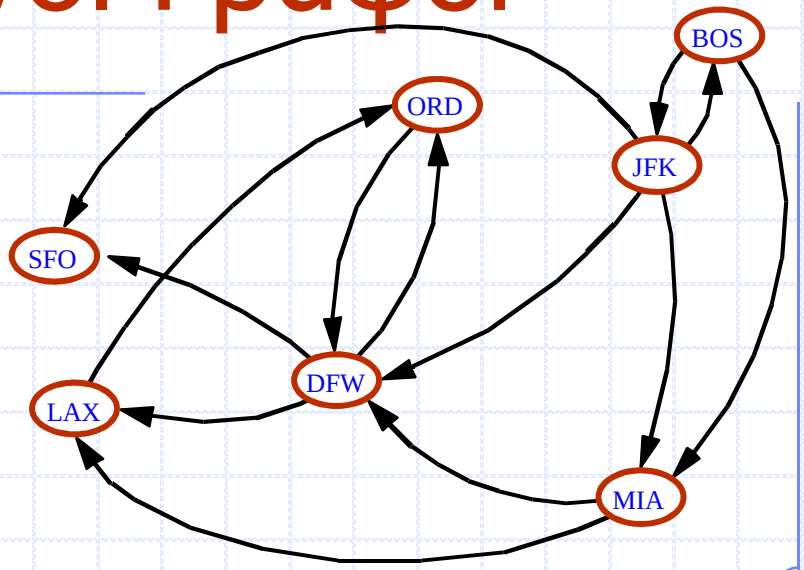
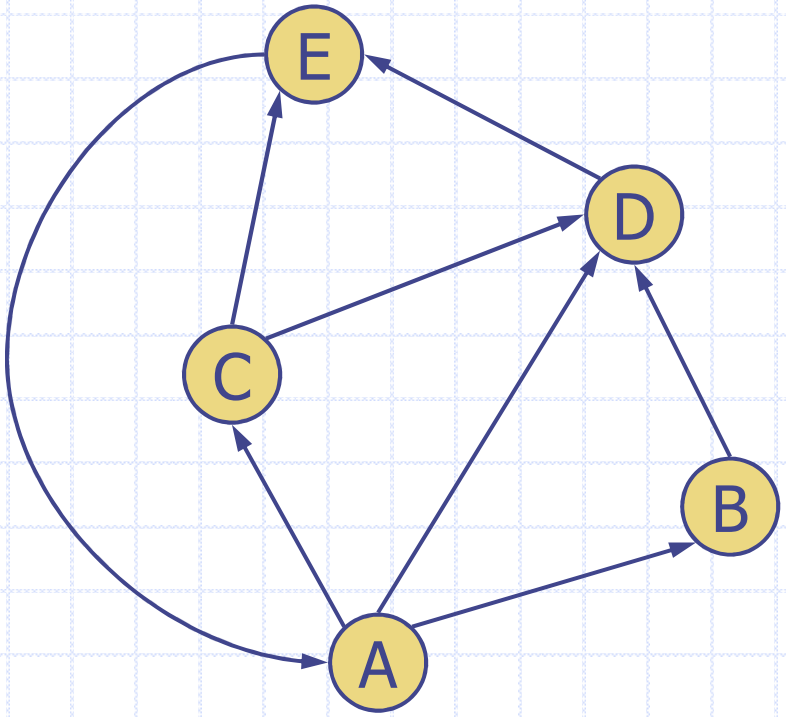
Κατευθυνόμενοι Γράφοι

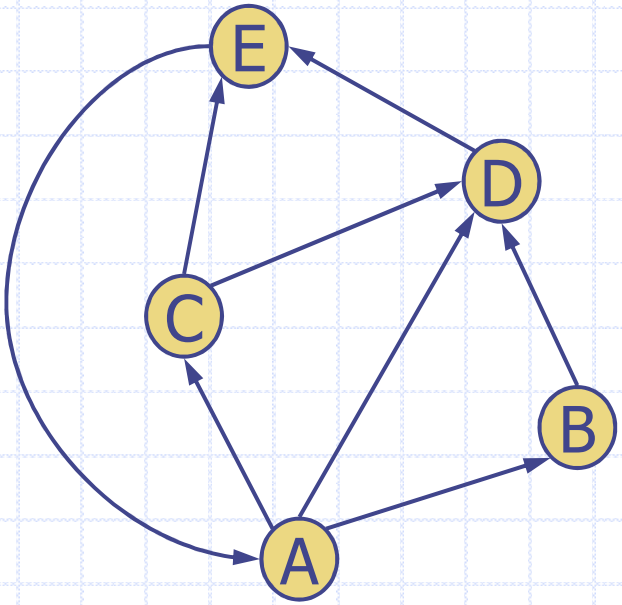


Κατευθυνόμενοι Γράφοι (Digraphs-Δίγραφοι)

- Ένας δίγραφος είναι ένας γράφος με όλες τις ακμές κατευθυνόμενες
  - Συντομία του "directed graph"
- Εφαρμογές
  - δρόμοι μιας κατεύθυνσης
  - πτήσεις
  - χρονοδρομολόγηση



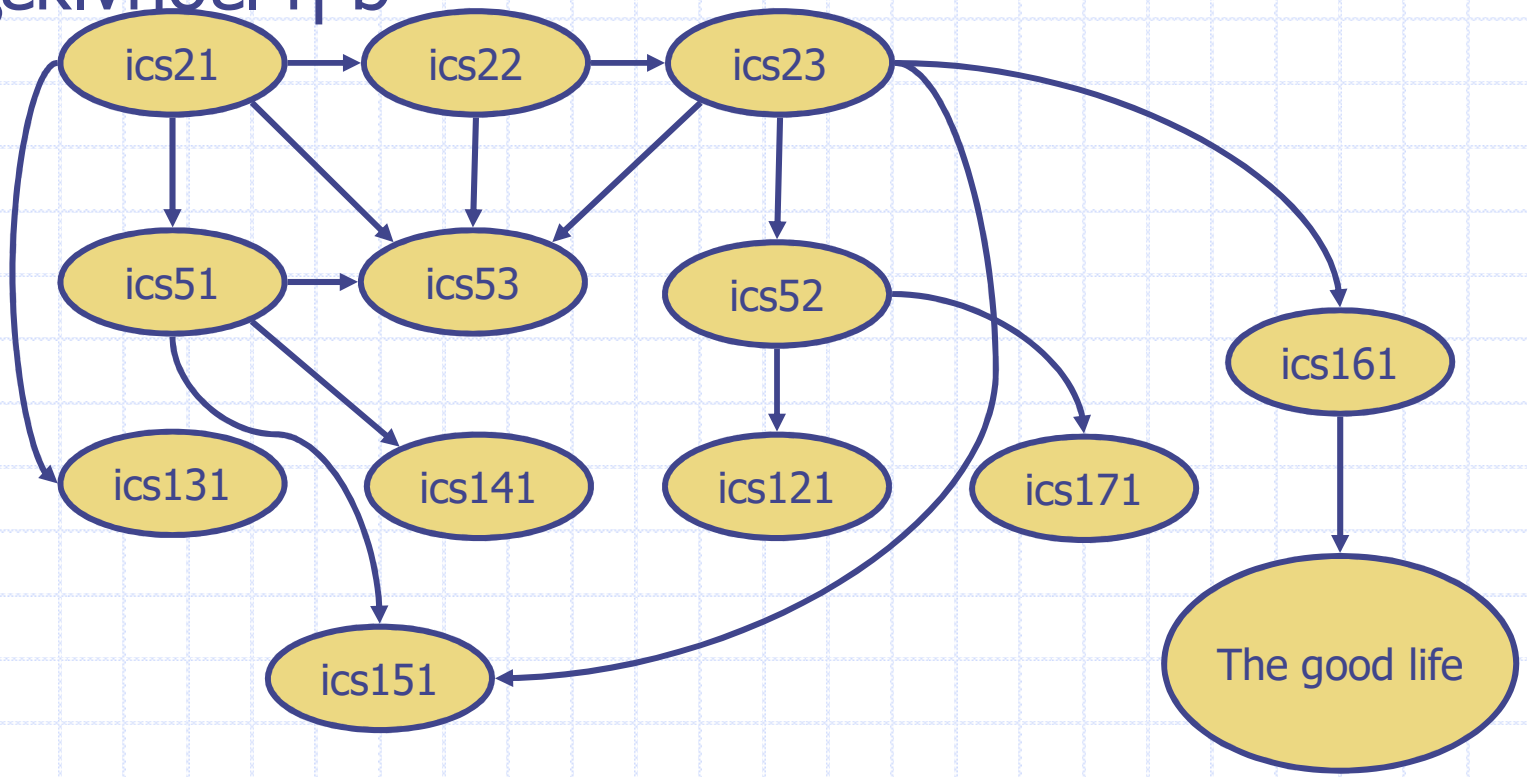
Ιδιότητες Δίγραφων



- Ένα γράφος G=(V,E) έτσι ώστε
  - Κάθε ακμή έχει μια κατεύθυνση:
  - Η ακμή (a,b) πάει από το a στο b, αλλά όχι από το b στο a
- Αν ο G είναι απλός, $m \leq n \cdot (n-1)$
- Αν καταχωρούμε τις ακμές που προσπίπτουν και αυτές που φεύγουν σε ξεχωριστές λίστες γειτνίασης, μπορούμε να καταγράψουμε τις προσπίπτουσες και τις πλευρές που φεύγουν σε χρόνο ανάλογο του μεγέθους τους

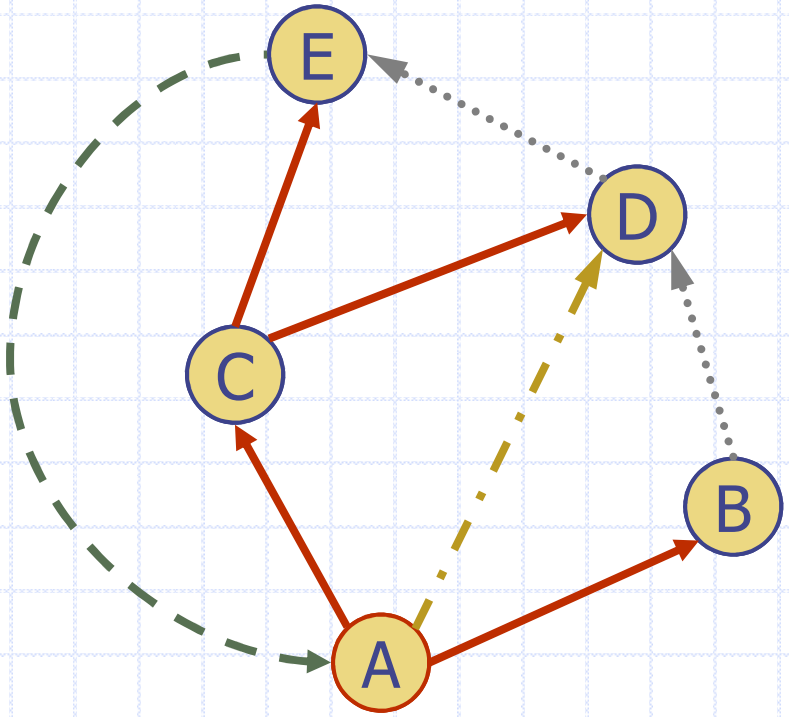
Εφαρμογή

- Δρομολόγηση: η πλευρά (a,b) σημαίνει ότι η εργασία a πρέπει να ολοκληρωθεί πριν ξεκινήσει η b



