Διάλεξη 19: Εισαγωγή στους Γράφους

Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:

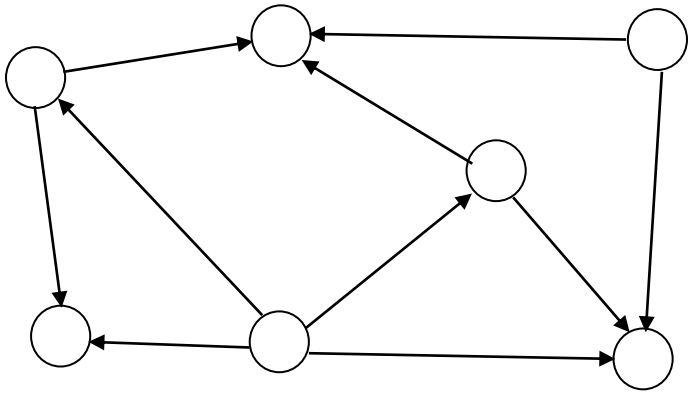
Γράφοι - ορισμοί και υλοποίηση

Διάσχιση Γράφων

Διδάσκων: Δημήτρης Ζεϊναλιπούρ

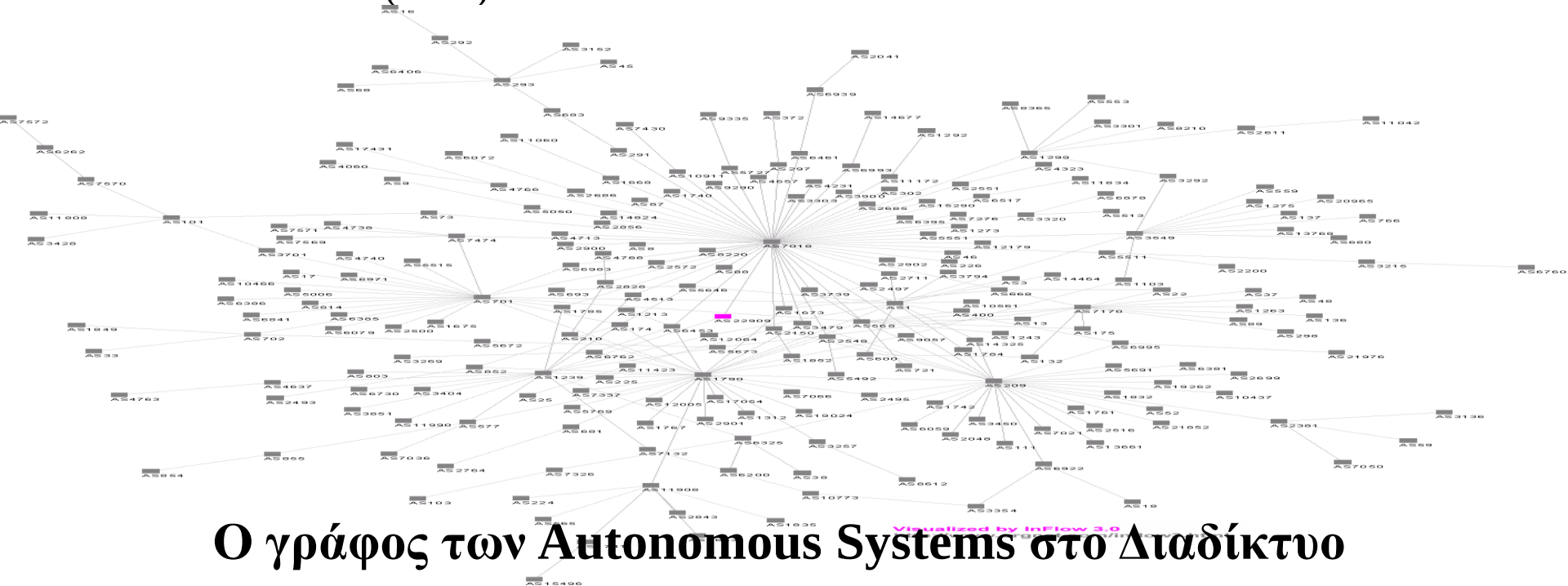
Εισαγωγή στους Γράφους

- **Η πιο γενική μορφή δομής δεδομένων, με την έννοια ότι όλες οι προηγούμενες δομές μπορούν να θεωρηθούν ως περιπτώσεις γράφων.**
- Ένα γράφος **G(V,E)** αποτελείται από
  - ένα σύνολο **V κορυφών (vertices),** ή σημείων, ή κόμβων , και
  - ένα σύνολο **E ακμών (edges),** ή τόξων, ή γραμμών. Μια ακμή είναι ένα ζεύγος (u,v) από κορυφές.
- Παράδειγμα γράφου:



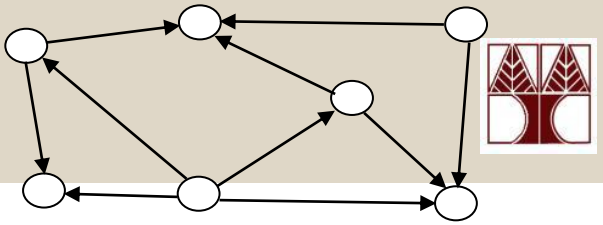
Γράφοι

- Οι γράφοι προσφέρουν μια χρήσιμη μέθοδο για τη διατύπωση και λύση πολλών προβλημάτων, σε
  - δίκτυα και συστήματα τηλεπικοινωνιών (π.χ. το Internet),
  - χάρτες - επιλογή δρομολογίων ,
  - προγραμματισμό εργασιών (scheduling),
  - ανάλυση προγραμμάτων (flow charts).
- Η θεωρία των γράφων θεωρείται ότι ξεκίνησε από τον Euler στις αρχές του 18ου αιώνα (1736).



Ο γράφος των Autonomous Systems στο Διαδίκτυο

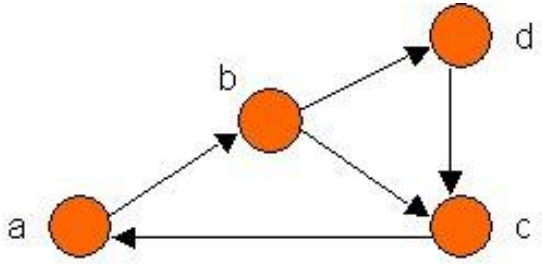
Ορισμοί



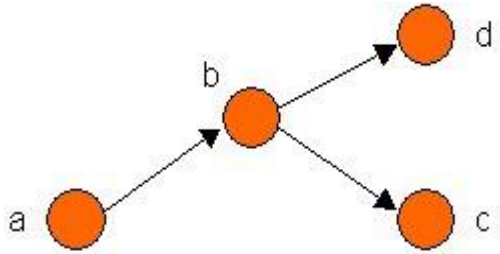
- Ένας γράφος ονομάζεται **κατευθυνόμενος (directed graph, digraph)** αν κάθε μια από τις ακμές του είναι προσανατολισμένη προς μία κατεύθυνση. (πχ: ένα οδικό δίκτυο με λωρίδες μονής κατεύθυνσης)
- Ένας γράφος ονομάζεται **μη-κατευθυνόμενος (undirected)** αν οι ακμές του δεν είναι προσανατολισμένες.(πχ: ένα οδικό δίκτυο με λωρίδες διπλής κατεύθυνσης)
- Αν (u,v) είναι ακμή τότε λέμε ότι οι κορυφές u και v είναι **γειτονικές (adjacent)** ή ότι *γειτνιάζουν*.
- **Μονοπάτι ή διαδρομή (path)** ενός γράφου μήκους n, είναι μια ακολουθία κόμβων $v_0, v_1, \ldots, v_n$, όπου για κάθε i , $0 \leq i < n$, (v_i , v_{i+1}) είναι ακμή του γράφου.
- **Μήκος** ενός μονοπατιού είναι ο αριθμός ακμών που περιέχει (εκτός και εάν ορίζεται διαφορετικά)
- **Βραχύτερο Μονοπάτι μεταξύ δυο κορυφών (Shortest Path):** Το μονοπάτι μεταξύ δυο κορυφών που έχει το ελάχιστο μήκος.
- Μια *διαδρομή* ενός γράφου ονομάζεται **απλή (simple path)** αν όλες οι κορυφές της είναι διαφορετικές μεταξύ τους, εκτός από την πρώτη και την τελευταία οι οποίες μπορούν να είναι οι ίδιες.
- **Κυκλική Διαδρομή (cycle)** ονομάζεται μια *διαδρομή* με μήκος >1 οπου $v_0 = v_n$.

