ΠΕΡΙΕΧΟΜΕΝΑ:

Σπύρος Παπαγιάκουμος

Χρυσός Χορηγός Μαθήματος

Χρήστος Σ.

Ασημένιος Χορηγός Μαθήματος

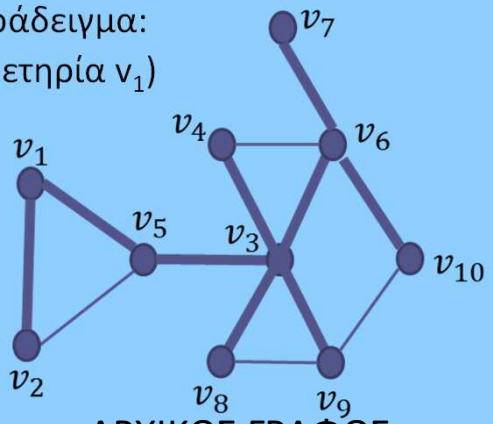
ΜΑΘΗΜΑ 5: Διαπεράσεις Γράφων 1. Διαπέραση Πρώτα Κατά Πλάτος (1. Περιγραφή) Αλγόριθμοι σε C psounis

- **Διαπέραση Γραφου:**
  - Είναι ένας συστηματικός τρόπος για να επισκεφθούμε όλες τις κορυφές ενός γραφήματος
  - (Αναλυτικά: Βλέπε Θεωρία Γράφων: Μάθημα 6.2)

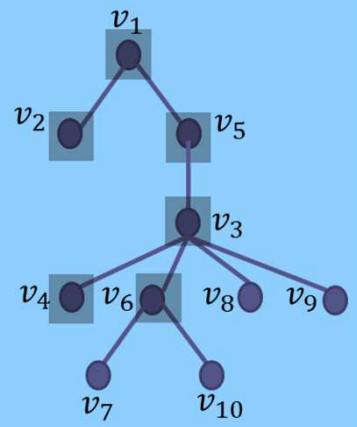
Διαπέραση πρώτα κατά πλάτος (Breadth First Search):

- **Αρχικοποίηση:**
  - Τοποθετούμε αυθαίρετα μία κορυφή στο συνδετικό δένδρο
- **Σε κάθε βήμα:**
  - Επιλέγουμε τρέχουσα κορυφή (με βάση τη σειρά με την οποία μπήκε στο συνδετικό δένδρο)
  - Κάθε γειτονική της κορυφή (που δεν έχει ήδη μπει στο δένδρο) την θέτουμε ως παιδί της
- **Τερματισμός:**
  - Όταν όλες οι κορυφές μπουν στο συνδετικό δένδρο

Παράδειγμα:
(Αφετηρία v_1)



ΑΡΧΙΚΟΣ ΓΡΑΦΟΣ



ΣΥΝΔΕΤΙΚΟ ΔΕΝΔΡΟ

Παρατηρήσεις για την υλοποίηση:

- Επιλέγουμε την αποτύπωση του γράφου με λίστες γειτνίασης (λόγω πιο γρήγορης πρόσβασης στους επόμενους μιας κορυφής)
- Χρησιμοποιούμε έναν πίνακα discovered για να μαρκάρουμε τις κορυφές που έχουμε ήδη επισκεφθεί.
- Επίσης χρησιμοποιούμε μία ουρά για να διατηρήσουμε την σειρά με την οποία μπήκαν οι κορυφές στο συνδετικό δένδρο.

Αλγόριθμος σε Ψευδογλώσσα (~ από Wikipedia):

```
procedure BFS(G, ρίζα):
    Q: ουρά
    discovered: πίνακας ακεραίων αρχικοποιημένος σε 0

    enqueue(Q, ρίζα)
    discovered[ρίζα] = 1
    while (Q δεν είναι άδεια)
        v = dequeue(Q)
        Για κάθε γειτονική κορυφή w της v:
            Αν η w δεν έχει εξερευνηθεί:
                discovered[w] = 1
                Θέσε την ακμή [v,w] στο συνδετικό δένδρο
                enqueue(Q, w)
    Επέστρεψε το συνδετικό δένδρο
```