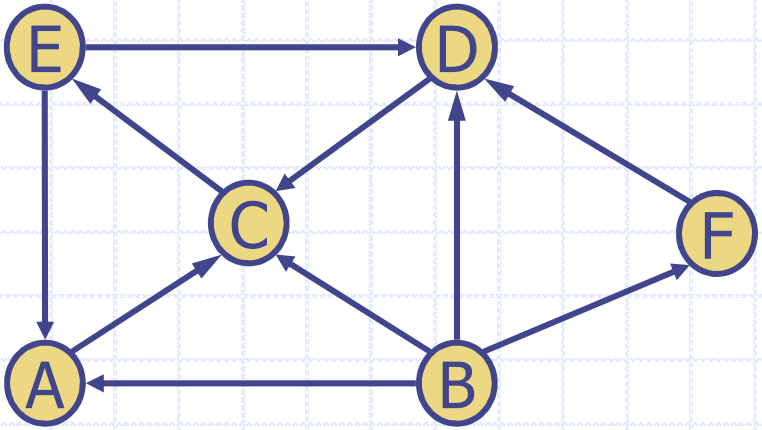
Κατευθυνόμενο DFS

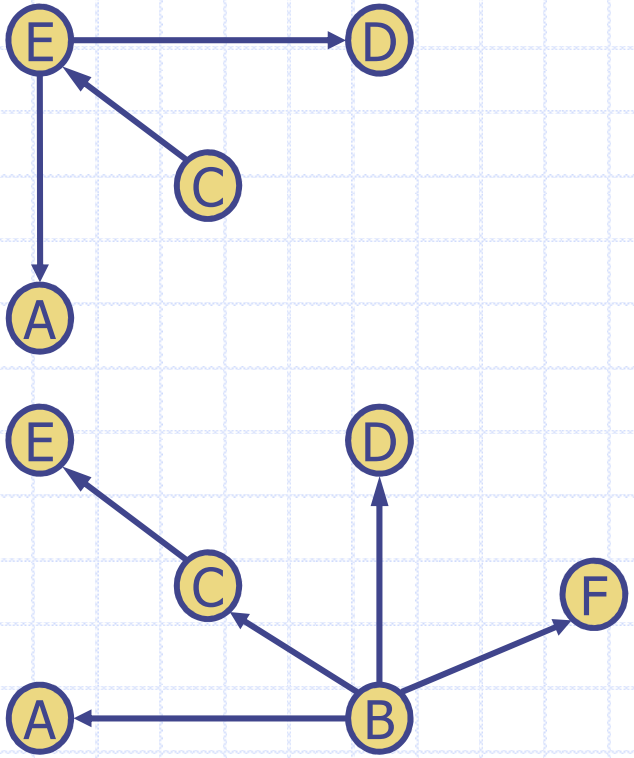
- Μπορούμε να εξειδικεύσουμε τους αλγόριθμους σάρωσης (DFS και BFS) σε δίγραφους σαρώνοντας τις ακμές μόνο κατά την φορά τους
- Στο κατευθυνόμενο DFS αλγόριθμο, έχουν τεσσάρων ειδών τύπους ακμών
  - ακμές που έχουν ανακαλυφθεί
  - ακμές οπισθοχώρησης
  - ακμές προώθησης
  - ακμές διασταύρωσης
- Ένα κατευθυνόμενο DFS που ξεκινάει από την κορυφή s καθορίζει τις κορυφές που είναι προσπελάσιμες από το s



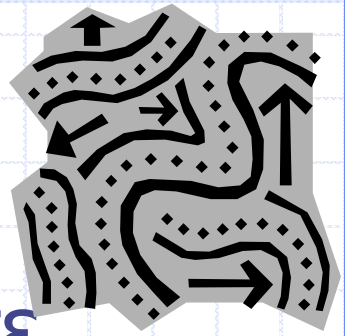
Προσβασιμότητα

- Το DFS δένδρο με ρίζα στο v: οι προσβάσιμες κορυφές από την v μέσω κατευθυνόμενων διαδρομών

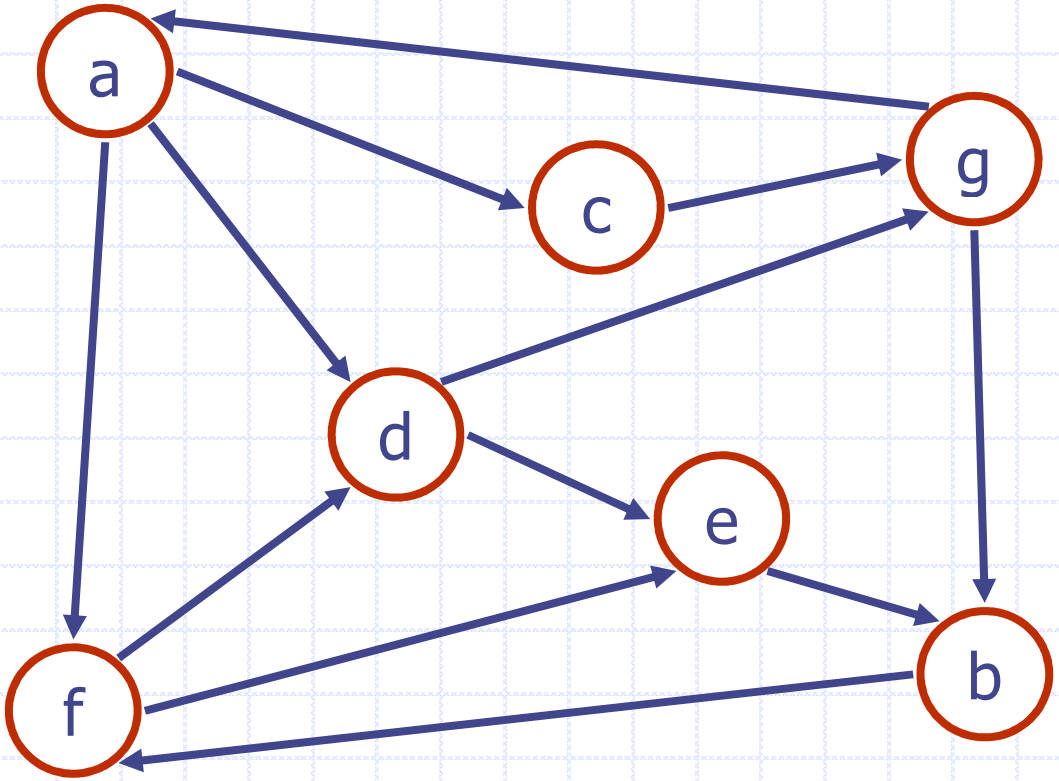




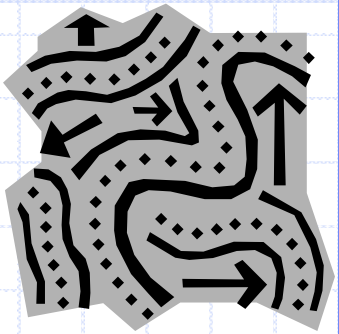
Ισχυρή συνεκτικότητα



- Κάθε κορυφή μπορεί να φθάσει σε όλες τις άλλες κορυφές

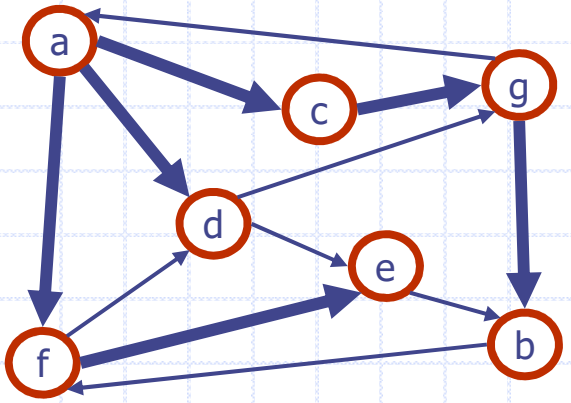


Αλγόριθμος Ισχυρής Συνεκτικότητας

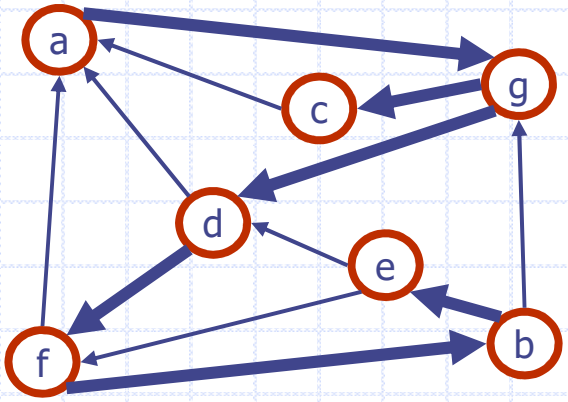


- Επιλέγουμε νια κορυφή v στον G
- Εκτελούμε μια DFS από την v στον G
  - Αν υπάρχει κάποια κορυφή w που δεν έχουμε επισκεφθεί, τύπωσε 'όχι'
- Έστω G' ο γράφος G με τις ακμές αντεστραμμένες
- Εκτέλεση ενός DFS από την v στο G'
  - Αν υπάρχει κάποια κορυφή w που δεν έχουμε επισκεφθεί, τύπωσε 'όχι'
  - Διαφορετικά, τύπωσε "ναι"
- Χρόνος τρεξίματος: O(n+m)

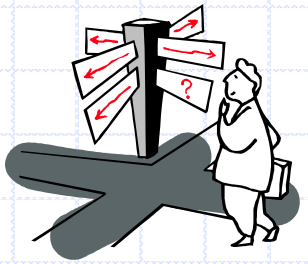
G:



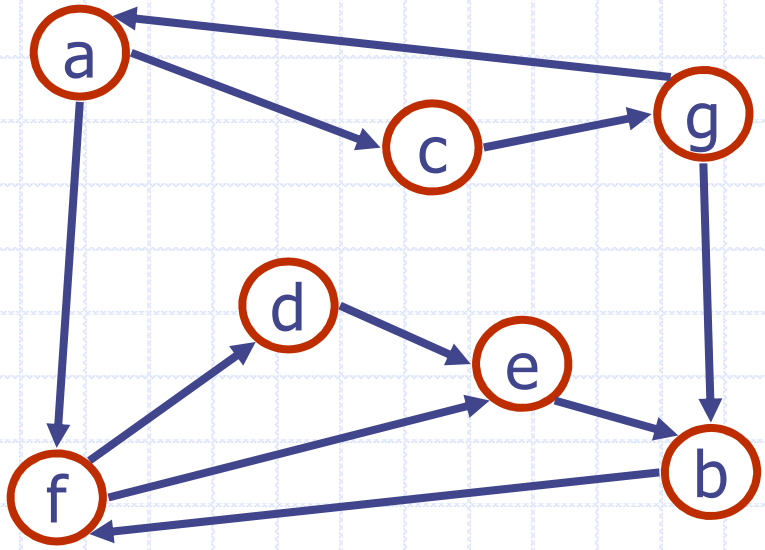
G':



Ισχυρά Συνδεδεμένες Συνιστώσες



- Maximal υπογράφοι έτσι που κάθε κορυφή μπορεί να προσπελάσει όλες τις άλλες κορυφές
- Μπορεί να γίνει σε χρόνο O(n+m) με DFS, αλλά είναι πιό πολύπλοκο

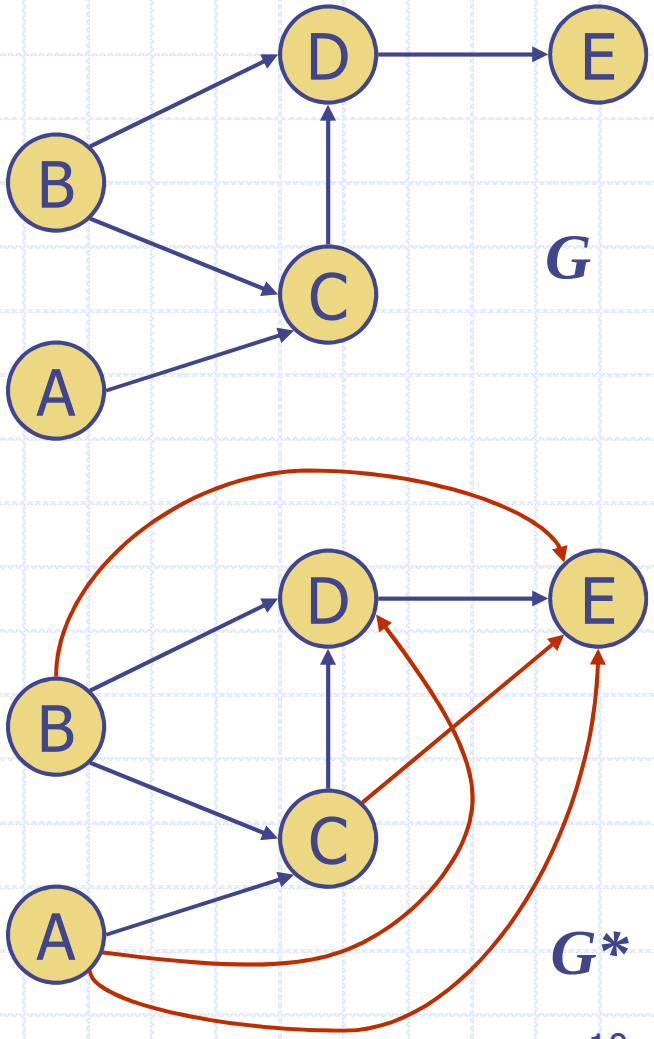


{ a , c , g }

{ f , d , e , b }

Μεταβατική Κλειστότητα

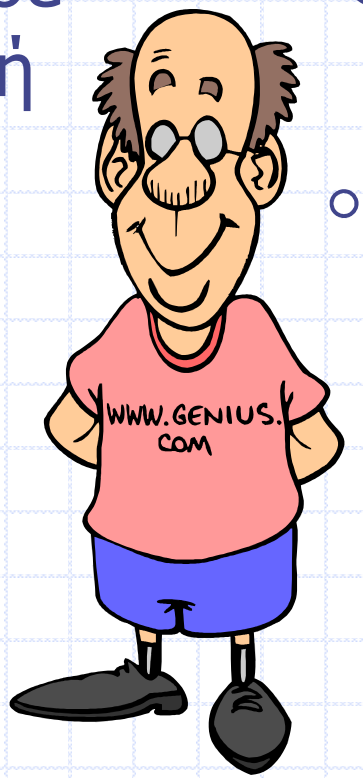
- Δίδεται ένας δίγραφος G, η μεταβατική κλειστότητα του G είναι ο δίγραφος G^* έτσι ώστε
  - G^* έχει τις ίδιες κορυφές με τον G
  - αν ο G έχει μια κατευθυνόμενη διαδρομή από την u στη v ($u \neq v$), ο G^* έχει μια κατευθυνόμενη ακμή από την u στη v
- Η μεταβατική κλειστότητα παρέχει πληροφορίες προσβασιμότητας για το δίγραφο



