∅ :
    u ← extract-min(Q);
    S ← S ∪ {u}
```



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

Παράδειγμα

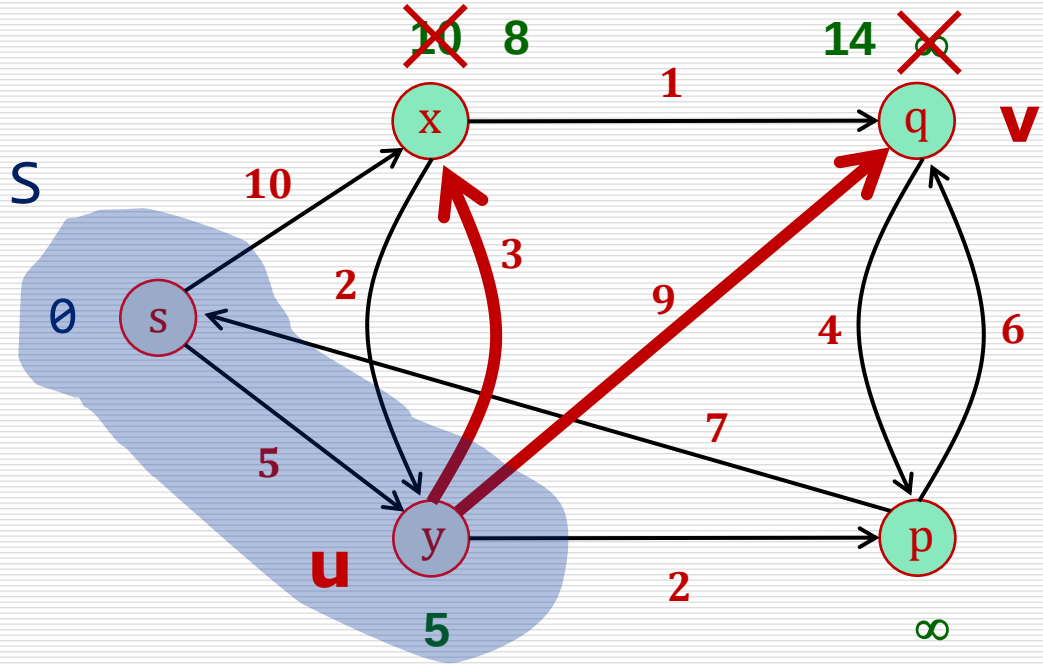



```
Q = {x,y,p,q}
while Q ≠ ∅ :
    u ← extract-min(Q);
    S ← S ∪ {u}
    για κάθε κόμβο v∈N(u)
        Update(u,v,c)
```

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

Παράδειγμα

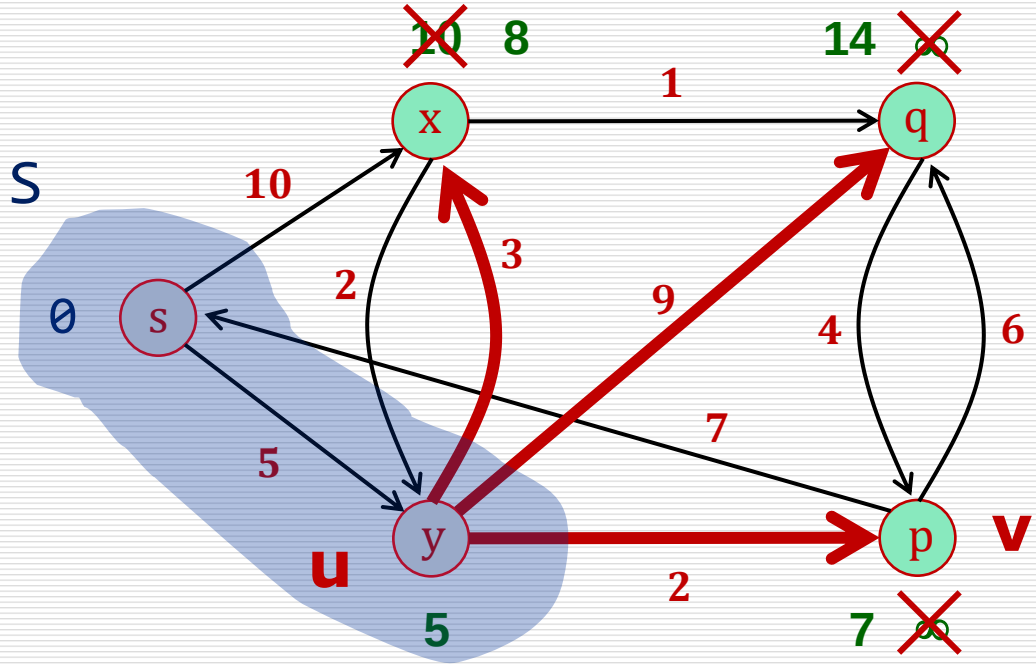



```
Q = {x,y,p,q}
while Q ≠ ∅ :
    u ← extract-min(Q);
    S ← S ∪ {u}
    για κάθε κόμβο v∈N(u)
        Update(u,v,c)
```

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

Παράδειγμα



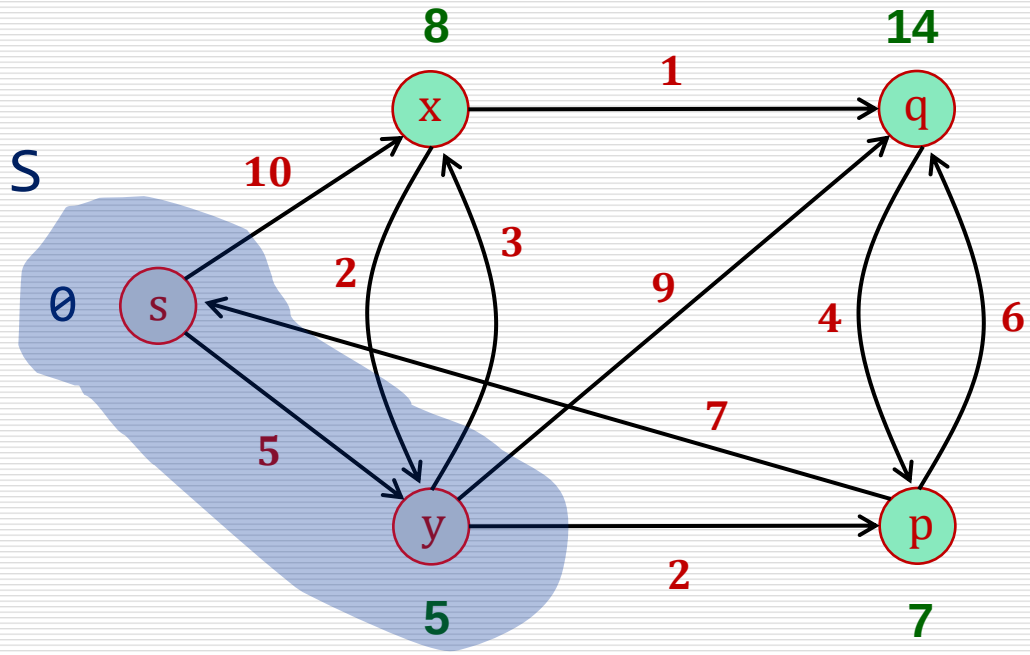

```
Q = {x,y,p,q}
while Q ≠ ∅ :
    u ← extract-min(Q);
    S ← S ∪ {u}
    για κάθε κόμβο v∈N(u)
        Update(u,v,c)
```

d(s)=0	d(x)=8	d(y)=5	d(p)=7	d(q)=14
prev(s)=NIL	prev(x)=y	prev(y)=s	prev(p)=y	prev(q)=y

3ο παράδειγμα Dijkstra

- Παράδειγμα

```
Q = {x,p,q}
while Q ≠ ∅ :
   u ← extract-min(Q);
```

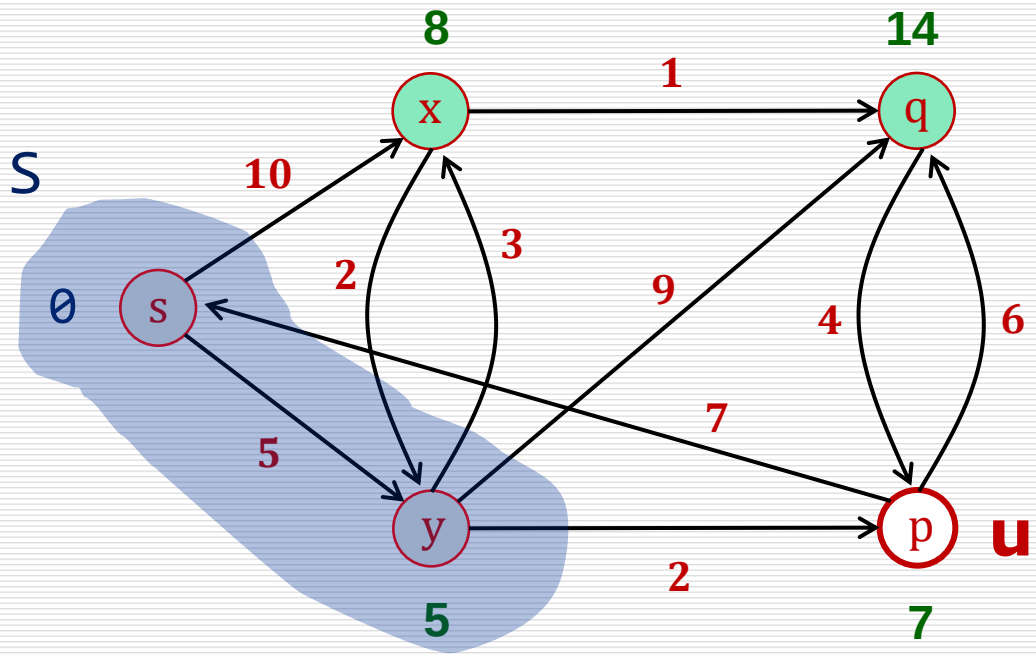


d(s)=0	d(x)=8	d(y)=5	d(p)=7	d(q)=14
prev(s)=NIL	prev(x)=y	prev(y)=s	prev(p)=y	prev(q)=y

3ο παράδειγμα Dijkstra

Παράδειγμα

```
Q = {x,q}
while Q ≠ ∅ :
   u ← extract-min(Q);
```

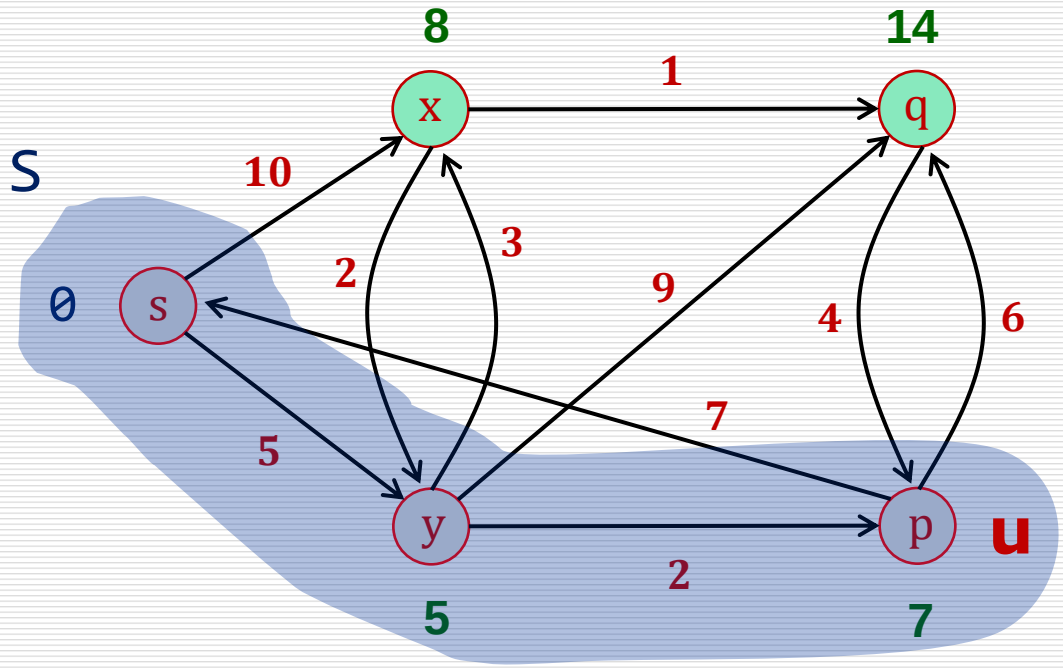


d(s)=0	d(x)=8	d(y)=5	d(p)=7	d(q)=14
prev(s)=NIL	prev(x)=y	prev(y)=s	prev(p)=y	prev(q)=y

3ο παράδειγμα Dijkstra

- Παράδειγμα

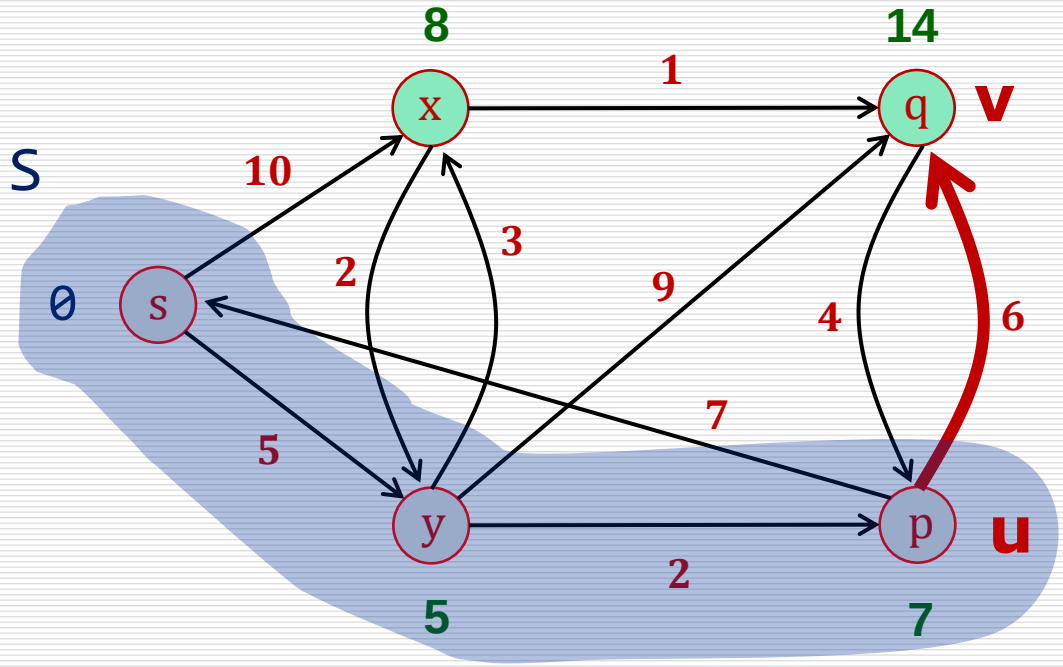
```
Q = {x,q}
while Q ≠ ∅ :
    u ← extract-min(Q);
    S ← S ∪ {u}
```



d(s)=0	d(x)=8	d(y)=5	d(p)=7	d(q)=14
prev(s)=NIL	prev(x)=y	prev(y)=s	prev(p)=y	prev(q)=y

3ο παράδειγμα Dijkstra

Παράδειγμα

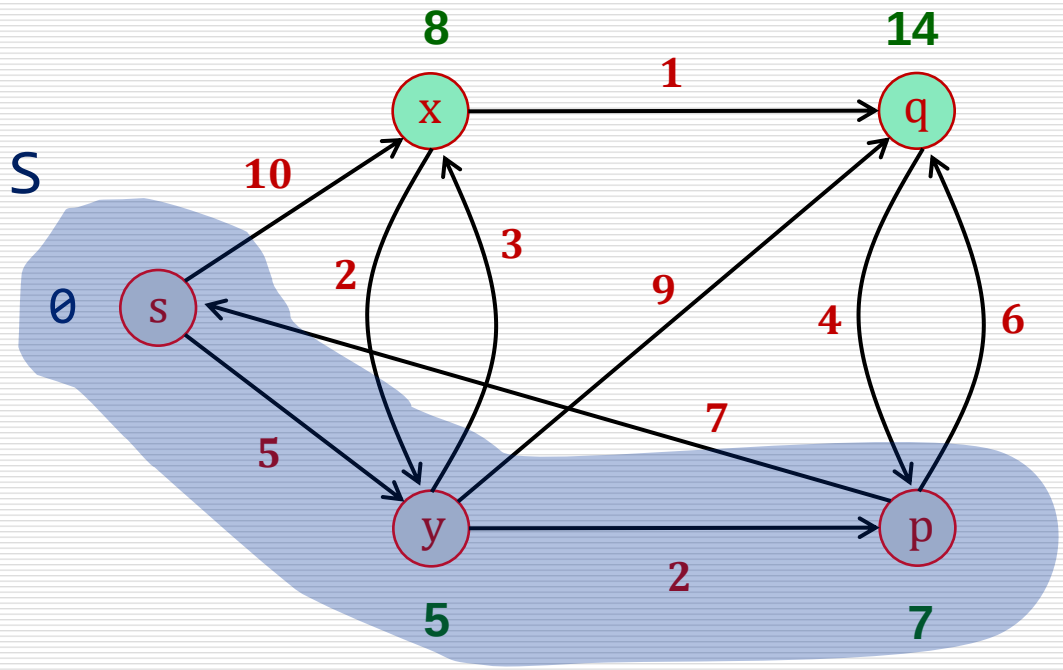