Πολυπλοκότητα:

- Η πολυπλοκότητα του αλγορίθμου είναι O(m)

Υλοποίηση σε C (βλέπε project BFSSpanningTree):

```
GRAPH DFSSpanningTree(GRAPH g, int start)
{
  GRAPH st;
  QUEUE qu;
  int *discovered;
  int i, v, cnt, vertices, neighbors_length;
  int stop_loop, *neighbors;

  vertices = GR_vertices_count(g);
  GR_init(&st, vertices);

  discovered = (int *)malloc(sizeof(int)*vertices);
  if (!discovered)
  {
     printf("Error Allocating Memory!");
     exit(0);
  }
  for (i=0; i<vertices; i++)
     discovered[i] = 0;

  QU_init(&qu);
  QU_enqueue(&qu, start);
  discovered[start]=1;
  cnt=1; stop_loop=FALSE;
  while(!QU_empty(qu)&&!stop_loop)
  {
     QU_dequeue(&qu, &v);
     GR_neighbors(g, v, &neighbors_length, &neighbors);
     for (i=0; i<neighbors_length; i++)
     {
        if (!discovered[neighbors[i]])
        {
           discovered[neighbors[i]]=1;
           QU_enqueue(&qu, neighbors[i]);
           GR_add_edge(st, v, neighbors[i]);
           cnt++;
           if (cnt==vertices)
           {
              stop_loop=TRUE;
              break;
           }
        }
     }
     free(neighbors);
  }
  QU_destroy(&qu);
  free(discovered);
  return st;
}
```

Παρατήρηση:

- Ο αλγόριθμος της διαπέρασης πρώτα κατά πλάτος, μπορεί να χρησιμοποιηθεί για τον εντοπισμό του συντομότερου μονοπατιού σε έναν απλό μη κατευθυνόμενο γράφο, μεταξύ 2 κορυφών

Παρατήρηση:

- Η σειρά με την οποία οι κορυφές μπαίνουν στο συνδετικό δένδρο είναι σημαντική.
- Μπορεί να χρησιμοποιηθεί σε άλλους αλγόριθμους (όπως π.χ. η τοπολογική ταξινόμηση)

Άσκηση 1: Συντομότερο Μονοπάτι με Κατά Πλάτος

Κατασκευάστε τη συνάρτηση BFSShortestPath η οποία με ορίσματα έναν απλό μη κατευθυνόμενο γράφο, μία αφετηρία και έναν προορισμό:

- Επιστρέφει έναν πίνακα με το συντομότερο μονοπάτι από την αφετηρία στον προορισμό, ως μια ακολουθία των διαδοχικών κόμβων που χρησιμοποιούνται.

Άσκηση 2: Ακολουθία κατά πλάτος

Κατασκευάστε τη συνάρτηση BFSSequence η οποία με όρισμα έναν απλό μη κατευθυνόμενο γράφο και μία αφετηρία:

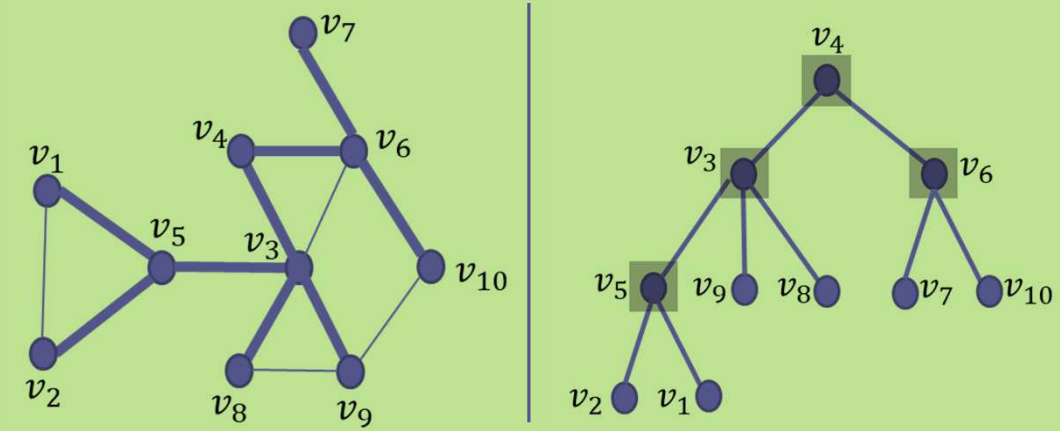
- Επιστρέφει τη σειρά επίσκεψης (σειρά με την οποία μπήκαν στο συνδετικό δένδρο)

