

Εισαγωγή στην Επιστήμη των Υπολογιστών

4^ο εξάμηνο ΣΗΜΜΥ

6^η ενότητα:

Γράφοι: προβλήματα και αλγόριθμοι

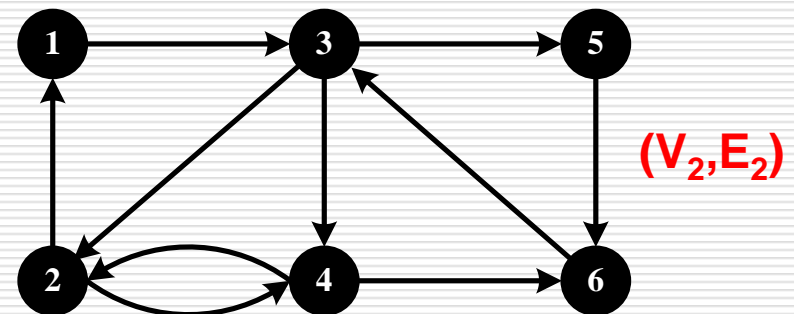
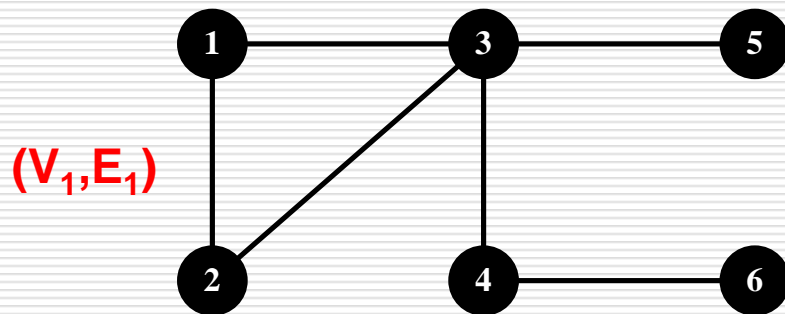
Επιμέλεια διαφανειών:

Στάθης Ζάχος, Άρης Παγουρτζής, Δημήτρης Φωτάκης



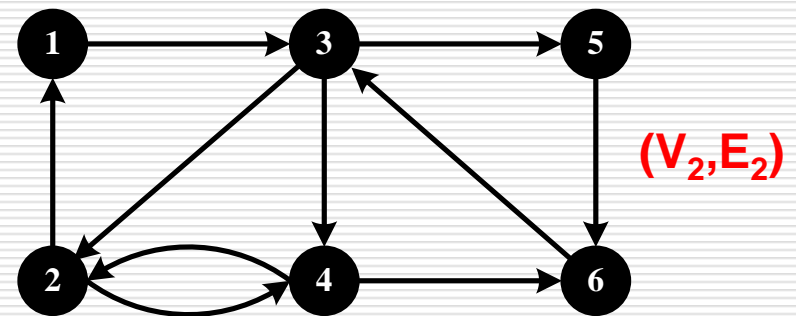
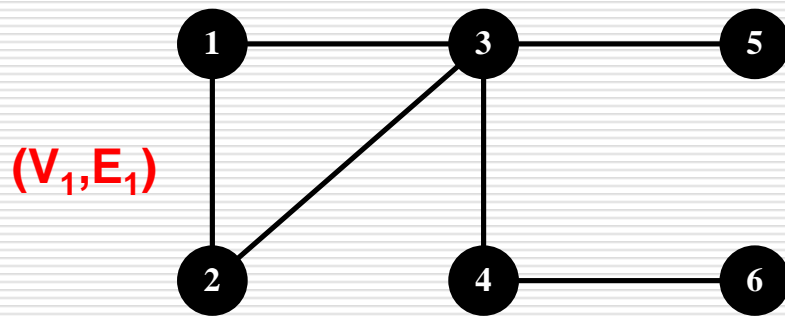
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Εθνικό Μετσόβιο Πολυτεχνείο

Γράφοι: κατευθυνόμενοι και μη



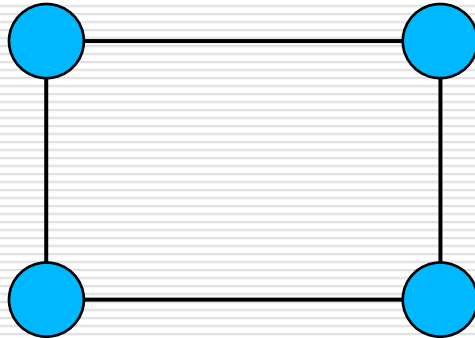
- **Γράφος (ή γράφημα):** ζεύγος (V, E) , V ένα μη κενό σύνολο, E διμελής σχέση πάνω στο V
- **Μη κατευθυνόμενος γράφος:** σχέση E συμμετρική
- **V : κορυφές (vertices), κόμβοι (nodes)**
- **E : ακμές (edges)**
 - $E_1 = \{\{1,2\}, \{1,3\}, \{2,3\}, \{3,4\}, \{3,5\}, \{4,6\}\}$
 - $E_2 = \{(1,3), (2,1),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

Γράφοι: ορολογία



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

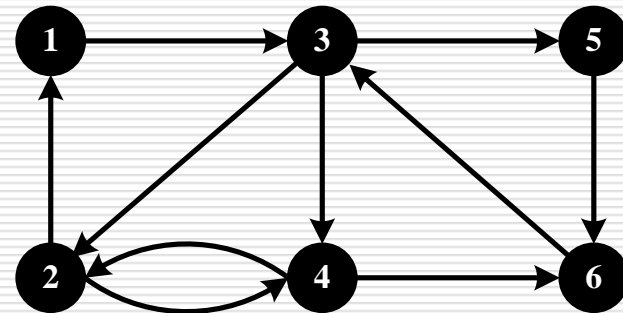
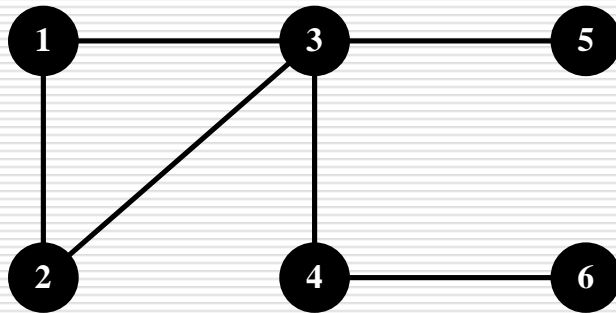
Γράφοι: ορολογία



2-κανονικός γράφος

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

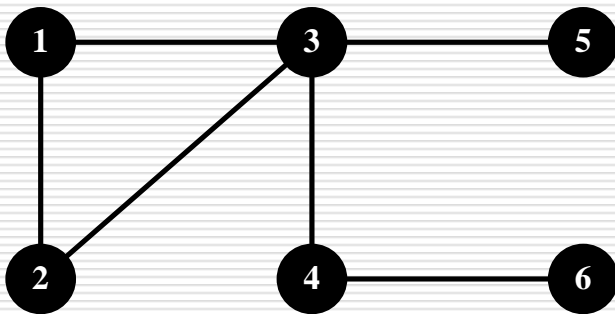
Διαδρομές σε γράφους



- **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

Αναπαράσταση γράφων

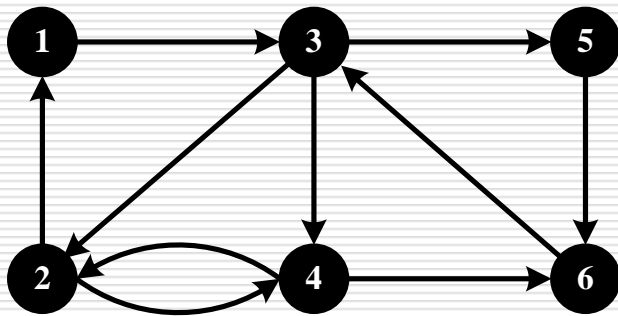
- ... με **πίνακα γειτνίασης**: $A[i, j] = \begin{cases} 1 & (v_i, v_j) \in E \\ 0 & (v_i, v_j) \notin E \end{cases}$
 - Αν έχουμε βάρη, $A[i, j] = w(v_i, v_j)$
 - Μη-κατευθυνόμενος: **συμμετρικός** πίνακας
 - Χώρος: $\Theta(n^2)$
 - Προσπέλαση γειτόνων: $\Theta(n)$
 - Άμεσος έλεγχος ύπαρξης ακμής: $O(1)$



	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

Αναπαράσταση γράφων

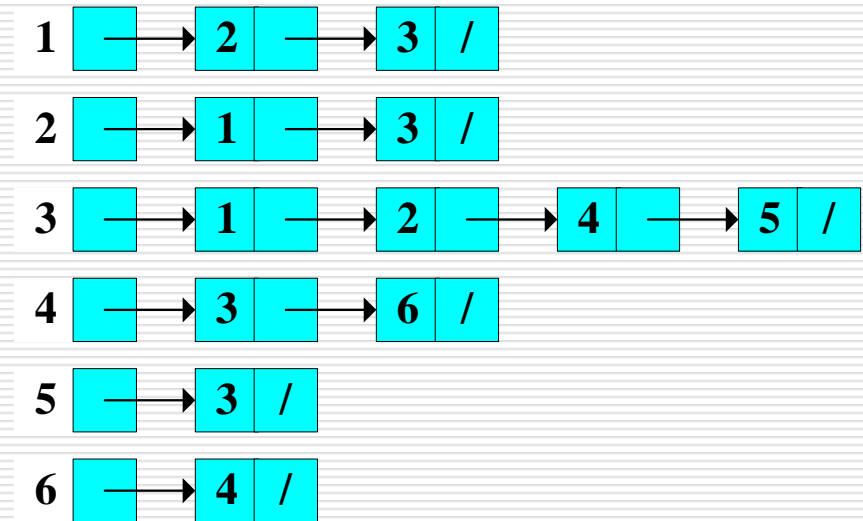
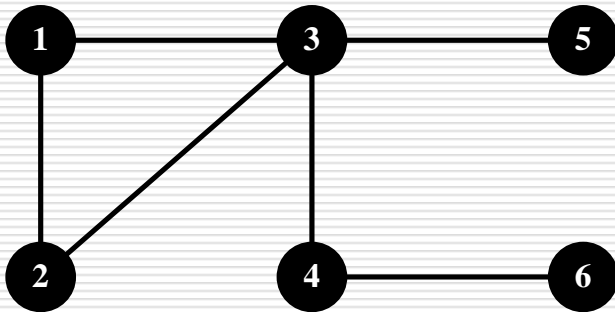
- ... με **πίνακα γειτνίασης**: $A[i, j] = \begin{cases} 1 & (v_i, v_j) \in E \\ 0 & (v_i, v_j) \notin E \end{cases}$
 - Αν έχουμε βάρη, $A[i, j] = w(v_i, v_j)$
 - Κατευθυνόμενος: **μη-συμμετρικός** πίνακας
 - Χώρος: $\Theta(n^2)$
 - Προσπέλαση γειτόνων: $\Theta(n)$
 - Άμεσος έλεγχος ύπαρξης ακμής: $O(1)$



	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

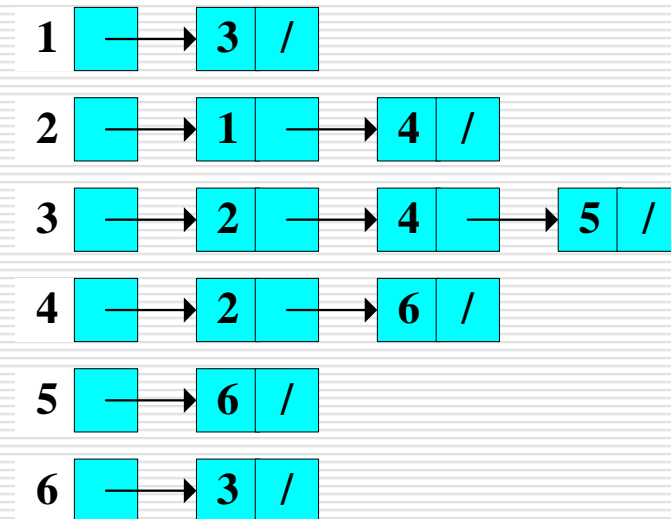
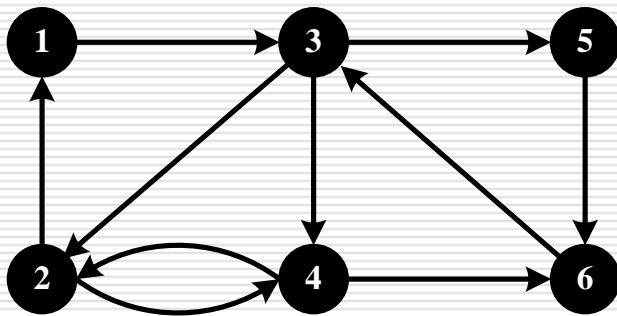
Αναπαράσταση γράφων

- ... με **λίστες γειτνίασης**: γειτονικές κορυφές σε λίστες
 - Αν έχουμε βάρη, τα αποθηκεύουμε στους κόμβους
 - Χώρος: $\Theta(m)$
 - Προσπέλαση γειτόνων: $\Theta(\text{deg}(u))$
 - Έλεγχος ύπαρξης ακμής: $O(\text{deg}(u))$



Αναπαράσταση γράφων

- ... με **λίστες γειτνίασης**: γειτονικές κορυφές σε λίστες
 - Αν έχουμε βάρη, τα αποθηκεύουμε στους κόμβους
 - Χώρος: $\Theta(m)$
 - Προσπέλαση γειτόνων: $\Theta(\text{deg}(u))$
 - Έλεγχος ύπαρξης ακμής: $O(\text{deg}(u))$



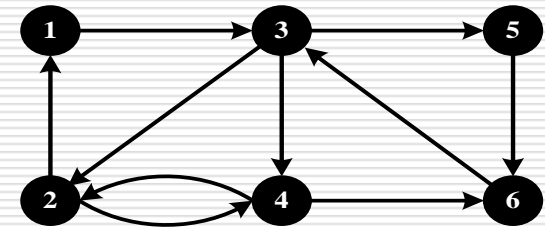
Γράφοι: συνεκτικοί και μη

- 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)**

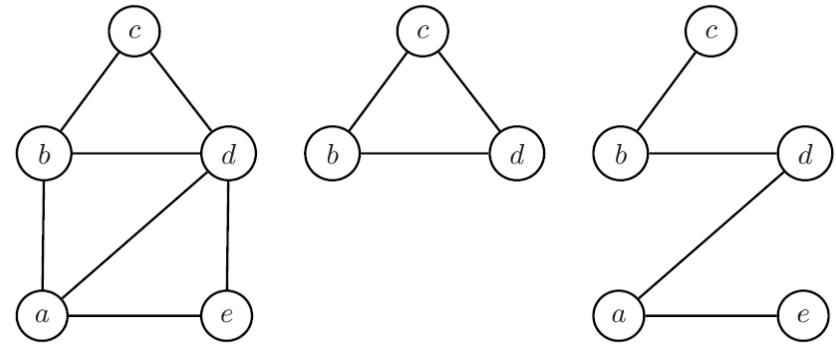
αν υπάρχει δρόμος μεταξύ οποιωνδήποτε δύο κορυφών του ακολουθώντας τις κατευθύνσεις των ακμών



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

Άλλες έννοιες

- **Παράγων υπογράφος** (spanning subgraph)



- **Παραγόμενος υπογράφος** (induced subgraph)
- **Συνεκτικές συνιστώσες** (connected components)
- **Πλήρης γράφος (K_n), διμερής γράφος (πλήρης $K_{n,m}$)**
- **Επίπεδος γράφος:** αν δεν περιέχει ως υπογράφους τα K_5 , $K_{3,3}$ - ούτε γράφους που προκύπτουν από αυτά με υποδιαίρεσεις των ακμών τους
- **Δένδρο (tree):** συνεκτικός γράφος χωρίς κύκλους