```
Q = {x,q}
while Q ≠ ∅ :
   u ← extract-min(Q);
   S ← S ∪ {u}
   για κάθε κόμβο v∈N(u)
      Update(u,v,c)
```

d(s)=0	d(x)=8	d(y)=5	d(p)=7	d(q)=14
prev(s)=NIL	prev(x)=y	prev(y)=s	prev(p)=y	prev(q)=y

3ο παράδειγμα Dijkstra

- Παράδειγμα

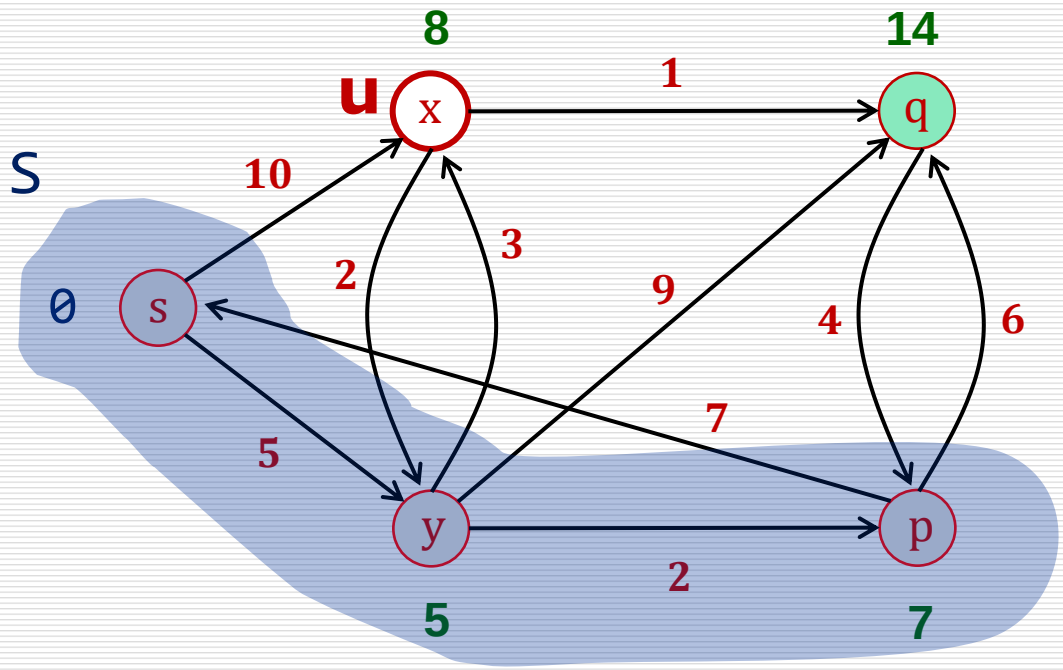


d(s)=0	d(x)=8	d(y)=5	d(p)=7	d(q)=14
prev(s)=NIL	prev(x)=y	prev(y)=s	prev(p)=y	prev(q)=y

3ο παράδειγμα Dijkstra

- Παράδειγμα

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

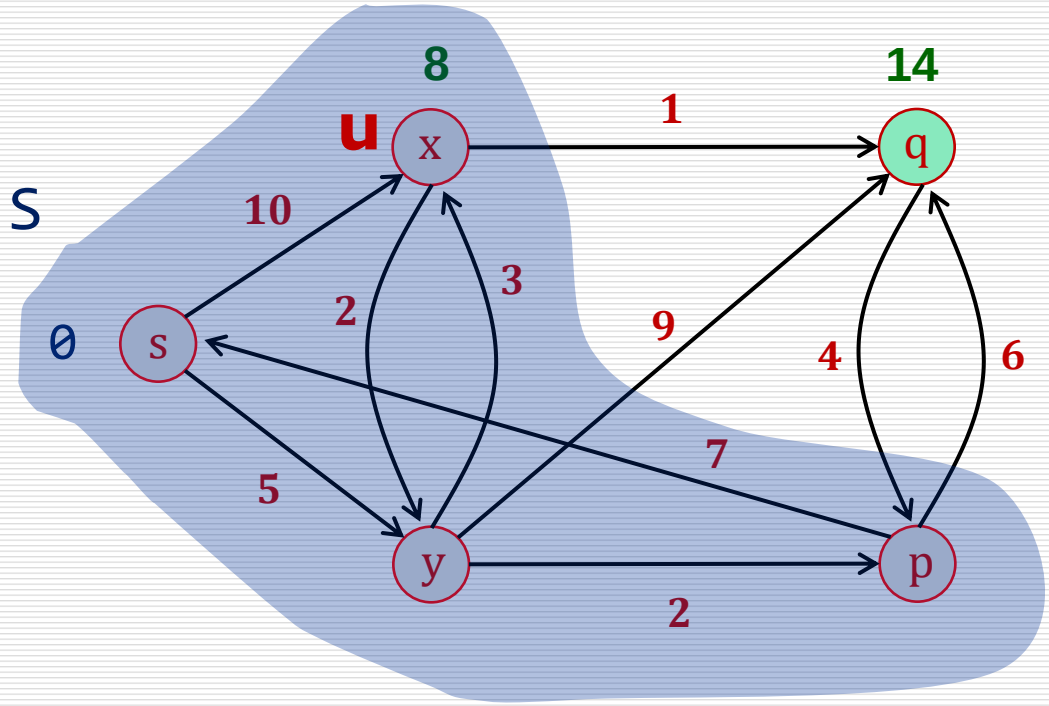


d(s)=0	d(x)=8	d(y)=5	d(p)=7	d(q)=14
prev(s)=NIL	prev(x)=y	prev(y)=s	prev(p)=y	prev(q)=y

3ο παράδειγμα Dijkstra

Παράδειγμα

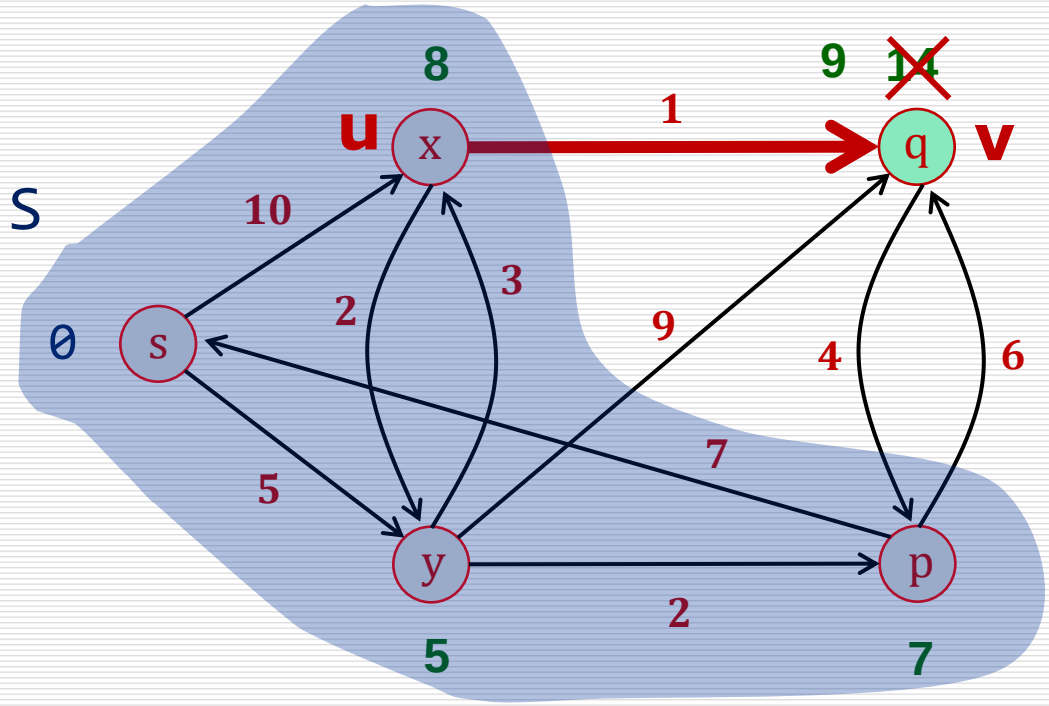
```
Q = {q}
while Q ≠ ∅ :
    u ← extract-min(Q);
    S ← S ∪ {u}
```



d(s)=0	d(x)=8	d(y)=5	d(p)=7	d(q)=14
prev(s)=NIL	prev(x)=y	prev(y)=s	prev(p)=y	prev(q)=y

3ο παράδειγμα Dijkstra

Παράδειγμα



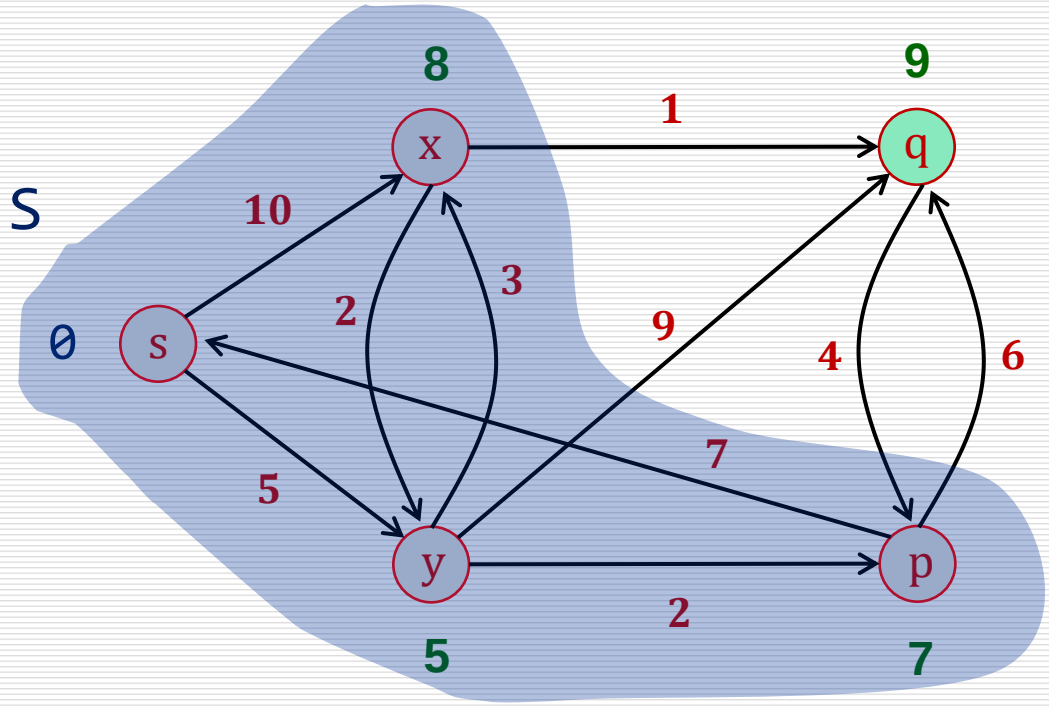