Υπολογισμός Μεταβατικής Κλειστότητας

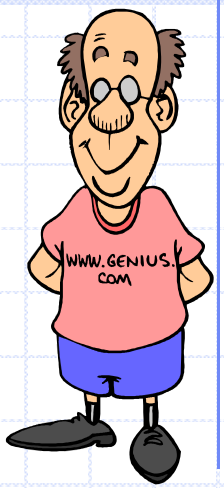
- Μπορούμε να εκτελέσουμε DFS ξεκινώντας σε κάθε κορυφή
  - O(n(n+m))

Αν υπάρχει τρόπος να πάμε από το A στο B και από το B στο C, τότε υπάρχει τρόπος και από το A στο C

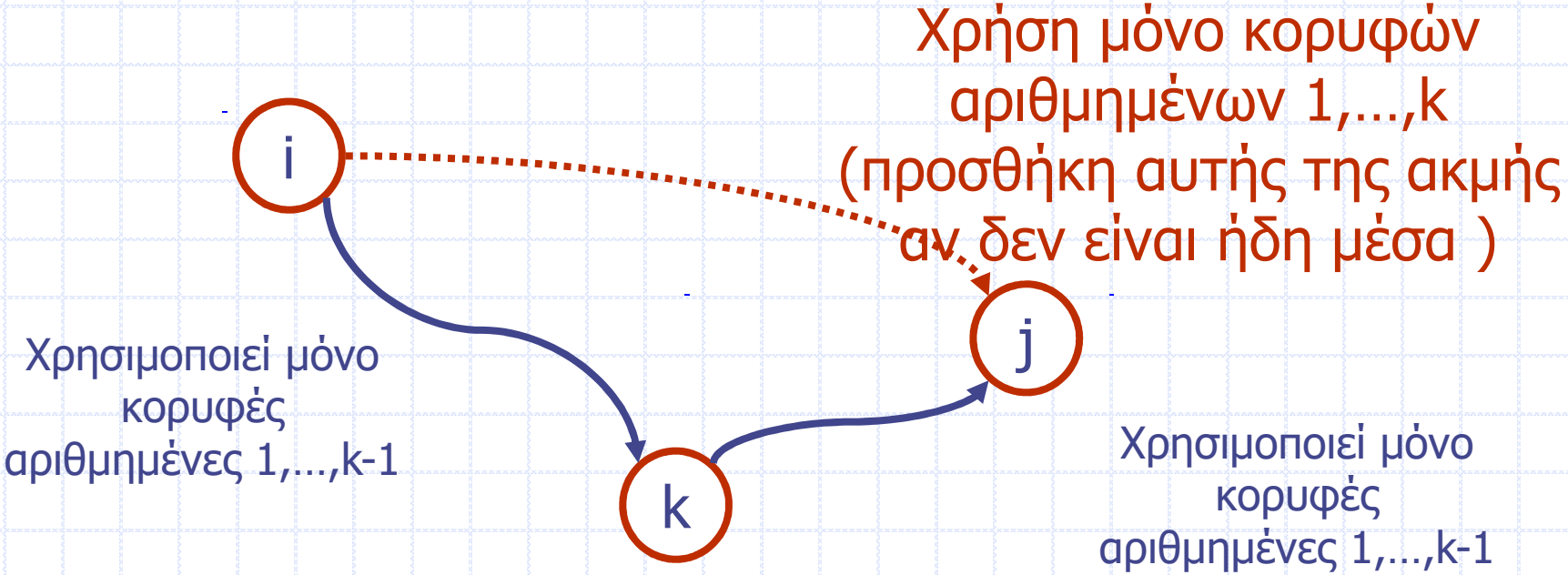


Εναλλακτικά ... Χρήση δυναμικού προγραμματισμού: Ο αλγόριθμος Floyd-Warshall

Floyd-Warshall Μεταβατική Κλειστότητα



- Ιδέα #1: Αρίθμησε τις κορυφές 1, 2, …, n.
- Ιδέα #2: Θεωρείστε τις διαδρομές που χρησιμοποιούν μόνο κορυφές αριθμημένες 1, 2, …, k, σαν ενδιάμεσες κορυφές:

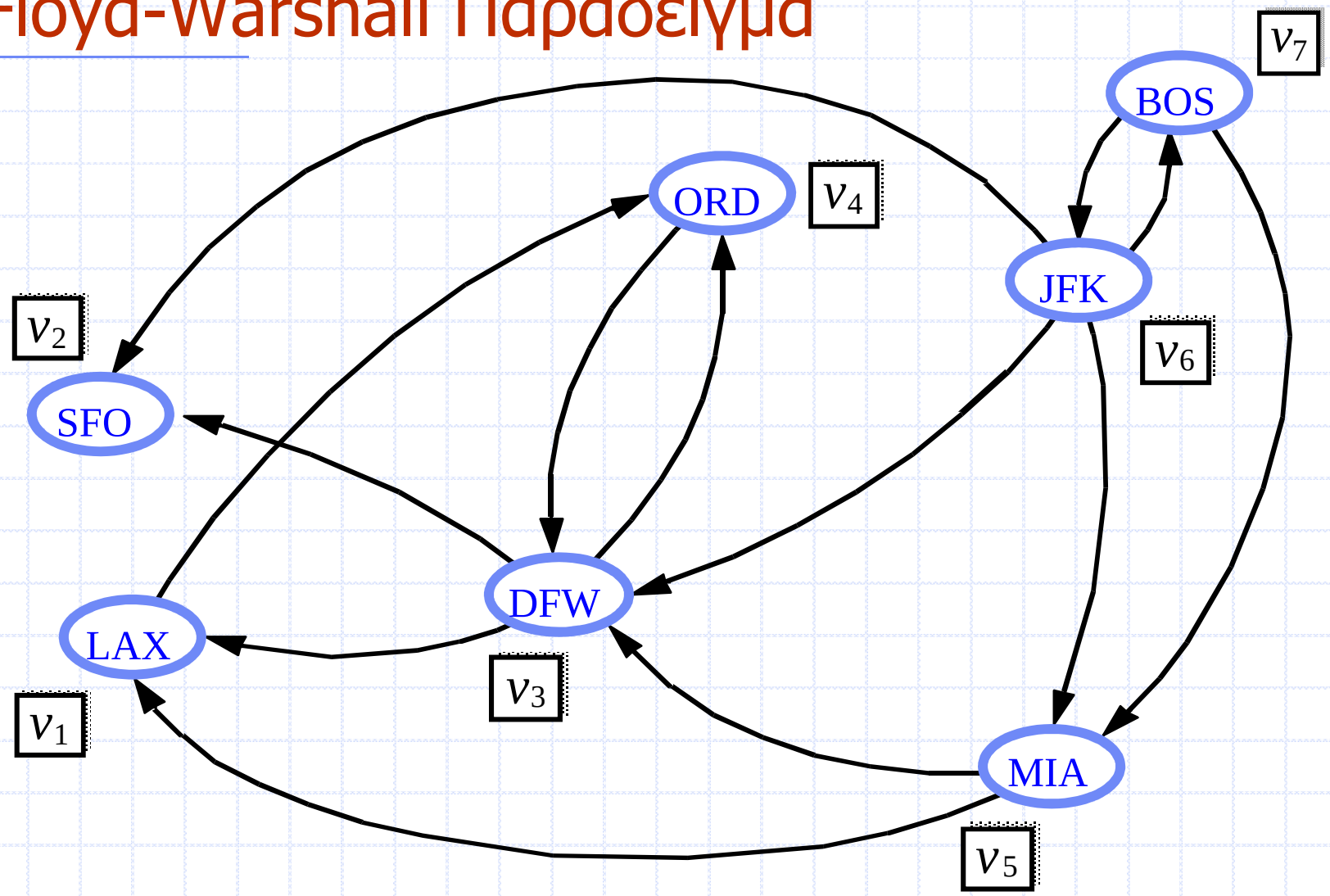


Αλγόριθμος Floyd-Warshall

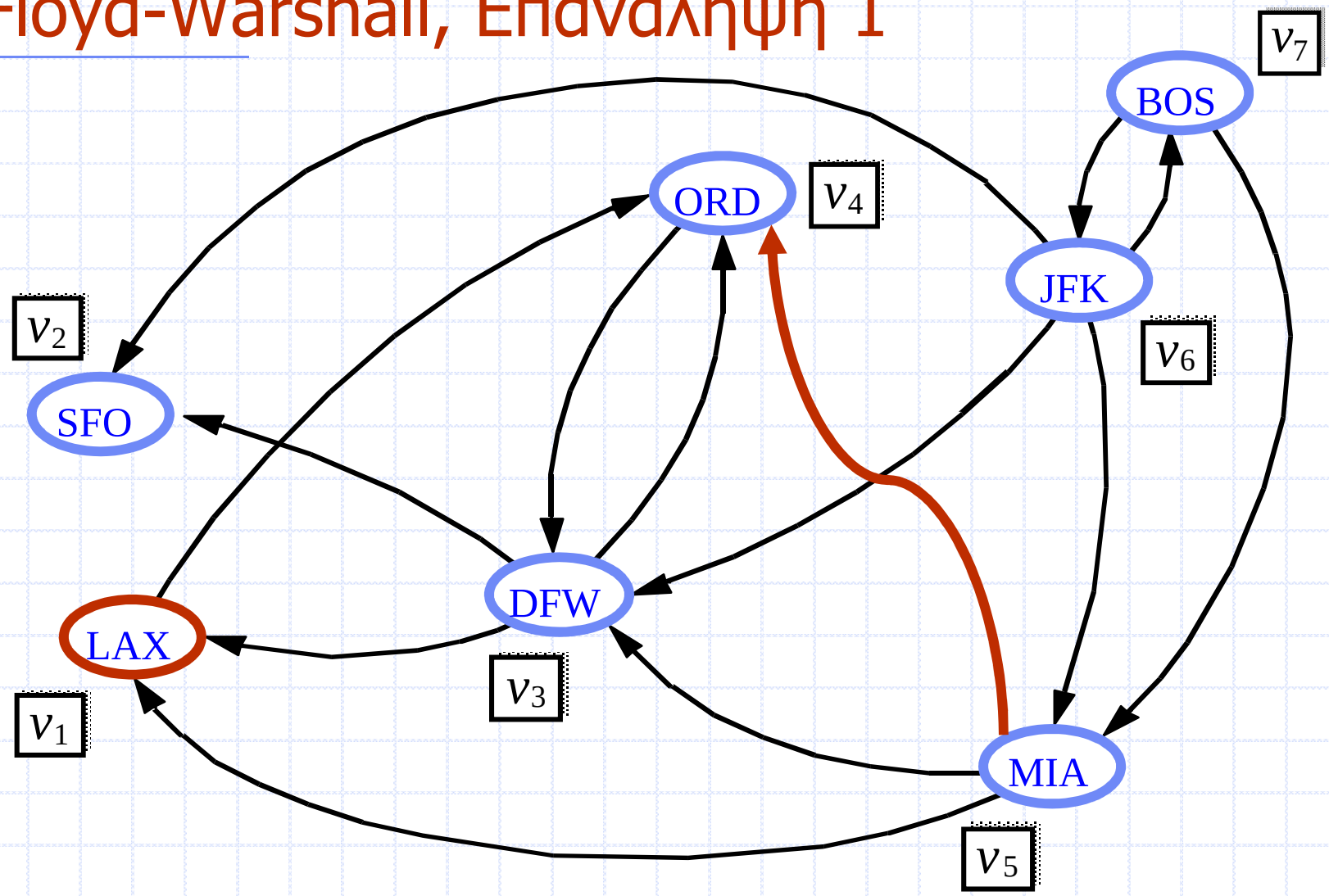
- Αρίθμησε τις κορυφές $v_1, \ldots, v_n$
- Υπολόγισε τους δίγραφους $G_0, \ldots, G_n$
  - $G_0 = G$
  - *ο* G_k έχει την κατευθυνόμενη ακμή (v_i, v_j) αν ο G έχει μια κατευθυνόμενη διαδρομή από την v_i στην v_j με ενδιάμεσες κορυφές στις $\{v_1, \ldots, v_k\}$
- Έχουμε ότι $G_n = G^*$
- Στην φάση k, υπολογίζεται ο δίγραφος G_k από τον G_{k-1}
- Χρόνος τρεξίματος: $O(n^3)$, υποθέτοντας ότι η areAdjacent είναι $O(1)$ (δηλ., πίνακας γειτνίασης)