Παρατήρηση:

- Είναι εμφανές ότι στην κατασκευή του συνδετικού δένδρου, έχει μεγάλη σημασία η σειρά των γειτόνων μιας κορυφής.
- Έτσι ορίζεται μια παραλλαγή, όπου μας δίνεται πρόσθετα μια διάταξη των κορυφών, έτσι ώστε να τη συμβουλευόμαστε, όταν πρόκειται να βάζουμε τους επόμενους μιας κορυφής στην ουρά.

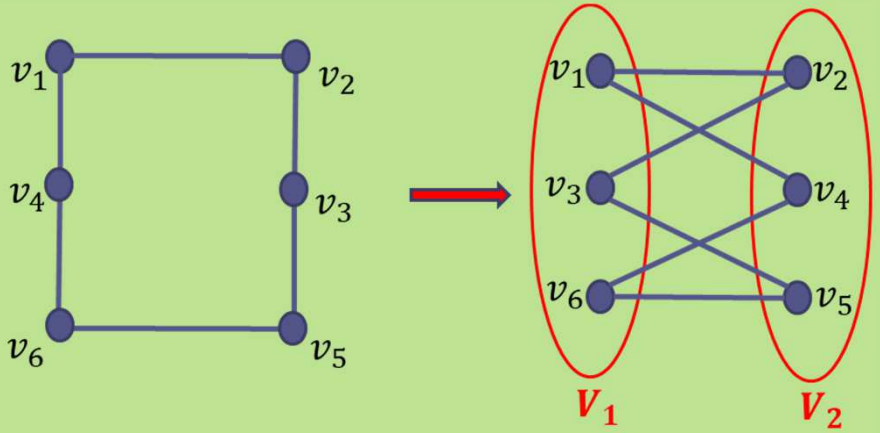
Παράδειγμα: διαπέρασης κατά πλάτος από τη v4 με τη διάταξη

$(v_4\ v_7\ v_5\ v_2\ v_9\ v_{10}\ v_8\ v_1\ v_3\ v_6)$



Παρατήρηση:

- Η διάσχιση κατά πλάτος μπορεί να χρησιμοποιηθεί για να ελεγχθεί αν ένα γράφημα είναι διχοτομίσιμο (ή διμερές).
- Διχοτομίσιμο είναι ένα γράφημα ανν οι κορυφές μπορούν να χωριστούν σε δύο μερίδια, έτσι ώστε κάθε ακμή να έχει το ένα της άκρο στο ένα μερίδιο και το άλλο της άκρο στο άλλο μερίδιο.



Άσκηση 3: Προτεραιότητα στις κορυφές

Κατασκευάστε τη συνάρτηση BFSSpanningTreePriorities() η οποία με ορίσματα έναν απλό μη κατευθυνόμενο γράφο, μία αφετηρία και έναν πίνακα προτεραιοτήτων:

- Επιστρέφει το συνδετικό δένδρο της κατά πλάτος, δεδομένων των προτεραιοτήτων.

Άσκηση 4: Συνδετικό Δένδρο

Κατασκευάστε τη συνάρτηση BFSBipartite η οποία με όρισμα έναν απλό μη κατευθυνόμενο γράφο:

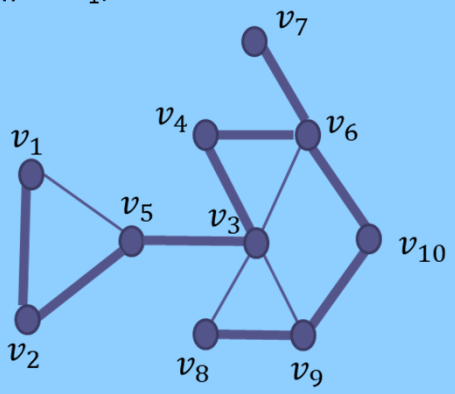
- Επιστρέφει TRUE/FALSE ανάλογα αν το γράφημα είναι διχοτομίσιμο ή όχι.
- Αν το γράφημα είναι διμερές να τυπώνει και τη διαμέριση.