```
Q = {q}
while Q ≠ ∅ :
    u ← extract-min(Q);
    S ← S ∪ {u}
    για κάθε κόμβο v∈N(u)
        Update(u,v,c)
```

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

- Παράδειγμα

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

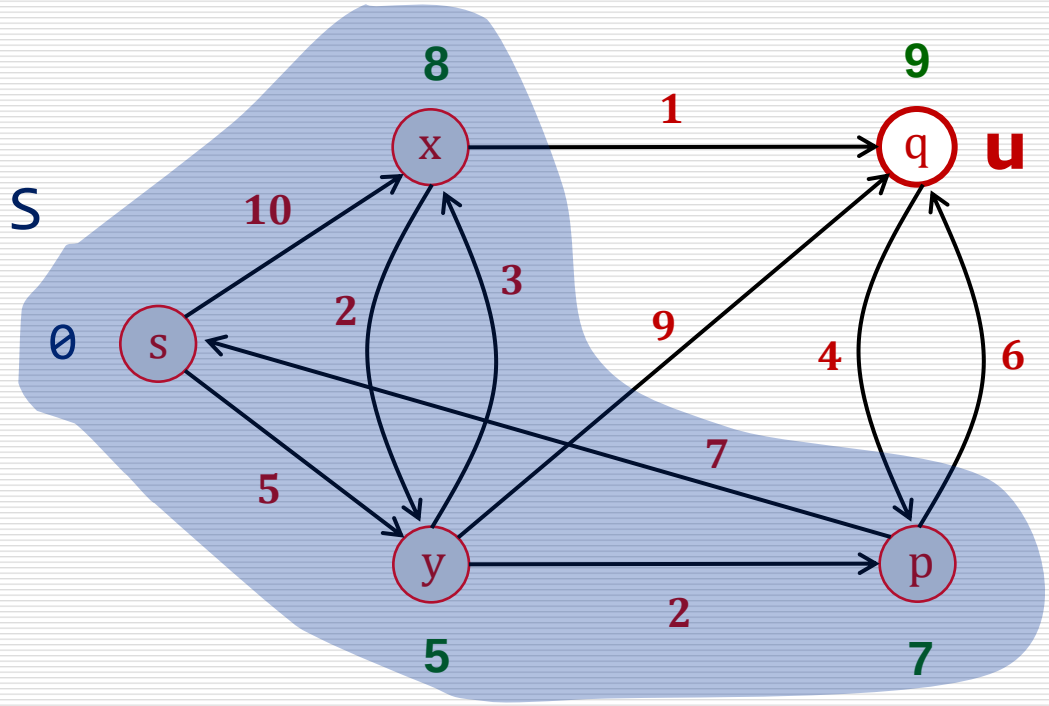


d(s)=0	d(x)=8	d(y)=5	d(p)=7	d(q)=9
prev(s)=NIL	prev(x)=y	prev(y)=s	prev(p)=y	prev(q)=x

3ο παράδειγμα Dijkstra

- Παράδειγμα

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

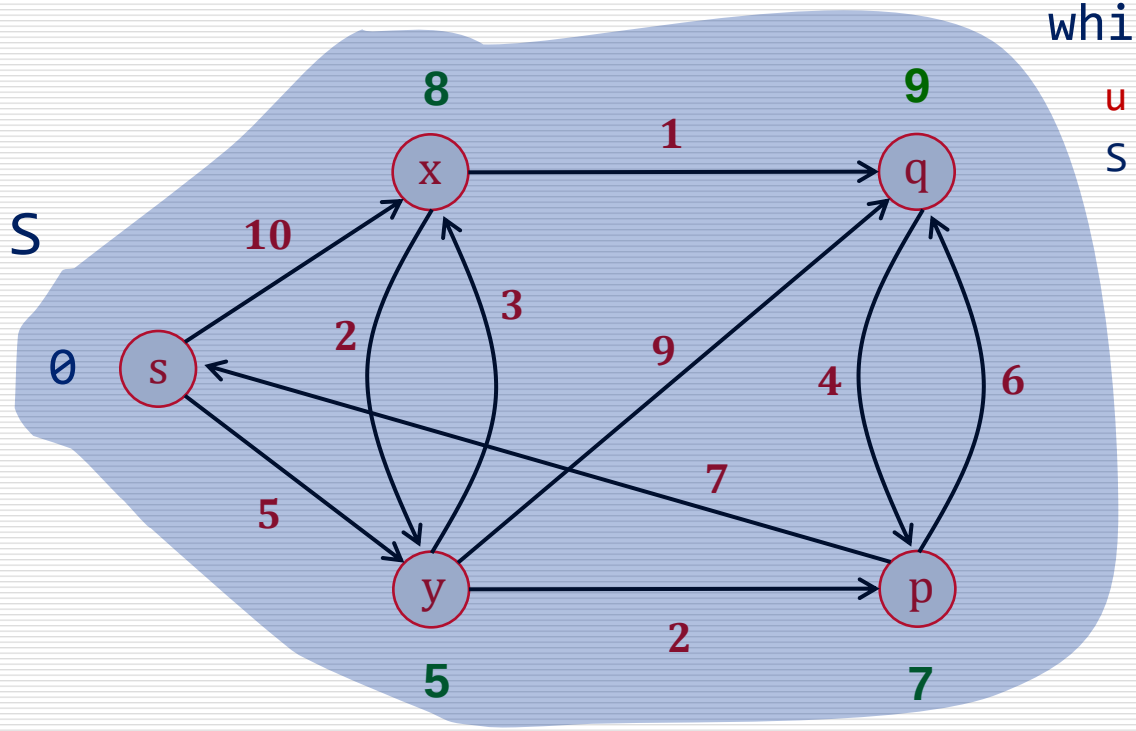


d(s)=0	d(x)=8	d(y)=5	d(p)=7	d(q)=9
prev(s)=NIL	prev(x)=y	prev(y)=s	prev(p)=y	prev(q)=x

3ο παράδειγμα Dijkstra

- Παράδειγμα

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

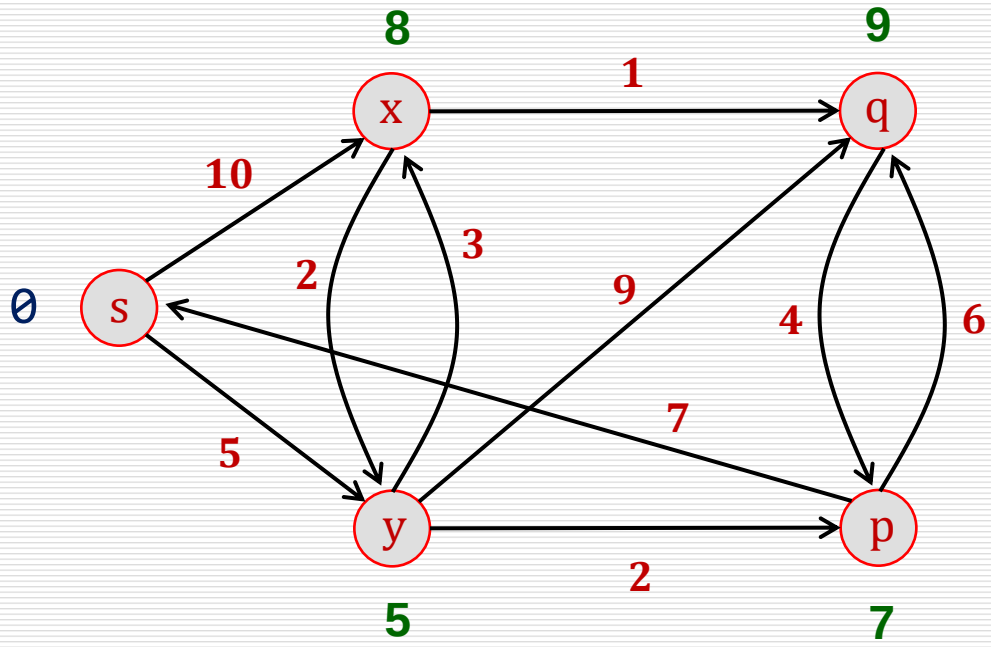


d(s)=0	d(x)=8	d(y)=5	d(p)=7	d(q)=9
prev(s)=NIL	prev(x)=y	prev(y)=s	prev(p)=y	prev(q)=x

3ο παράδειγμα Dijkstra

- Παράδειγμα