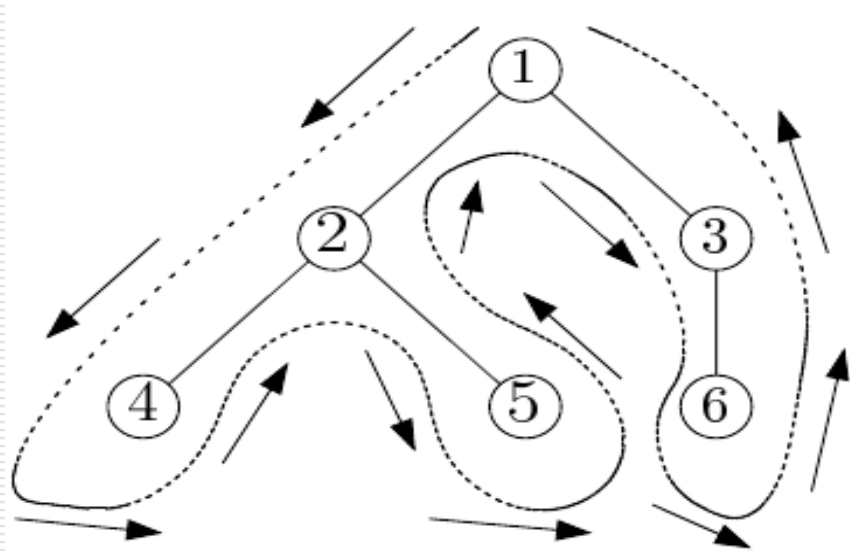
Κλάσεις πολυπλοκότητας

- **P**: προβλήματα απόφασης για τα οποία η σωστή απάντηση μπορεί να **βρεθεί** σε πολυωνυμικό χρόνο
- **NP**: προβλήματα απόφασης για τα οποία η σωστή απάντηση, αν είναι *καταφατική*, μπορεί να **επαληθευθεί** σε πολυωνυμικό χρόνο (με χρήση *σύντομου πιστοποιητικού*)
- Συχνά εννοούμε τα αντίστοιχα προβλήματα **αναζήτησης** ή / και **βελτιστοποίησης**

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)
- **Τέλειο ταίριασμα** (perfect matching)
- **Χρωματισμός ακμών διμερούς γράφου** (bipartite edge coloring)

Διάσχιση δένδρων



- **Προδιατ/νη:** καταγραφή κόμβου την 1^η φορά που τον συναντάμε
- **Ενδοδιατ/νη:** καταγραφή κόμβου τη 2^η φορά που τον συναντάμε (φύλλα: την 1^η)
- **Μεταδιατ/νη:** καταγραφή κόμβου την τελευταία φορά που τον συναντάμε

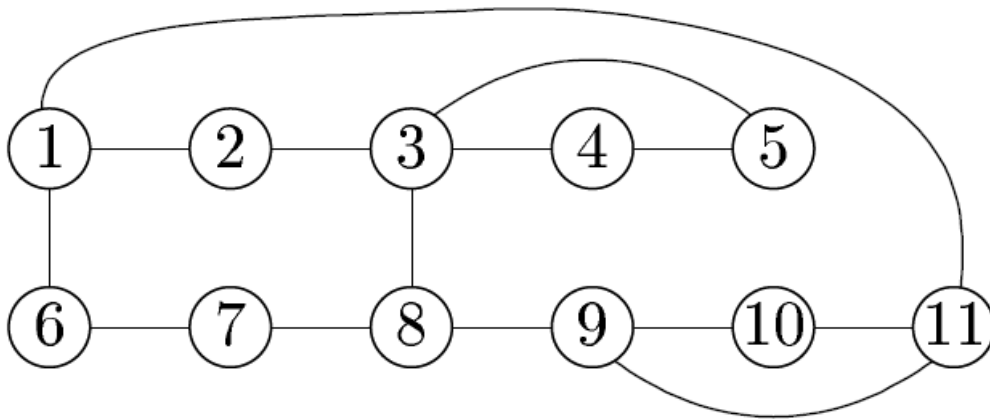
- **Προδιατεταγμένη (preorder):** 1 2 4 5 3 6
- **Ενδοδιατεταγμένη (inorder):** 4 2 5 1 6 3
- **Μεταδιατεταγμένη (postorder):** 4 5 2 6 3 1

Αναζήτηση Κατά Βάθος (DFS)

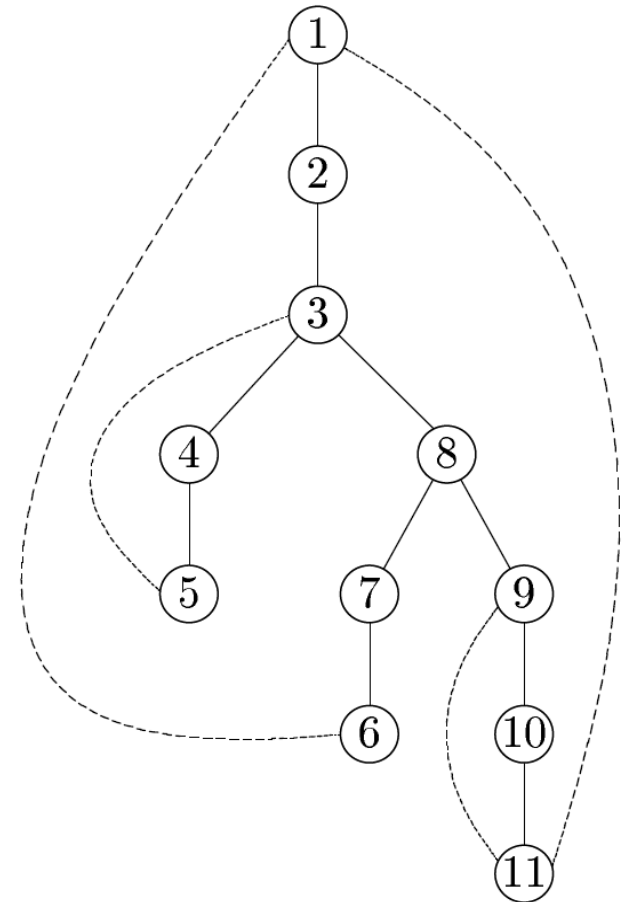
```
procedure dfs(v:vertex);  
begin  
    visited[v]:=true;  
    for all vertices u adjacent to v do  
        if not visited[u] then dfs(u)  
    end
```

Πολυπλοκότητα $O(|V| + |E|)$: σε κάθε κόμβο $O(\text{deg}(v))$ έλεγχοι και κλήσεις της dfs (με ποια αναπαράσταση;)

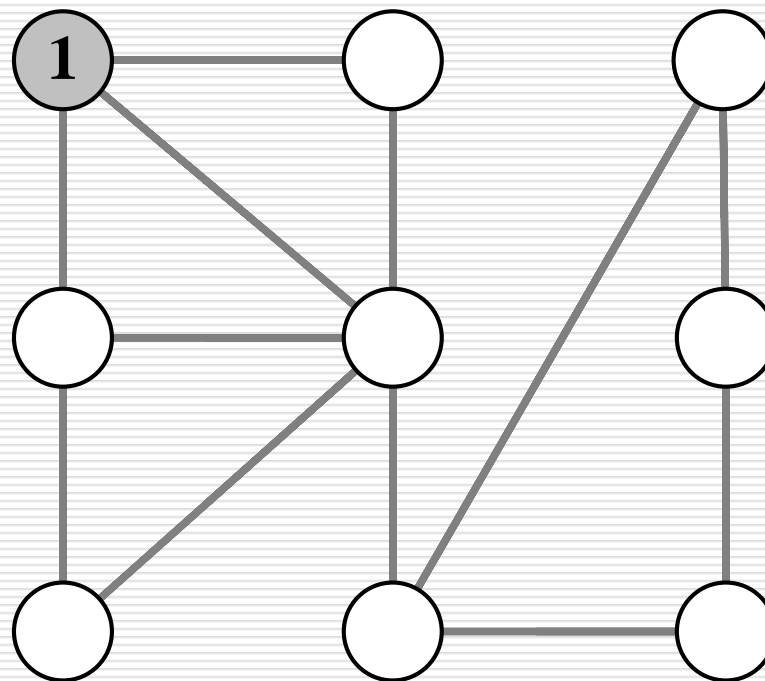
Παράδειγμα DFS



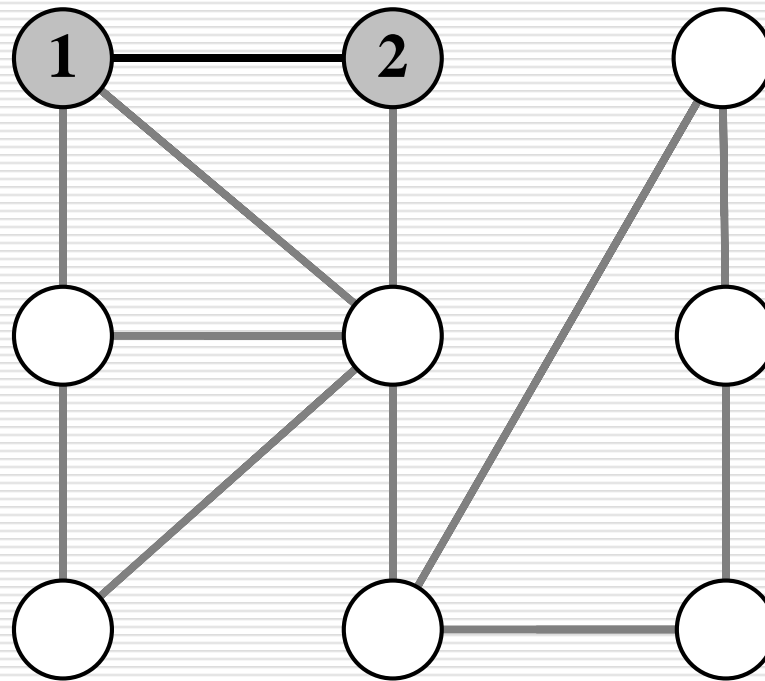
```
procedure dfs(v:vertex);  
begin  
    visited[v]:=true;  
    for all vertices u adjacent to v do  
        if not visited[u] then dfs(u)  
    end
```