Algorithm *FloydWarshall*(G)
  Input δίγραφος G
  Output μεταβατική κλειστότητα G^* του G
  $i \leftarrow 1$
  for all $v \in$ *G.vertices*()
    denote v as v_i
    $i \leftarrow i + 1$
  $G_0 \leftarrow G$
  for $k \leftarrow 1$ **to** n **do**
    $G_k \leftarrow G_{k-1}$
    for $i \leftarrow 1$ **to** n $(i \neq k)$ **do**
      for $j \leftarrow 1$ **to** n $(j \neq i, k)$ **do**
        if G_{k-1}.*areAdjacent*$(v_i, v_k) \wedge$
            G_{k-1}.*areAdjacent*(v_k, v_j)
          if $\neg G_k$.*areAdjacent*(v_i, v_j)
            G_k.*insertDirectedEdge*(v_i, v_j, k)
    return G_n

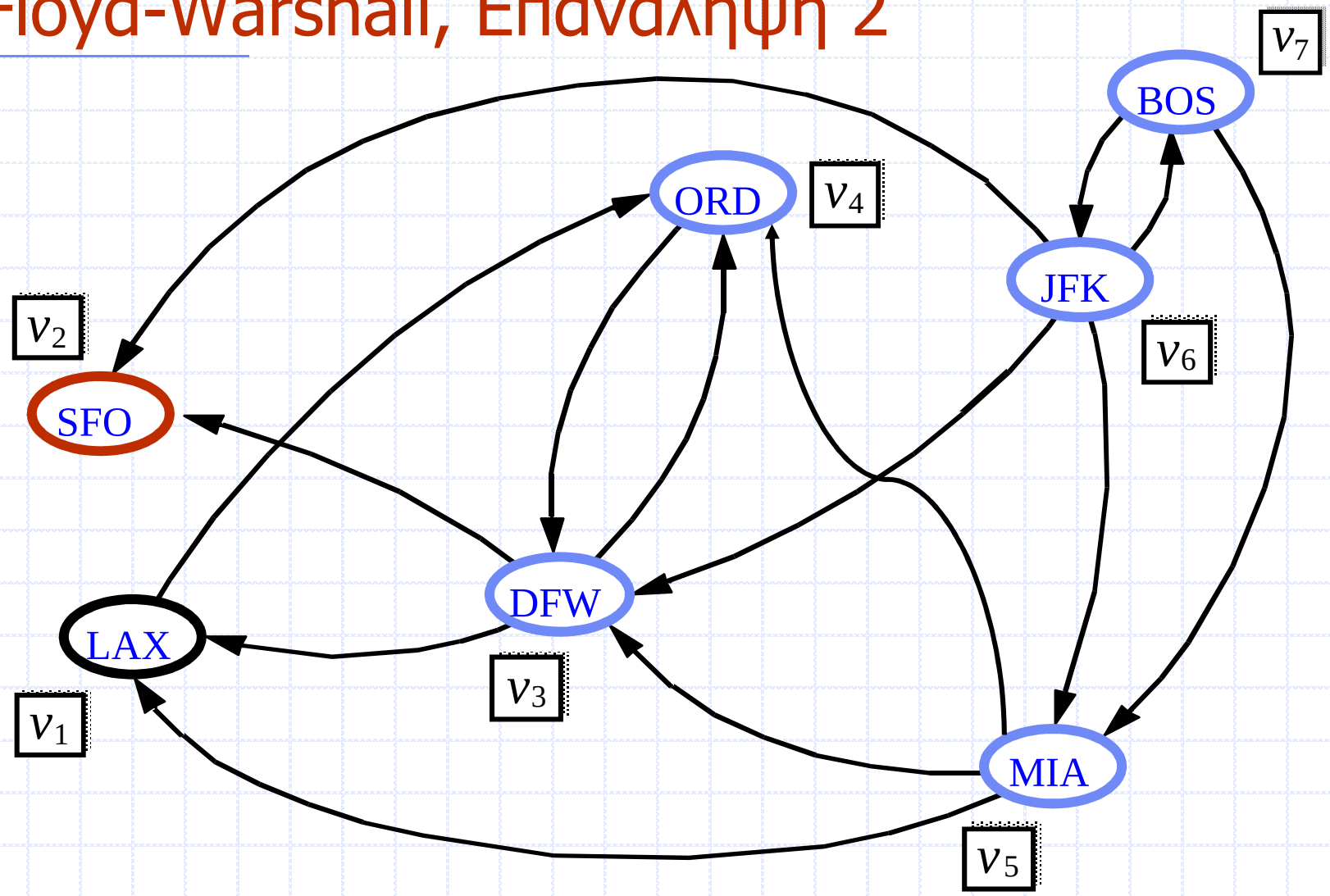
Floyd-Warshall Παράδειγμα