```
Q = {}
while Q ≠ ∅:
    return d(), prev()
```



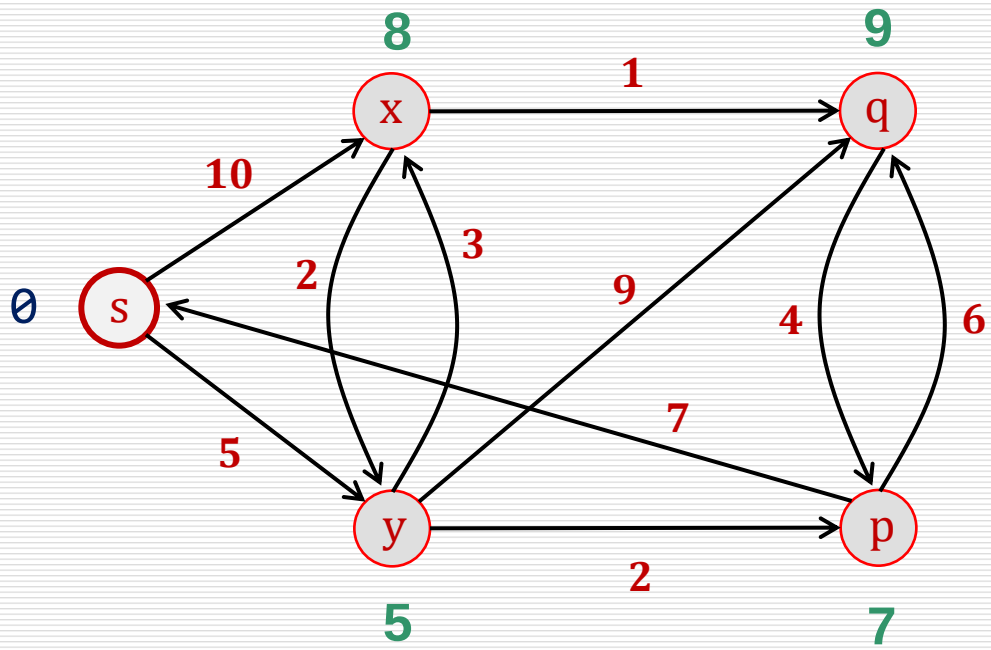
d(s)=0	d(x)=8	d(y)=5	d(p)=7	d(q)=9
prev(s)=NIL	prev(x)=y	prev(y)=s	prev(p)=y	prev(q)=x

3ο παράδειγμα Dijkstra

Παράδειγμα

Έξοδος Αλγόριθμου

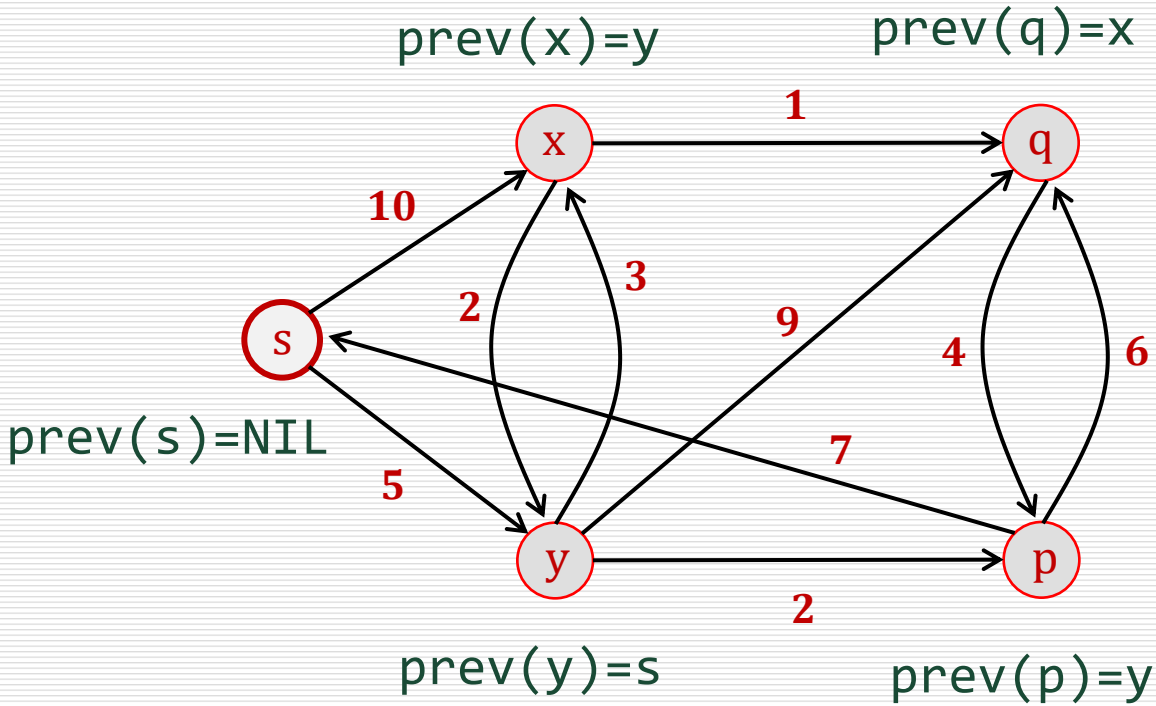
Κόστος ελάχιστων διαδρομών



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

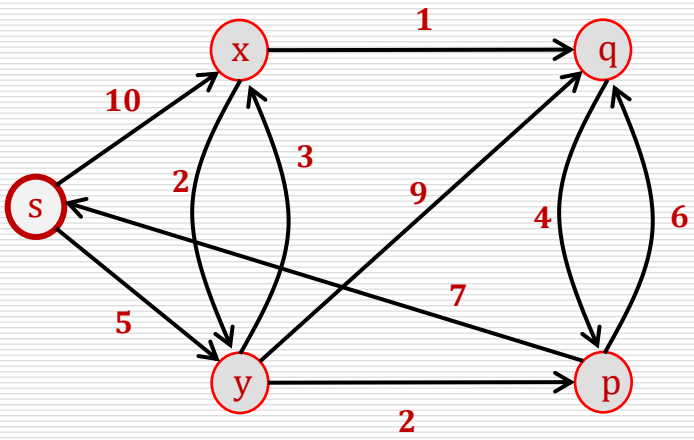


Έξοδος Αλγόριθμου

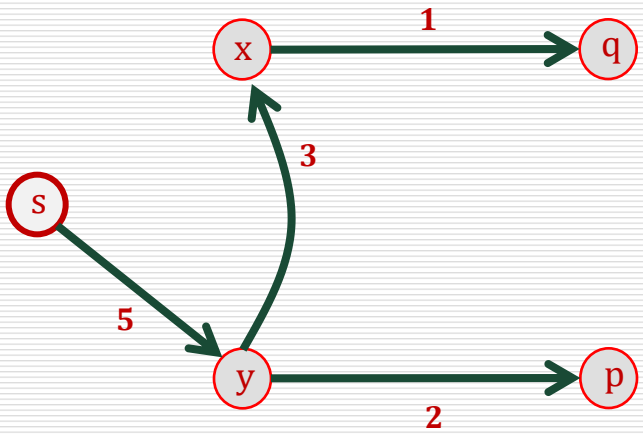
Προηγούμενοι κόμβοι στις ελάχιστες διαδρομές

3ο παράδειγμα Dijkstra

- Παράδειγμα



Δένδρο Ελάχιστων Διαδρομών

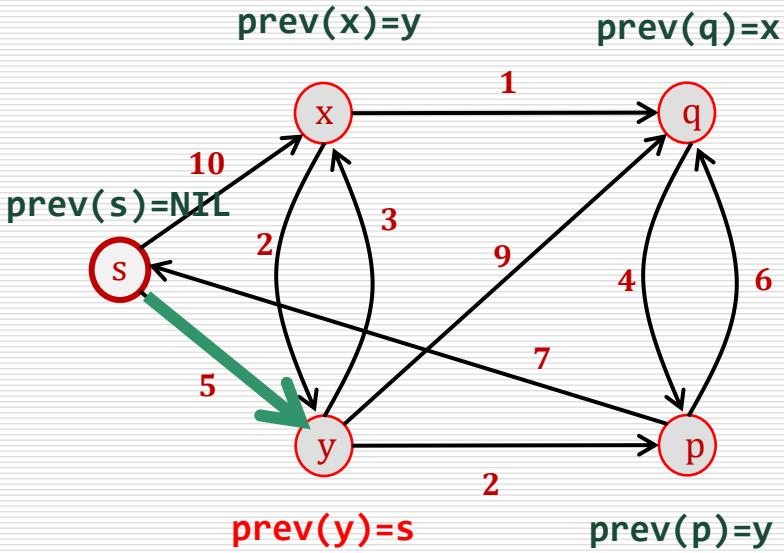


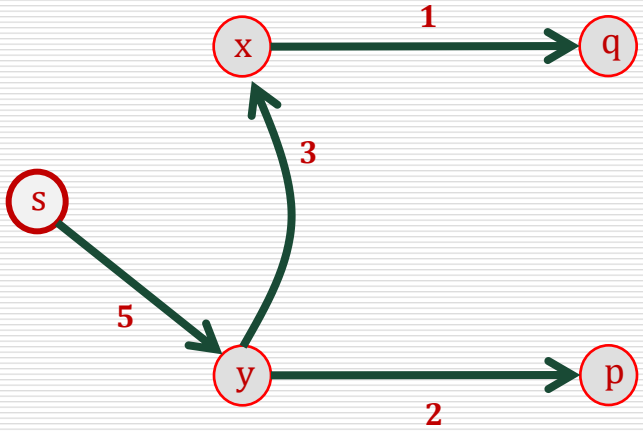
3ο παράδειγμα Dijkstra

- Παράδειγμα

Δένδρο Ελάχιστων Διαδρομών

ΚΑΤΑΣΚΕΥΗ



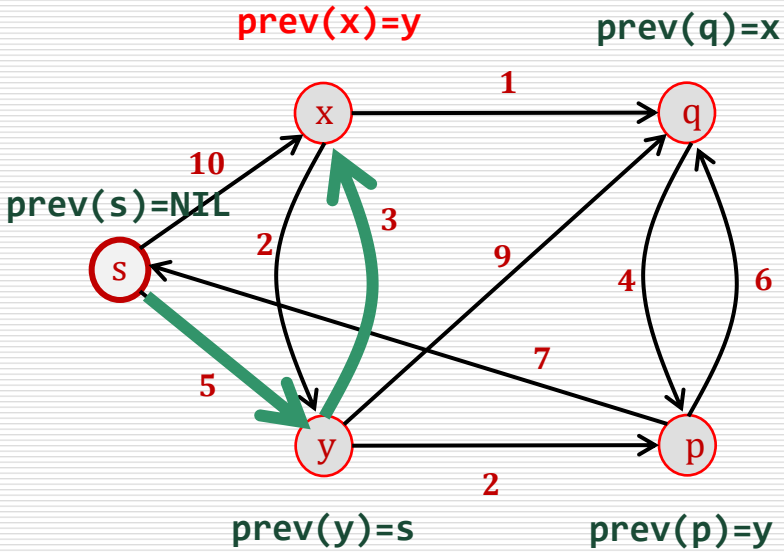


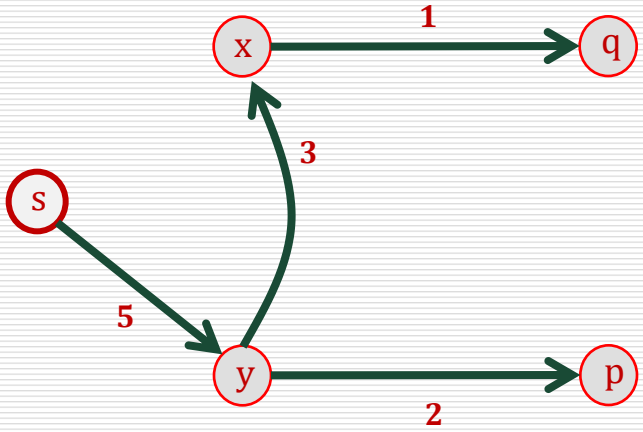
3ο παράδειγμα Dijkstra

- Παράδειγμα

Δένδρο Ελάχιστων Διαδρομών

ΚΑΤΑΣΚΕΥΗ



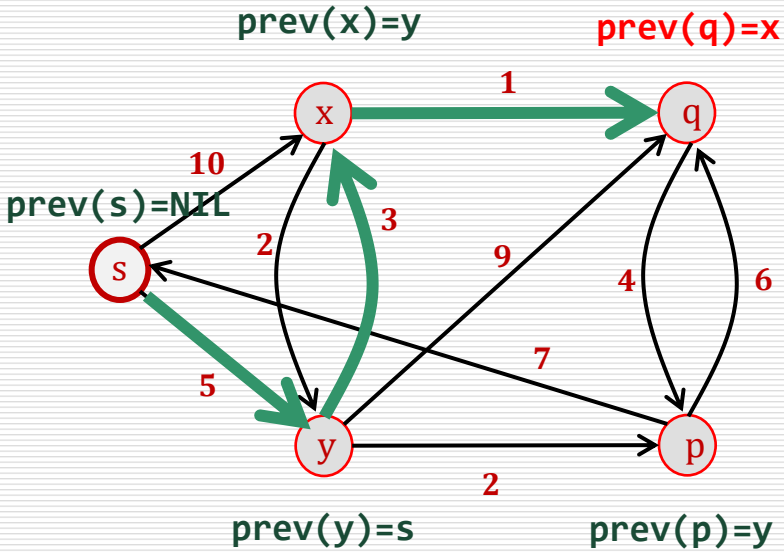


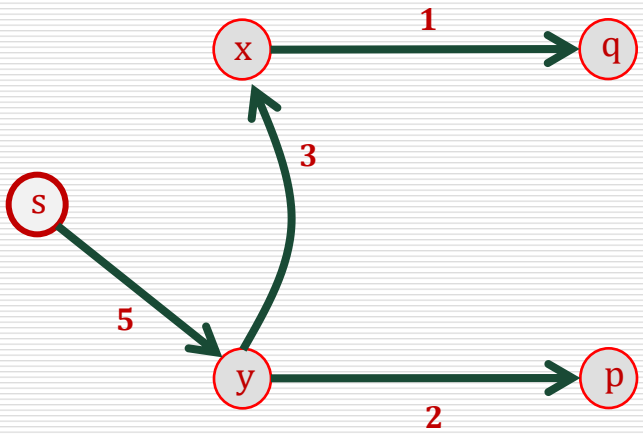
3ο παράδειγμα Dijkstra

- Παράδειγμα

Δένδρο Ελάχιστων Διαδρομών

ΚΑΤΑΣΚΕΥΗ



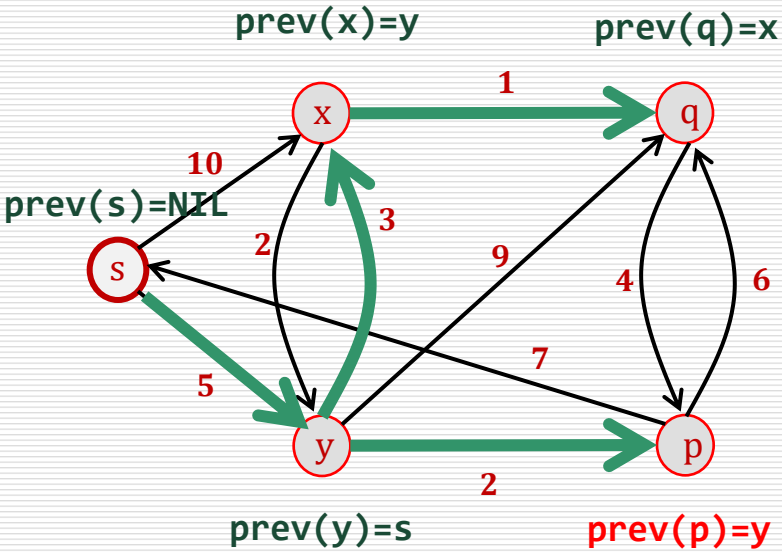


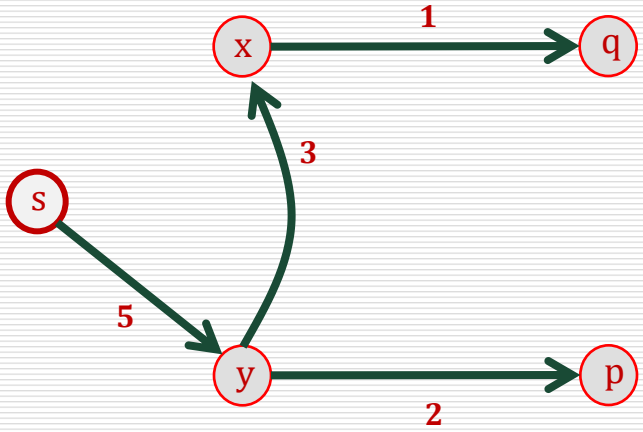
3ο παράδειγμα Dijkstra

- Παράδειγμα

Δένδρο Ελάχιστων Διαδρομών

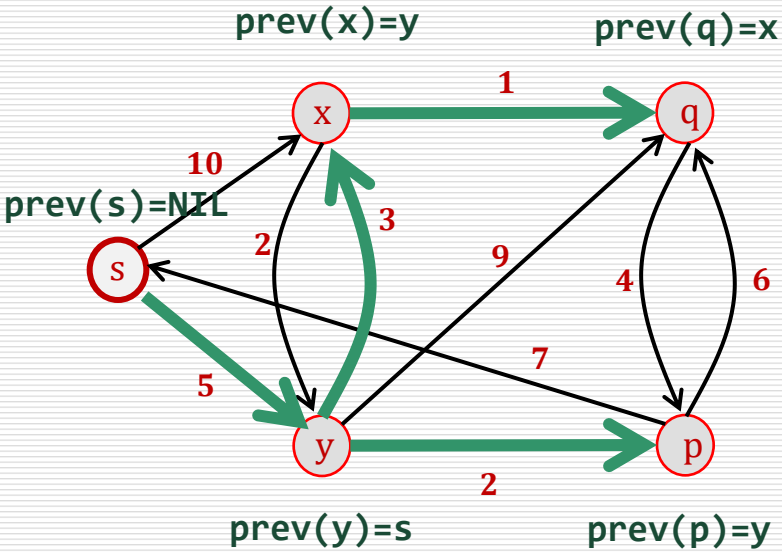
ΚΑΤΑΣΚΕΥΗ





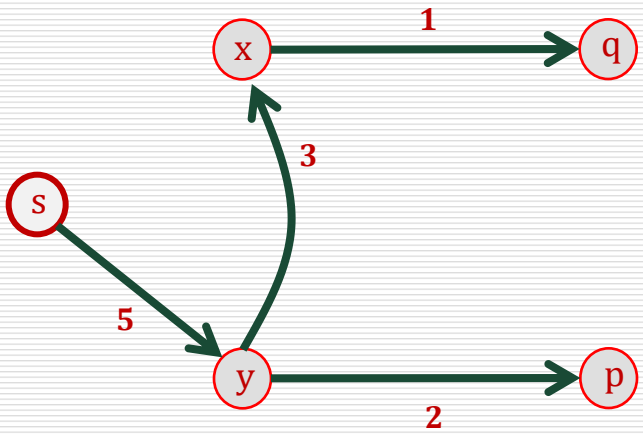
3ο παράδειγμα Dijkstra

Παράδειγμα



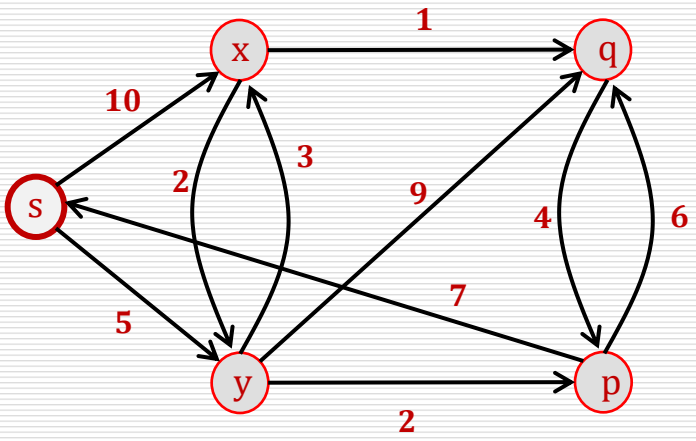
Δένδρο Ελάχιστων Διαδρομών

ΚΑΤΑΣΚΕΥΗ



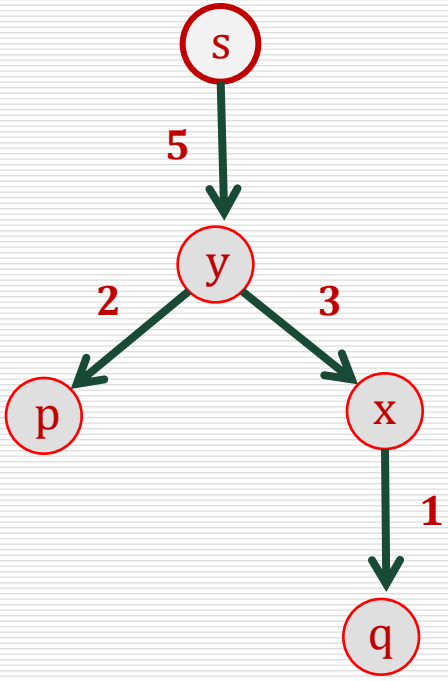
3ο παράδειγμα Dijkstra

- Παράδειγμα



Δένδρο Ελάχιστων Διαδρομών

ΚΑΤΑΣΚΕΥΗ



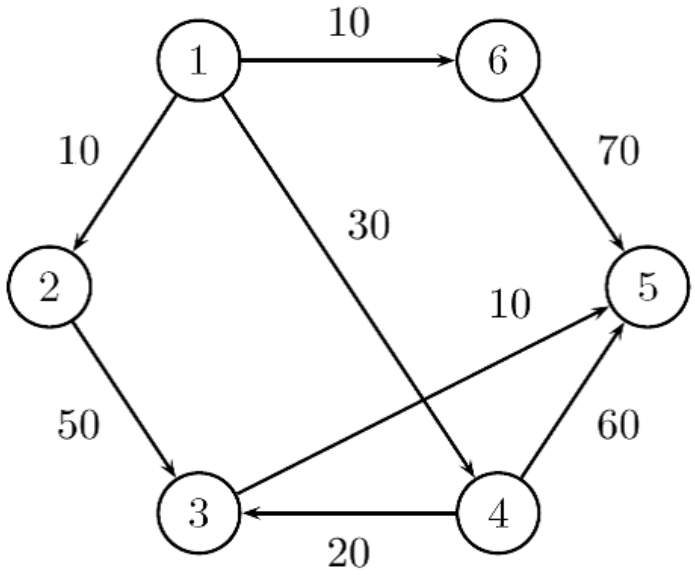
Συντομότερες διαδρομές (Dijkstra)

```
procedure Dijkstra;
begin (* Αρχικοποίηση *)
    S := {1}; for i:=2 to n do begin D[i]:=cost[1,i]; P[i]:=1 end;
    for i:=2 to n-1 do
   begin
      select w from V − S such that D[w] is minimum;
      S := S + {w};
      for all v in V − S do
         if D[v] > D[w] + C[w,v] then
            P[v] := w;
            D[v] := D[w]+C[w,v]
    end
end
```

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

Παράδειγμα Dijkstra

