ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΕΣ ΤΕΧΝΙΚΕΣ

http://www.softlab.ntua.gr/~nickie/Courses/progtech/

Διδάσκοντες: Γιάννης Μαΐστρος (maistros@cs.ntua.gr)
Νίκος Παπασπύρου (nickie@softlab.ntua.gr)
Βασίλης Βεσκούκης (bxb@cs.ntua.gr)
Πέτρος Στεφανέας (petros@mail.ntua.gr)

ΓΡΑΦΟΙ (GRAPHS)

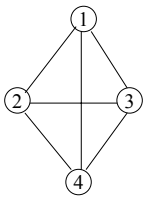
ΟΡΙΣΜΟΙ (1)

- Γράφοι: G = (V,E), όπου V σύνολο και E διμελής σχέση πάνω στο V. Τα στοιχεία του V: *κορυφές* ή *κόμβοι*. Τα στοιχεία του E: *τόξα* ή *ακμές*
- Όταν τα στοιχεία του E είναι διατεταγμένα ζεύγη ο γράφος ονομάζεται *κατευθυνόμενος*
- Δύο κορυφές χ,ψ ονομάζονται *γειτονικές* εάν (χ,ψ) στοιχείο του E.
- Ο *βαθμός* μιας κορυφής χ είναι ίσος με τον αριθμό των γειτονικών κορυφών της

ΟΡΙΣΜΟΙ (2)

- Μονοπάτι: ακολουθία ακμών που δεν χρησιμοποιεί την ίδια ακμή περισσότερες από μια φορές.
- $v_1,\ldots,v_k$ κύκλος αν και μόνο εάν $v_1 = v_k$
- Όταν ο γράφος δεν περιέχει κύκλο λέγεται άκυκλος
- Ο γράφος ονομάζεται συνδεδεμένος ή συνεκτικός εάν για κάθε δύο κορυφές του υπάρχει τουλάχιστον ένα μονοπάτι ανάμεσά τους

ΠΑΡΑΔΕΙΓΜΑΤΑ ΓΡΑΦΩΝ



G_1

G_3

ΛΕΙΤΟΥΡΓΙΕΣ ΓΡΑΦΩΝ

- **Join:** *σύνδεση δύο κορυφών μέσω μιας ακμής*
- **Remove:** *αποσύνδεση δύο κορυφών*
- **Adjacent:** *έλεγχος εάν δύο κορυφές είναι διπλανές*

ΑΝΑΠΑΡΑΣΤΑΣΕΙΣ ΓΡΑΦΩΝ

- ΠΙΝΑΚΑΣ ΔΙΠΛΑΝΩΝ ΚΟΡΥΦΩΝ (ADJACENCY MATRIX)
- ΛΙΣΤΑ ΔΙΠΛΑΝΩΝ ΚΟΡΥΦΩΝ

ΠΙΝΑΚΑΣ ΔΙΠΛΑΝΩΝ ΚΟΡΥΦΩΝ

Σε κάθε γράφο G = (V, E) αντιστοιχεί ένας πίνακας (διπλανών κορυφών) A(G) = [a_{ij}] ως εξής: a_{ij} = 1 αν και μόνο εάν (v_i, v_j) ανήκει στο E και a_{ij} = 0, διαφορετικά.

ΠΙΝΑΚΕΣ ΔΙΠΛΑΝΩΝ ΚΟΡΥΦΩΝ

$$\begin{vmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{vmatrix}$$

G_1

$$\begin{vmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{vmatrix}$$

G_3

ΠΙΝΑΚΑΣ ΔΙΠΛΑΝΩΝ ΚΟΡΥΦΩΝ

```
typedef bool graph[N][N];

void join (graph * g,
           int node1, int node2)
{
   (*g)[node1][node2] = true;
}

void remove (graph * g,
             int node1, int node2)
{
   (*g)[node1][node2] = false;
}
```


ΛΙΣΤΕΣ ΔΙΠΛΑΝΩΝ ΚΟΡΥΦΩΝ (1)

```
typedef struct node_tag {
  int vertex;
  struct node_tag * link;
} * graph[N];

void join (graph * g, int node1, int node2)
{  /* Assume: the graph is directed */
   struct node_tag * temp = (*g)[node1];
   struct node_tag * new =
     (struct node_tag *)
        malloc(sizeof(struct node_tag));
   if (new == NULL) {
      fprintf(stderr, "Out of memory\n");
      exit(1);
   }
   new->link = temp; temp = new;
}
```