Ορισμοί

- Ένας γράφος που δεν περιέχει κύκλους ονομάζεται **άκυκλος (acyclic graph)**
- Έστω **G=(V,E)** και **G΄= (V΄, E΄)** γράφοι, όπου V΄⊆V και E΄⊆ E. Τότε ο γράφος G΄ είναι **υπογράφος (subgraph)** του γράφου G.
- **Απόσταση Κορυφών (Vertex Distance):** Το μήκος της συντομότερης διαδρομής που οδηγεί από τη μια κορυφή στην άλλη.
- **Συνεκτικός Γράφος (Connected Graph):** Ένας *μη-κατευθυνόμενος* γράφος στον οποίο υπάρχει τουλάχιστο μια διαδρομή μεταξύ όλων των κορυφών.
- **Ισχυρά Συνεκτικός Γράφος (Strongly Connected Graph):** Ένας *κατευθυνόμενος* γράφος στον οποίο υπάρχει μια διαδρομή από οποιαδήποτε κορυφή σε οποιανδήποτε άλλη.
- **Ελαφρά Συνεκτικός Γράφος (Weakly Connected Graph):** Ένας *κατευθυνόμενος* (συνεκτικός) γράφος στον οποίο **δεν** υπάρχει μια διαδρομή από οποιαδήποτε κορυφή σε οποιανδήποτε άλλη.



Strongly Connected Directed Graph

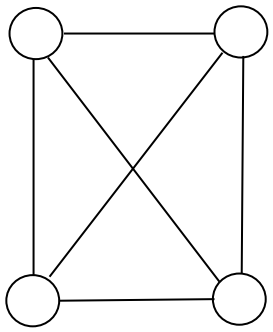


Weakly Connected Directed Graph

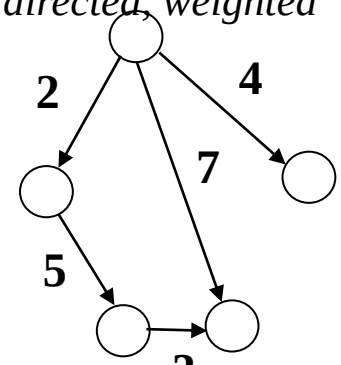
Ορισμοί

- Πόσες **ακμές** μπορεί να έχει ένας γράφος με **n** κορυφές;
  Α) Κατευθυνόμενος: **nx(n-1)** Β) Μη-κατευθυνόμενος: **nx(n-1)/2**
- **Αραιός Γράφος (Sparse Graph):** Αν ο αριθμός των ακμών του είναι της τάξης **O(n),** όπου n είναι ο αριθμός κορυφών του, διαφορετικά λέγεται **Πυκνός Γράφος (Dense Graph)**
- **Γράφος με Βάρη (Weighted Graph):** Συχνά συσχετίζουμε κάθε ακμή ενός γράφου με κάποιο βάρος (weight).
- Ποιες ιδιότητες ικανοποιούν οι πιο κάτω γράφοι;

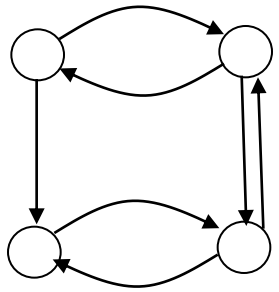
Connected, undirected, unweighted



Weakly connected, directed, weighted



strongly connected directed, unweighted,

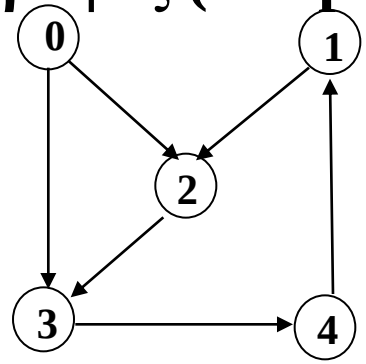


Αναπαράσταση γράφων στην Μνήμη

Αναπαράσταση γράφων με Πίνακες Γειτνίασης (Adjacency Matrix)