```
procedure Dijkstra;
begin (* Αρχικοποίηση *)
   S := {1}; for i:=2 to n do begin D[i]:=cost[1,i]; P[i]:=1 end;
   for i:=2 to n-1 do
  begin
     select w from V − S such that D[w] is minimum;
     S := S + {w};
     for all v in V − S do
        if D[v] > D[w] + C[w,v] then
           P[v] := w;
           D[v] := D[w]+C[w,v]
   end
end
```



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										

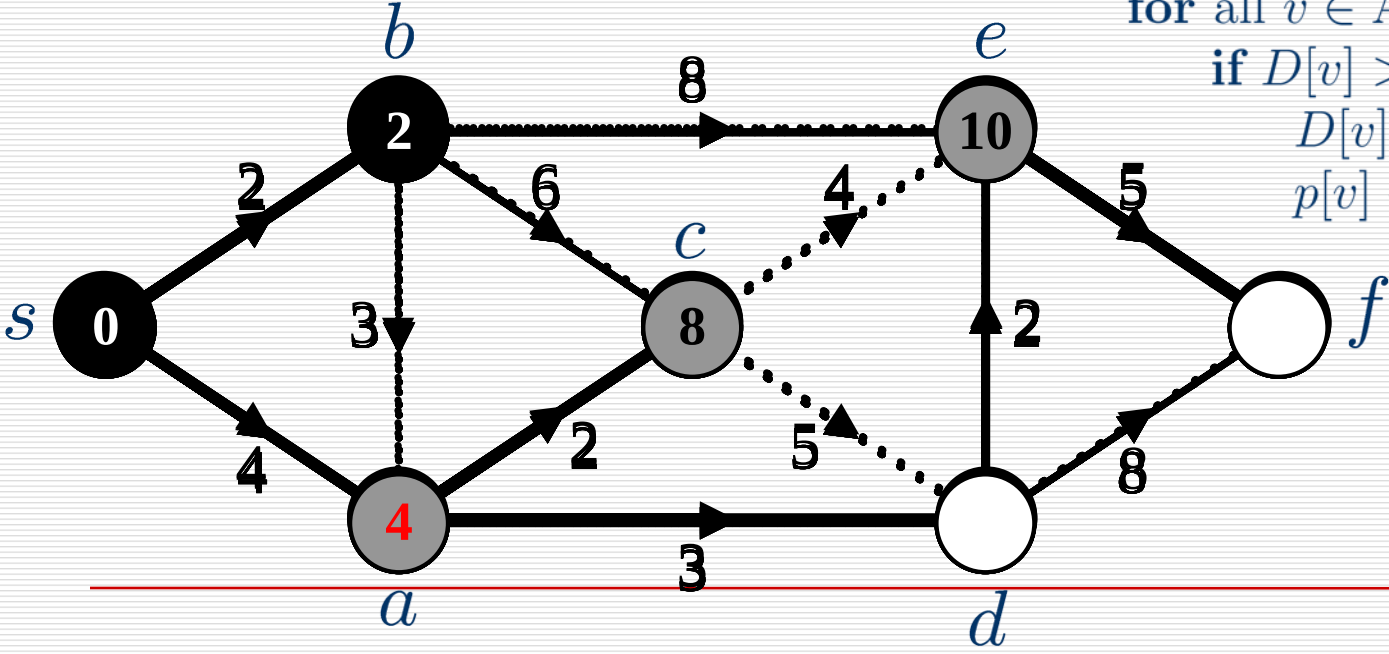
Αλγόριθμος Dijkstra

- Γράφημα με θετικά κόστη στις ακμές.
  Αρχική κορυφή s.
- ∀u, διατηρεί D(u) εκτίμηση απόστασης από s.
  - Αρχικά D(s) = 0, D(u) = ∞ για κάθε u ≠ s.
  - Μήκος συντ. s – u μονοπ. που έχει «ανακαλύψει» ο αλγόριθμος.
  - Κορυφές στο S έχουν D(u) = d(s, u) (εκτίμηση απόστασης είναι οριστική).
- Κάθε επανάληψη επιλέγει διαθέσιμη (εκτός S) κορυφή u με **ελάχιστη** εκτίμηση απόστασης.
  - D(u) οριστικοποιείται και προστίθεται στο S (μη διαθέσιμη)
  - Ενημέρωση εκτιμήσεων απόστασης για γείτονες της u.

$\text{Dijkstra}(G(V, E, c), s)$
for all $u \in V$ **do**
$D[u] \leftarrow \infty;\ p[u] \leftarrow \text{NULL};$
$D[s] \leftarrow 0;\ S \leftarrow \emptyset;$
while $|S| < |V|$ **do**
$u \notin S : D[u] = \min_{v \notin S}\{D[v]\};$
$S \leftarrow S \cup \{u\};$
for all $v \in \text{AdjList}[u]$ **do**
if $D[v] > D[u] + c(u, v)$ **then**
$D[v] \leftarrow D[u] + c(u, v);$
$p[v] \leftarrow u;$

Αλγόριθμος Dijkstra: Επανάληψη

```
Dijkstra(G(V, E, c), s)
    for all u ∈ V do
        D[u] ← ∞;  p[u] ← NULL;
    D[s] ← 0;  S ← ∅;
    while |S| < |V| do
        u ∉ S : D[u] = min_{v∉S}{D[v]};
        S ← S ∪ {u};
        for all v ∈ AdjList[u] do
            if D[v] > D[u] + c(u, v) then
                D[v] ← D[u] + c(u, v);
                p[v] ← u;
```



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 μονοπατιών. Το δένδρο αρχικοποιείται με τον αρχικό κόμβο *s*.
- Σε κάθε επανάληψη επιλέγεται ο κόμβος ***w*** με την **ελάχιστη** εκτίμηση απόστασης από τον *s*.
- Η απόδειξη στηρίζεται σε δύο αναλλοίωτες συνθήκες βρόχου.

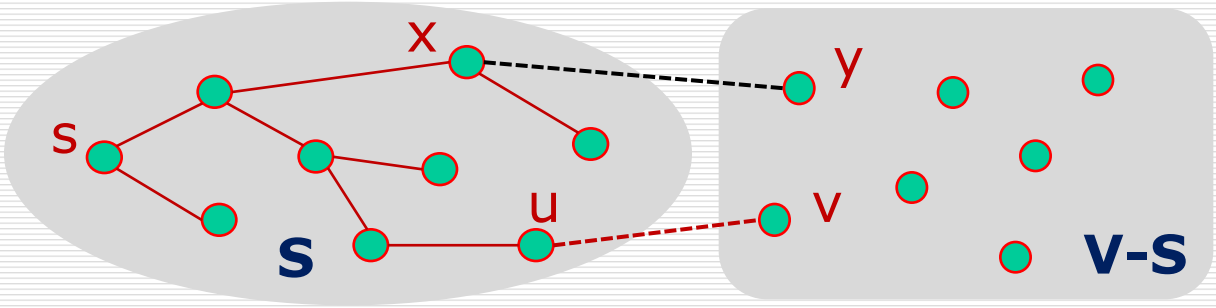
Ορθότητα αλγορίθμου Dijkstra

- 1η αναλλοίωτη βρόχου: *πριν από κάθε επανάληψη του εξωτερικού βρόχου,* ***εκτίμηση απόστασης*** *(κάθε) κόμβου* ***D(v)*** *= κόστος συντομότερης διαδρομής από τον s στον v, μεταξύ όλων των διαδρομών που περνούν μόνο από κόμβους του συνόλου S (δηλ. του μέχρι στιγμής δένδρου).*
- 2η αναλλοίωτη βρόχου: *πριν από κάθε επανάληψη του εξωτερικού βρόχου, για τον κόμβο w που ανήκει στο V-S και έχει* ***ελάχιστη*** *(μεταξύ κόμβων του V-S) εκτίμηση απόστασης από τον s, η απόσταση αυτή είναι η* ***τελική*** *(ελάχιστη) του w από τον s.*
- Οι δύο αναλλοίωτες αποδεικνύονται **μαζί**, χρησιμοποιώντας επαγωγή.

Ορθότητα αλγορίθμου Dijkstra

Λήμμα (βήμα επαγωγής)

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 γνωρίζουμε ελάχιστη απόσταση $D[u] = d(s, u)$ $\forall u \in S$.



Εάν (u, v) είναι η ακμή που ελαχιστοποιεί την ποσότητα

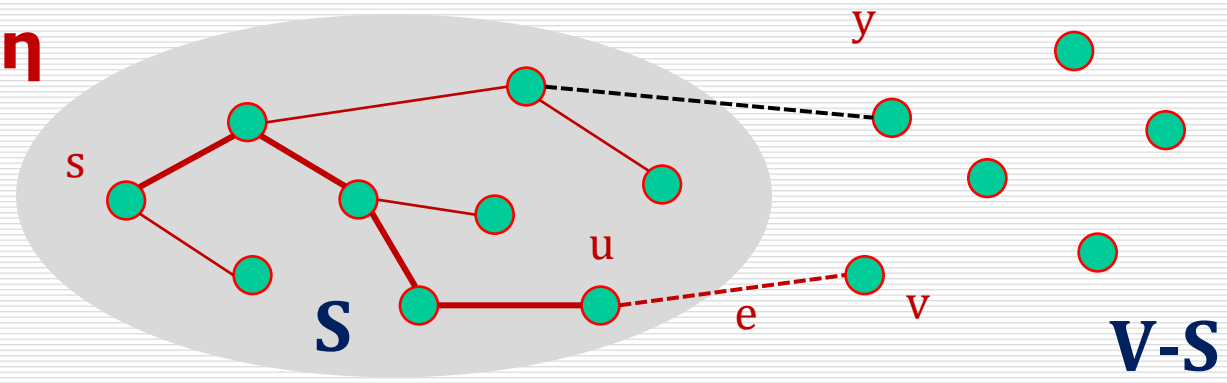
$$\mathbf{D[v] = d(s, u) + c(u, v)}$$

μεταξύ όλων των ακμών (x, y) με $x \in S$ και $y \in V\text{-}S$, τότε

P = (s, …, u, v) είναι ε.δ. $s \rightsquigarrow v$.

Ορθότητα αλγορίθμου Dijkstra

Απόδειξη



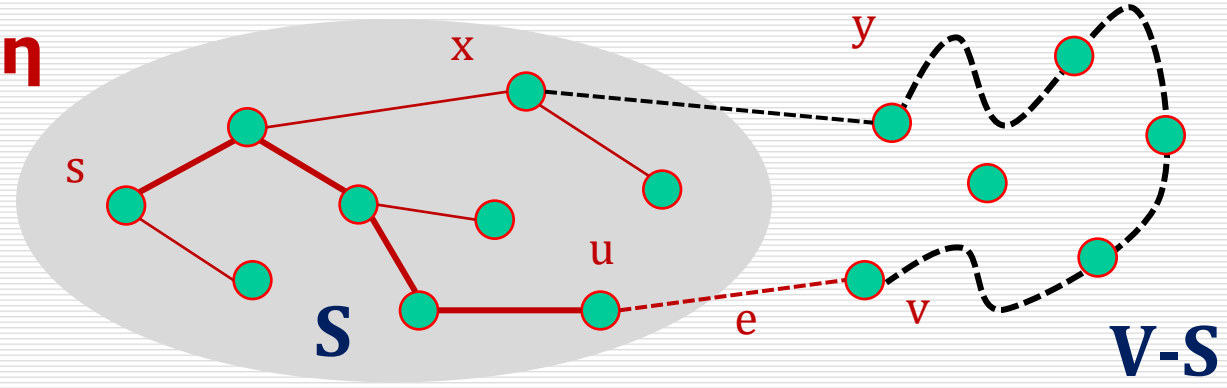
Έστω e = (u, v) και έστω (s, …, u) η ελάχιστη διαδρομή από τον s στον u.

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

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

Ορθότητα αλγορίθμου Dijkstra

Απόδειξη



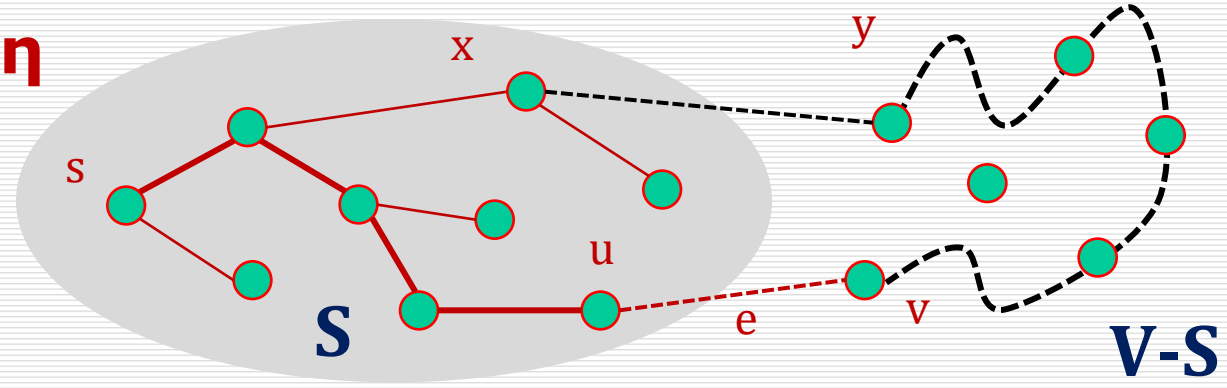
Έστω Q = (s, …, x, y, …, v) μια ελάχιστη διαδρομή από τον s στον v, και

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

Θα δείξουμε ότι **c(P) ≤ c(Q)**

Ορθότητα αλγορίθμου Dijkstra

Απόδειξη

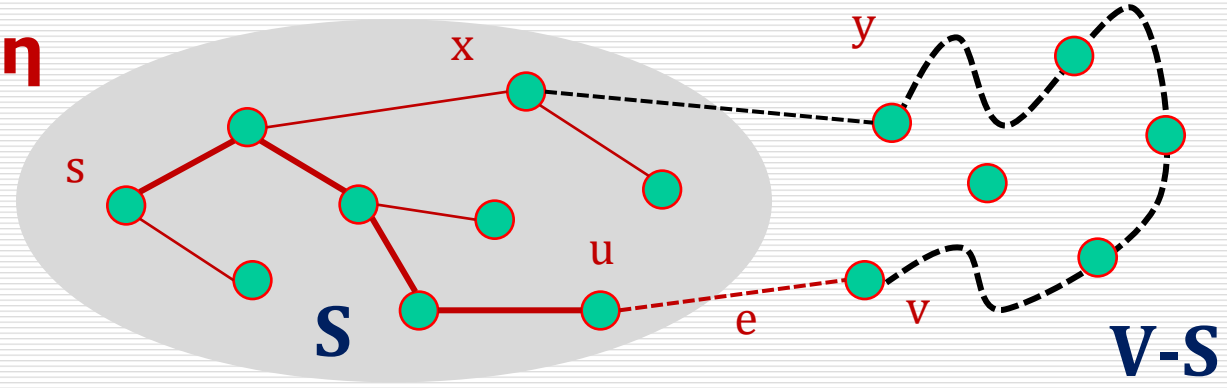


Από (1) και από επιλογή της ακμής e = (u, v), έχουμε:

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

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

- *Άσκηση*: βρείτε αντιπαράδειγμα.

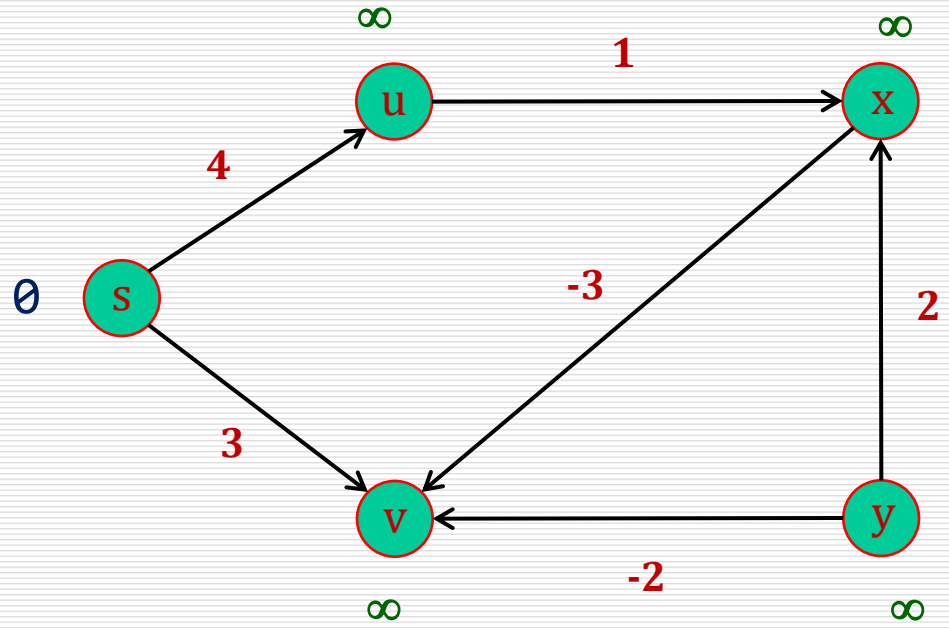
Αλγόριθμος Bellman-Ford