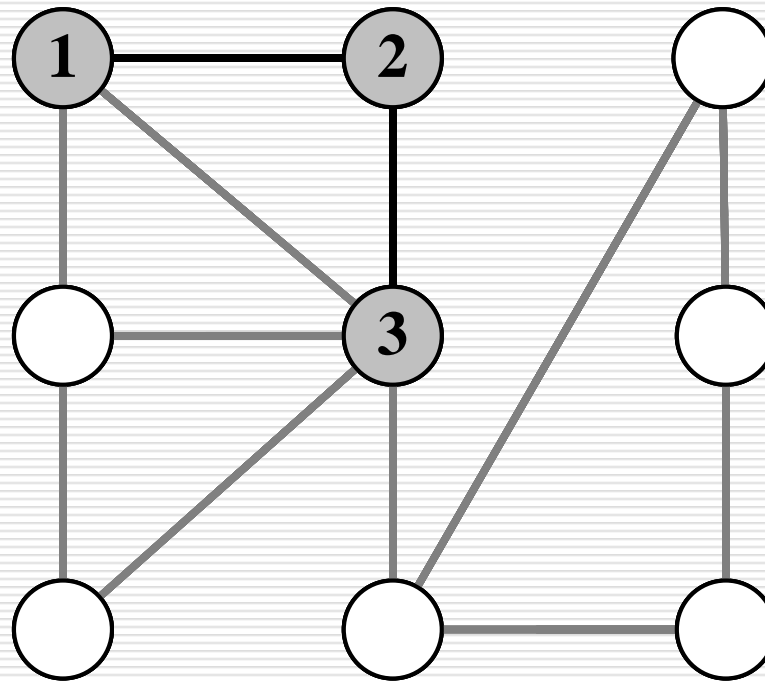
2^ο παράδειγμα DFS



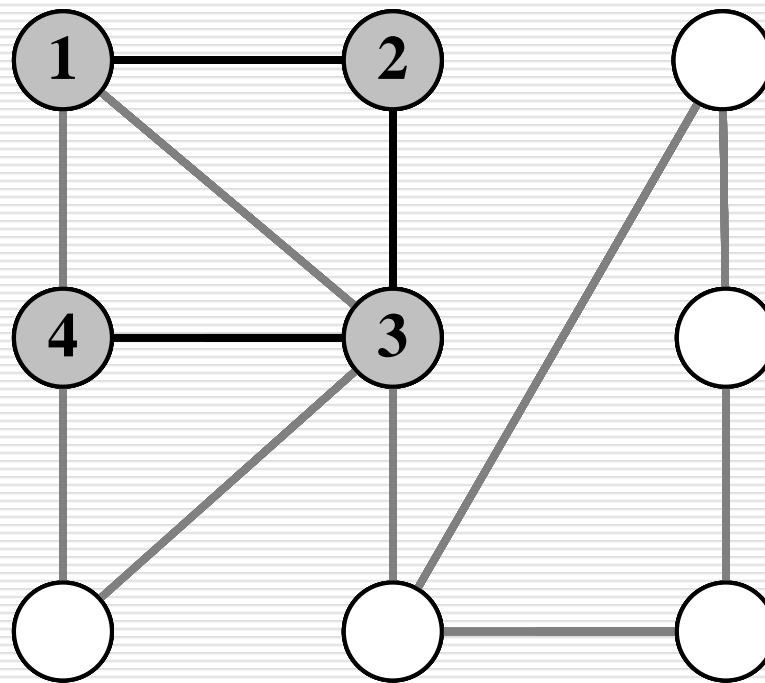
2^ο παράδειγμα DFS



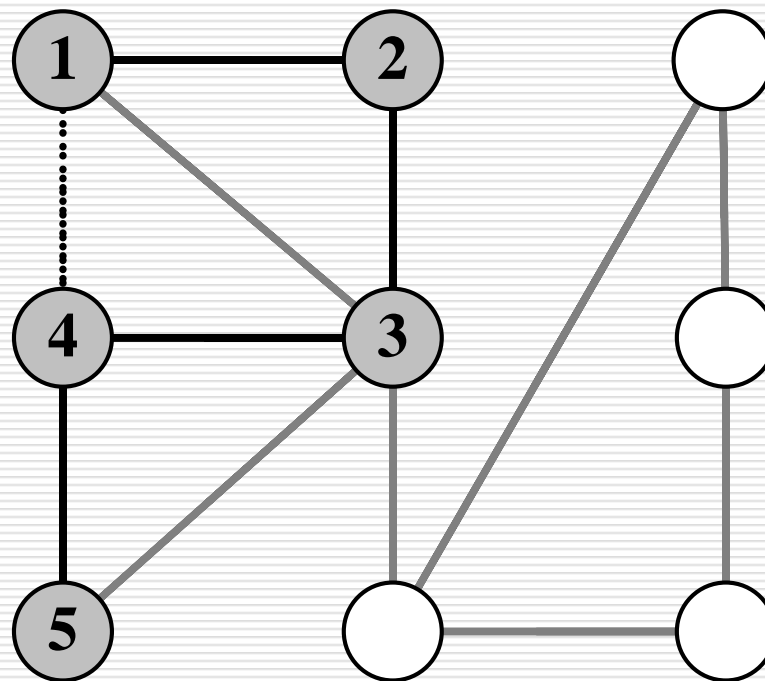
2^ο παράδειγμα DFS



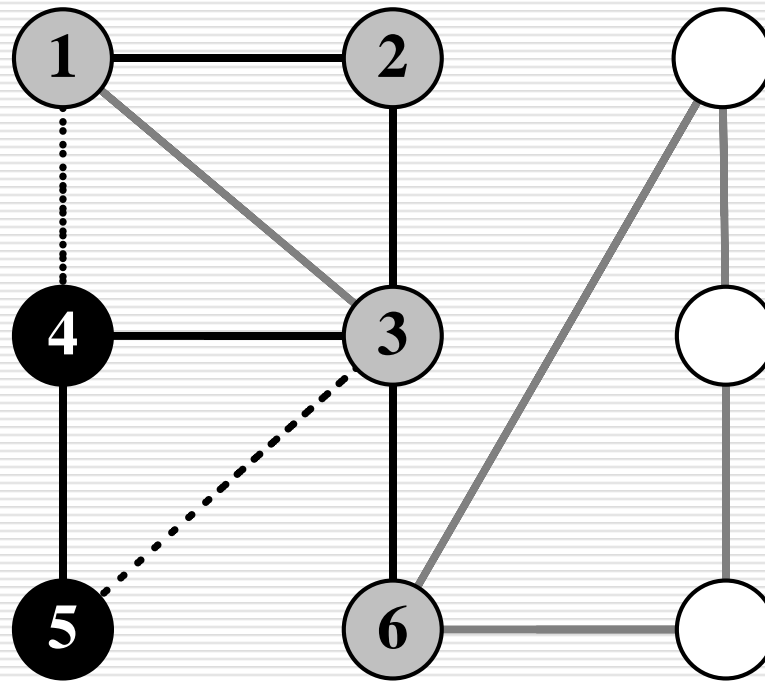
2^ο παράδειγμα DFS



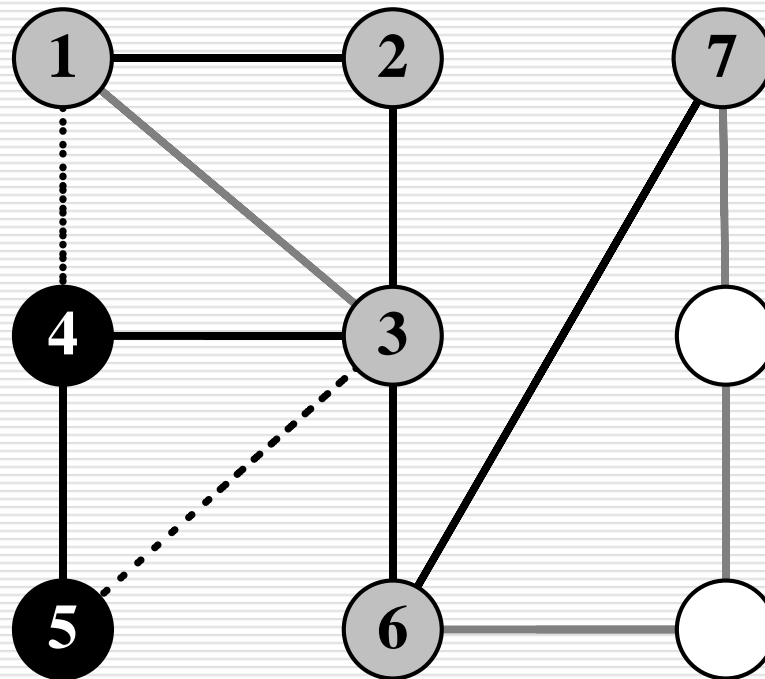
2^ο παράδειγμα DFS



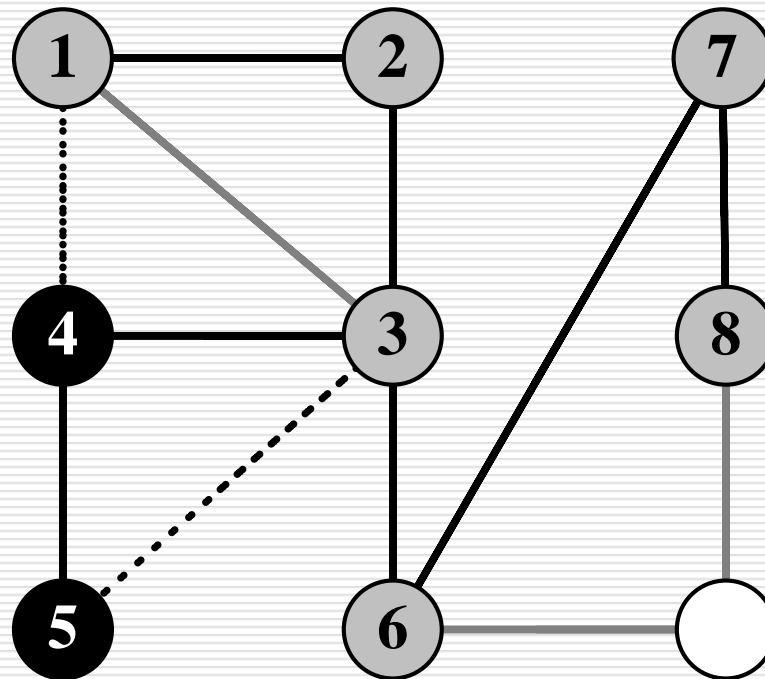
2^ο παράδειγμα DFS



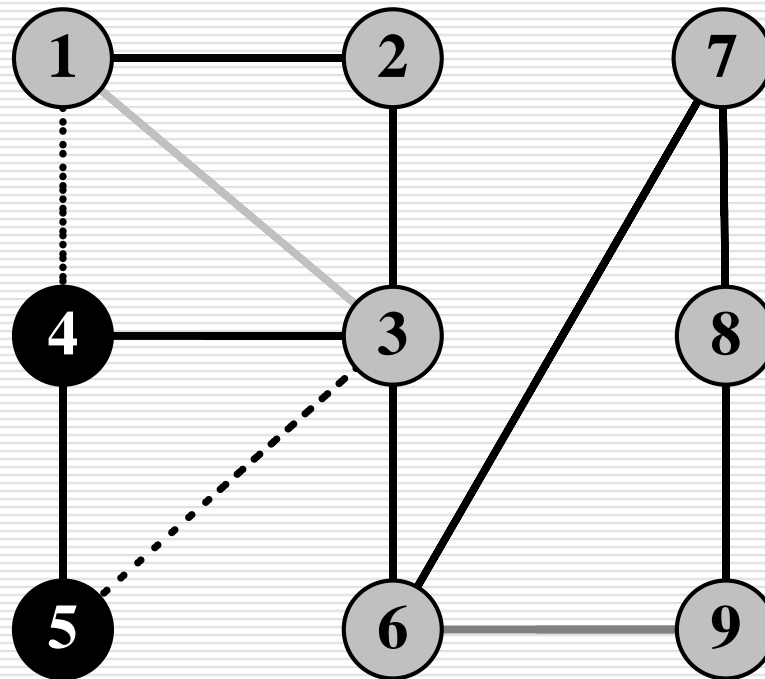
2^ο παράδειγμα DFS



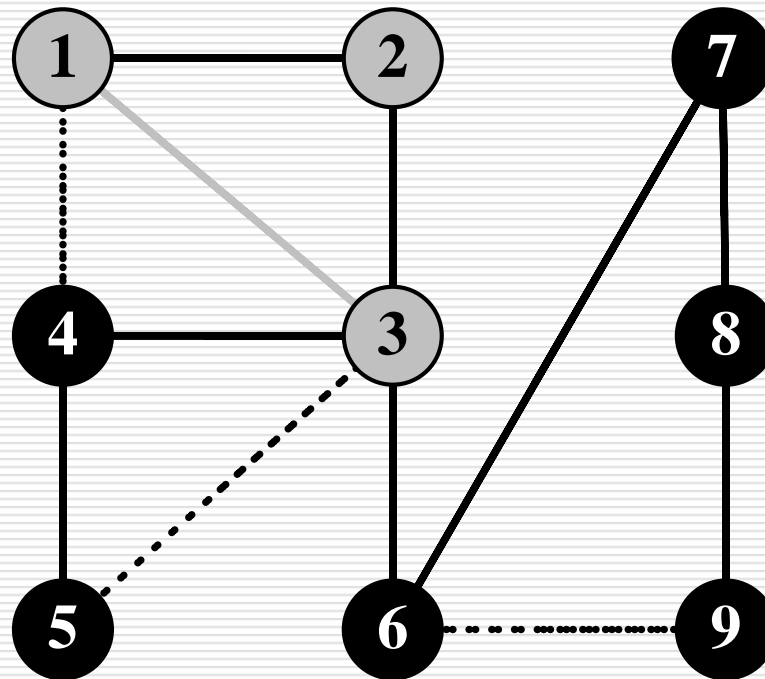
2^ο παράδειγμα DFS



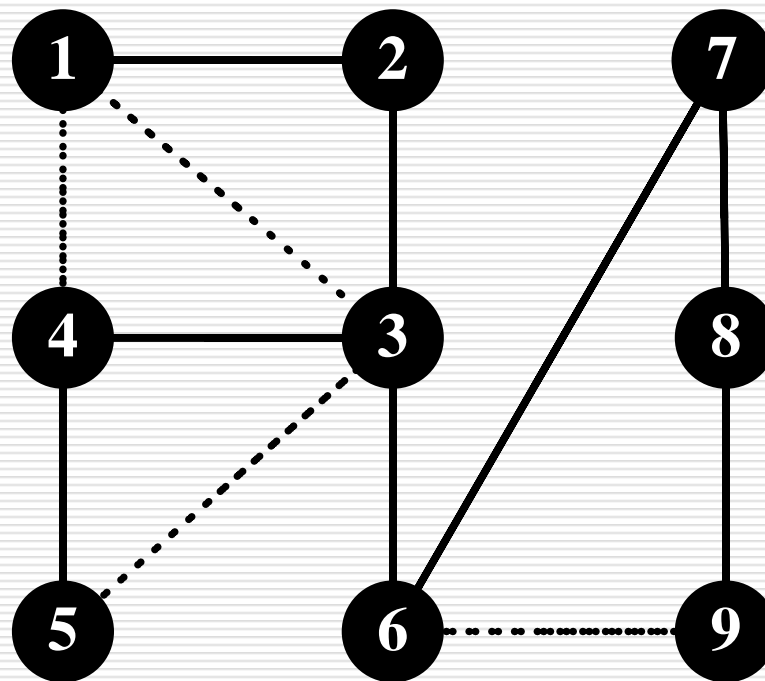
2^ο παράδειγμα DFS



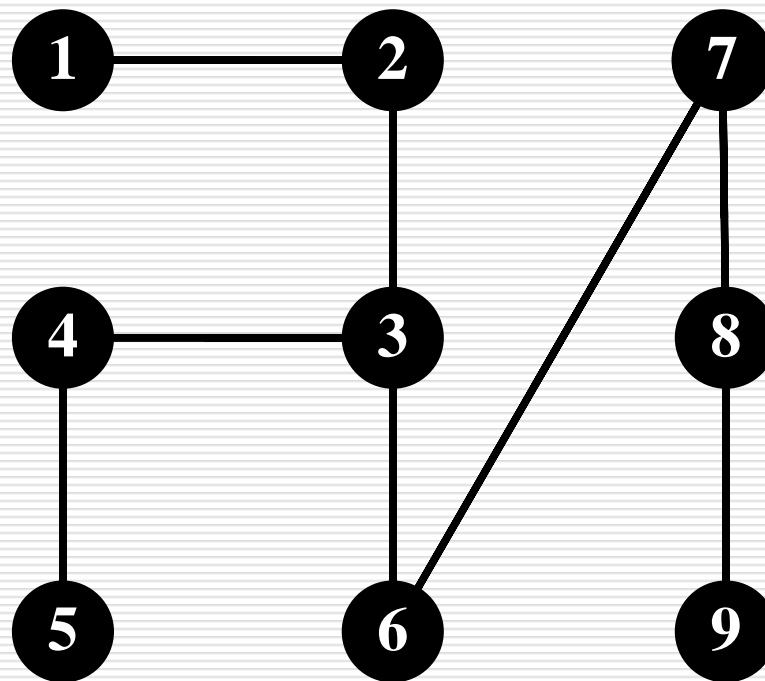
2^ο παράδειγμα DFS



2^ο παράδειγμα DFS



2^ο παράδειγμα DFS



Επιπλέον εφαρμογές DFS

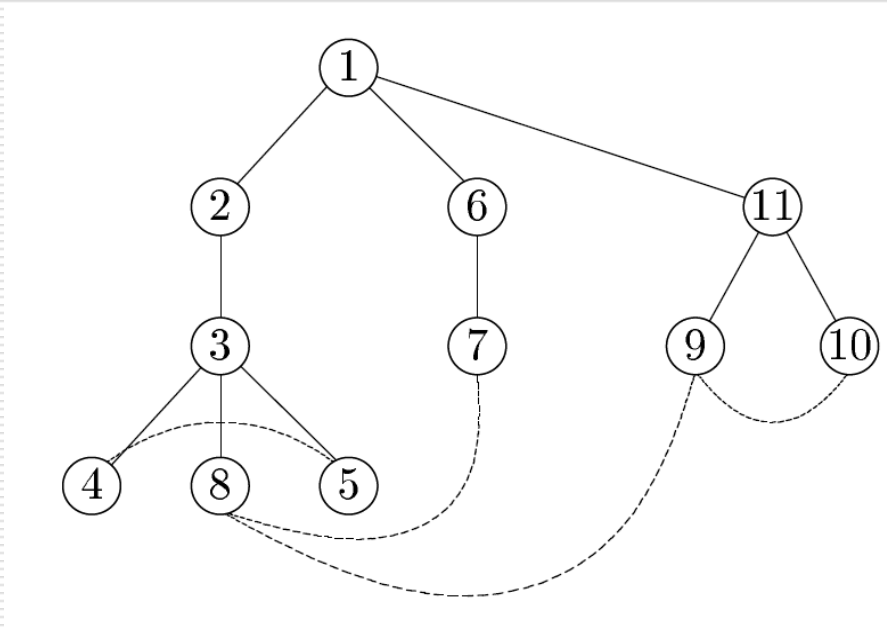
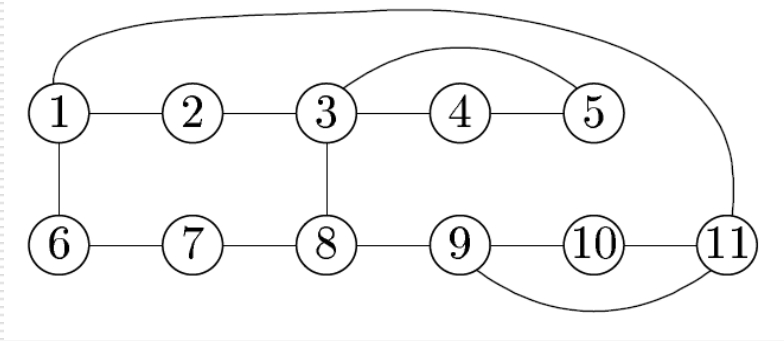
- **Έλεγχος συνεκτικότητας**
- **Εύρεση συνεκτικών συνιστωσών** (πώς;)
- **Εντοπισμός / εύρεση κύκλων:** με έλεγχο και χρήση *μη δενδρικών* ακμών
- **Αποδοτική εξερεύνηση άγνωστης περιοχής**

Αναζήτηση Κατά Πλάτος (BFS)

```
procedure bfs(v:vertex);
begin
  initialize queue with v; visited[v]:=true;
  repeat dequeue(u);
    for all vertices w adjacent to u do
      if not visited[w] then
        begin visited[w] := true; enqueue(w) end
  until queue is empty
end
```

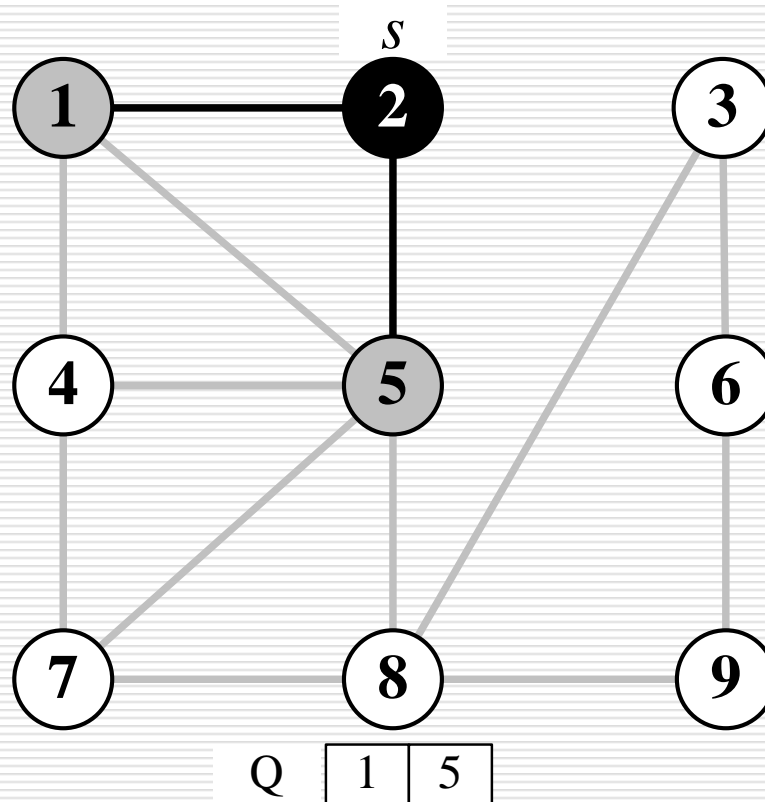
Πολυπλοκότητα $O(|V| + |E|)$: σε κάθε κόμβο v ,
 $O(\text{deg}(v))$ έλεγχοι και εισαγωγές στην ουρά