- Ένας γράφος G=(V,E) με n κορυφές μπορεί να αναπαρασταθεί ως ένας **n×n** πίνακας που περιέχει τις τιμές 0 και 1, και όπου
  - αν η (i,j) είναι ακμή τότε A[(i,j)]=1, διαφορετικά A[(i,j)]=0.
- Αν ο γράφος είναι γράφος με βάρη, και το βάρος κάθε ακμής είναι τύπου t, τότε για την αναπαράσταση του γράφου μπορεί να χρησιμοποιηθεί πίνακας τύπου t με
  - A[(i,j)] = βάρος(i,j), αν υπάρχει ακμή (i,j)
  - A[(i,j)] = ∞, αν δεν υπάρχει ακμή (i,j)
- Αυτή η αναπαράσταση απαιτεί χώρο $\Theta(n^2)$, όπου n=|V|.
- Αν ο γράφος είναι αραιός (δηλαδή τάξης O(n)) η μέθοδος οδηγεί σε σπάταλη χώρου.

Γράφος (Graph)



Πίνακας Γειτνίασης (Adjacency Matrix)

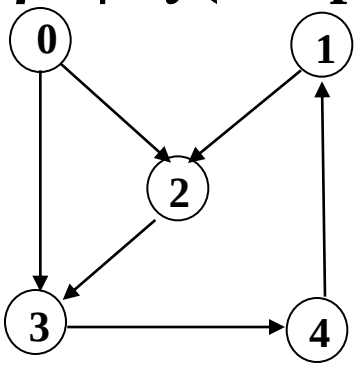
From: \ To:	0	1	2	3	4
0			1	1	
1			1		
2				1	
3					1
4		1			

Αναπαράσταση γράφων

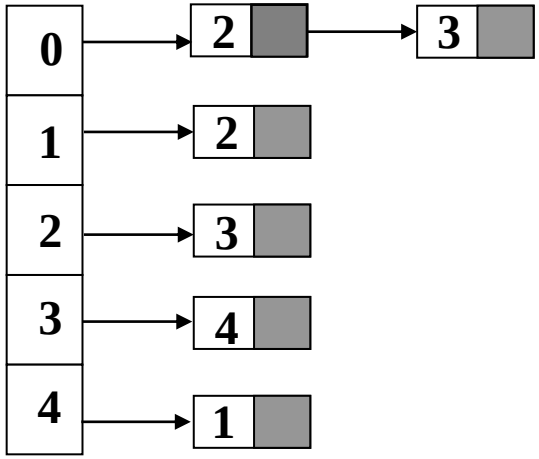
Αναπαράσταση γράφων με Λίστες Γειτνίασης (Adjacency Lists)

- Ένας γράφος G=(V,E) αναπαρίσταται ως ένας μονοδιάστατος πίνακας A.
- Για κάθε κορυφή **v**, **A[v]** είναι ένας δείκτης σε μια συνδεδεμένη λίστα στην οποία αποθηκεύονται οι κορυφές που γειτνιάζουν με την **v**.
- Η μέθοδος απαιτεί χώρο Θ(|V|+ |E|).
- **Επιτυγχάνεται εξοικονόμηση χώρου για αραιούς γράφους.**
- Στην περίπτωση γράφων με βάρη στη λίστα γειτνίασης αποθηκεύουμε επίσης το βάρος κάθε ακμής.

Γράφος (Graph)



Λίστα Γειτνίασης



Διάσχιση Γράφων

- Αν θέλουμε να επισκεφτούμε όλους τους κόμβους ενός γράφου μπορούμε να χρησιμοποιήσουμε έναν από πολλούς τρόπους, οι οποίοι διαφέρουν στη σειρά με την οποία εξετάζουν τους κόμβους.

- Διαδικασίες διάσχισης χρησιμοποιούνται για τη διακρίβωση ύπαρξης μονοπατιού μεταξύ δύο κόμβων κ.α.