```
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

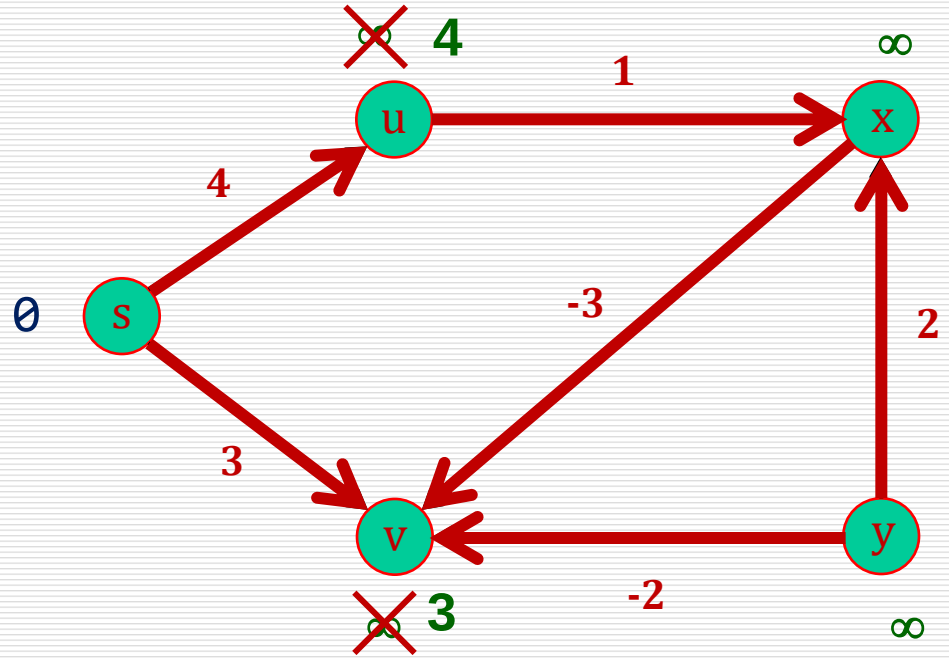
Update στις
6 ακμές
4 φορές

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

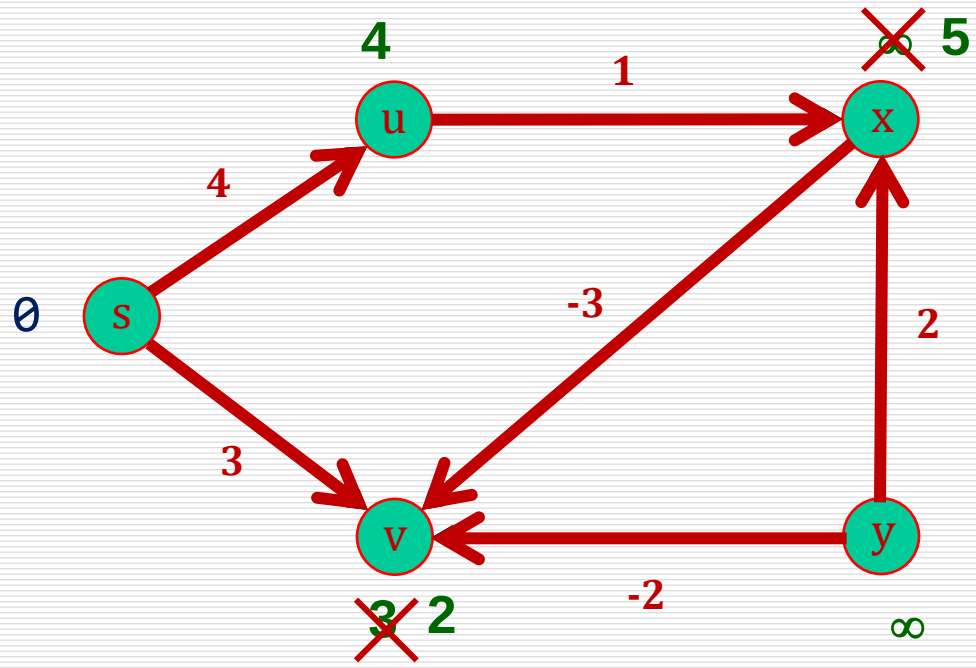
1η Επανάληψη

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

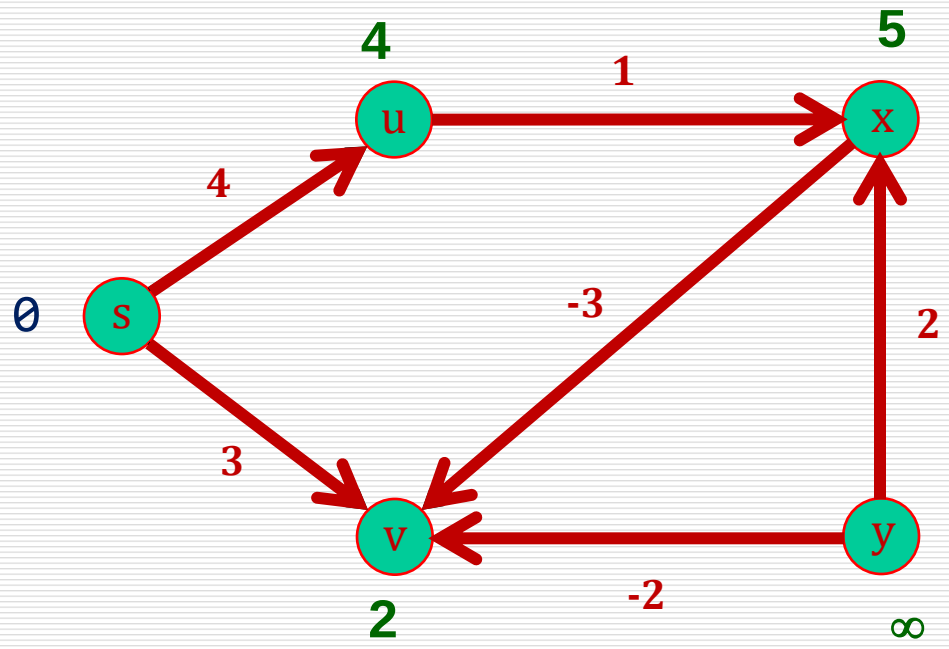
2η Επανάληψη

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

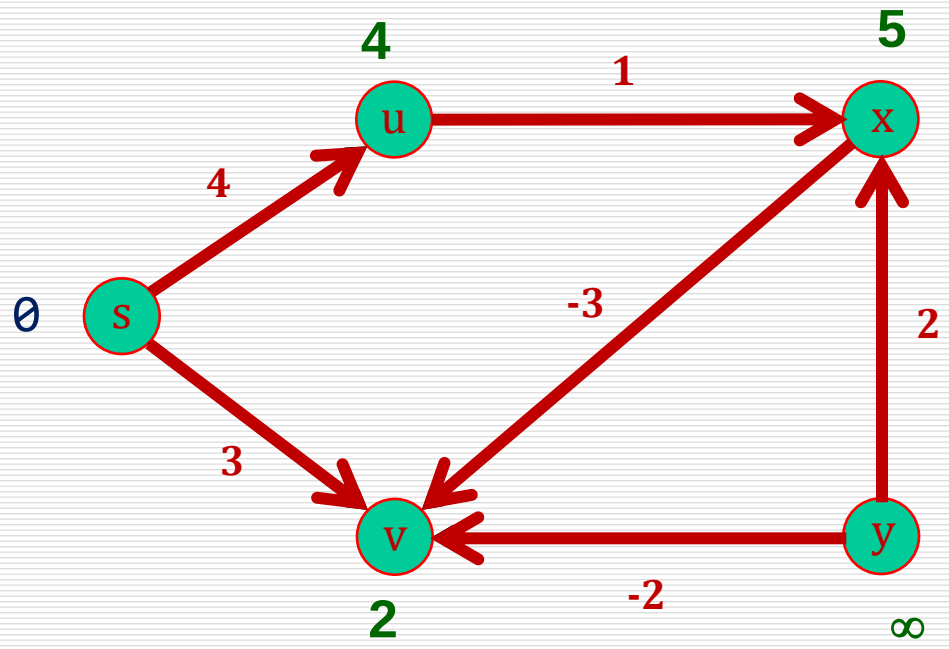
Καμία Αλλαγή
Γιατί ?

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

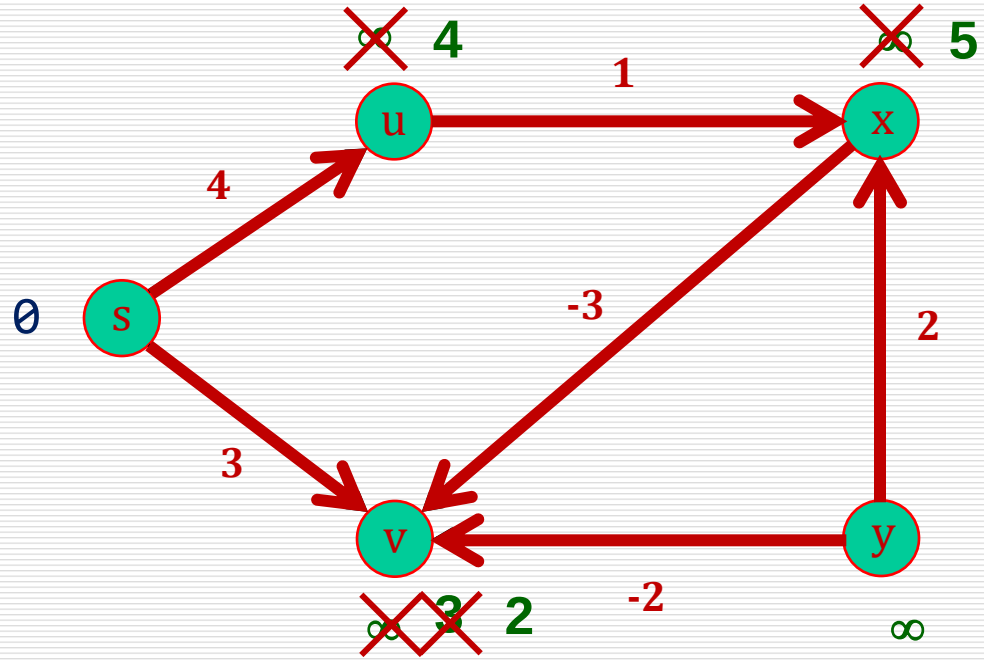
Καμία Αλλαγή

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

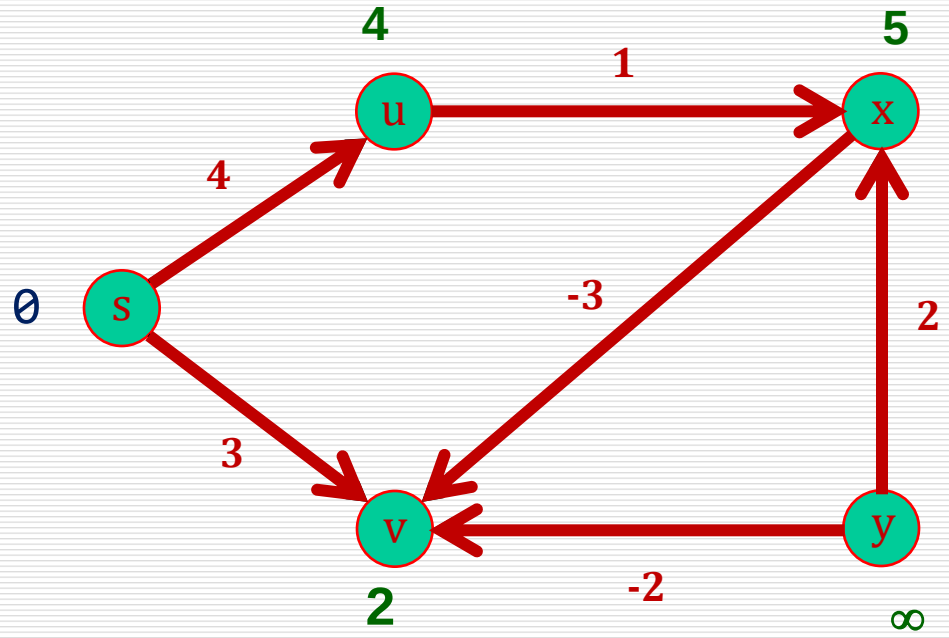
1η Επανάληψη

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

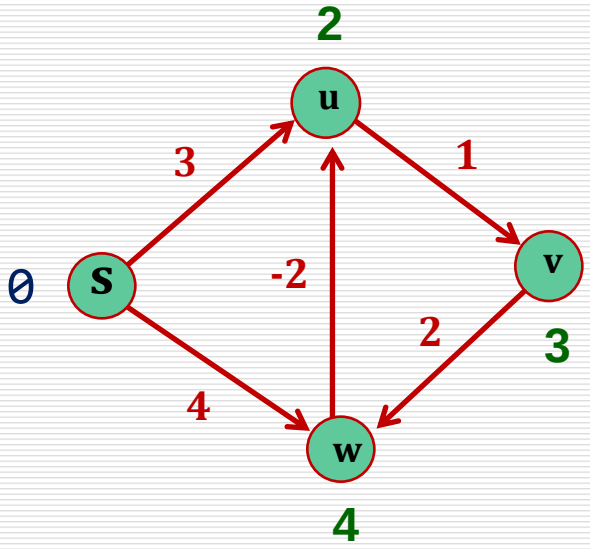
Καμία Αλλαγή
Γιατί ?

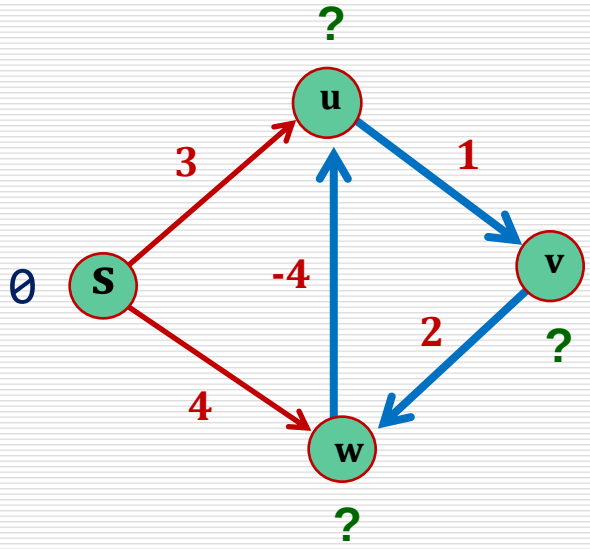
S2: (s,u), (u,x), (y,x), (s,v), (x,v), (y,v)

✓ ✓ ✓ ✓ ✓ ✓

Αλγόριθμος Bellman-Ford

- **Αρνητικοί Κύκλοι**



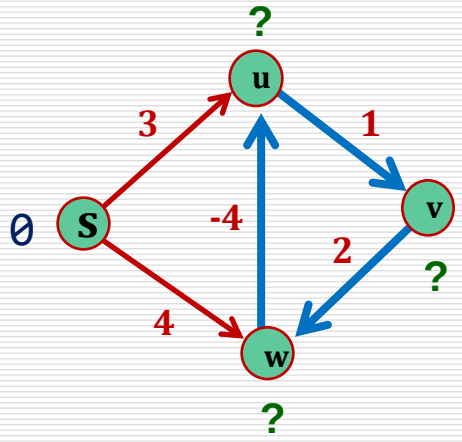


Εάν υπάρχει αρνητικός κύκλος, δεν έχει νόημα να αναζητούμε συντομότερες διαδρομές !!!