ΜΑΘΗΜΑ 5: Διαπεράσεις Γράφων 2. Διαπέραση Πρώτα Κατά Βάθος (1. Περιγραφή) Αλγόριθμοι σε C psounis

Διαπέραση πρώτα κατά βάθος (Depth First Search):

- **Αρχικοποίηση:**
  - Τοποθετούμε μία "βολίδα" σε μια (αυθαίρετη) κορυφή. Την κορυφή την τοποθετούμε στο συνδετικό δένδρο
- **Σε κάθε βήμα:**
  - Αν υπάρχει γειτονική κορυφή που δεν έχει επισκεφθεί, μεταβαίνει και την τοποθετεί στο συνδετικό δένδρο μαζί με την ακμή μετάβασης.
  - Αν δεν υπάρχει κορυφή που δεν έχει επισκεφθεί, πηγαίνει στην ακριβώς προηγούμενη κορυφή που είχε επισκεφθεί.
- **Τερματισμός:**
  - Όταν όλες οι κορυφές μπουν στο συνδετικό δένδρο

Παράδειγμα:
(Αφετηρία v_1)



ΑΡΧΙΚΟΣ ΓΡΑΦΟΣ

ΣΥΝΔΕΤΙΚΟ ΔΕΝΔΡΟ

Παρατηρήσεις για την υλοποίηση:

- Επιλέγουμε την αποτύπωση του γράφου με λίστες γειτνίασης (λόγω πιο γρήγορης πρόσβασης στους επόμενους μιας κορυφής)
- Χρησιμοποιούμε έναν πίνακα discovered για να μαρκάρουμε τις κορυφές που έχουμε ήδη επισκεφθεί.
- Επίσης χρησιμοποιούμε μία στοίβα για να διατηρήσουμε την σειρά με την οποία θα οπισθοδρομήσουμε στο δένδρο.

Αλγόριθμος σε Ψευδογλώσσα (~ από Wikipedia):

```
procedure DFS(G, ρίζα):
  S: στοίβα
  discovered: πίνακας ακεραίων αρχικοποιημένος σε 0

 discovered[ρίζα] = 1
  push(S, (γείτονας, ρίζα)) για κάθε γείτονα της ρίζας
  while (S δεν είναι άδεια και δεν έχουν εξερευνηθεί όλες οι κορυφές)
      (v, γονέας(v)) = pop(S)
      Αν η v δεν έχει εξερευνηθεί:
          discovered[v] = 1
          Πρόσθεσε την ακμή (v, γονέας(v)) στο συνδετικό δένδρο
          Για κάθε γειτονική κορυφή w της v:
                push(S, (w, v))
  Επέστρεψε το συνδετικό δένδρο
```

Πολυπλοκότητα:

- Η πολυπλοκότητα του αλγορίθμου είναι O(m)