Άλλα Παραδείγματα

1. **Οι μηχανές αναζήτησης** (search engines π.χ. Google), χρησιμοποιούν προγράμματα τα οποία ονομάζονται crawlers, για να διασχίσουν όλες τις ιστοσελίδες του WWW. Αυτές εκτελούνε αναδρομικά την εξής διαδικασία: a) Κατέβασε την σελίδα X, b) Βρες όλους του επόμενους συνδέσμους (links) από την X, c) Για κάθε σύνδεσμο εκτέλεσε τα βήματα a-c.
2. **Προγράμματα ανταλλαγής αρχείων** (P2P File-sharing Systems, π.χ. Gnutella), χρησιμοποιούνε αλγόριθμους διάσχισης για να μπορούνε οι χρήστες να αποστέλλουν την αναζήτηση τους (query) στους κόμβους του δικτύου. Αν κάποιος κόμβος έχει κάποιο αρχείο το οποίο ψάχνουμε τότε στέλνει κάποια απάντηση στο query.

Διάσχιση Γράφων

- Οι πιο διαδεδομένοι τύποι διάσχισης είναι:

1. *Κατά Βάθος Διάσχιση(Depth-First Search (DFS))*
2. *Κατά Πλάτος Διάσχιση (Breadth-First Search (BFS))*

Depth-First Search

- Γενίκευση της **προθεματικής διάσχισης δένδρων**: Ξεκινώντας από ένα κόμβο v, επισκεπτόμαστε πρώτα τον v και ύστερα καλούμε αναδρομικά τη διαδικασία στο καθένα από τα παιδιά του. Δηλαδή:

```
Print_Preorder(treenode u)
     Output data at u;
     for each child v of u
            Print_Preorder(v)
```

- Σε ένα δένδρο δεν έχουμε κυκλικές διαδρομές. Σε ένα γράφο όμως έχουμε κυκλικές διαδρομές. **Άρα πως θα διασχίσουμε τον γράφο;**

- Θα διατηρήσουμε ένα πίνακα `Visited` ο οποίος θα κρατά πληροφορίες ως προς το ποιους κόμβους έχουμε επισκεφθεί ανά πάσα στιγμή. Δεν θα επισκεπτόμαστε ξανά τον ίδιο κόμβο

Διαδικασία Προθεματικής Διάσχισης

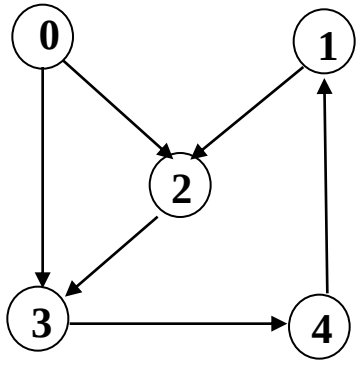
```
DepthFirstSearch(graph G,vertex v){
   for each vertex w in G
       visited[w] = False;
   DFS(v);
}

DFS(vertex v){
   visited[v] = True;
   print "Visited node v"
   for each w adjacent to v
       if (visited[w]==False)
            DFS(w)
}
```

Αρχικοποίηση Πίνακα Visited

Χρόνος Εκτέλεσης: Θ(|V| + |E|)

Θα επισκεφτούμε ακριβώς μια φορά όλες τις κορυφές και ακμές

Υπόθεση: Ο γράφος είναι συνεκτικός (connected).