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(.)
```



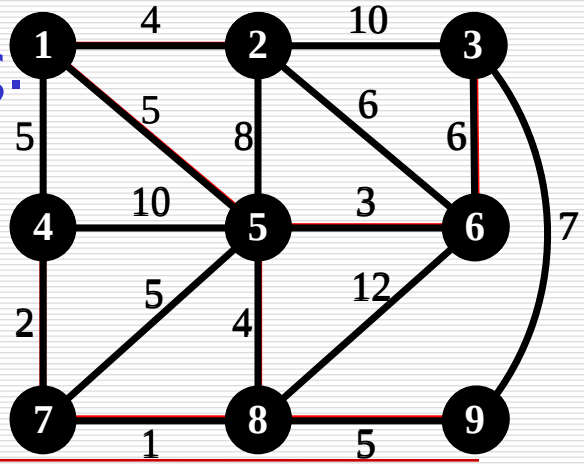
Αλγόριθμος Bellman-Ford

- **Ορθότητα:** στο τέλος της k-οστής επανάληψης έχουν υπολογιστεί σωστά όλες οι ελάχιστες διαδρομές που χρησιμοποιούν το πολύ k ακμές
(*άσκηση*: αποδείξτε το).

- **Πολυπλοκότητα**: **O(|V||E|)**

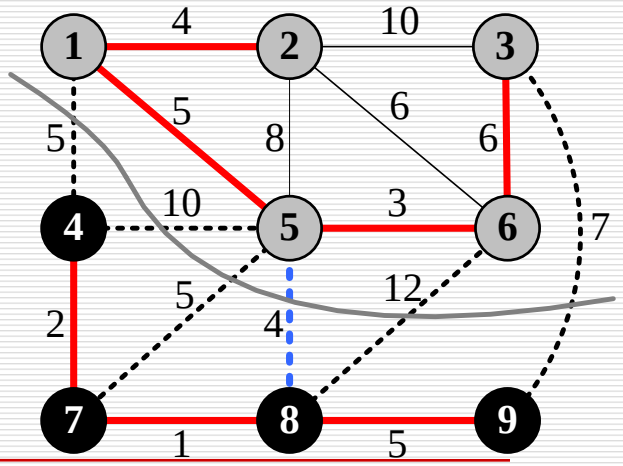
Ελάχιστο Συνδετικό Δέντρο (MST)

- Συνεκτικό G(V, E, c) με βάρη στις ακμές.
- Ζητούμενο: ελάχιστου συνολικού βάρους **συνεκτικό** υπογράφημα που καλύπτει όλες τις κορυφές.
  - Συνεκτικό (εξ' ορισμού) + ακυκλικό (ελάχιστο) ⇒ Δέντρο.
  - Minimum Spanning Tree (MST, ΕΣΔ).
- Πολλές και σημαντικές εφαρμογές.



Τομές, Σύνολα Τομής, και ΕΣΔ

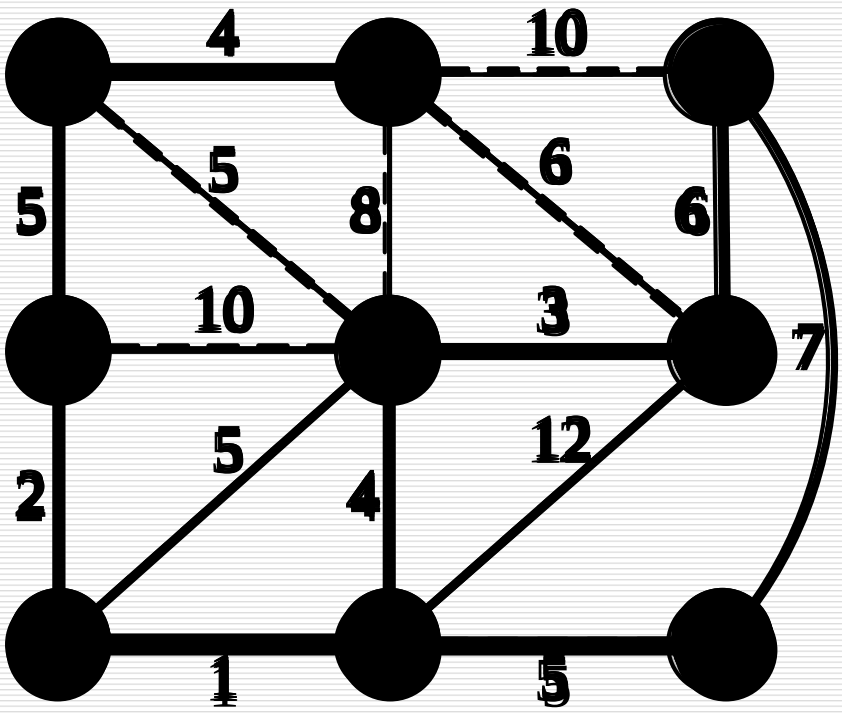
- Τομή (S, V \ S): διαμέριση κορυφών σε 2 σύνολα S, V \ S.
- Σύνολο τομής δ(S, V \ S): ακμές ένα άκρο στο S και άλλο άκρο στο V \ S.
  - δ(S, V \ S): όλες οι ακμές που **διασχίζουν** τομή (S, V \ S).
- Σύνολο ακμών E' **διασχίζει** τομή (S, V \ S) αν $E' \cap \delta(S, V \setminus S) \neq \emptyset$.
- (Ε)ΣΔ ορίζεται από σύνολο ακμών (ελάχιστου) βάρους που διασχίζει όλες τις τομές.
  - Άπληστη στρατηγική: ενόσω «αγεφύρωτη» τομή, διάσχισέ την με ακμή ελάχιστου βάρους.



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



MST-Prim$(G(V, E, c), s)$

for all $u \in V$ **do**

$D[u] \leftarrow \infty$; $p[u] \leftarrow$ NULL;

$D[s] \leftarrow 0$; $S \leftarrow \emptyset$;

while $|S| < |V|$ **do**

$u \notin S : D[u] = \min_{v \notin S}\{D[v]\}$;

$S \leftarrow S \cup \{u\}$;

for all $v \in$ AdjList$[u]$ **do**

if $v \notin S$ **and** $c(u, v) < D[v]$ **then**

$D[v] \leftarrow c(u, v)$;

$p[v] \leftarrow u$;

Αλγόριθμος Prim

- Επιλέγεται αρχικός κόμβος, έστω s. Αρχικοποίηση:

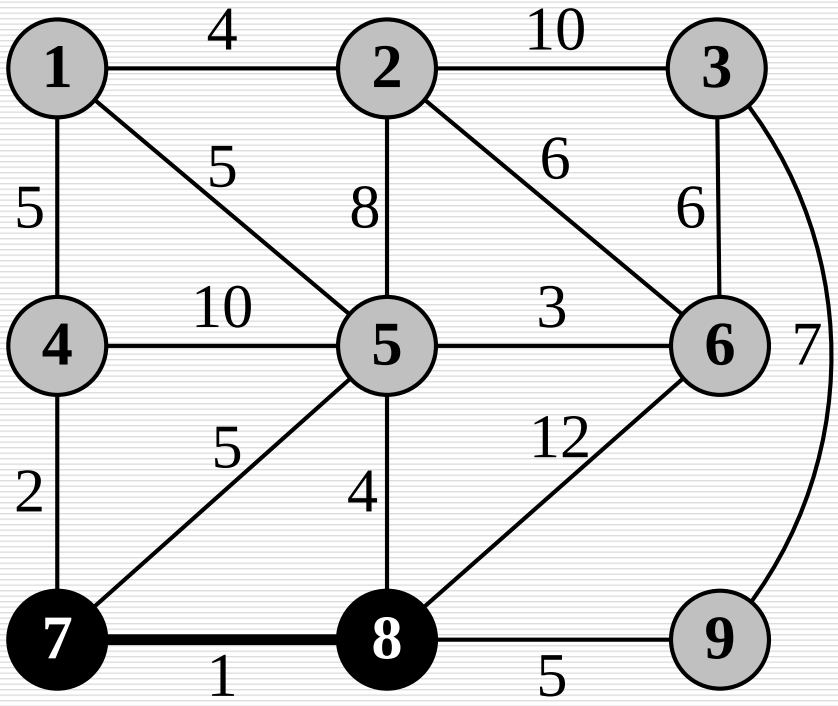
```
d(s):= 0; for each v<>s do d(v) := ∞
```

- Κάθε φορά επιλέγεται ο κόμβος, έστω w, με ελάχιστο κόστος σύνδεσης στο μέχρι στιγμής κατασκευασμένο δένδρο, και προστίθεται στο δένδρο. Ενημερώνονται οι αποστάσεις των υπόλοιπων κόμβων από το δένδρο με βάση το κόστος των ακμών (w,u_i):

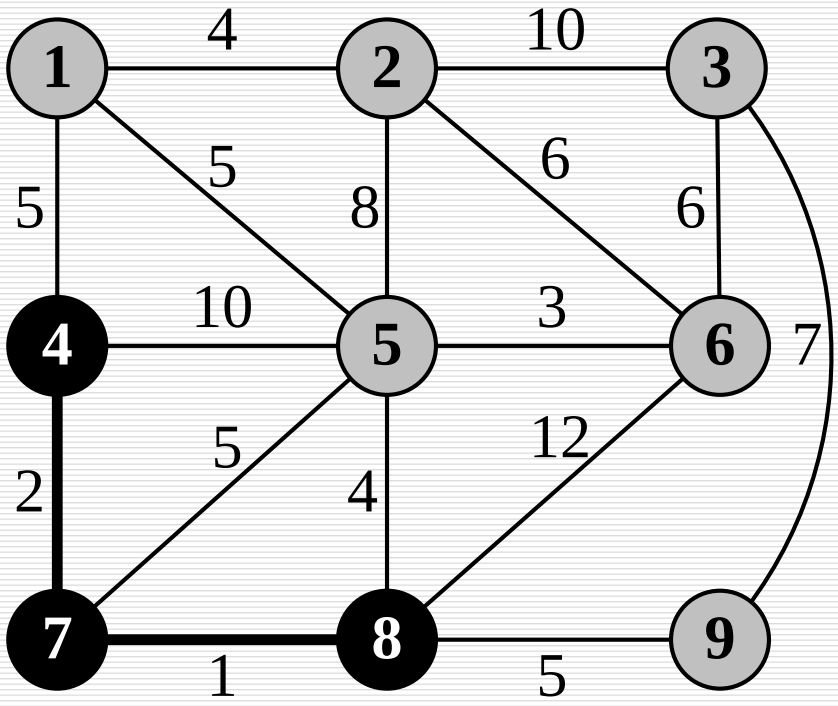
```
if c(w,u_i) < d(u_i) then
    d(u_i) := c(w,u_i)
```

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

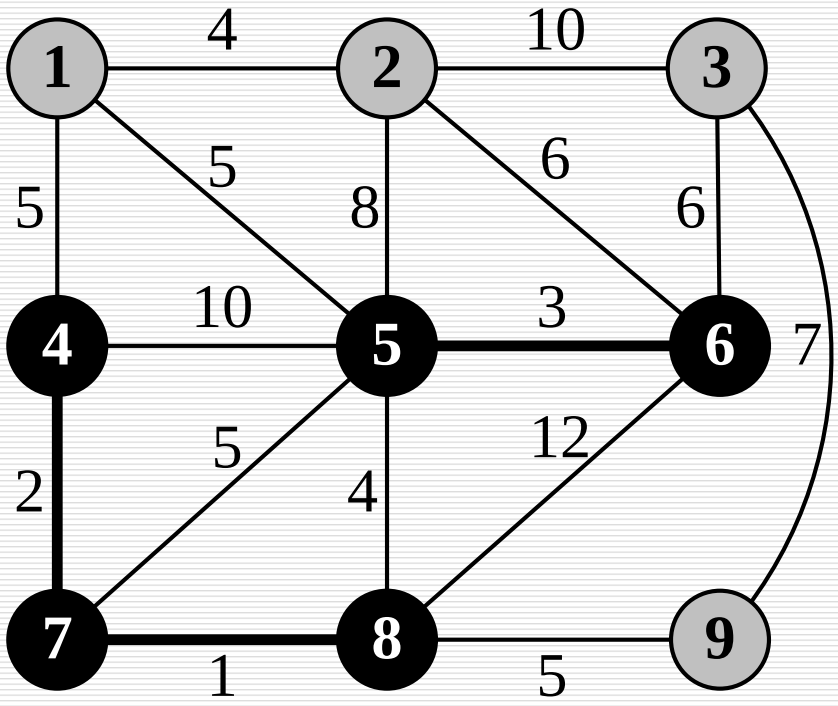
Αλγόριθμος Kruskal: παράδειγμα