Παράδειγμα BFS

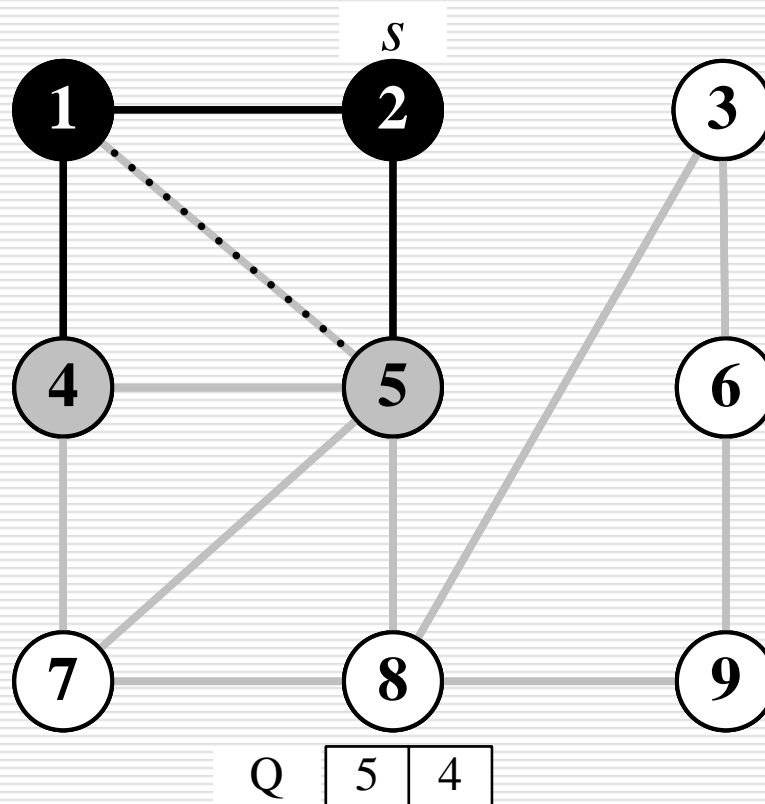


visited nodes	Queue
1	1
2	2
6	2-6
11	2-6-11
3	6-11-3
7	11-3-7
9	3-7-9
10	3-7-9-10
4	7-9-10-4
8	7-9-10-4-8
5	7-9-10-4-8-5
-	9-10-4-8-5
-	10-4-8-5
-	4-8-5
-	8-5
-	5
-	\emptyset

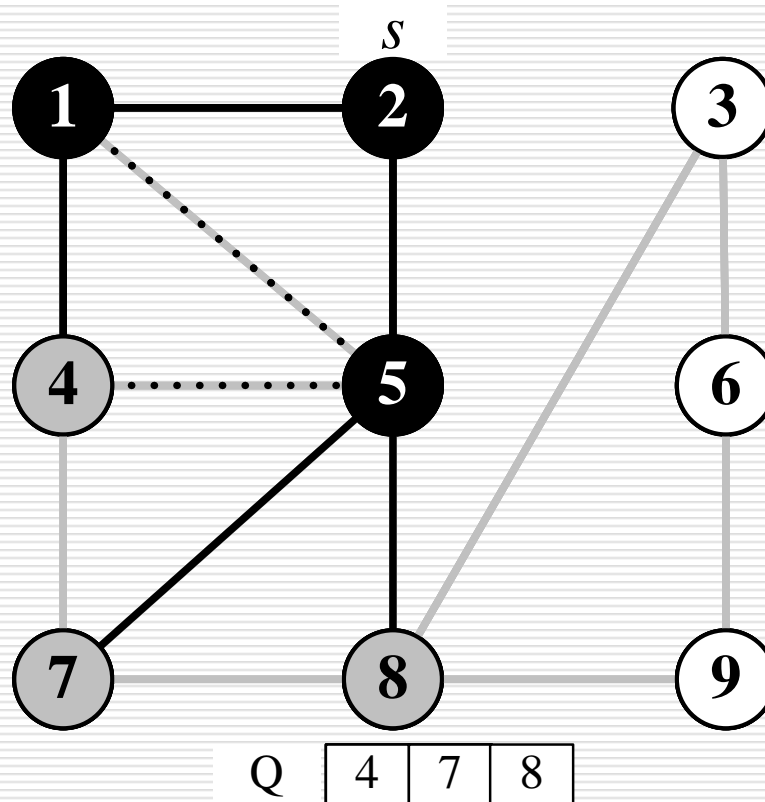
2^ο παράδειγμα BFS



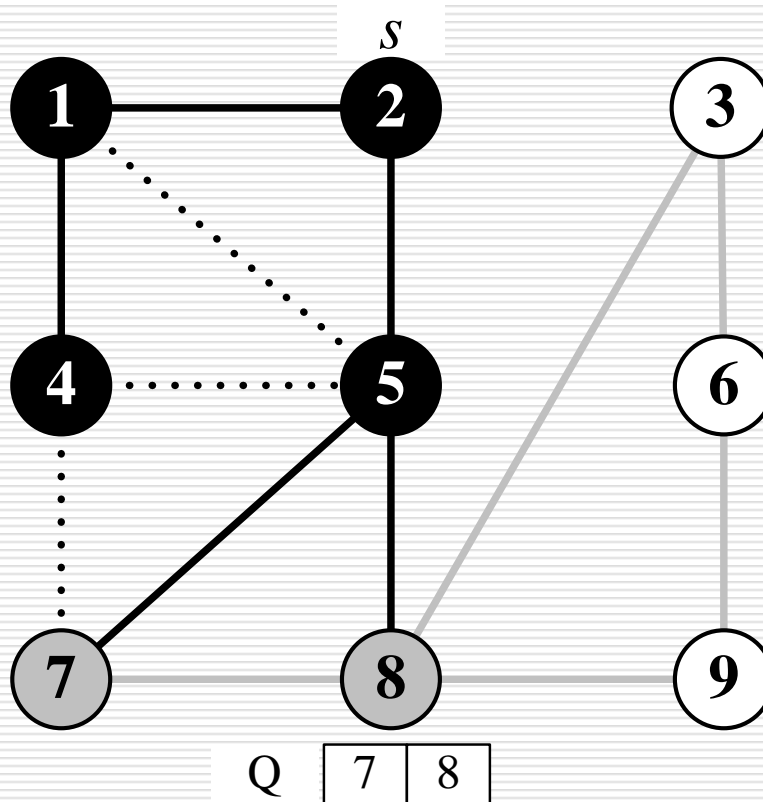
2^ο παράδειγμα BFS



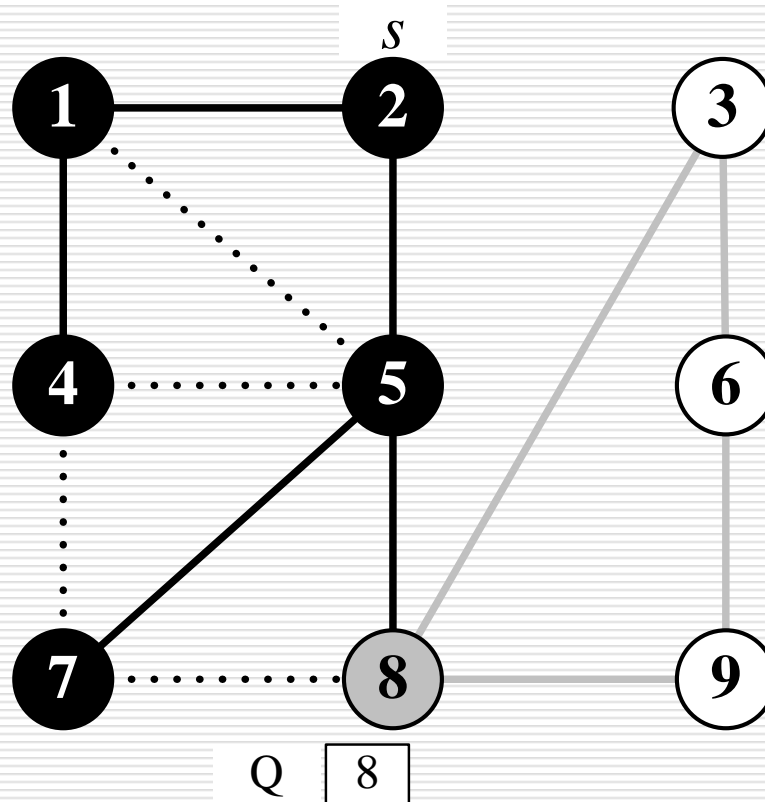
2^ο παράδειγμα BFS



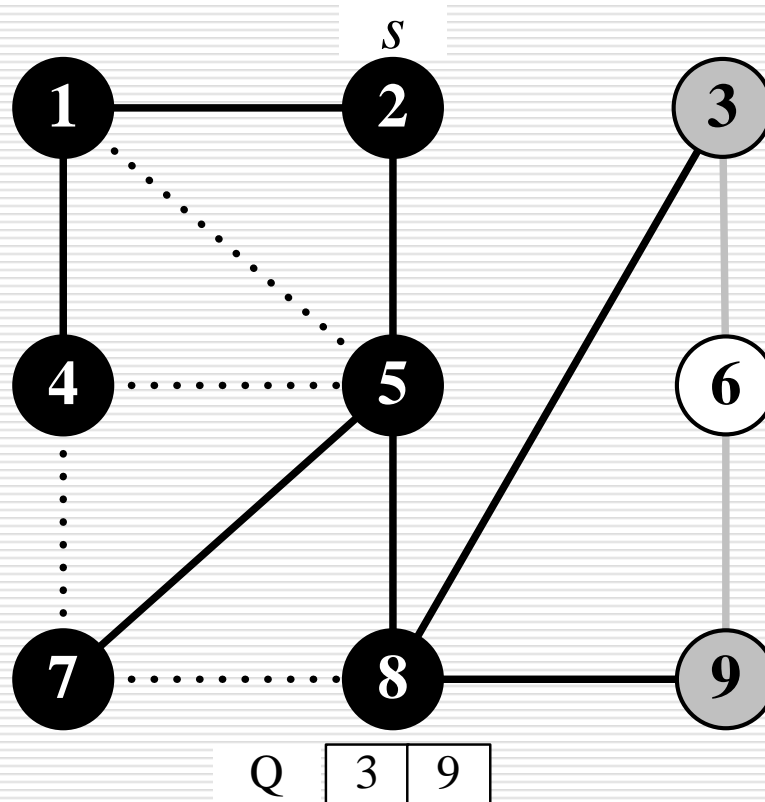
2^ο παράδειγμα BFS



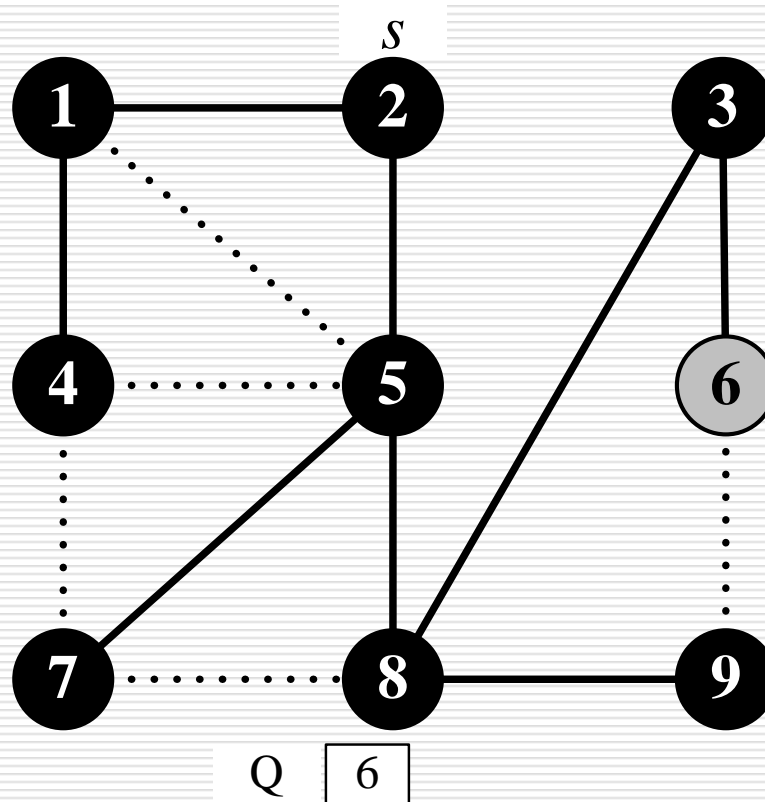
2^ο παράδειγμα BFS



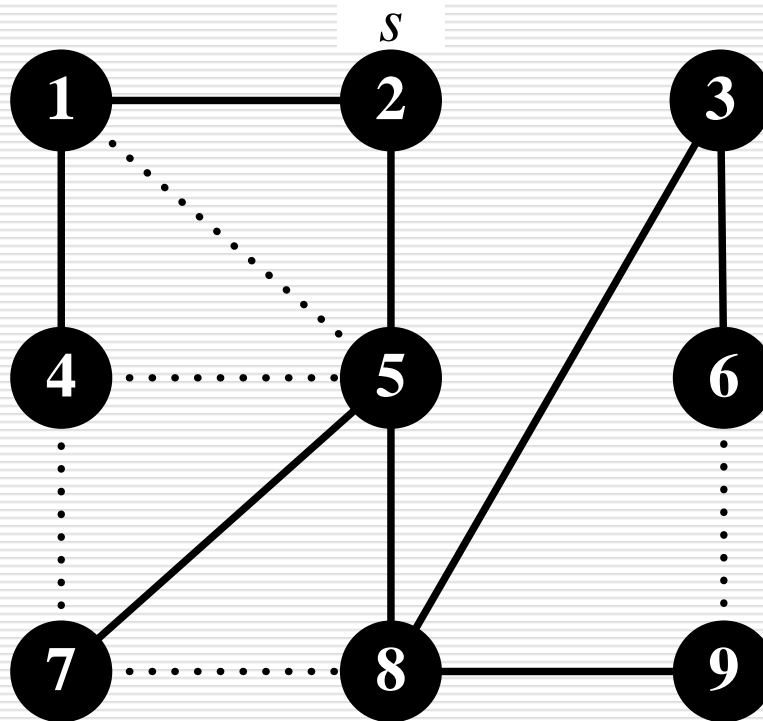
2^ο παράδειγμα BFS



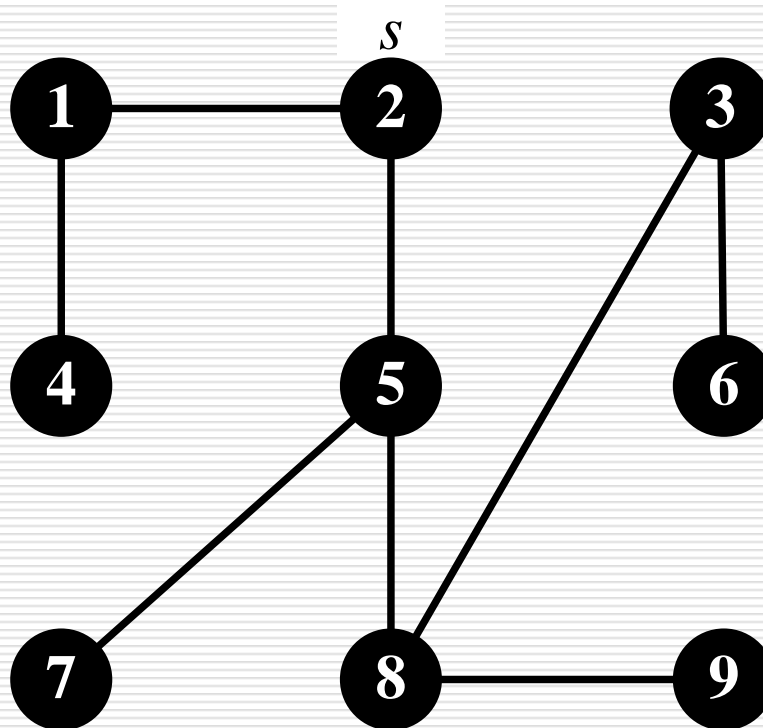
2^ο παράδειγμα BFS



2^ο παράδειγμα BFS



2^ο παράδειγμα BFS

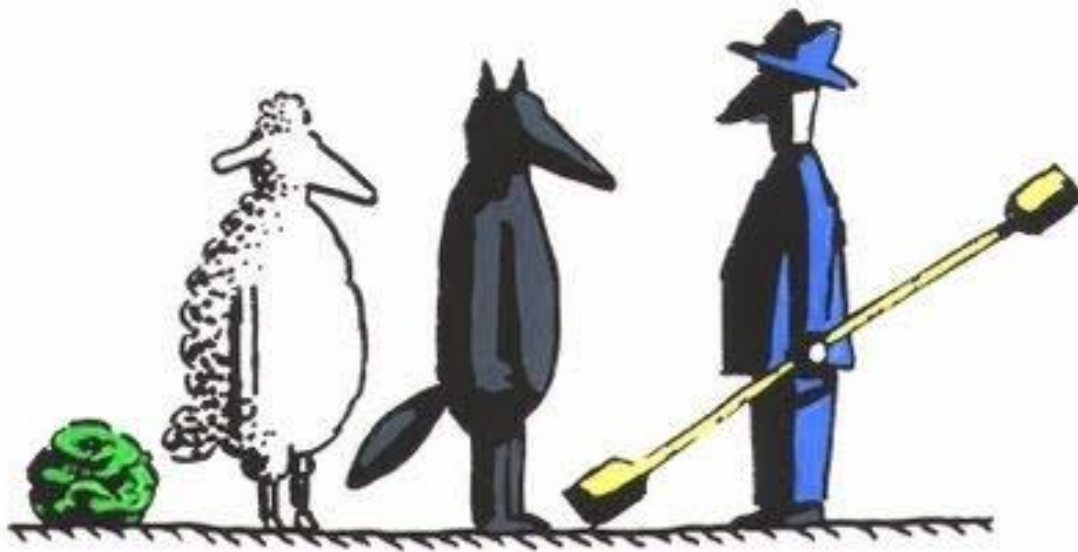


Επιπλέον εφαρμογές BFS

- **Έλεγχος συνεκτικότητας**
- **Εύρεση συνεκτικών συνιστωσών**
- **Εντοπισμός / εύρεση κύκλων:** με έλεγχο και χρήση *μη δενδρικών* ακμών
- **Μέτρηση αποστάσεων** από αρχικό κόμβο (σε πλήθος ακμών)

