Διάλεξη 21: Γράφοι ΙΙ - Τοπολογική Ταξινόμηση

Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:

- Τοπολογική Ταξινόμηση
- Εφαρμογές, Παραδείγματα, Αλγόριθμοι

Τοπολογική Ταξινόμηση (Topological Sort)

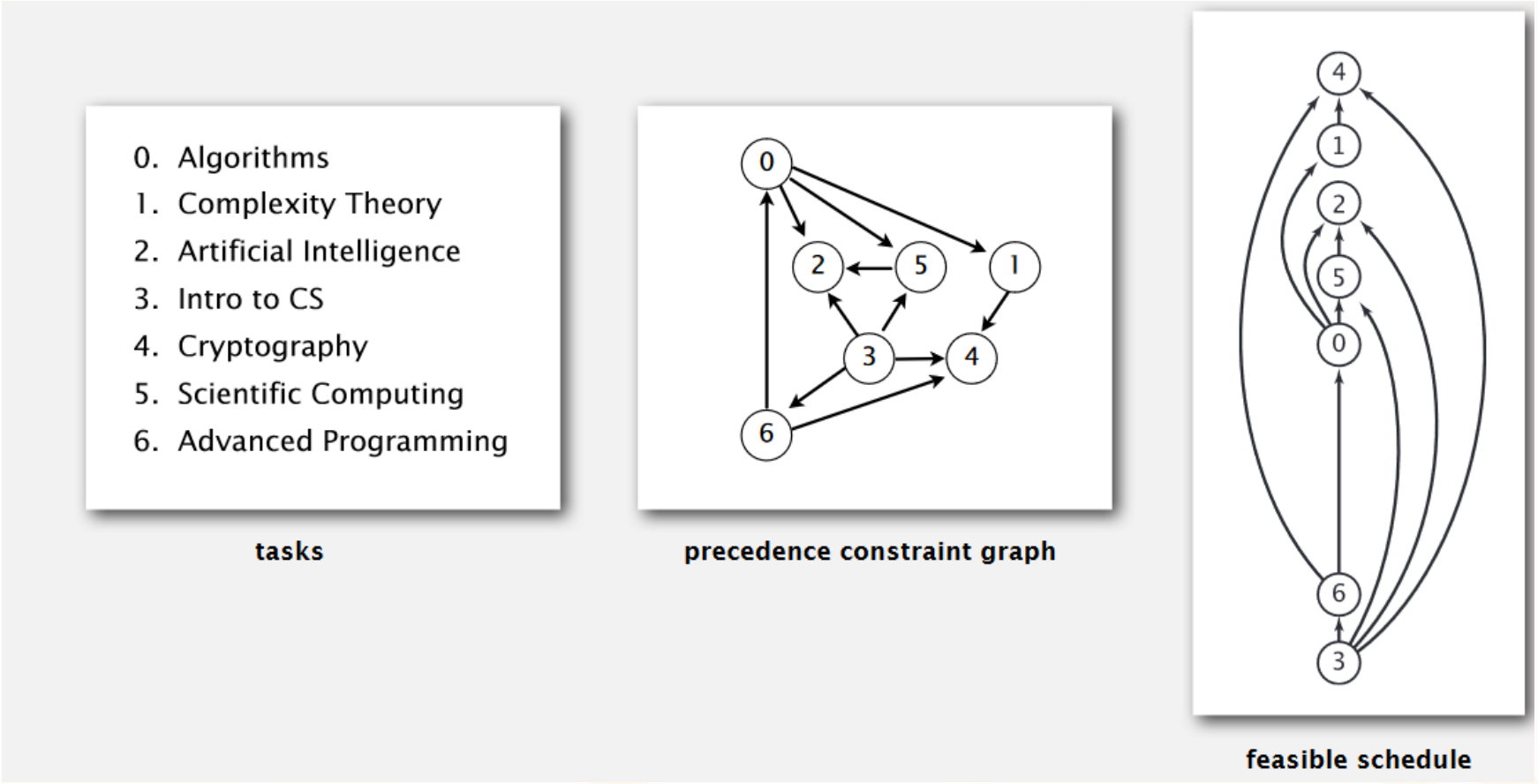
- Δίνεται ένα **σύνολο εργασιών** και θέλουμε να ορίσουμε τη **σειρά με την οποία πρέπει να εκτελέσει** τις εργασίες ένας επεξεργαστής, δεδομένης της **ύπαρξης περιορισμών** ως προς την προτεραιότητά τους.

- Κάθε εργασία έχει ένα σύνολο προαπαιτούμενων εργασιών, δηλαδή δεν μπορεί να εκτελεσθεί προτού συμπληρωθεί κάθε μια από τις προαπαιτούμενες.