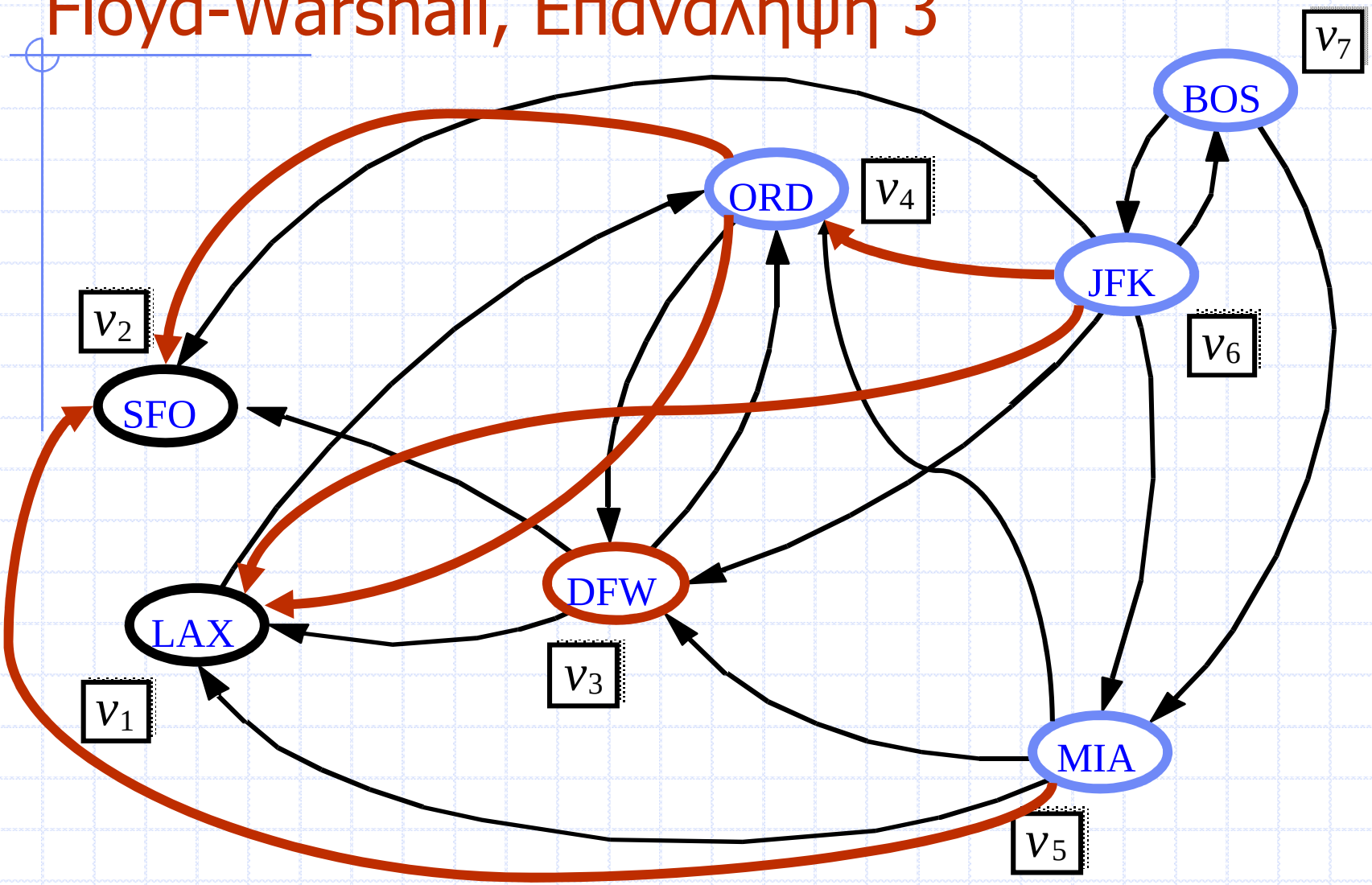
Floyd-Warshall, Επανάληψη 1



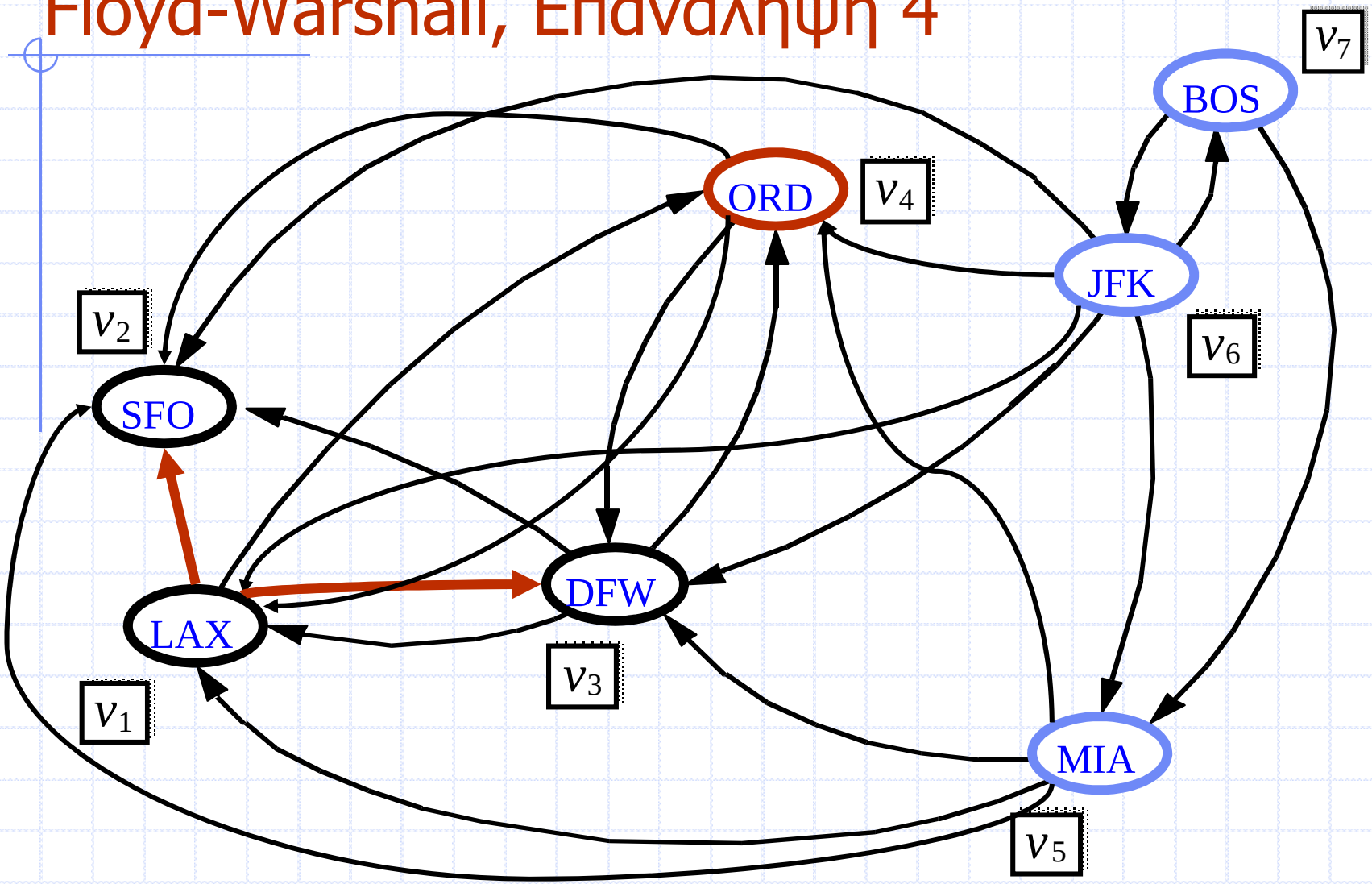
Floyd-Warshall, Επανάληψη 2



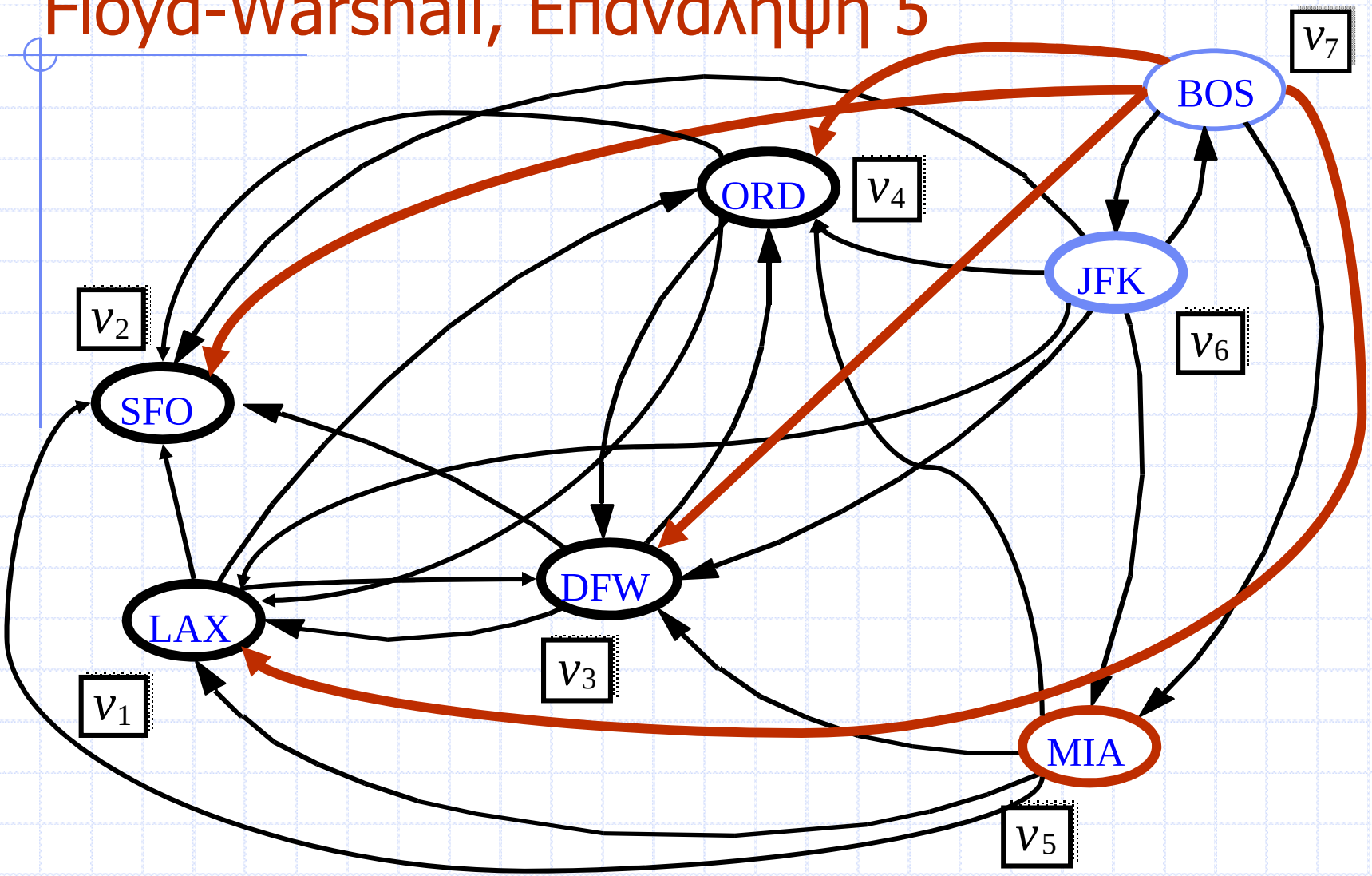
Floyd-Warshall, Επανάληψη 3



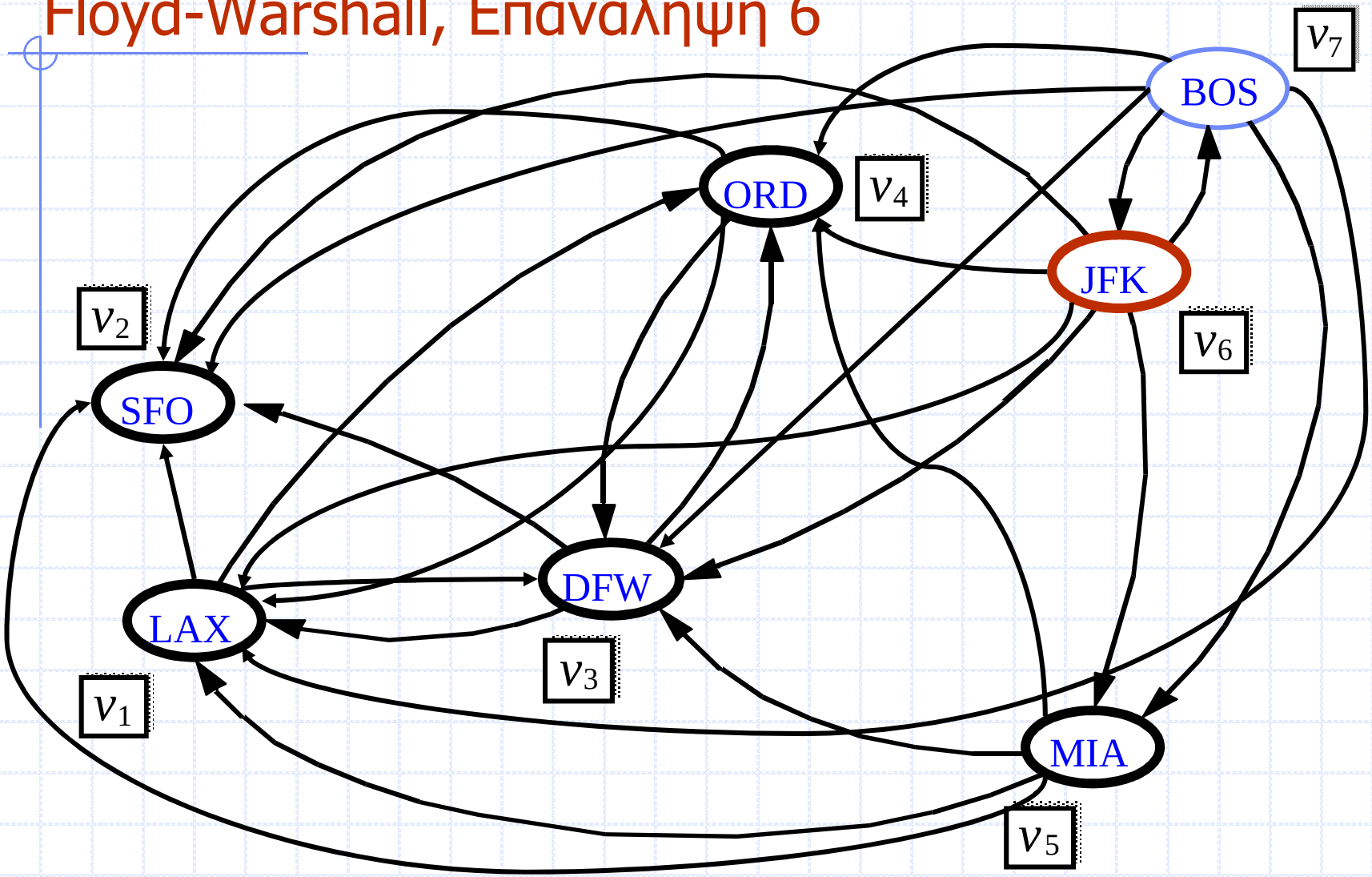
Floyd-Warshall, Επανάληψη 4



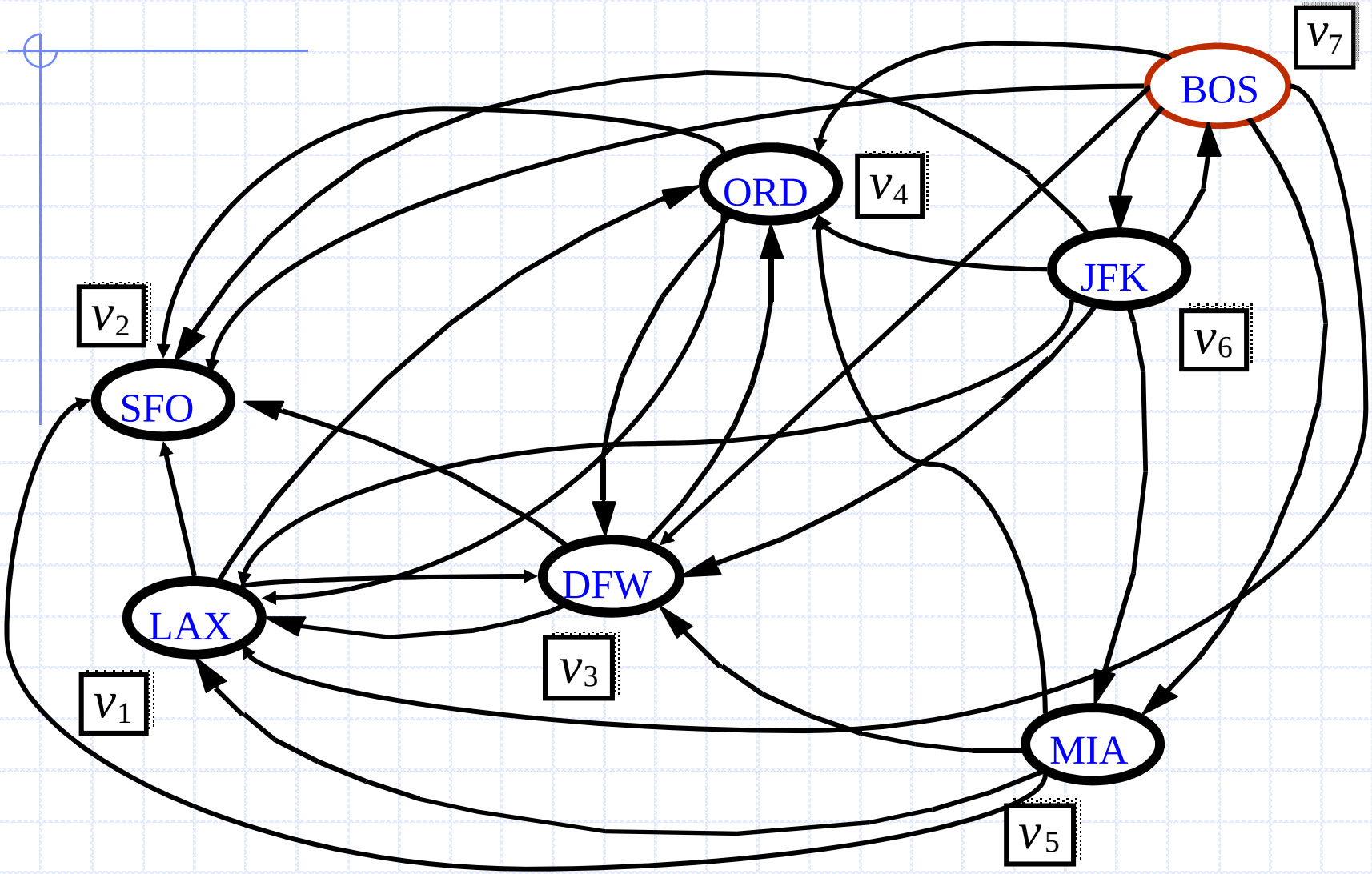
Floyd-Warshall, Επανάληψη 5



Floyd-Warshall, Επανάληψη 6



Floyd-Warshall, Ολοκλήρωση



Κατευθυνόμενοι γράφοι χωρίς κύκλους και Τοπολογική Ταξινόμηση

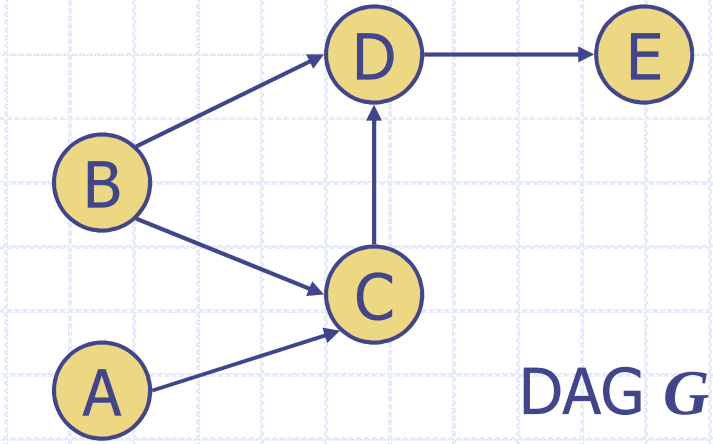
- Ένας άκυκλος κατευθυνόμενος γράφος (DAG) είναι ένας δίγραφος χωρίς κατευθυνόμενους κύκλους
- Μια τοπολογική διάταξη ενός δίγραφου είναι μια αρίθμηση