Υλοποίηση σε C (βλέπε project DFSSpanningTree):

```
GRAPH DFSSpanningTree(GRAPH g, int start)
{
   GRAPH st; STACK s;  selem edge;
   int *discovered;
   int i, v, cnt, vertices, neighbors_length;
   int stop_loop, *neighbors;

   vertices = GR_vertices_count(g);
   GR_init(&st, vertices);

   discovered = (int *)malloc(sizeof(int)*vertices);
   if (!discovered) {
      printf("Error Allocating Memory!");
      exit(0);
   }
   for (i=0; i<vertices; i++)
      discovered[i] = 0;

   ST_init(&s);
   discovered[start]=1;
   cnt=1;
   GR_neighbors(g, start, &neighbors_length, &neighbors);
   for (i=neighbors_length-1; i>=0; i--)
   {
      edge.node = neighbors[i], edge.parent = start;
      ST_push(&s, edge);
   }
   free(neighbors);
   stop_loop=FALSE;
   while(!ST_empty(s)&&!stop_loop)
   {
      ST_pop(&s, &edge);
      v = edge.node;
      if (!discovered[v])
      {
         discovered[v]=1;
         GR_add_edge(st, v, edge.parent);
         cnt++;
         if (cnt==vertices)
         {
            stop_loop=TRUE;
            break;
         }
         GR_neighbors(g, v, &neighbors_length, &neighbors);
         for (i=neighbors_length-1; i>=0; i--)
         {
            edge.node = neighbors[i], edge.parent = v;
            ST_push(&s, edge);
         }
         free(neighbors);
      }
   }
   ST_destroy(&s);
               free(discovered);
               return st;
}
```

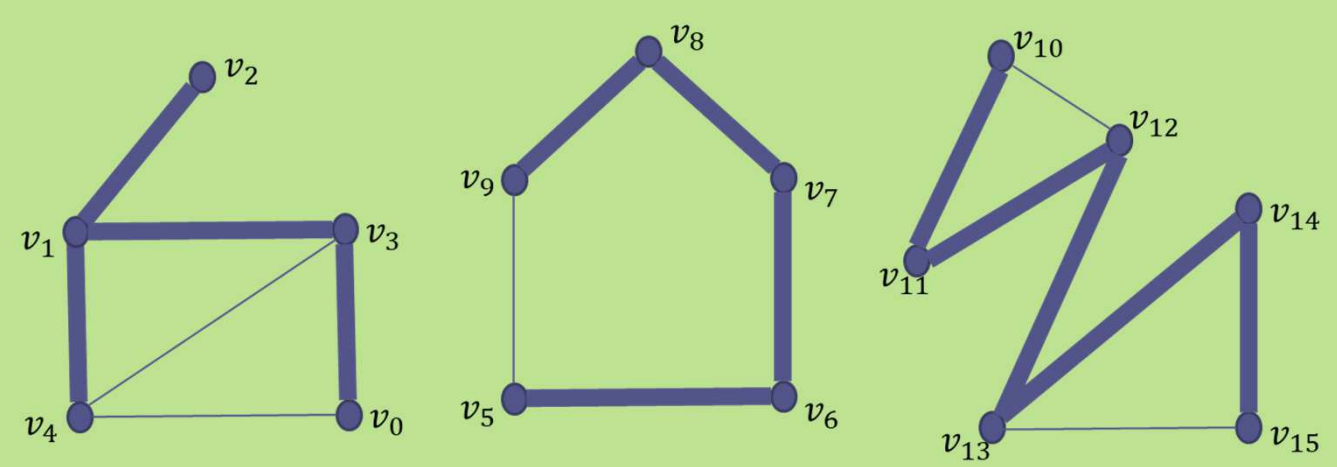
Άσκηση 5: Ακολουθία κατά βάθος

Κατασκευάστε τη συνάρτηση DFSSequence η οποία με όρισμα έναν απλό μη κατευθυνόμενο γράφο και μία αφετηρία:

- Επιστρέφει τη σειρά επίσκεψης (σειρά με την οποία μπήκαν στο συνδετικό δένδρο)

Παρατήρηση:

- Όλες οι παραλλαγές που εξετάσαμε αφορούσαν συνδεόμενα γραφήματα.
- Ωστόσο οι αλγόριθμοι αυτοί μπορούν να τρέξουν και για μη συνδεόμενα γραφήματα, παράγοντας ένα συνδετικό δάσος (συνεκτικές συνιστώσες που η κάθε μία είναι δένδρο)



Άσκηση 6: Μη Συνδεόμενα Γραφήματα

Τροποποιήστε κατάλληλα τον κώδικα της DFSSpanningTree, παράγοντας τη συνάρτηση DFSSpanningForest:

- Με είσοδο ένα μη συνδεόμενο γράφημα, θα εκτυπώνει ένα δάσος από συνδετικά δένδρα των αντίστοιχων συνιστωσών.