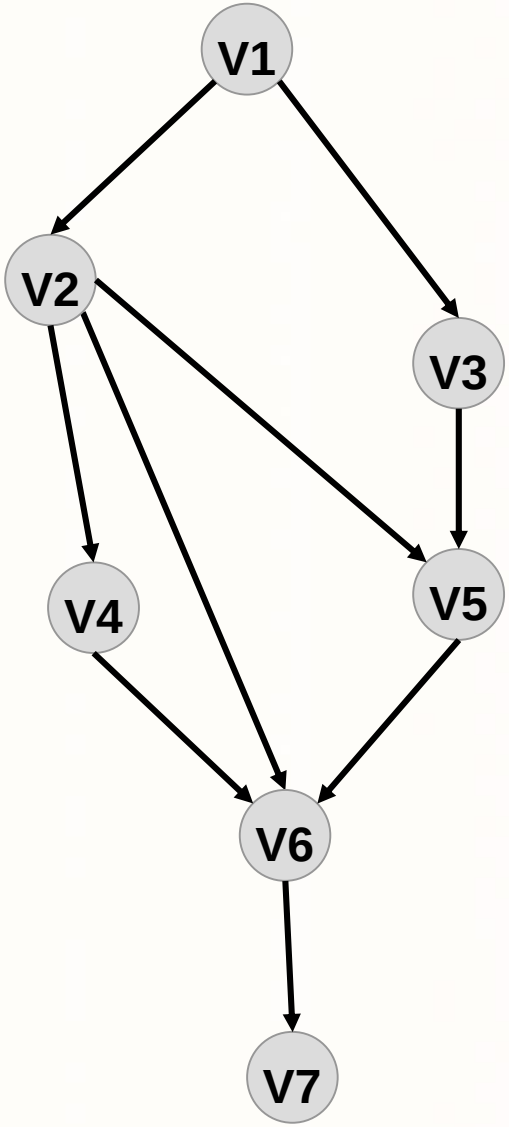
- Μπορούμε να παραστήσουμε το πρόβλημα ως έναν κατευθυνόμενο γράφο:
  - Οι **κορυφές** του γράφου αντιστοιχούν σε κάθε μια από τις εργασίες, και
  - η ύπαρξη **ακμής** από την κορυφή **A** στην κορυφή **B** δηλώνει ότι η εργασία A πρέπει να εκτελεστεί πριν από τη B.

- Τοπολογική ταξινόμηση του γράφου είναι **μια σειρά των κορυφών** του, v_1, …, v_n, ώστε αν (v_i, v_j) είναι ακμή του γράφου τότε i εκτελείται πριν το j (δηλαδή i<j).

- Η διαδικασία της τοπολογικής ταξινόμησης στην ουσία μας λέει την σειρά με την οποία μπορούμε να βάλουμε τους κόμβους σε μια σειρά έτσι ώστε καμία ακμή να μη δείχνει προς τα πίσω



Παράδειγμα Τοπολογικών Ταξινομήσεων



Τοπολογικές Ταξινομήσεις του γράφου:

1. V1, V2, V4, V3, V5, V6, V7
2. V1, V2, V3, V5, V4, V6, V7
3. V1, V3, V2, V5, V4, V6, V7
4. …

Αλγόριθμος για Τοπολογική Ταξινόμηση

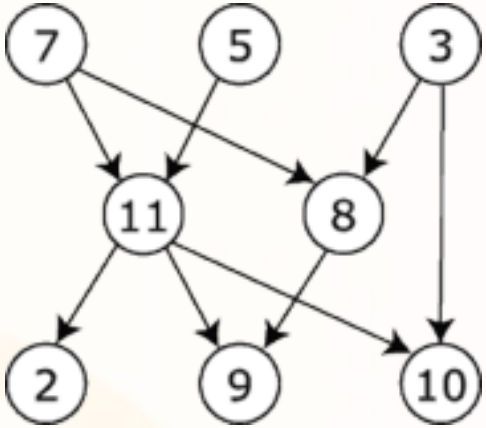
- **Βαθμός εισόδου (in-degree):** ενός κόμβου είναι ο αριθμός των ακμών που καταλήγουν στον κόμβο. (Στο πρόβλημα μας, ο αριθμός των προαπαιτούμενων εργασιών)
- Για κάθε κορυφή **u** έστω **I[u]** ο βαθμός εισόδου (αριθμός γονέων) της **u.**

ΑΛΓΟΡΙΘΜΟΣ

Επαναλαμβάνουμε τα εξής βήματα:

1. διαλέγουμε κορυφή Α με I[A]=0,
2. τυπώνουμε την Α,
3. για όλες τις κορυφές Β, όπου υπάρχει η ακμή (Α,Β) μειώνουμε την τιμή I[B] κατά 1.