  $v_1, \ldots, v_n$

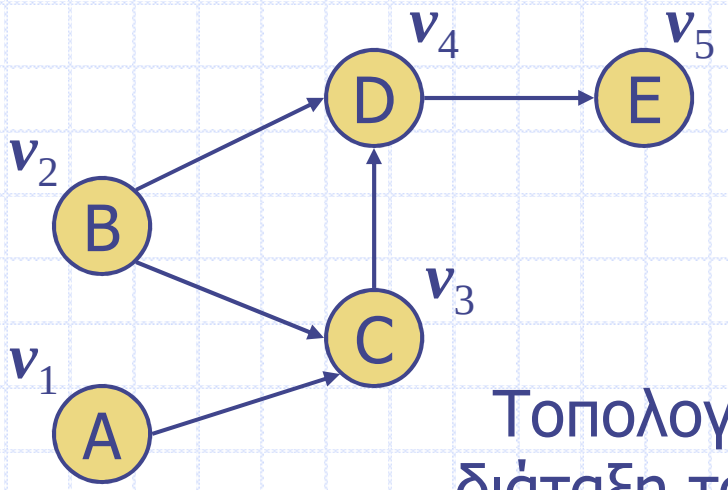
  των κορυφών έτσι που για κάθε ακμή (v_i, v_j), έχουμε $i < j$
- Παράδειγμα: σε ένα δίγραφο δρομολόγησης εργασιών, μια τοπολογική διάταξη είναι μια ακολουθία εργσιών που ικανοποιεί τους κανόνες προτεραιότητας

Θεώρημα

Ένας δίγραφος έχει τοπολογική διάταξη αν και μόνον αν είναι DAG

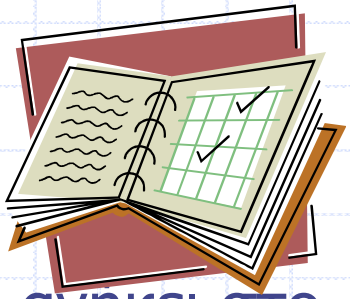


DAG G

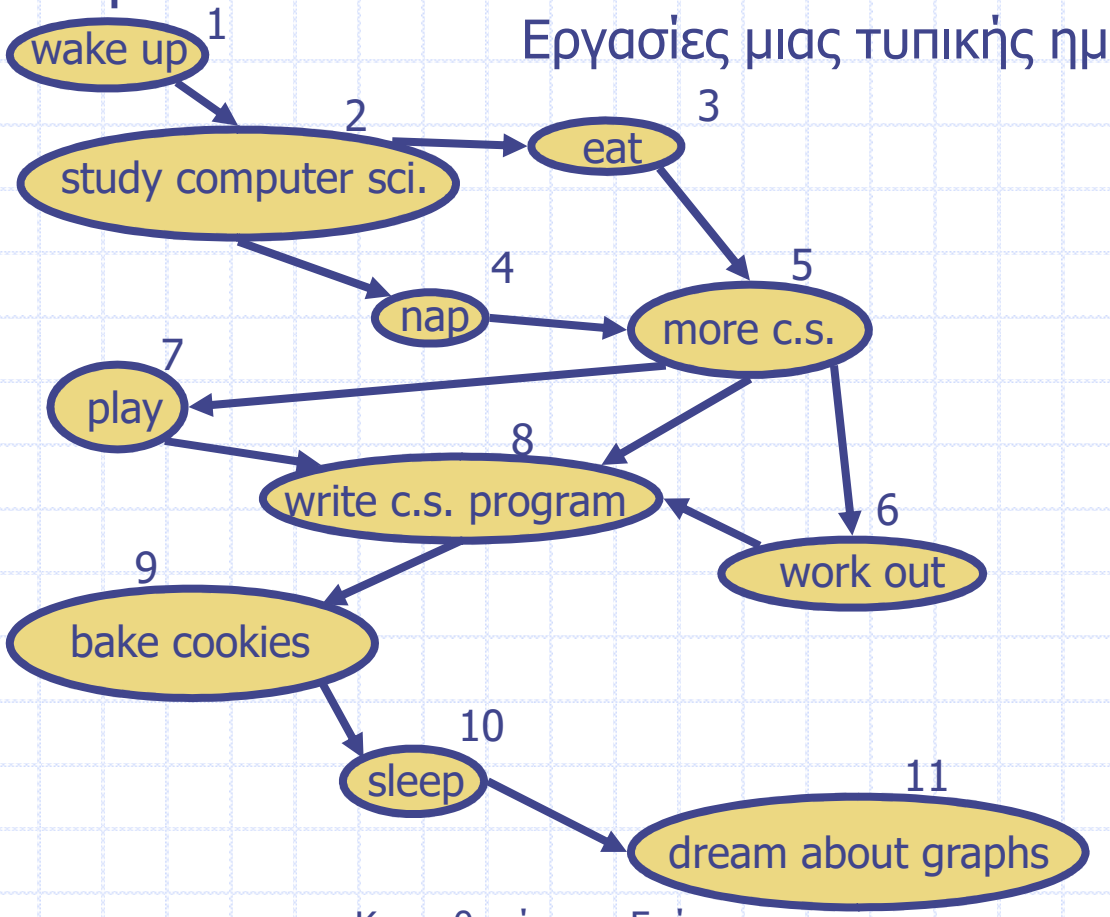


Τοπολογική διάταξη του G

Τοπολογική Ταξινόμηση



- Αρίθμηση των κορυφών, έτσι που αν η (u,v) ανήκει στο Ε συνεπάγεται u < v



Αλγόριθμος για Τοπολογική Ταξινόμηση

```
Algorithm TopologicalSort(G)
    H ← G          // Προσωρινό αντίγραφο του G
    n ← G.numVertices()
    while H is not empty do
        Let v be a vertex with no outgoing edges
        Label v ← n
        n ← n – 1
        Remove v from H
```

- Χρόνος τρεξίματος: O(n + m)

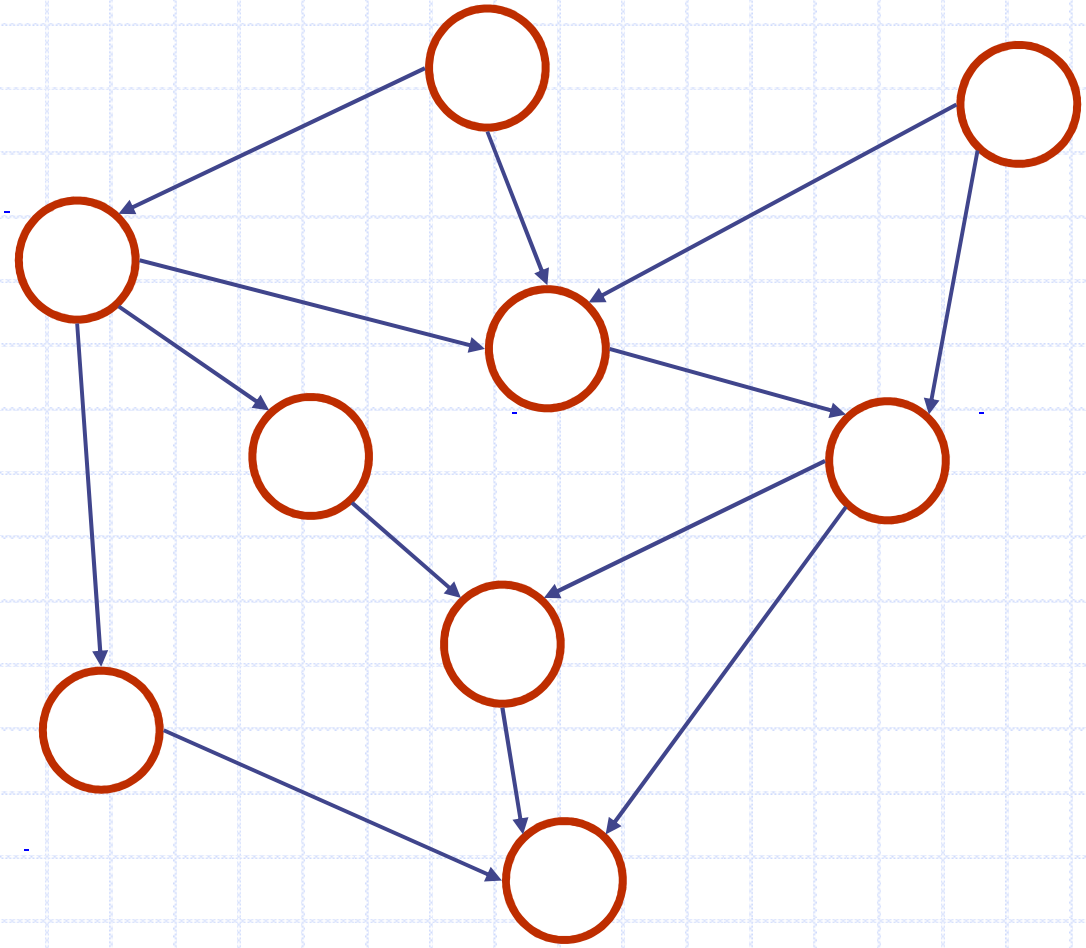
Υλοποίηση με DFS

- Προσομοίωση του αλγόριθμου με χρήση αναζήτησης με προτεραιότητα βάθους
- O(n+m) time.

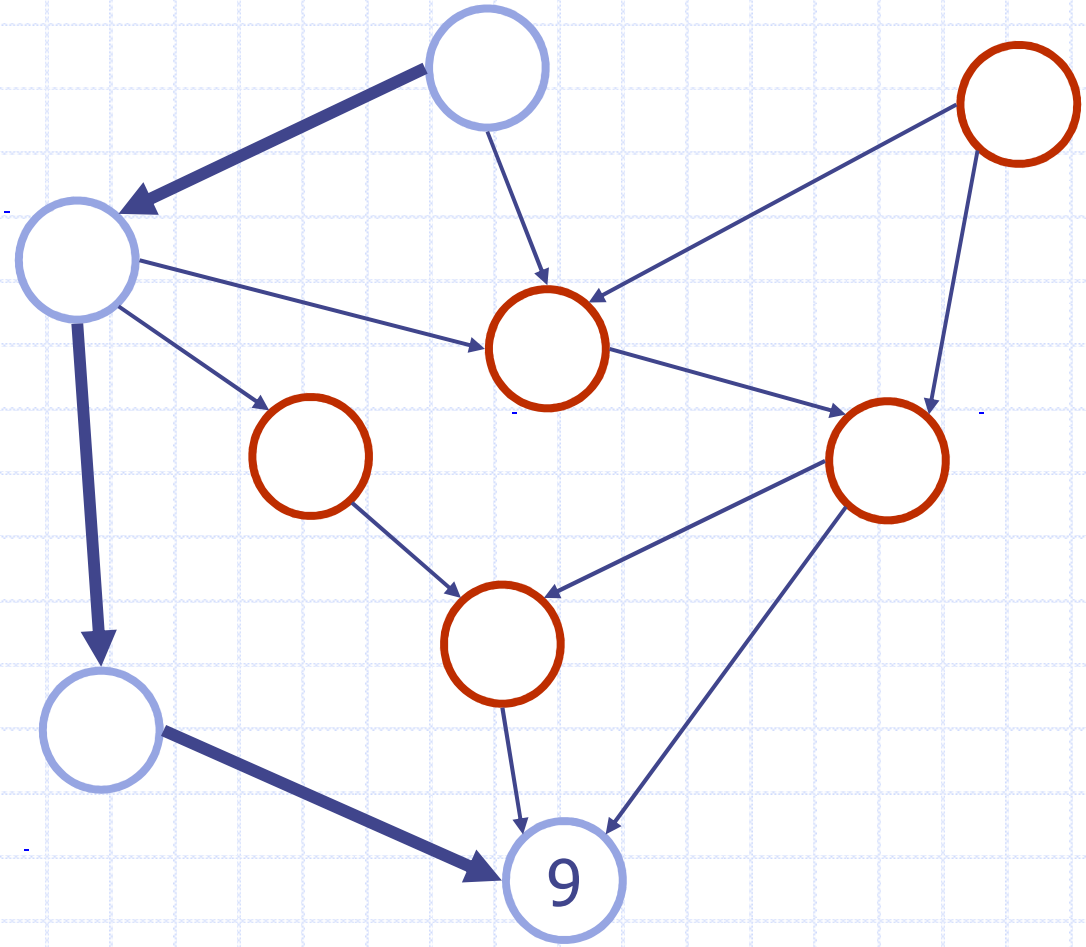
Algorithm *topologicalDFS*(*G*)
  Input dag *G*
  Output topological ordering of *G*
  $n \leftarrow$ *G.numVertices*()
  for all $u \in$ *G.vertices*()
    setLabel(*u*, *UNEXPLORED*)
  for all $v \in$ *G.vertices*()
    if *getLabel*(*v*) = *UNEXPLORED*
      topologicalDFS(*G*, *v*)