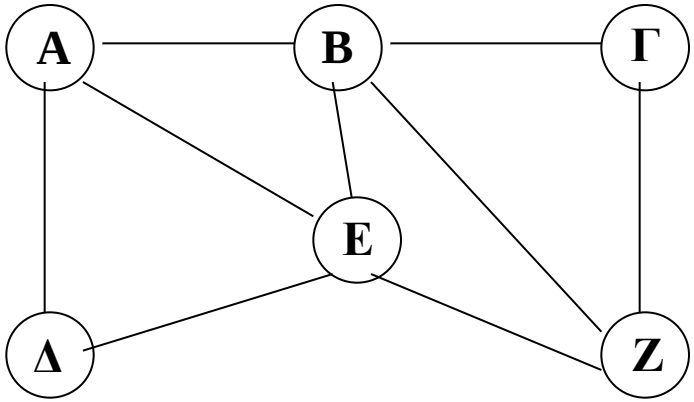


Προθεματική Διάσχιση

- Αν ο γράφος **δεν είναι συνεκτικός** (disconnected) αυτή η στρατηγική πιθανό να αγνοήσει μερικούς κόμβους. Αν ο στόχος μας είναι να επισκεφθούμε όλους τους κόμβους τότε μετά το τέλος της εκτέλεσης του DFS(v) θα πρέπει να ελέγξουμε τον πίνακα Visited να βρούμε τους κόμβους που δεν έχουμε επισκεφθεί και να καλέσουμε σε αυτούς τη διαδικασία DFS:

```
DepthFirstSearch(graph G,vertex v){
  for each vertex w in G
      Visited[w] = False;
  DFS(v);
  for each vertex w in G
      if (Visited[w]== False)
         DFS(w);
}
```

Παράδειγμα Depth-First-Search (σε Μη-Κατευθυνόμενο Γράφο)

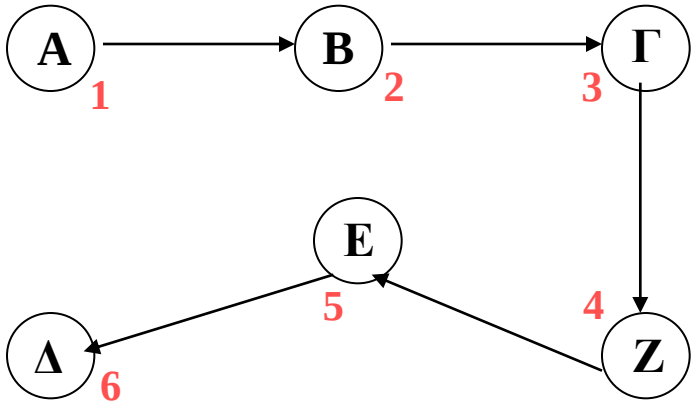


ΤΟ:

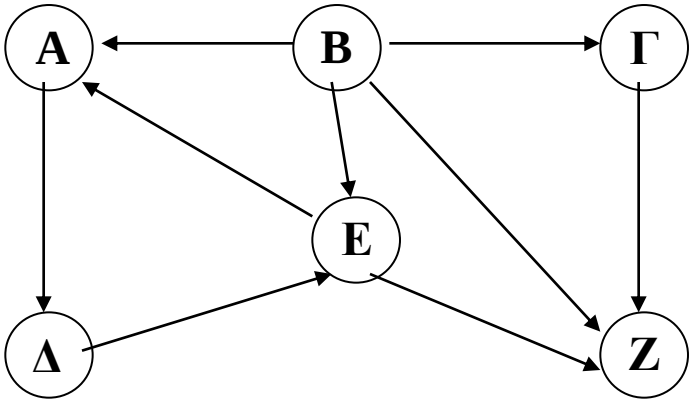
	Α	Β	Γ	Δ	Ε	Ζ	
	0	1	0	1	1	0	Α
	1	0	1	0	1	1	Β
	0	1	0	0	0	1	Γ
	1	0	0	0	1	0	Δ
	1	1	0	1	0	1	Ε
	0	1	1	0	1	0	Ζ

FROM

DepthFirstSearch(G, A)



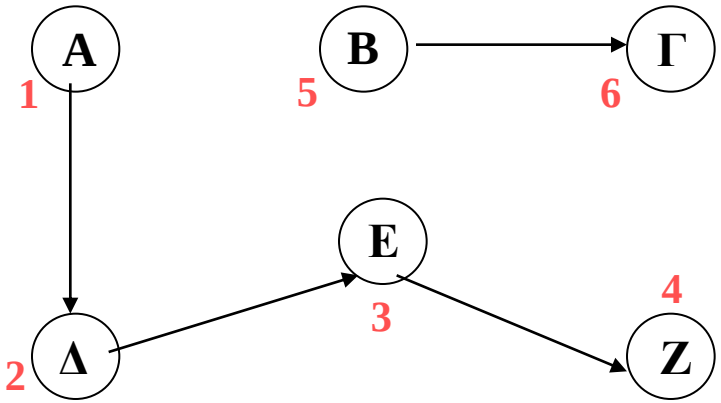
Παράδειγμα Depth-First-Search 2 (σε Κατευθυνόμενο & Ελαφρά Συνεκτικό Γράφο)