ΠΑΡΑΔΕΙΓΜΑ



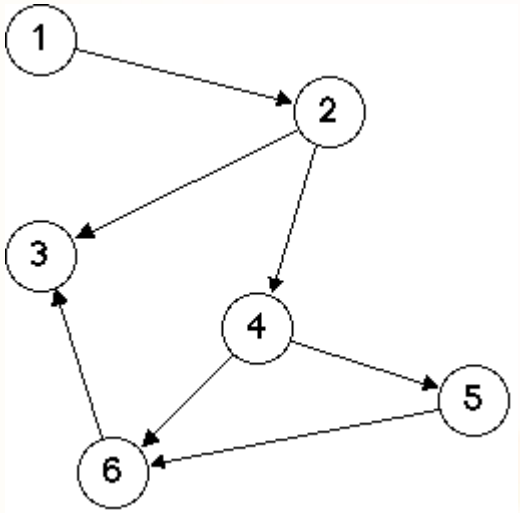
ΑΠΟΤΕΛΕΣΜΑΤΑ(Υπάρχουν αρκετές τοπολογικές ταξινομήσεις)

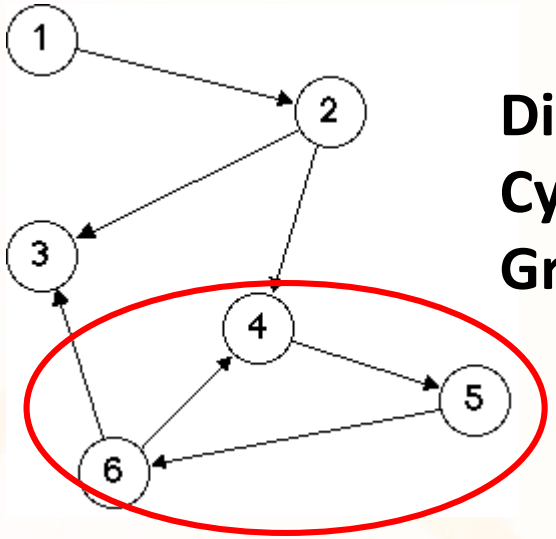
- **7,5,3,11,8,2,9,10**
- **7,5,11,2,3,10,8,9**
- …..

Κατευθυνόμενοι Μη-Κυκλικοί Γράφοι

- Ωστόσο η τοπολογική ταξινόμηση δουλεύει μόνο για μια ειδική κλάση γράφων η οποία ονομάζεται **DAGs**
- **DAG: Directed Acyclic Graphs (Κατευθυνόμενοι Μη-Κυκλικοί Γράφοι)**: Ένας γράφος στο οποίο κανένα μονοπάτι δεν ξεκινά και τελειώνει στον ίδιο κόμβο.

Directed Acyclic Graph





Directed Cyclic Graph

- Το γεγονός ότι δουλεύει μόνο για DAGs είναι αναμενόμενο διότι αν υπήρχαν κυκλικά μονοπάτια τότε δεν θα υπήρχε κάποια σειρά (ordering) μεταξύ των στοιχείων (αφού δεν θα ξέραμε ποια είναι η αρχή)

(Ψευδό)-Υλοποίηση 1

```
topSort1( graph G ){
  // αρχικοποίηση πίνακα μεγέθους |V|
  int I[|V|] = {};
  // μέτρηση in-degree για κάθε κόμβο
  for each vertex u
    for each edge (u,v)
       I[v]++;
// προσπέλαση του γράφου
  for (i=1; i <= |V|; i++){
    v = FindVertexOfIndegree0;
    if (v == NULL) {// δεν υπάρχει κόμβος
                    //με in-degree=0
          Error("Graph has a cycle");
          return;
    }
    print v;              // εκτύπωση κόμβου
    for each edge (v,w) // μείωση in-degree
       I[w]--;          // για κάθε παιδί
  }
```

O(|V| + |E|)

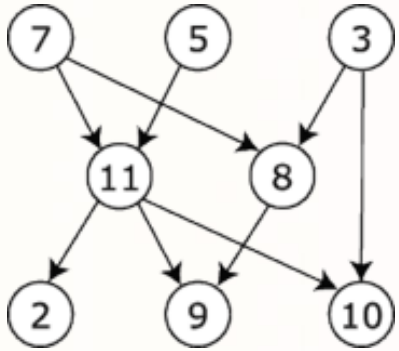
O(|V|)

O(|E|)

O(|V|² + |E|)

Χρόνος Εκτέλεσης:

O(|V|²+E)