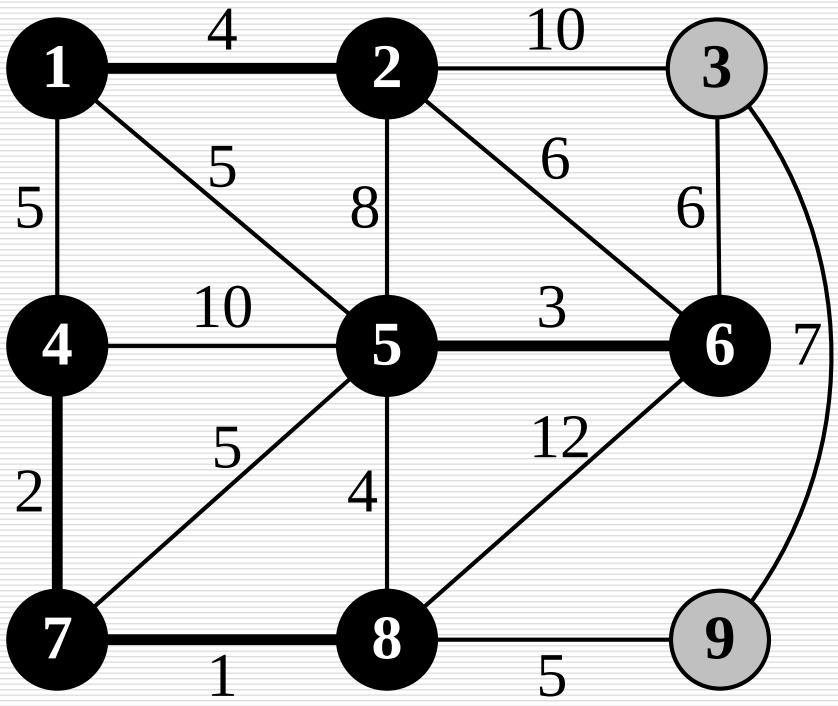
Αλγόριθμος Kruskal: παράδειγμα



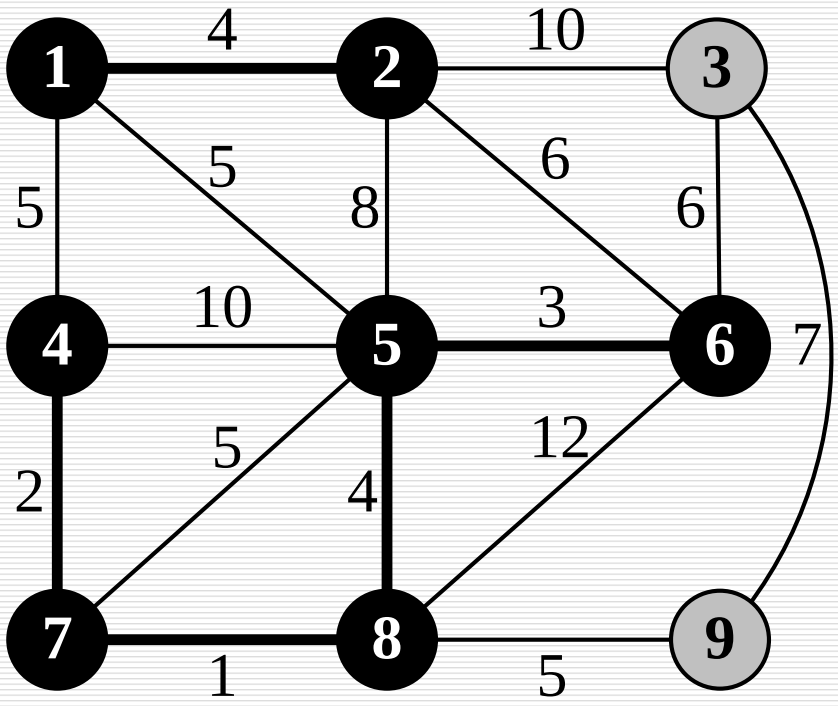
Αλγόριθμος Kruskal: παράδειγμα



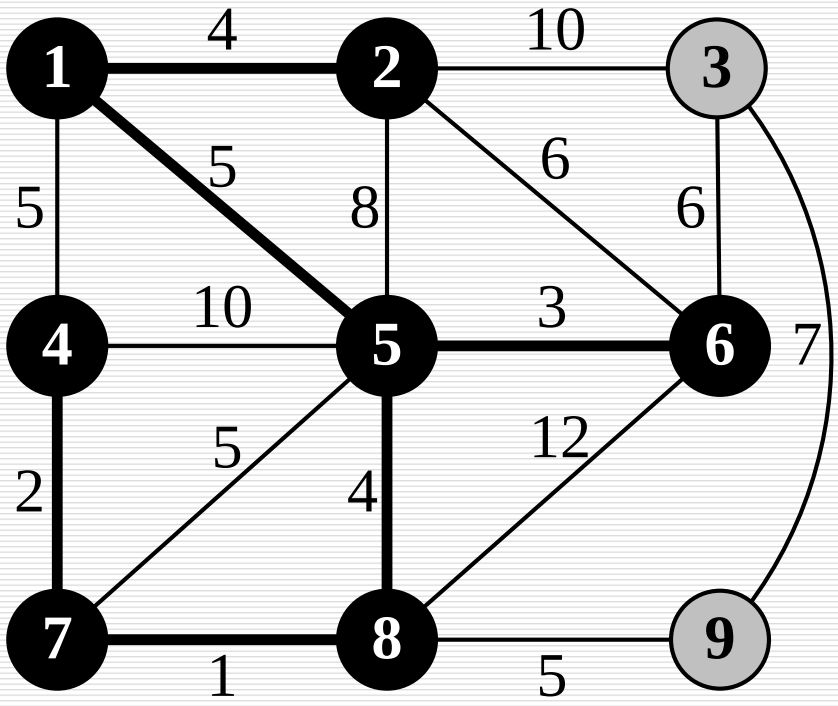
Αλγόριθμος Kruskal: παράδειγμα



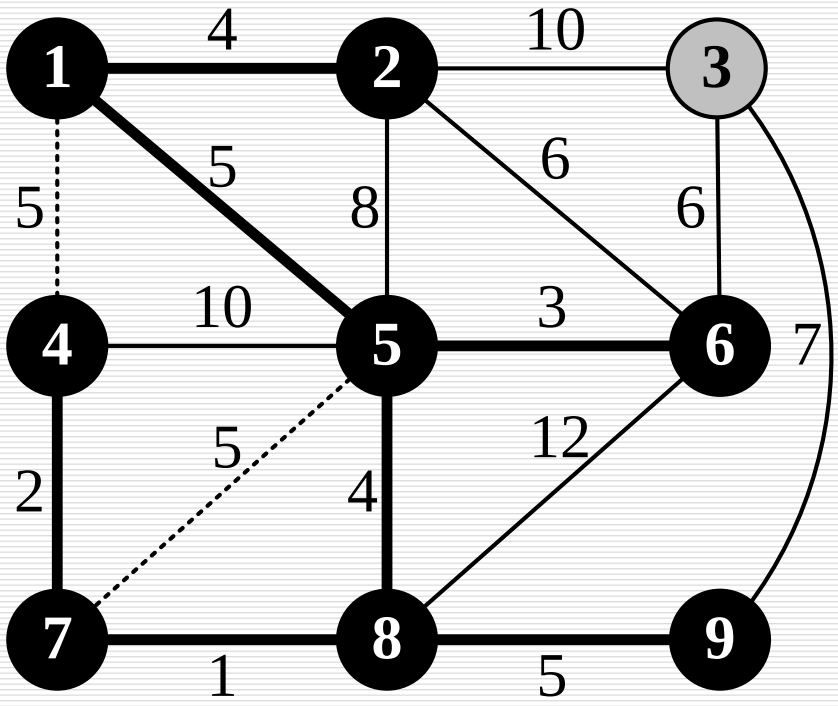
Αλγόριθμος Kruskal: παράδειγμα



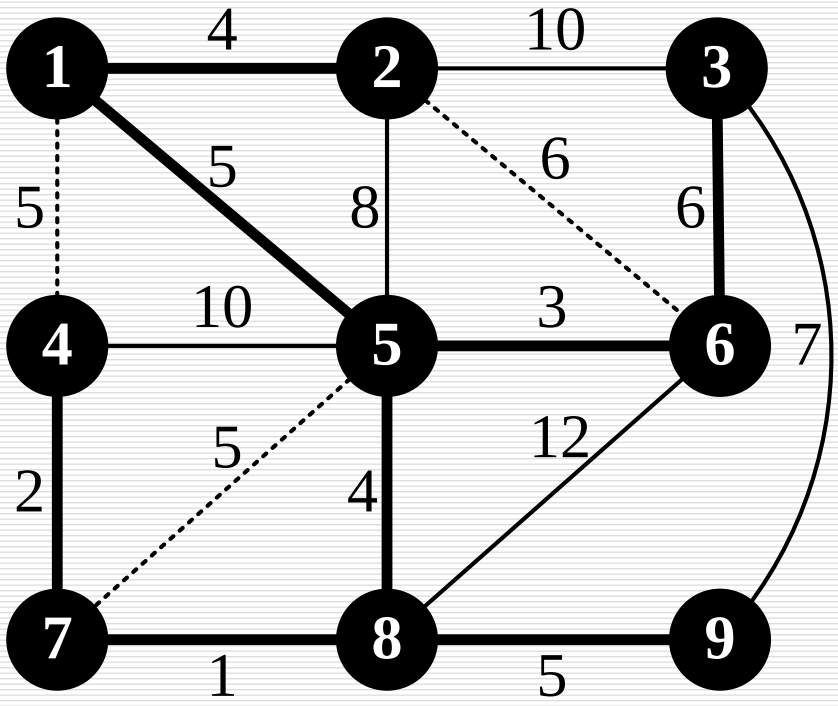
Αλγόριθμος Kruskal: παράδειγμα



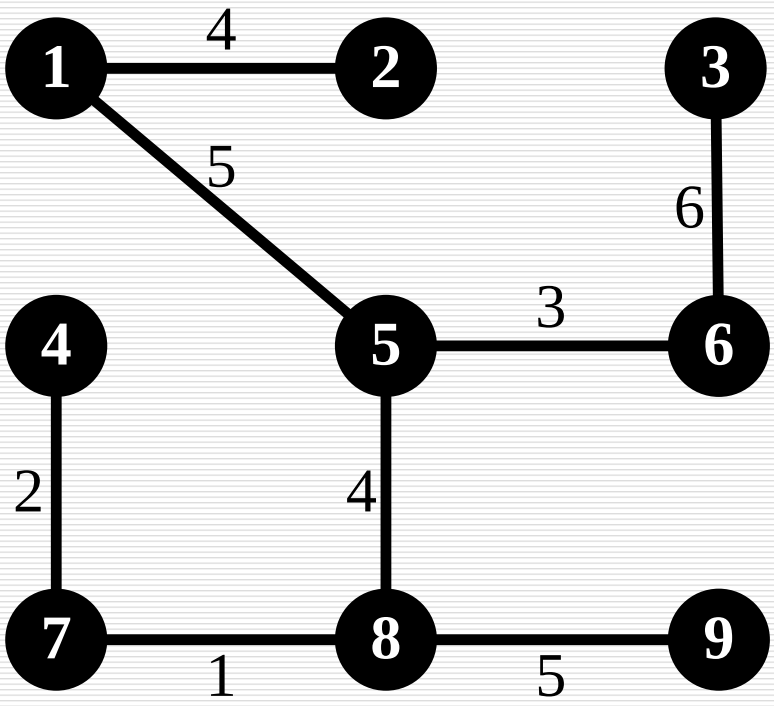
Αλγόριθμος Kruskal: παράδειγμα



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

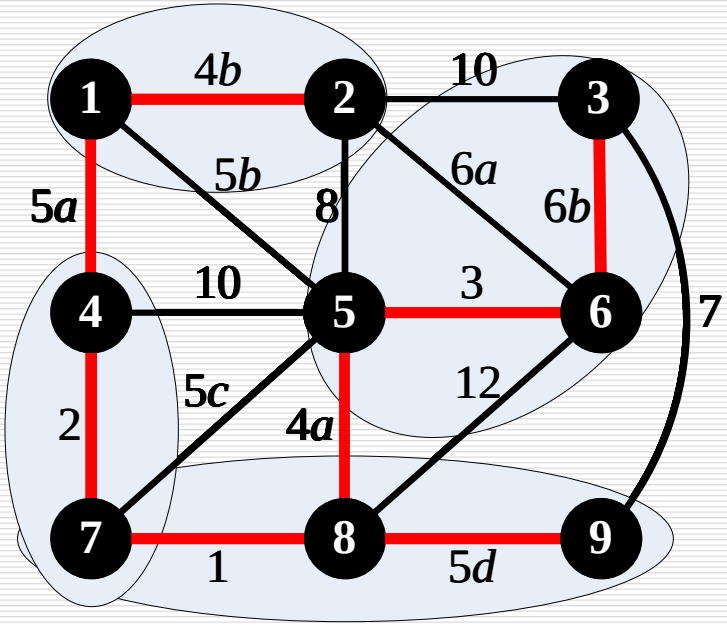
Bonus: αλγόριθμος Boruvka

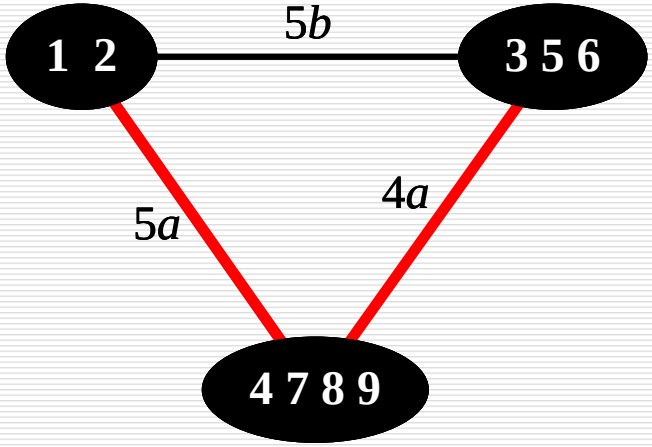
- Λειτουργεί σε γύρους. Αρχικά κάθε κόμβος είναι συνιστώσα μόνος του.
- Σε κάθε γύρο, *κάθε* συνεκτική συνιστώσα συνδέεται με την **ελαφρύτερη** δυνατή ακμή με κάποια από τις υπόλοιπες συνιστώσες. Χρειάζεται τρόπος επίλυσης 'ισοπαλιών'.

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

Προσφέρεται για παράλληλη / κατανεμημένη υλοποίηση.

Αλγόριθμος Boruvka: Παράδειγμα





Δύο σημαντικές τεχνικές

- **Άπληστοι** (greedy) αλγόριθμοι: «χτίσιμο» λύσης σταδιακά, από μικρότερα προς μεγαλύτερα υποπροβλήματα. Κάθε επιλογή βέλτιστη για αντίστοιχο υποπρόβλημα.
  - Dijkstra, Prim, Kruskal, Boruvka
- **Δυναμικός προγραμματισμός**: «χτίσιμο» λύσης συνδυάζοντας λύσεις (μικρότερων) υποπροβλημάτων με τρόπο που οδηγεί σε βέλτιστη λύση μεγαλύτερων υποπροβλημάτων.
  - Bellman-Ford
- Σύγκριση με «**διαίρει και βασίλευε**»: στη ΔκΚ τα υποπροβλήματα είναι ανεξάρτητα, στον ΔΠ και στους άπληστους τα υποπροβλήματα έχουν επικάλυψη.

Προβλήματα Γράφων στην Κλάση **P**

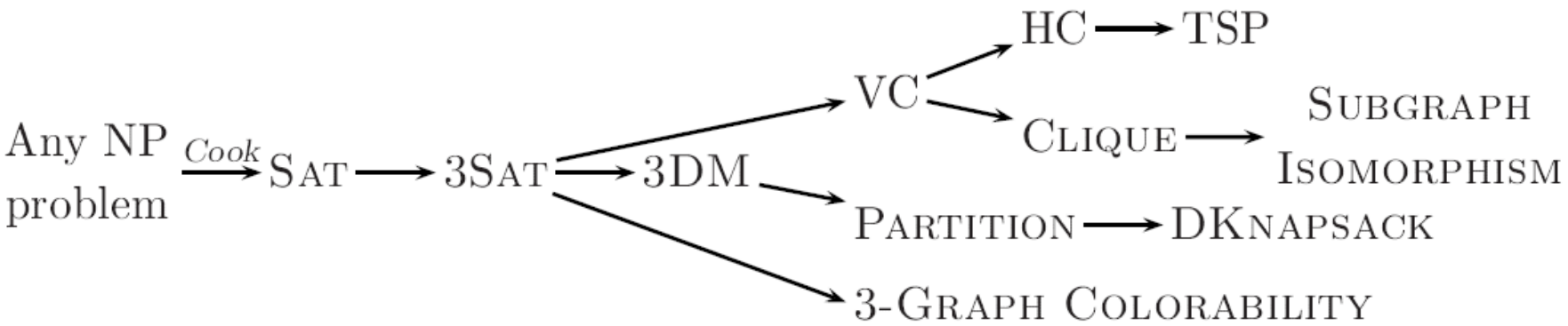
- Κύκλος Euler
- Προσβασιμότητα (reachability) + Διάσχιση (traversal): DFS, BFS, ...
- Συνεκτικές συνιστώσες (connected components)
- Συντομότερα μονοπάτια (shortest paths)
- Ελάχιστο συνδετικό δένδρο (minimum spanning tree)
- Μέγιστη ροή (maximum flow) [7ο εξ.]
- Τέλειο ταίριασμα (perfect matching)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

NP-πλήρη Προβλήματα Γράφων

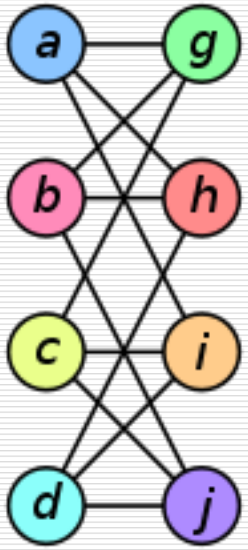
- Vertex Cover (VC)
- Clique
- Hamilton Circuit (HC)
- Traveling Salesman (TSP)
- 3-Colorability
- Subgraph Isomorphism
- 3-Dimensional Matching (3DM)

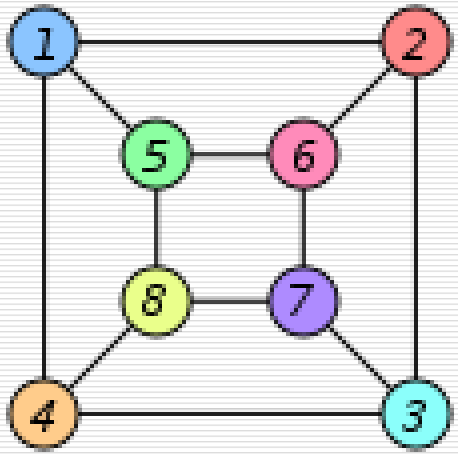
NP-πλήρη Προβλήματα Γράφων

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.