Εφαρμογή DFS/BFS

- Το Πρόβλημα του Βαρκάρη



Εφαρμογή DFS/BFS

- Το Πρόβλημα του Βαρκάρη
- Μοντελοποίηση καταστάσεων με κόμβους γράφου, π.χ.

$$(Λ,Π,Μ,Β / -) \rightarrow (Λ,Π / Β,Μ)$$

- Επίλυση: διαδρομή από αρχική κατάσταση προς τελική
- Πώς θα βρούμε την βέλτιστη σειρά κινήσεων;
- Πώς γενικεύεται το πρόβλημα;

Συντομότερα Μονοπάτια (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
    begin
      if  $D[v] > D[w] + C[w,v]$  then
      begin
        P[v] :=  $w$ ;  $D[v] := D[w] + C[w,v]$ 
      end
    end
  end
end
end
```

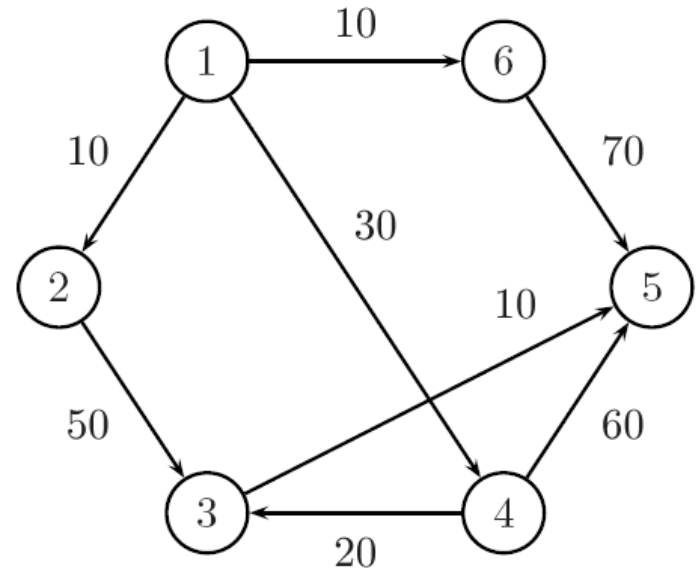
Πολυπλ/τα
 $O(|V|^2)$:

σε κάθε
επανάληψη
 $O(|V|)$ για
εύρεση
ελαχίστου,
 $O(|V|)$ για
ενημέρωση
αποστάσεων

Παράδειγμα Dijkstra

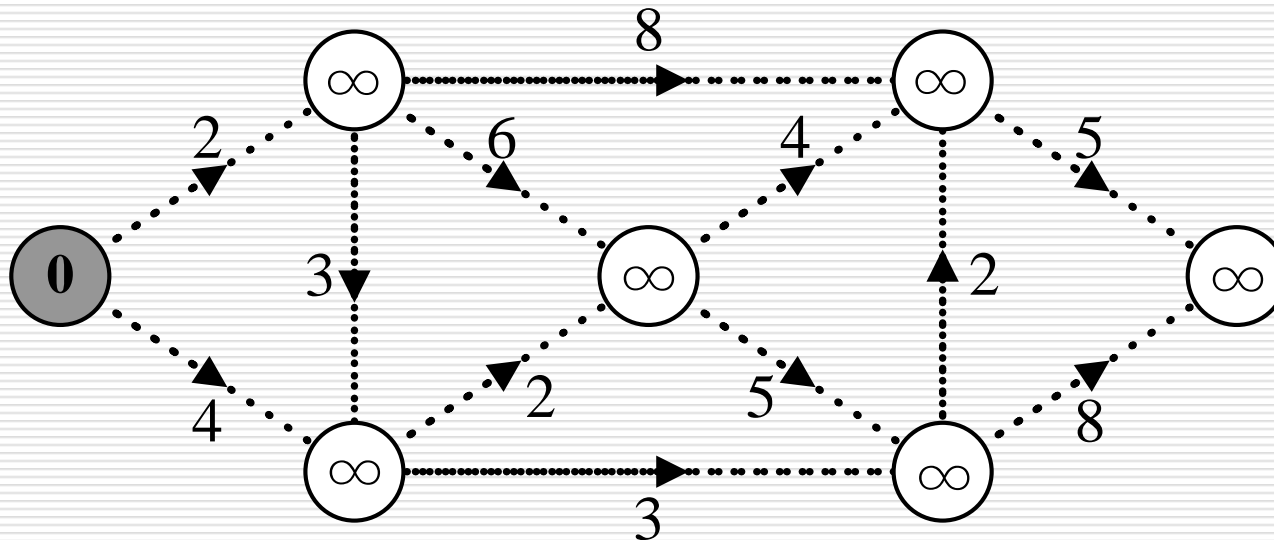
```