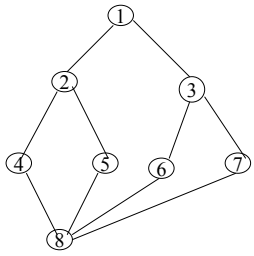

ΛΙΣΤΕΣ ΔΙΠΛΑΝΩΝ ΚΟΡΥΦΩΝ (2)

```
void remove (graph * g, int node1, int node2)
{
   (*g)[node1][node2] = false;
}
```


ΜΕΘΟΔΟΙ ΔΙΑΣΧΙΣΗΣ

- ΑΝΑΖΗΤΗΣΗ ΜΕ ΠΡΟΤΕΡΑΙΟΤΗΤΑ ΒΑΘΟΥΣ
- ΑΝΑΖΗΤΗΣΗ ΜΕ ΠΡΟΤΕΡΑΙΟΤΗΤΑ ΠΛΑΤΟΥΣ

ΑΝΑΖΗΤΗΣΗ ΜΕ ΠΡΟΤΕΡΑΙΟΤΗΤΑ ΒΑΘΟΥΣ



1 →2 → 4 → 8 → 5 → 6 → 3 → 7

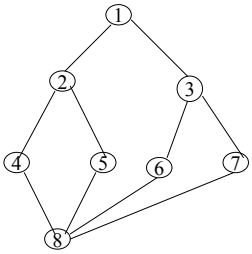
ΑΝΑΖΗΤΗΣΗ ΜΕ ΠΡΟΤΕΡΑΙΟΤΗΤΑ ΒΑΘΟΥΣ (1)

```
typedef struct node_tag {
  int vertex;
  struct node_tag * link;
} * graph[N];

void traverseDFS (graph g)
{
  bool visited[N];
  int i;

  for (i=0; i<N; i++)
    visited[i] = false;
  traverseDFS_aux(g, visited, 0);
}
```


ΑΝΑΖΗΤΗΣΗ ΜΕ ΠΡΟΤΕΡΑΙΟΤΗΤΑ ΒΑΘΟΥΣ (2)

```
void traverseDFS_aux (graph g,
                      bool visited[N],
                      int current)
{
  struct node_tag * adj;

  printf("Vertex %d\n", current);
  visited[current] = true;
  for (adj = g[current]; adj != NULL;
       adj = adj->link)
    if (!visited[adj->vertex])
       traverseDFS_aux(g, visited,
                       adj->vertex);
}
```


ΑΝΑΖΗΤΗΣΗ ΜΕ ΠΡΟΤΕΡΑΙΟΤΗΤΑ ΠΛΑΤΟΥΣ



1 →2 → 3 → 4 → 5 → 6 → 7 → 8

ΑΝΑΖΗΤΗΣΗ ΜΕ ΠΡΟΤΕΡΑΙΟΤΗΤΑ ΠΛΑΤΟΥΣ (1)

```
typedef struct node_tag {
  int vertex;
  struct node_tag * link;
} * graph[N];

void traverseBFS (graph g)
{
  struct node_tag * adj;
  bool visited[N];
  queue q;
  int current, i;

  for (i=0; i<N; i++)
    visited[i] = false;
```


ΑΝΑΖΗΤΗΣΗ ΜΕ ΠΡΟΤΕΡΑΙΟΤΗΤΑ ΠΛΑΤΟΥΣ (2)

```
  q = queueEmpty;
  queueInsert(&q, 0);

  while (!queueIsEmpty(q)) {
    int current = queueRemove(&q);

    printf("Vertex %d\n", current);
    visited[current] = true;

    for (adj = g[current]; adj != NULL;
         adj = adj->link)
      if (!visited[adj->vertex])
        queueInsert(&q, adj->vertex);
  }
}
```

ΤΟ ΠΡΟΒΛΗΜΑ ΤΟΥ ΣΥΝΤΟΜΟΤΕΡΟΥ ΜΟΝΟΠΑΤΙΟΥ

Αν υπάρχουν πολλά μονοπάτια από την κορυφή I στην κορυφή J ζητάμε το συντομότερο μονοπάτι ανάμεσα στις κορυφές.