Algorithm *topologicalDFS*(*G*, *v*)
  Input γράφος *G* και μια κορυφή εκκίνησης *v* του *G*
  Output επίθεση ετικετών στις κορυφές του *G* στη συνεκτική συνιστώσα του *v*
  setLabel(*v*, *VISITED*)
  for all $e \in$ *G.outEdges*(*v*)
    { outgoing edges }
    $w \leftarrow$ *opposite*(*v*,*e*)
    if *getLabel*(*w*) = *UNEXPLORED*
      { *e* is a discovery edge }
      topologicalDFS(*G*, *w*)
    else
      { *e* is a forward or cross edge }
  Label *v* with topological number *n*
  $n \leftarrow n - 1$

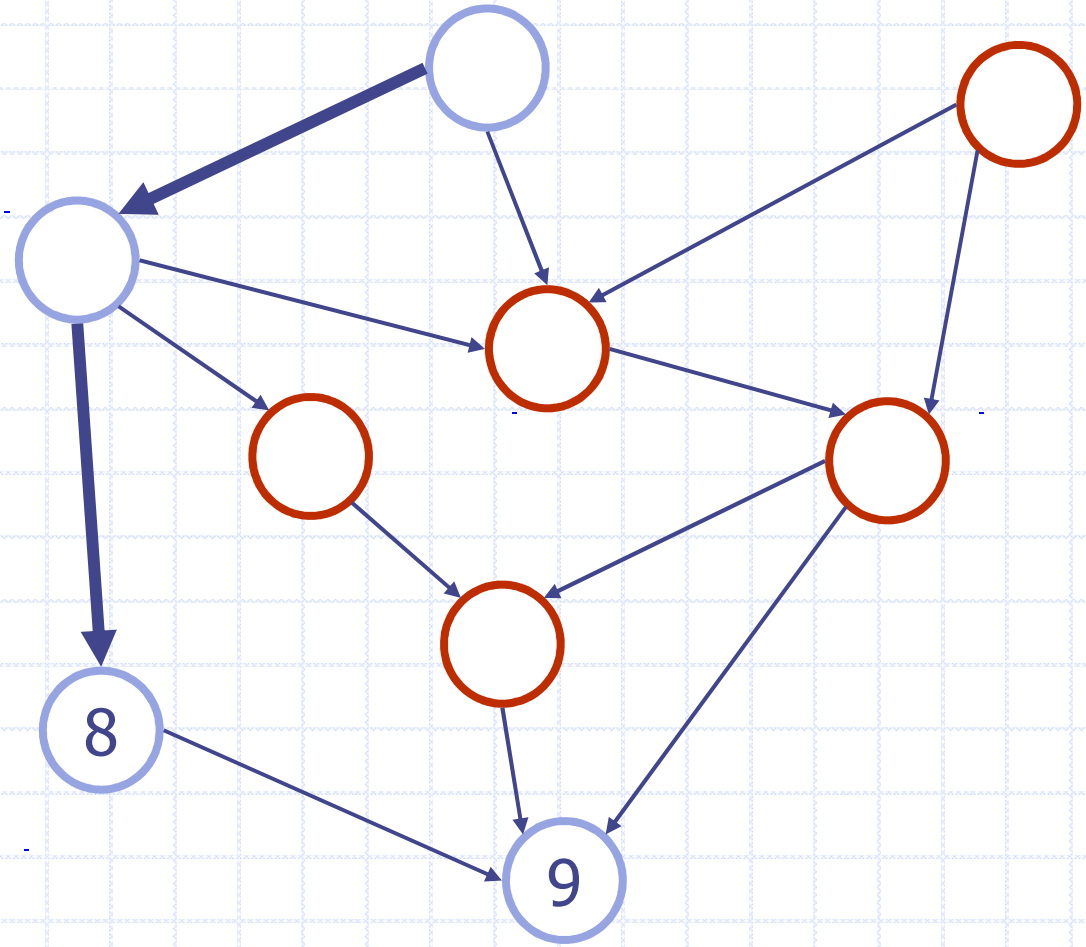
Παράδειγμα Τοπολογικής Ταξινόμησης