TO:

FROM	A	B	Γ	Δ	E	Z
A	0	0	0	1	0	0
B	1	0	1	0	1	1
Γ	0	0	0	0	0	1
Δ	0	0	0	0	1	0
E	1	0	0	0	0	1
Z	0	0	0	0	0	0

DepthFirstSearch(G, A)



Στον αρχικό γράφο δεν υπάρχει μια διαδρομή μεταξύ όλων των ζευγών

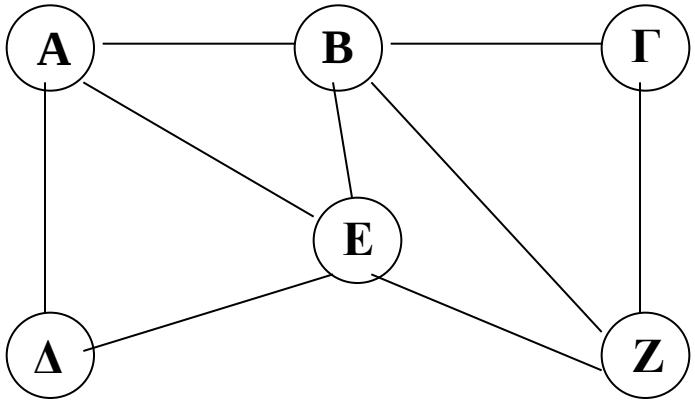
Π.χ. A->B

Μερικά Σχόλια

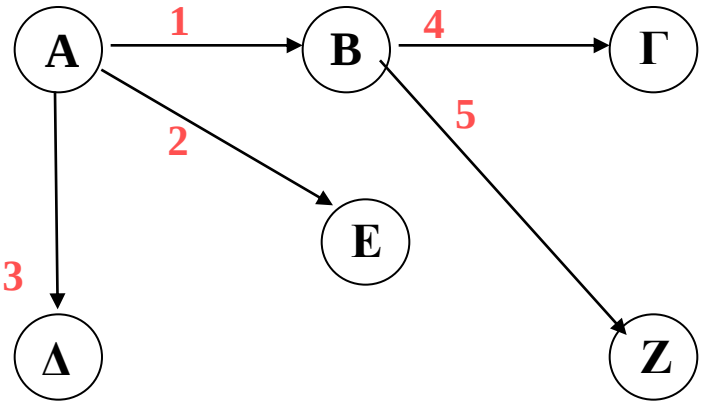
- Η διαδικασία καλείται σε κάθε κόμβο το πολύ μια φορά.
- Χρόνος Εκτέλεσης: $\Theta(|V| + |E|)$, δηλαδή γραμμικός ως προς τον αριθμό των ακμών και κορυφών.
- ***Αρίθμηση DFS*** των κορυφών ενός γράφου ονομάζεται η σειρά με την οποία επισκέπτεται η διαδικασία DepthFirstSearch τις κορυφές του γράφου.
- Η διαδικασία μπορεί να κληθεί και για μη-κατευθυνόμενους και για κατευθυνόμενους γράφους.
- Αντί με αναδρομή η διαδικασία μπορεί να υλοποιηθεί, ως συνήθως, με τη χρήση στοιβών.

Breadth-First Search (BFS)

- Ξεκινώντας από ένα κόμβο v, επισκεπτόμαστε πρώτα το v, ύστερα τους κόμβους που γειτνιάζουν με τον v, ύστερα τους κόμβους που βρίσκονται σε απόσταση 2 από τον v, και ούτω καθεξής.



Αρχικός Γράφος



Διάσχιση BFS

Διαδικασία Breadth-First Search

Διάσχιση Γράφου G ξεκινώντας από το v

```
BFSearch(graph G, vertex v){
Q=MakeEmptyQueue();
for each w in G
        Visited[w]=False;

Visited[v]= True;
Enqueue(v,Q);

while (!IsEmpty(Q)){
        w = Dequeue(Q);
        Print(w);
        for each u adjacent to w
                if (Visited[u]==False){
                        Visited[u]=True;
                        Enqueue(u,Q);
                }
}
```

Υλοποίηση BFS με χρήση ουράς

Χρόνος Εκτέλεσης: O(|V| + |E|)