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
  end
end
  
```



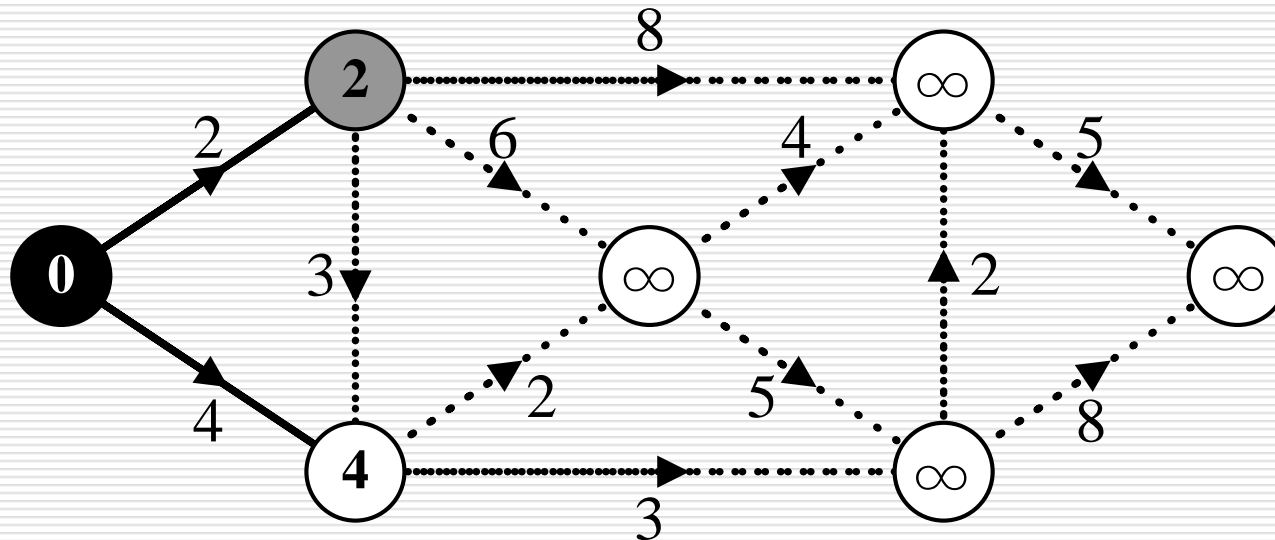
Βήμα	S	w	D					P				
			2	3	4	5	6	2	3	4	5	6
-	{1}	-	10	∞	30	∞	10	1	1	1	1	1
2	{1,2}	2		60	30	∞	10		2			
3	{1,2,6}	6		60	30	80					6	
4	{1,2,6,4}	4		50		80			4			
5	{1,2,6,4,3}	3				60						3
6	{1,2,6,4,3,5}	5										

2^ο παράδειγμα Dijkstra



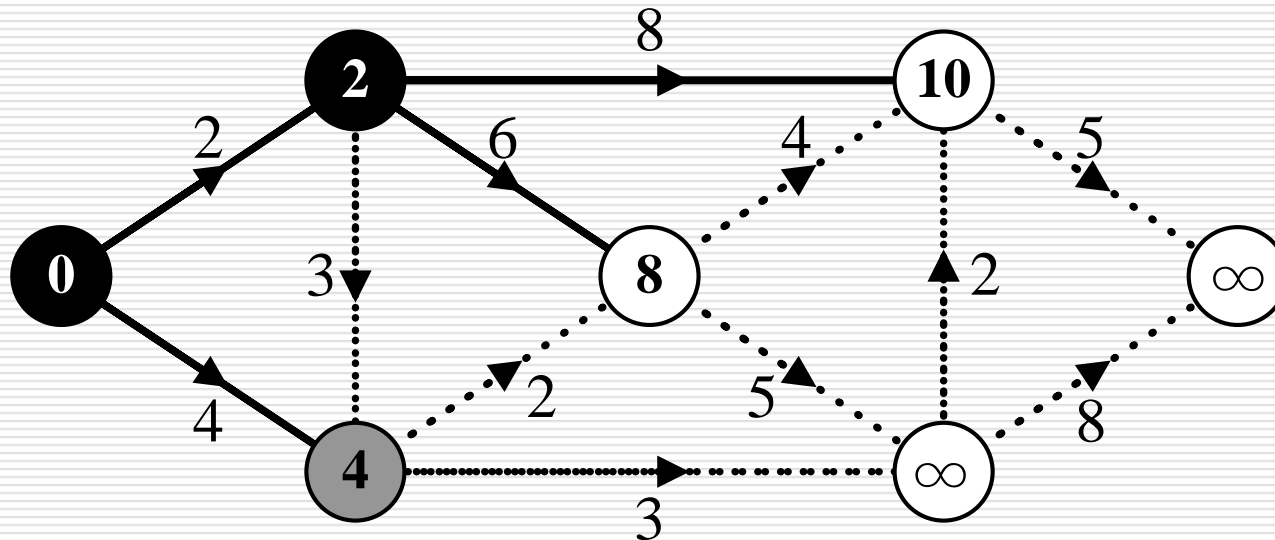
Οι ετικέτες των κόμβων δείχνουν την μέχρι στιγμής ελάχιστη απόσταση από τον αρχικό κόμβο (πίνακας D).

2^ο παράδειγμα Dijkstra



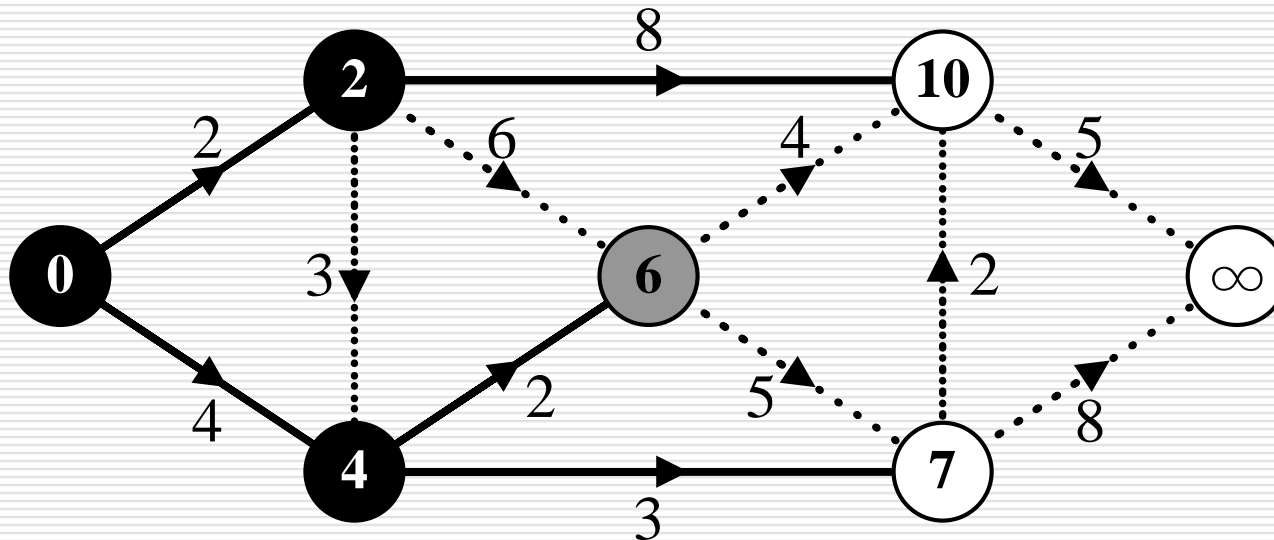
Οι ετικέτες των κόμβων δείχνουν την μέχρι στιγμής ελάχιστη απόσταση από τον αρχικό κόμβο (πίνακας D), οι συνεχείς ακμές δείχνουν ποιος είναι ο προηγούμενος κόμβος (πίνακας P).

2^ο παράδειγμα Dijkstra



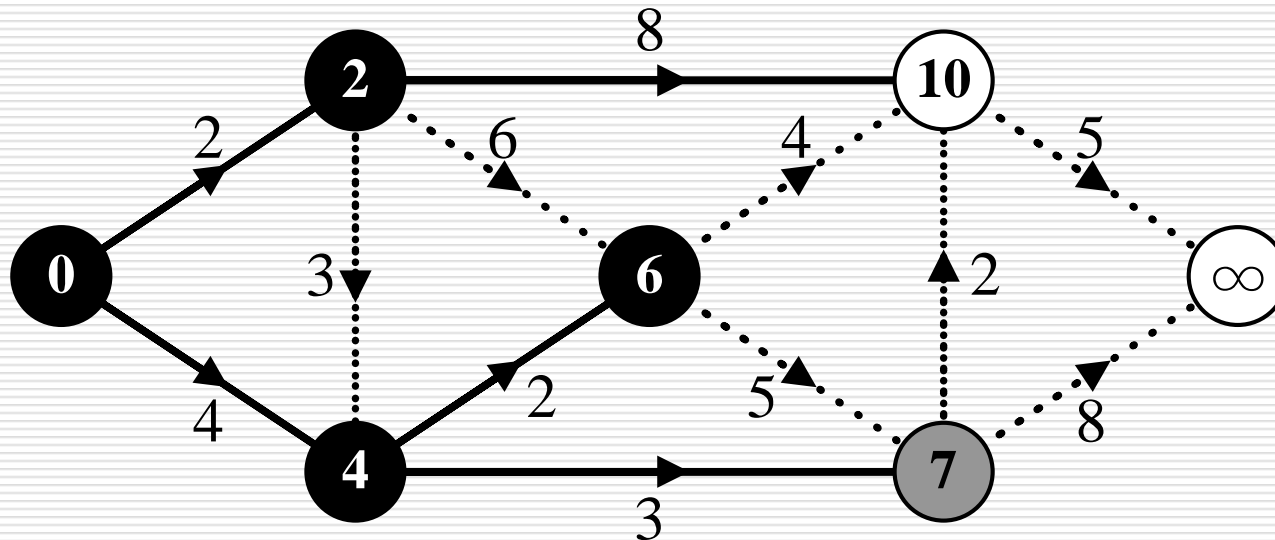
Οι ετικέτες των κόμβων δείχνουν την μέχρι στιγμής ελάχιστη απόσταση από τον αρχικό κόμβο (πίνακας D), οι συνεχείς ακμές δείχνουν ποιος είναι ο προηγούμενος κόμβος (πίνακας P).

2^ο παράδειγμα Dijkstra



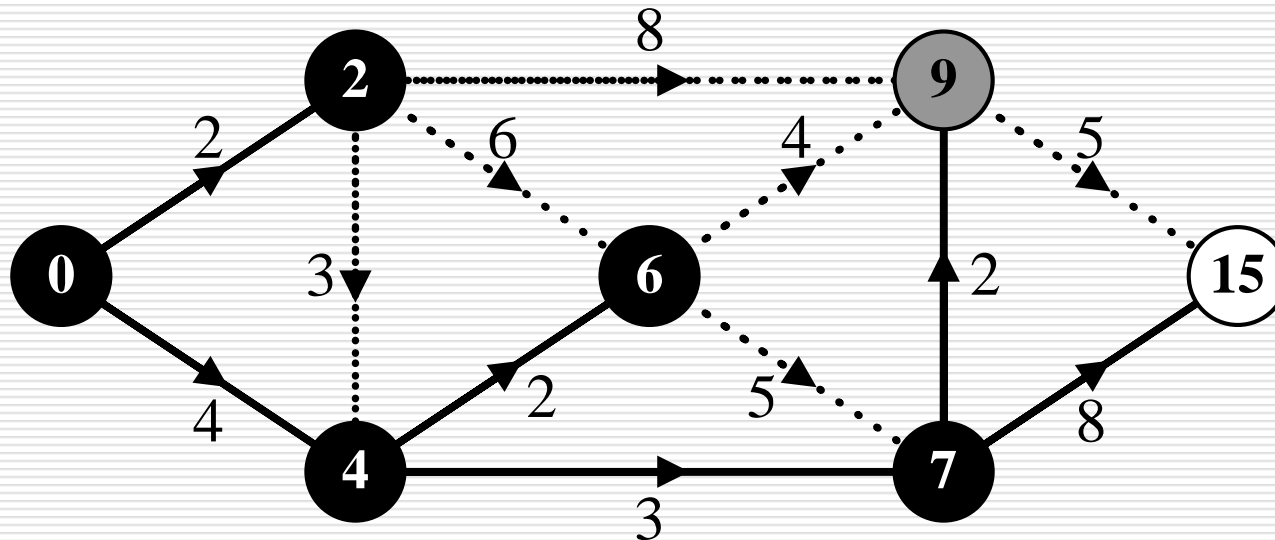
Οι ετικέτες των κόμβων δείχνουν την μέχρι στιγμής ελάχιστη απόσταση από τον αρχικό κόμβο (πίνακας D), οι συνεχείς ακμές δείχνουν ποιος είναι ο προηγούμενος κόμβος (πίνακας P).

2^ο παράδειγμα Dijkstra



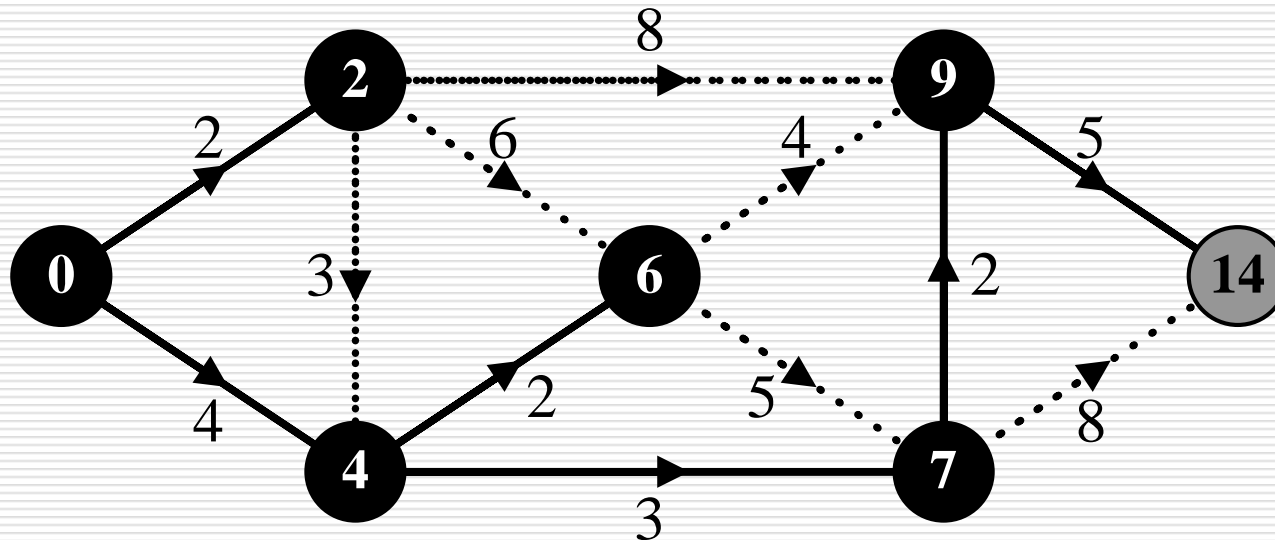
Οι ετικέτες των κόμβων δείχνουν την μέχρι στιγμής ελάχιστη απόσταση από τον αρχικό κόμβο (πίνακας D), οι συνεχείς ακμές δείχνουν ποιος είναι ο προηγούμενος κόμβος (πίνακας P).

2^ο παράδειγμα Dijkstra



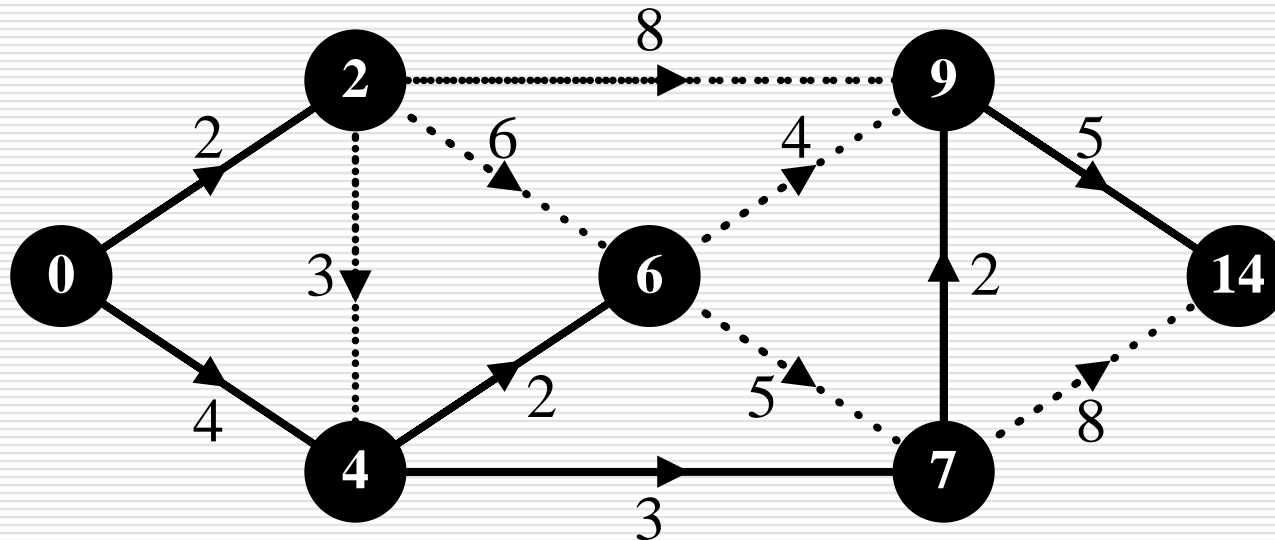