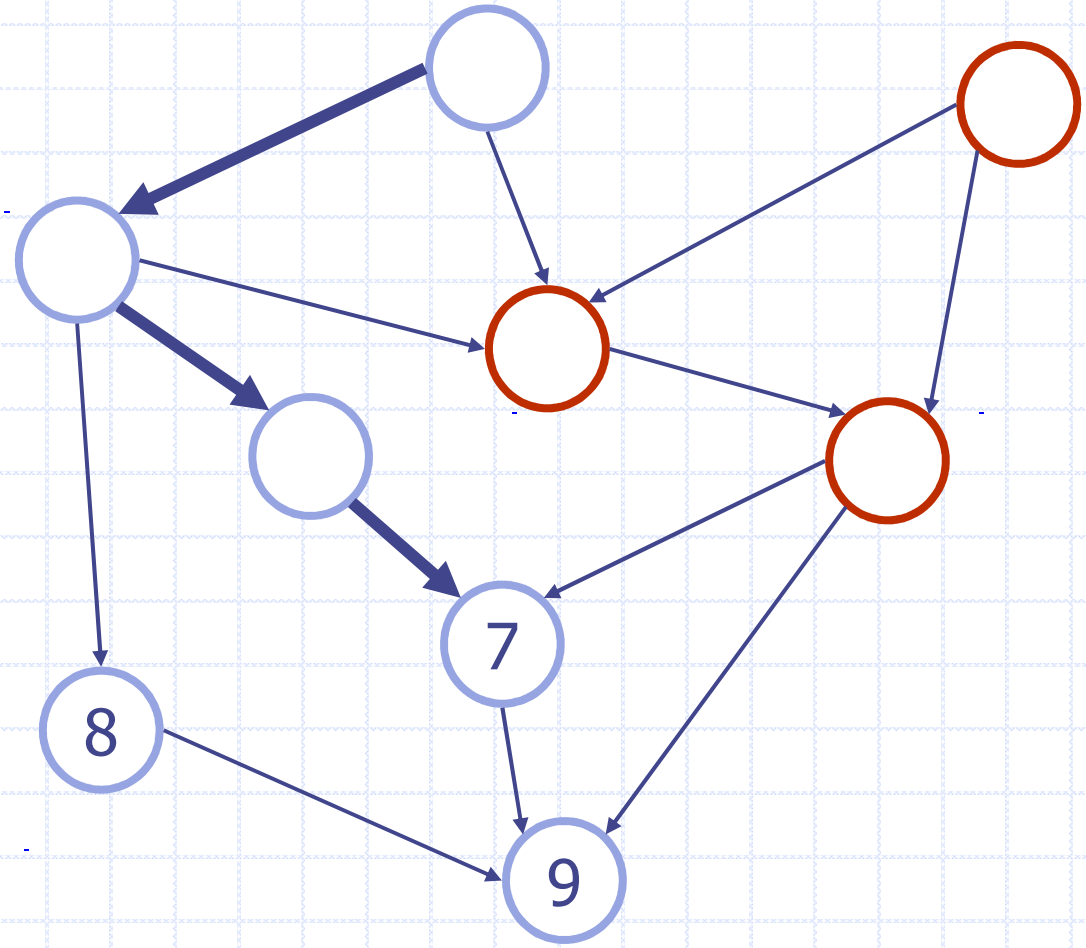
Παράδειγμα Τοπολογικής Ταξινόμησης



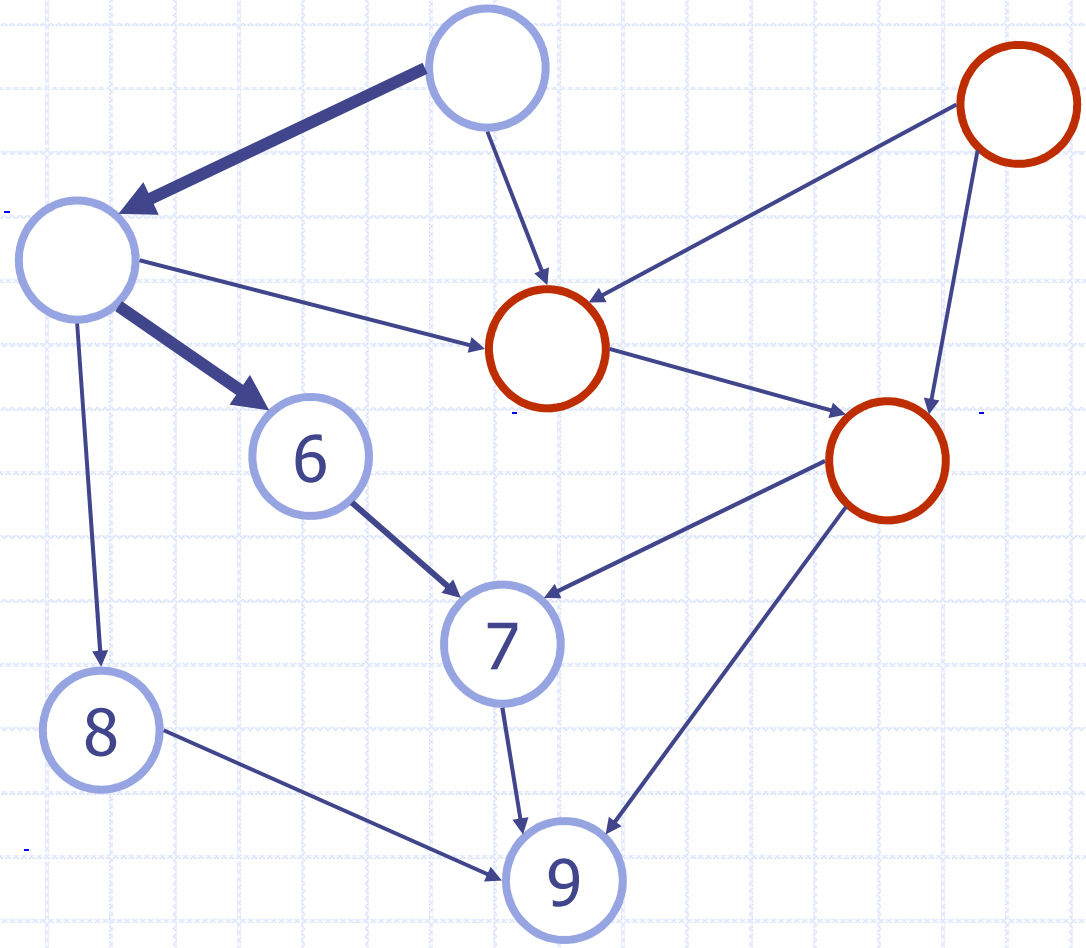
Παράδειγμα Τοπολογικής Ταξινόμησης



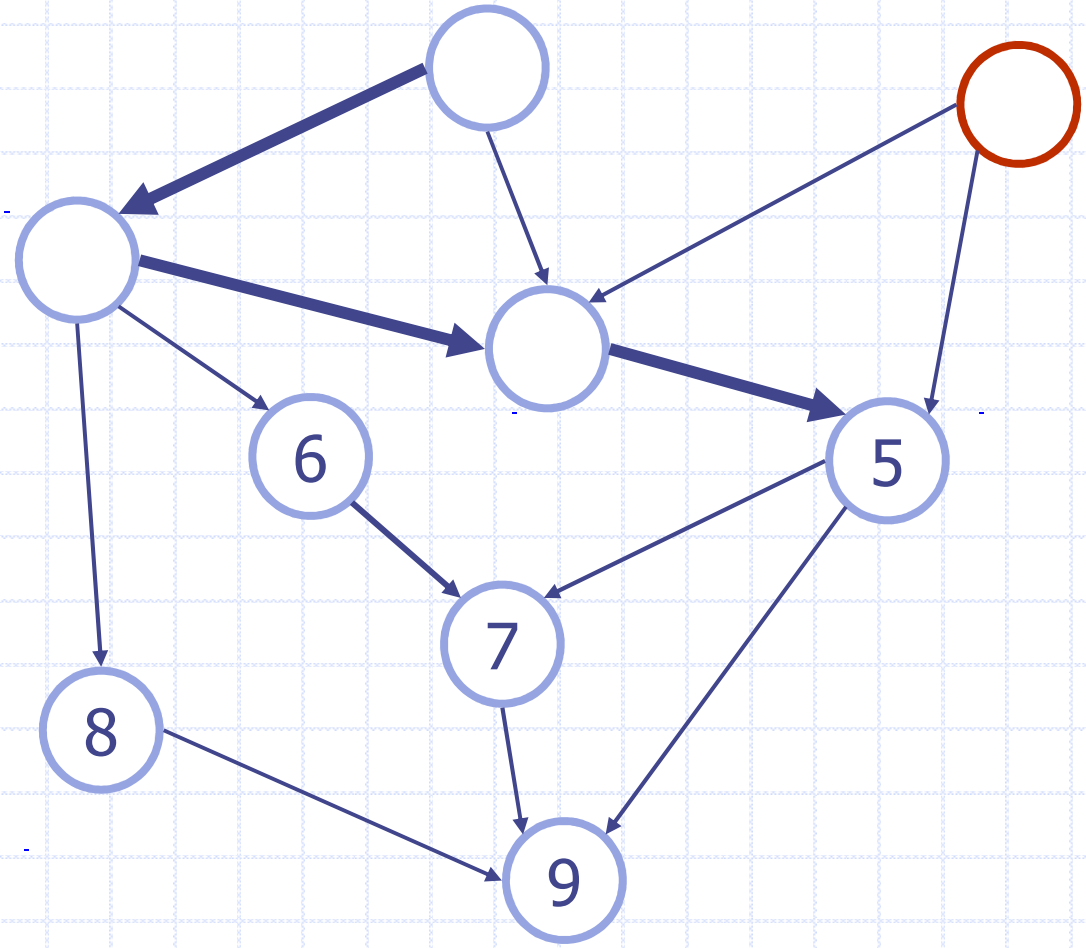
Παράδειγμα Τοπολογικής Ταξινόμησης



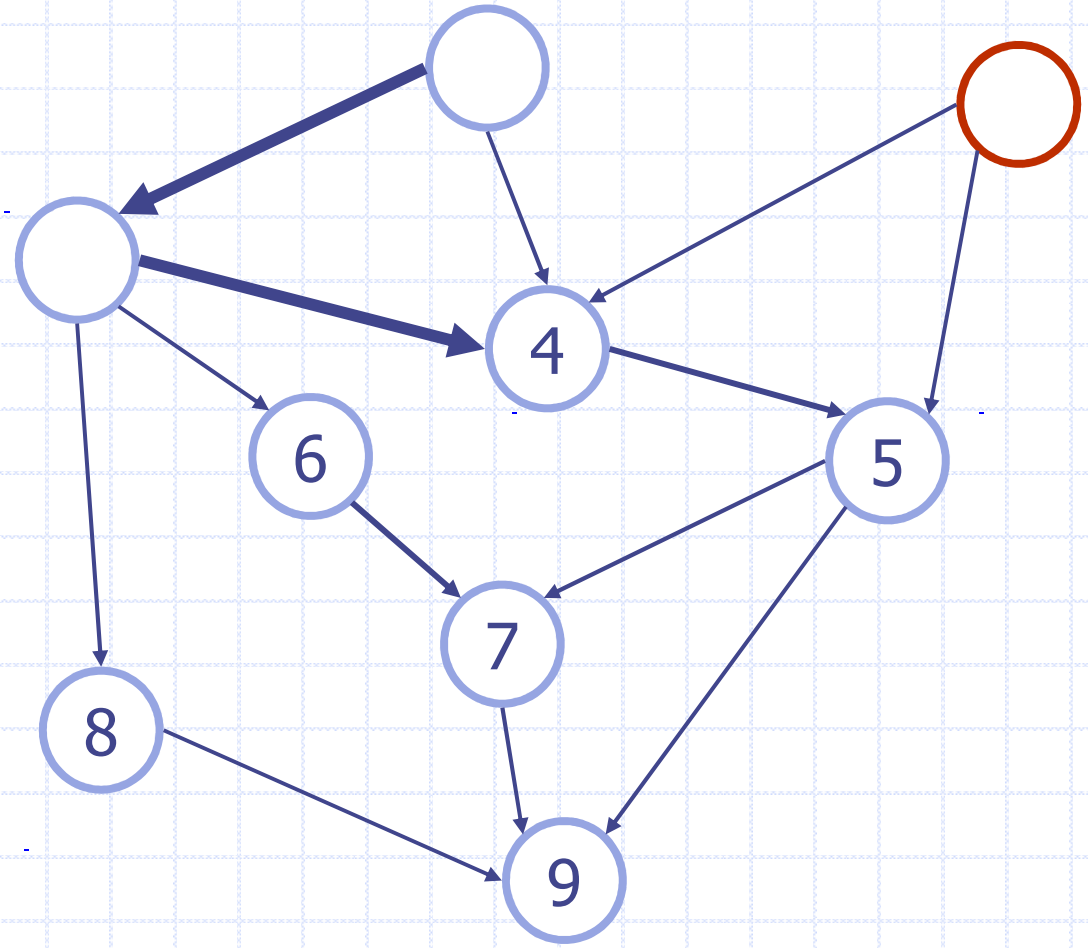
Παράδειγμα Τοπολογικής Ταξινόμησης



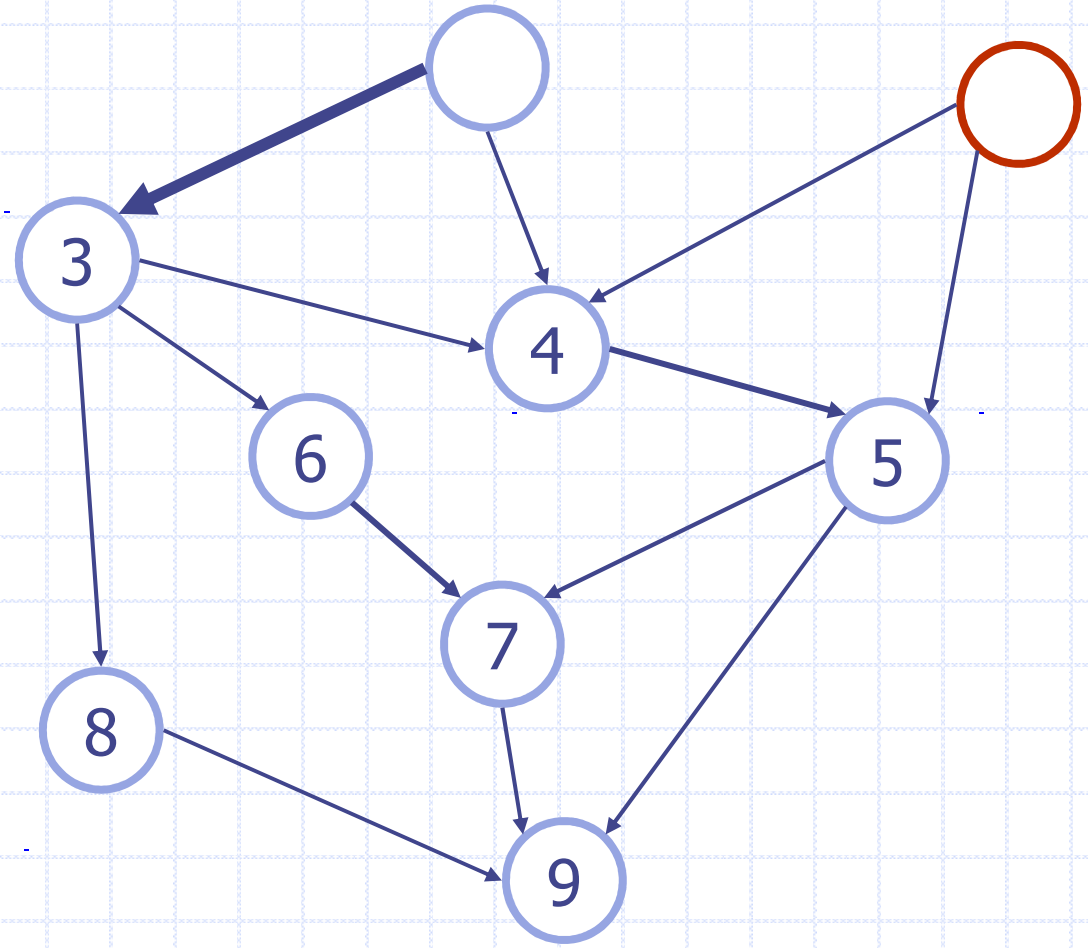
Παράδειγμα Τοπολογικής Ταξινόμησης



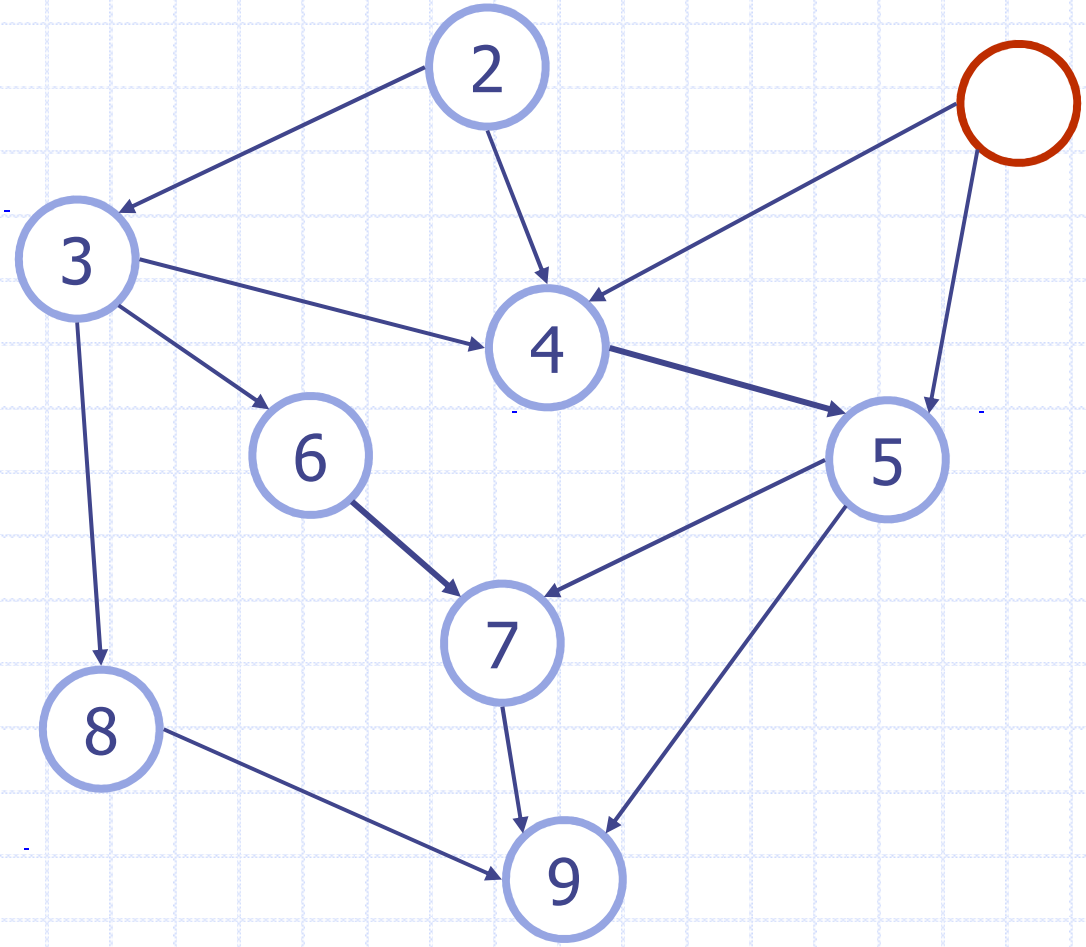
Παράδειγμα Τοπολογικής Ταξινόμησης



Παράδειγμα Τοπολογικής Ταξινόμησης



Παράδειγμα Τοπολογικής Ταξινόμησης



Παράδειγμα Τοπολογικής Ταξινόμησης