Οι ετικέτες των κόμβων δείχνουν την μέχρι στιγμής ελάχιστη απόσταση από τον αρχικό κόμβο (πίνακας D), οι συνεχείς ακμές δείχνουν ποιος είναι ο προηγούμενος κόμβος (πίνακας P).

2^ο 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 προηγούμενος κόμβος (πίνακας P).

2^ο 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 προηγούμενος κόμβος (πίνακας P).

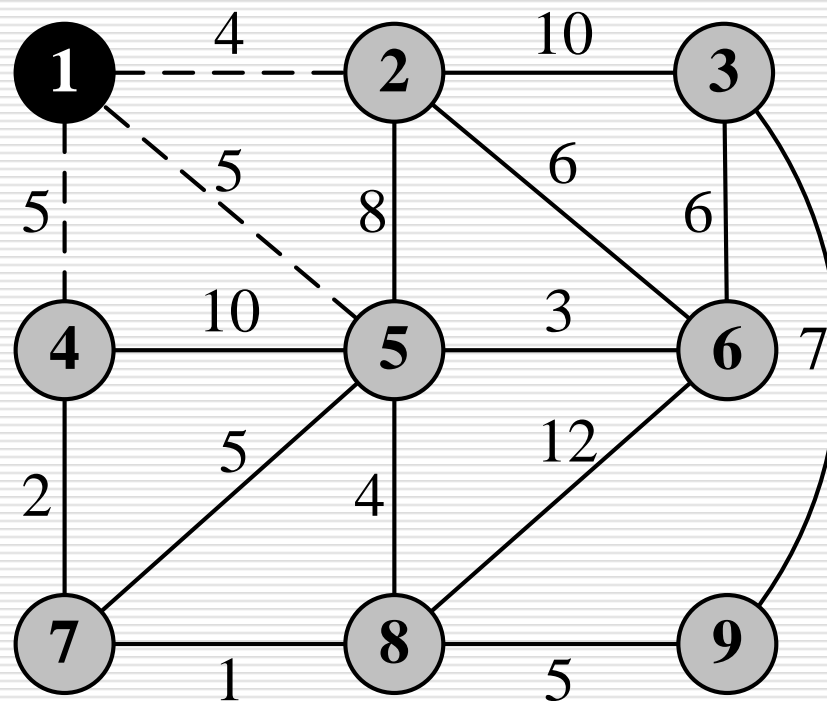
Ελάχιστο Συνδετικό Δένδρο (MST)

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

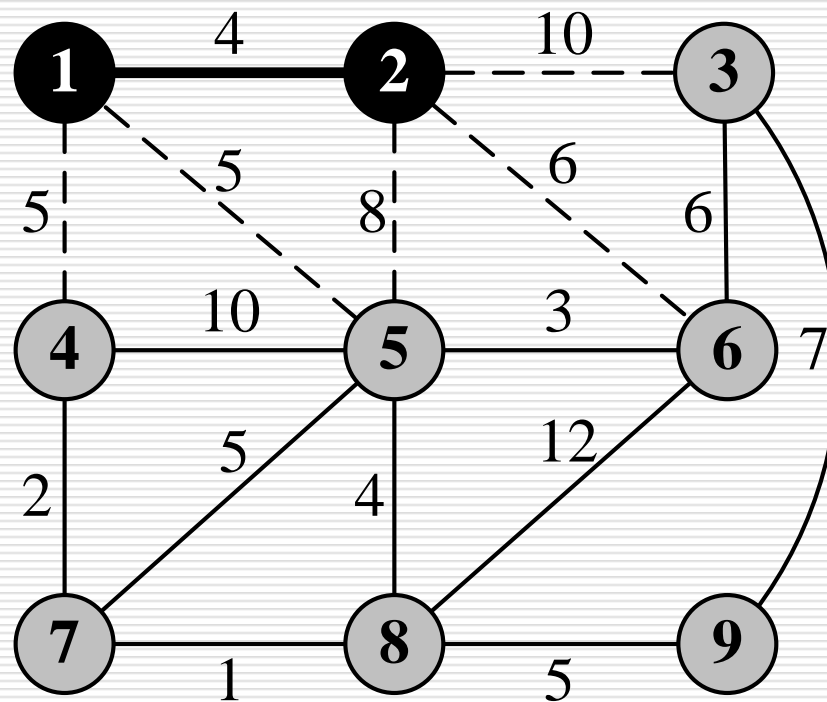
Αλγόριθμος Prim

- Επιλέγεται ένας αρχικός κόμβος, π.χ. ο κόμβος v . Η απόσταση του αρχικού κόμβου τίθεται στο 0 , ενώ των υπόλοιπων κόμβων στο ∞ .
- 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 υπόλοιπων κόμβων από το δένδρο με βάση το κόστος των ακμών (w, u_i) :
if $\text{cost}(w, u_i) < \text{dist}(u_i)$ **then** $\text{dist}(u_i) := \text{cost}(w, u_i)$
- **Πολυπλοκότητα:** $O(|V|^2)$, υλοποίηση παρόμοια με Dijkstra

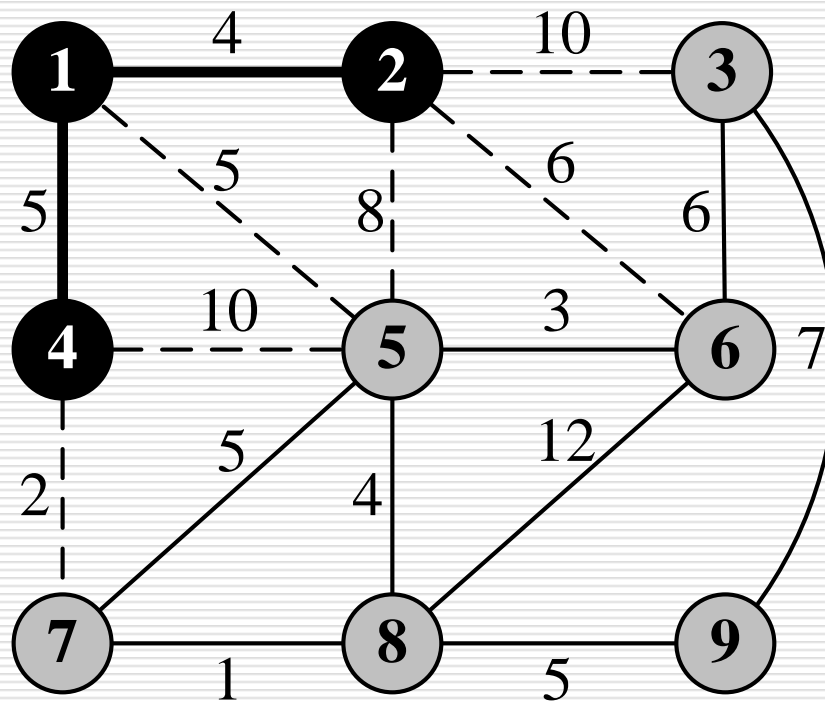
Αλγόριθμος Prim



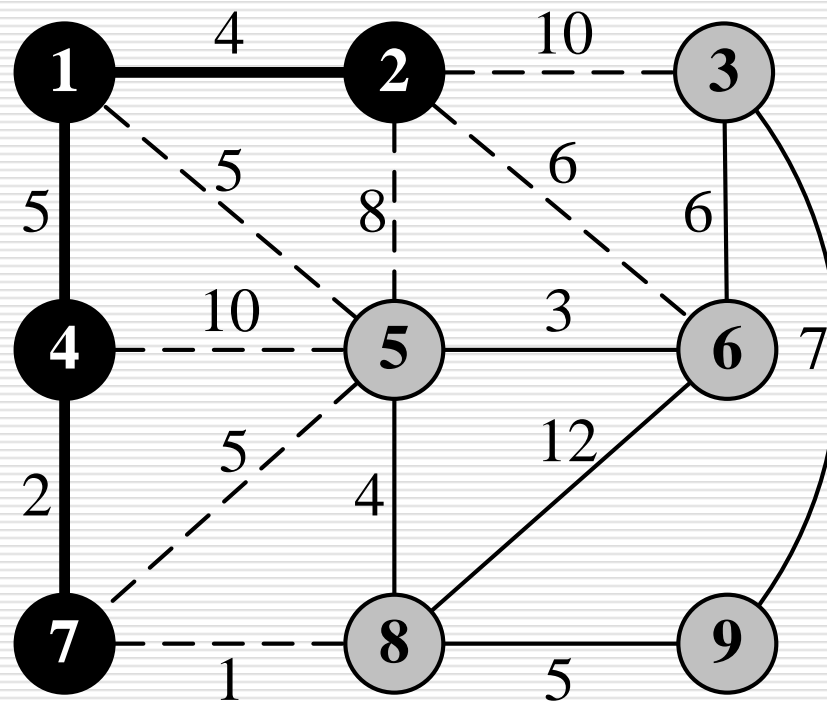
Αλγόριθμος Prim



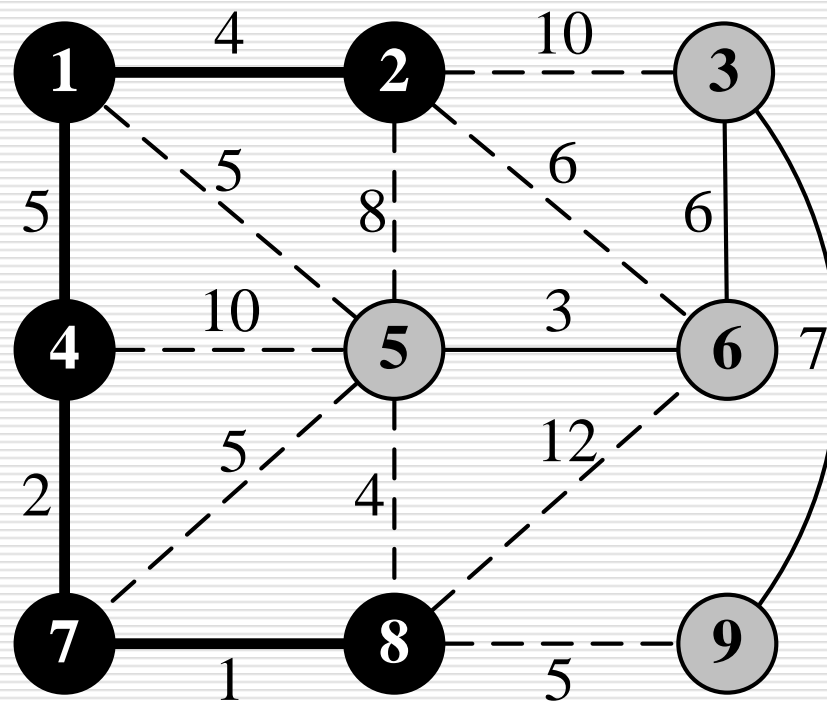
Αλγόριθμος Prim



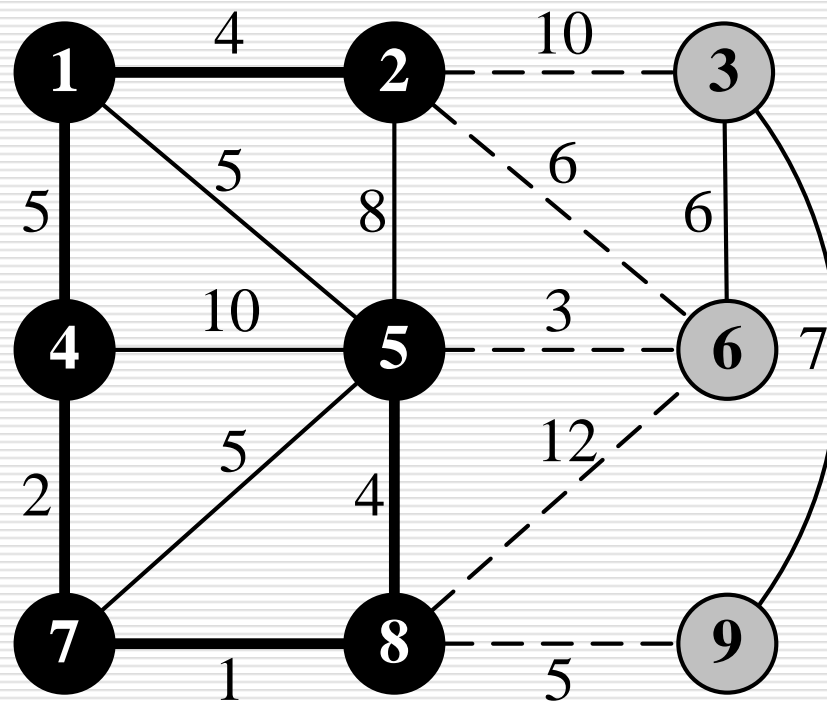
Αλγόριθμος Prim



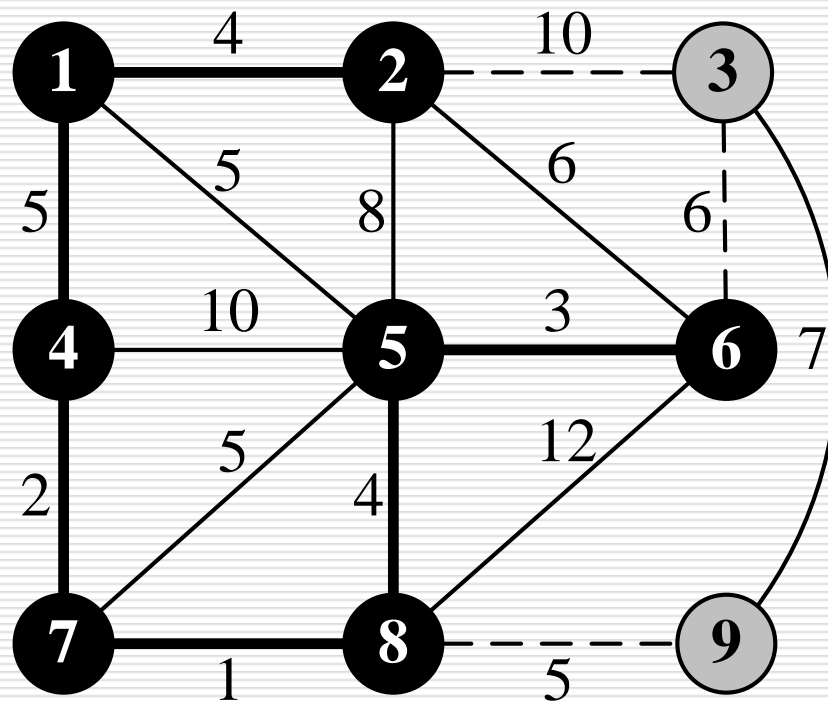
Αλγόριθμος Prim



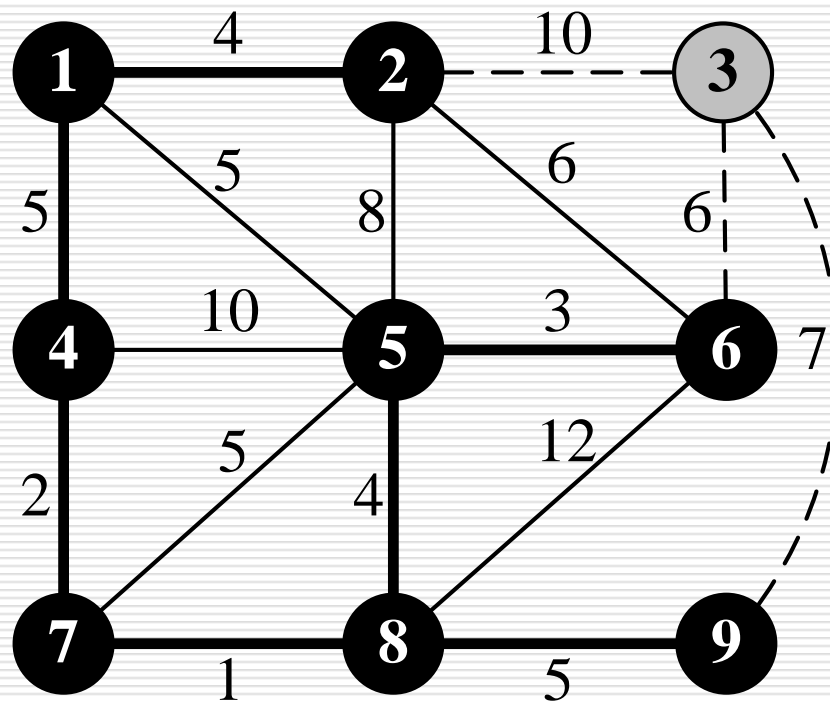
Αλγόριθμος Prim



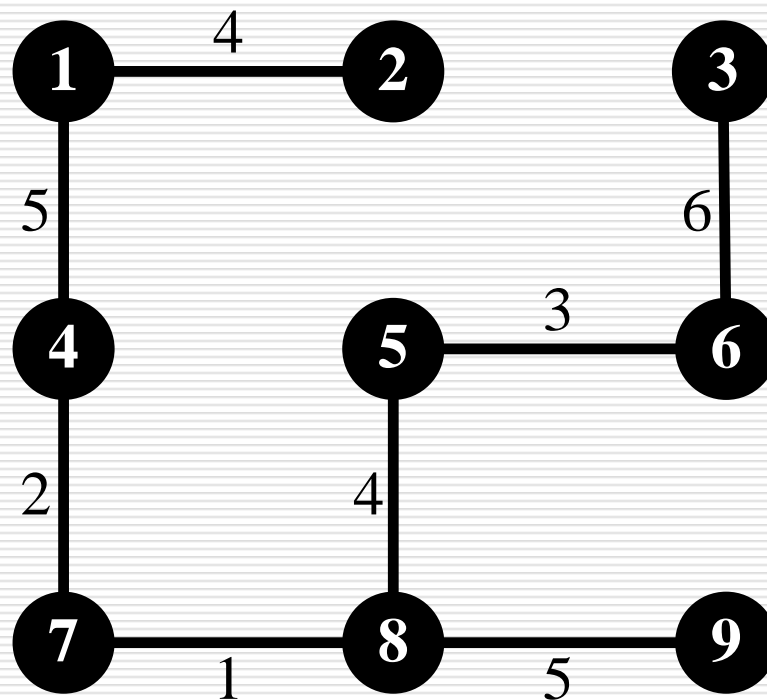
Αλγόριθμος Prim



Αλγόριθμος Prim



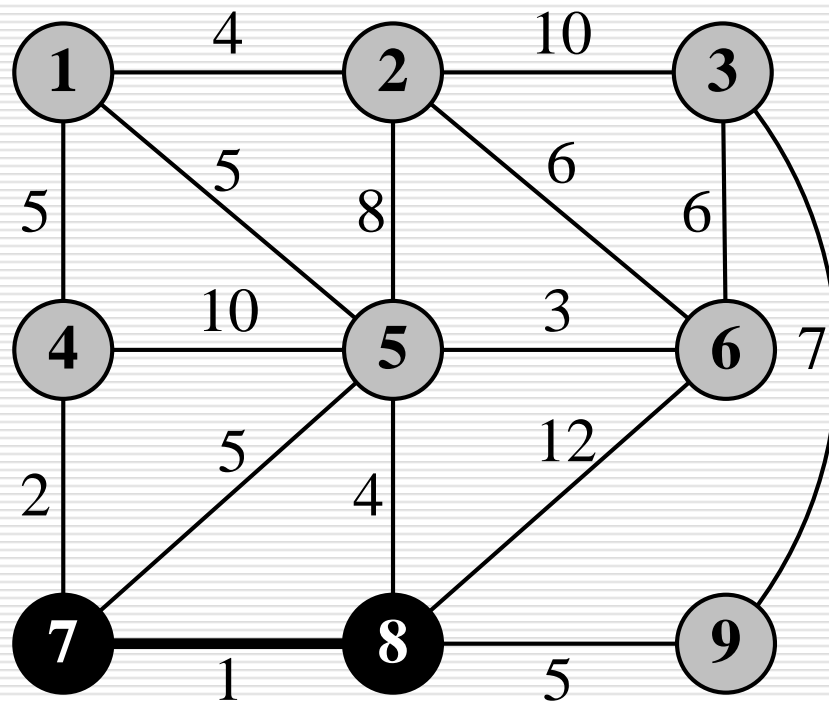
Αλγόριθμος Prim



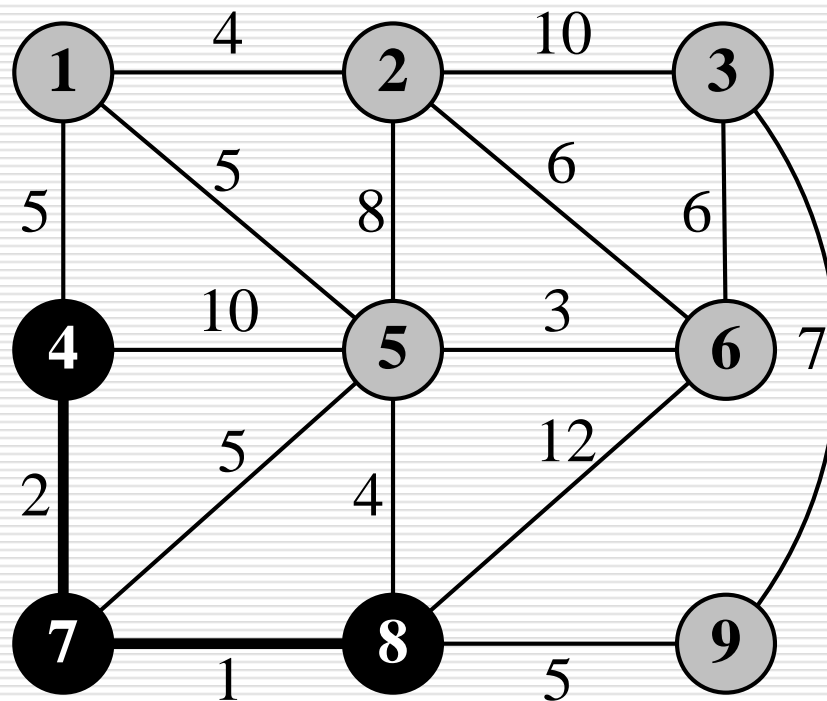
Αλγόριθμος Kruskal

- 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, Union by Rank).
- **Πολυπλοκότητα: $O(|E|\log|V|)$**

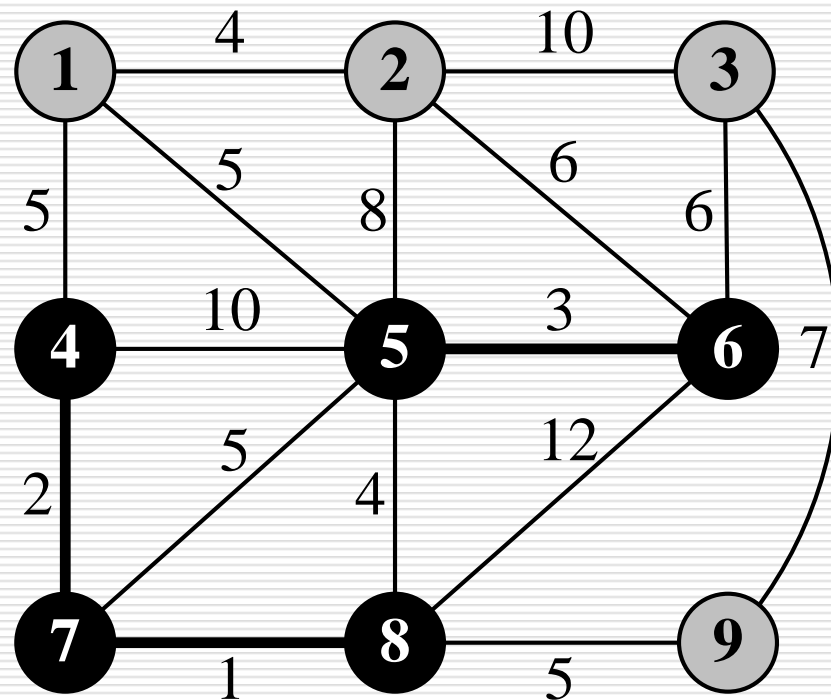
Αλγόριθμος Kruskal



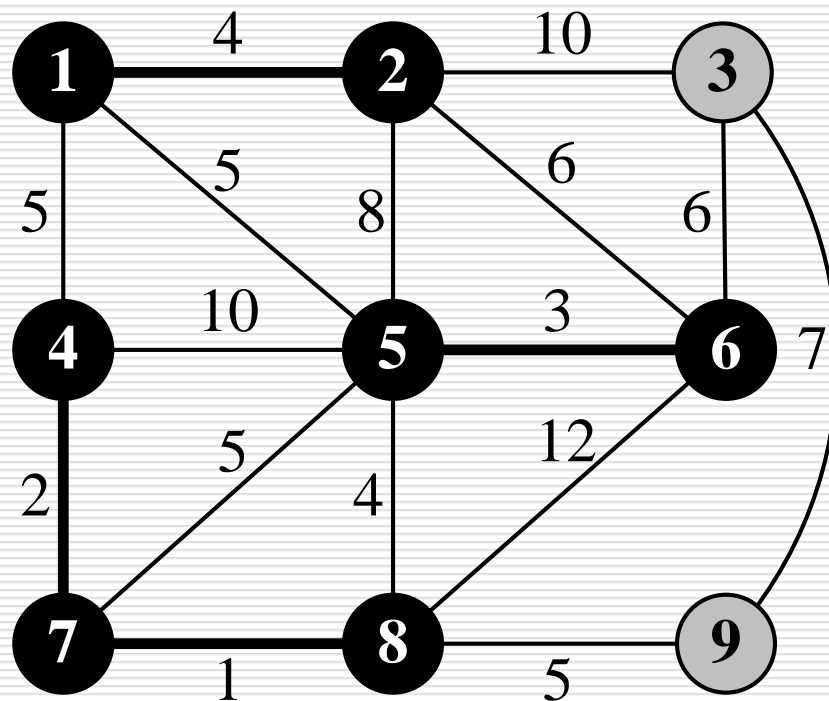
Αλγόριθμος Kruskal



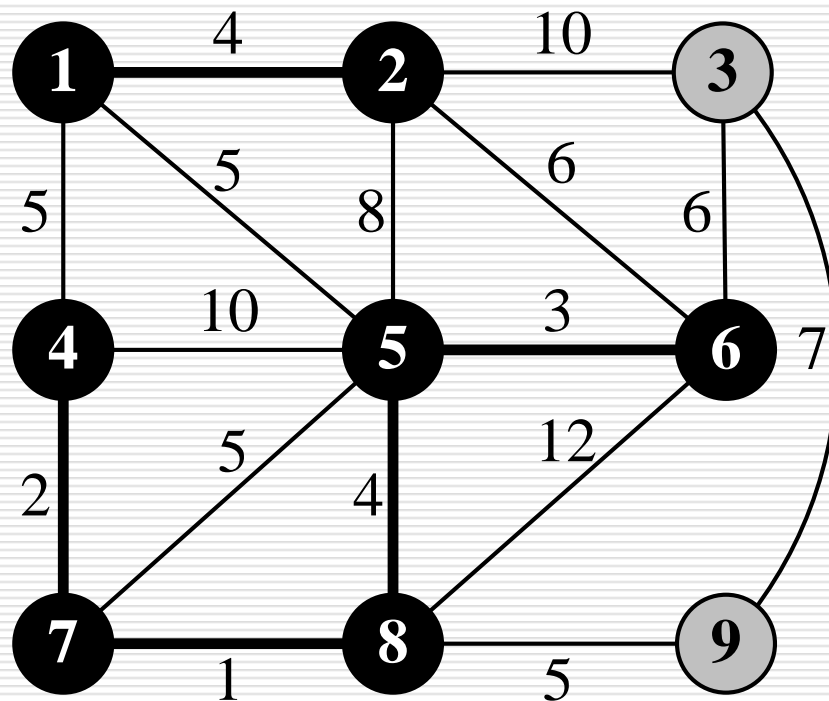
Αλγόριθμος Kruskal



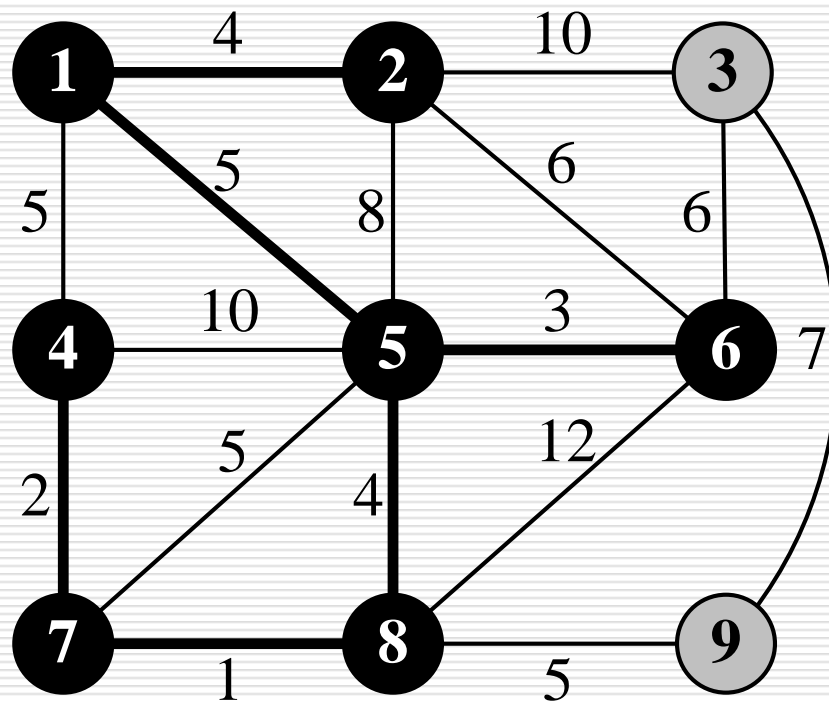
Αλγόριθμος Kruskal



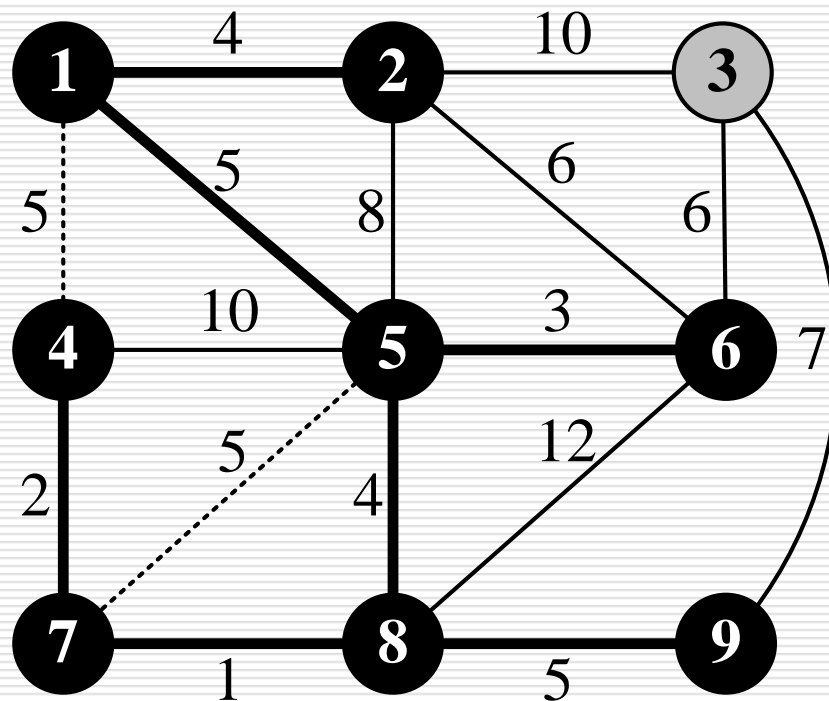
Αλγόριθμος Kruskal



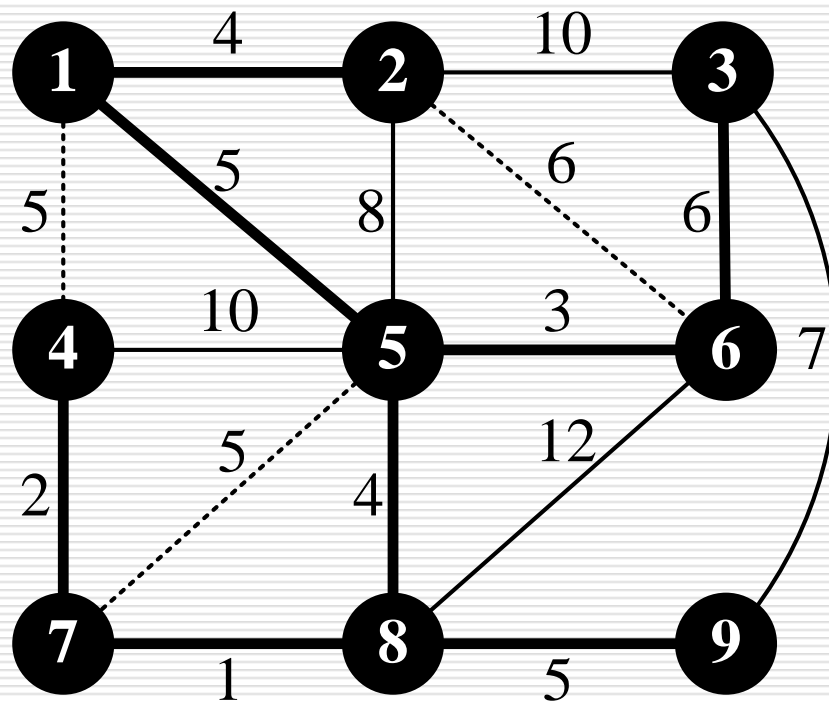
Αλγόριθμος Kruskal



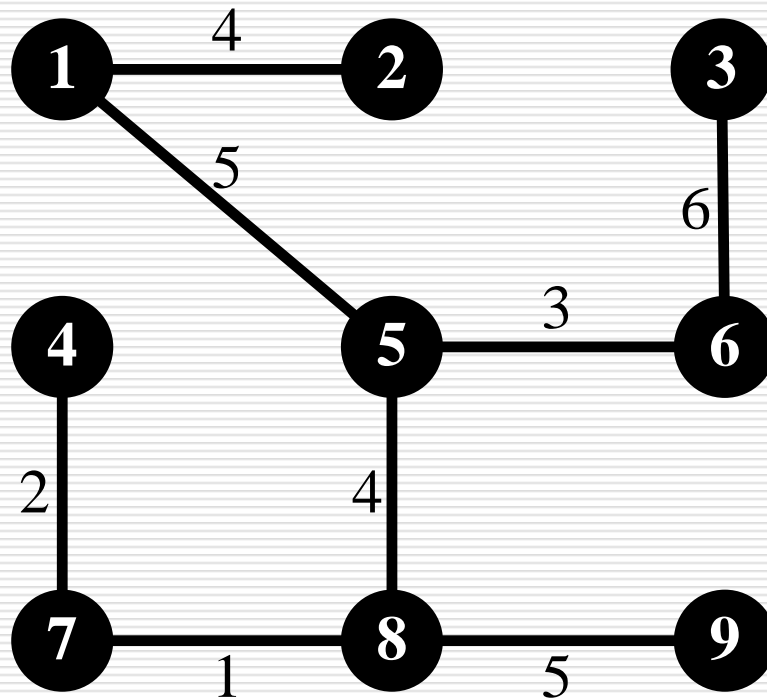
Αλγόριθμος Kruskal



Αλγόριθμος Kruskal



Αλγόριθμος Kruskal



Κοινή ιδέα Prim-Kruskal

Ξεκινώντας από τον γράφο χωρίς ακμές και ενώνοντας επαναληπτικά δύο **οποιαδήποτε** συμπληρωματικά υποσύνολα κόμβων S και $V \setminus S$, που ακόμη δεν έχουν ακμή μεταξύ τους, με την **ελαφρύτερη ακμή** καταλήγουμε σε ελάχιστο συνδετικό δένδρο

Γιατί δουλεύει η ιδέα;

Θεώρημα. Ένα σύνολο ακμών A που είναι *υποσχόμενο* (= υποσύνολο ενός MST) παραμένει υποσχόμενο αν του προσθέσουμε την *ελαφρύτερη* ακμή μεταξύ οποιασδήποτε συνεκτικής συνιστώσας (connected component) του τρέχοντος γράφου (που ορίζεται από τις ακμές του A) και του υπόλοιπου τρέχοντος γράφου.

Απόδειξη. Στον πίνακα.

Εφαρμογή: 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).

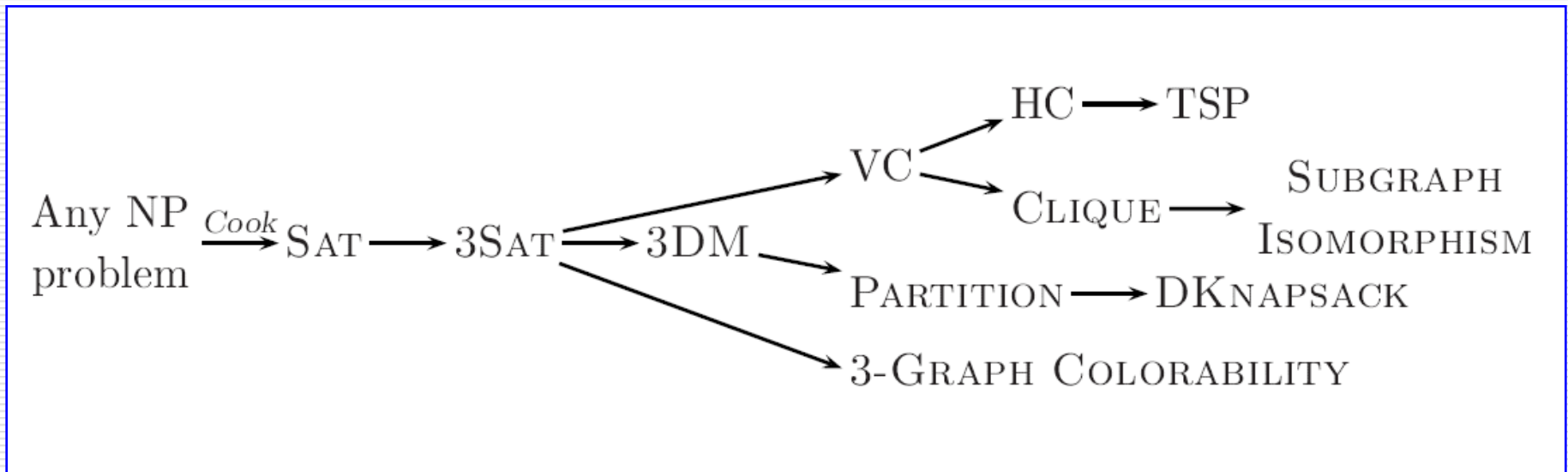
Προσφέρεται για *παράλληλη / κατανεμημένη* υλοποίηση.

NP-πλήρη Προβλήματα Γράφων

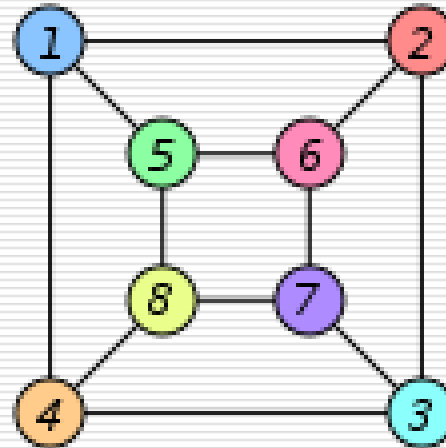
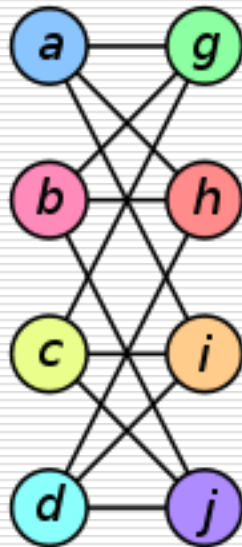
- VERTEX COVER (VC)
- CLIQUE
- HAMILTON CIRCUIT (HC)
- TRAVELING SALESMAN (TSP)
- 3-COLORABILITY
- SUBGRAPH ISOMORPHISM
- 3-DIMENSIONAL MATCHING (3DM)

NP-πλήρη Προβλήματα Γράφων

Απόδειξη NP-πληρότητας: *αναγωγές*



«Ενδιάμεση» Πολυπλοκότητα;



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