(Ψευδό)-Υλοποίηση 2 (Με Χρήση Ουράς)

```
topologicalSort( graph G ){
 Queue Q; // ορισμός βοηθητικής ουράς
 // αρχικοποίηση πίνακα μεγέθους |V|
 int I[|V|] = {};
 // μέτρηση indegree για κάθε κόμβο
 for each vertex u
    for each edge (u,v)
        I[v]++;
 // τοποθέτησε κάθε στοιχείο με indegree=0 σε μια ουρά
 for each vertex u
      if (I[u]==0) Enqueue(u, Q);
 while (! IsEmpty(Q)){
      u = Dequeue(Q);
      output u; number_of_outputs++;
      for each (u,v) {
         I[v]--;
         if (I[v]==0) Enqueue (v, Q);
      }
 }
 // Εάν δεν εκτυπώθηκαν όλοι οι κόμβοι τότε είχε κύκλο ο γράφος γιατί
  κάποιοι κόμβοι δεν πήραν ποτέ indegree = 0;
 if (number_of_outputs != |V|)
     Error("Graph has a cycle");
```

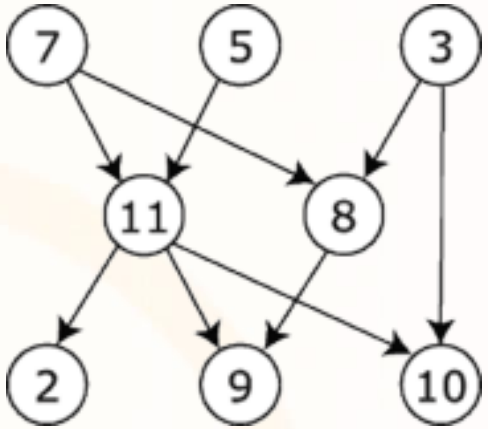
Θ(|V| + |E|)

Θ(|V|)

Θ(|V| + |E|)

Χρόνος Εκτέλεσης:

Θ(|V| + |E|)

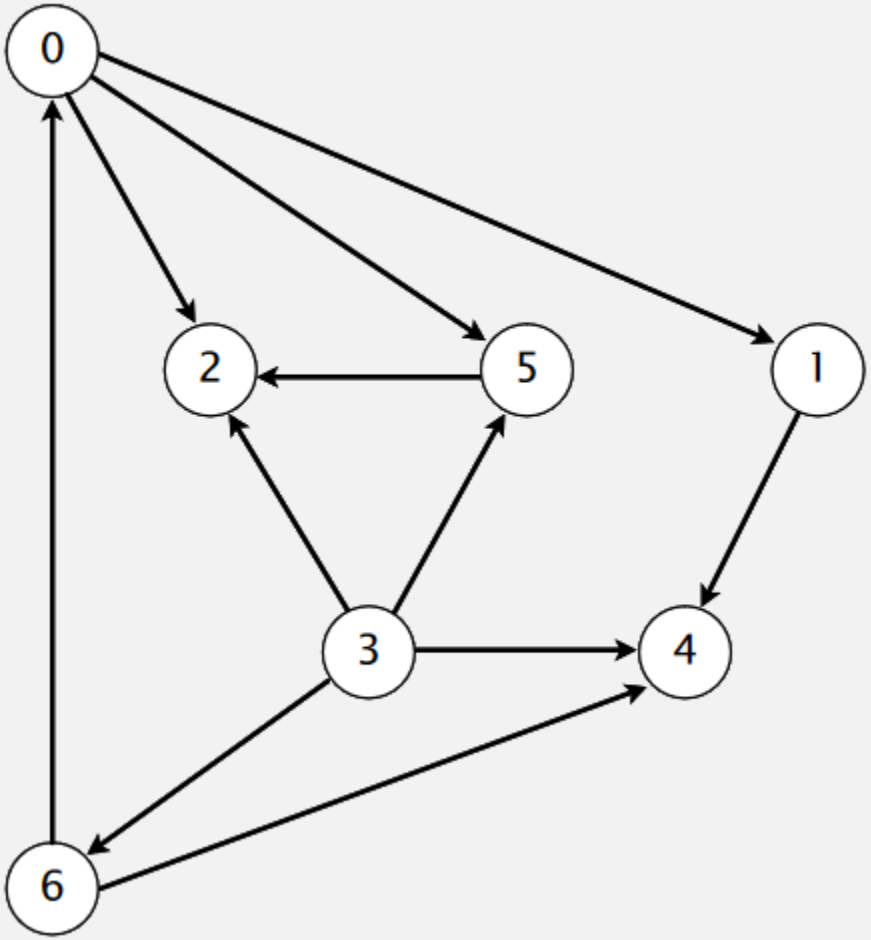


Αλγόριθμος 3… DFS + Stack!

```
public class DepthFirstOrder
{
   private boolean[] marked;
   private Stack<Integer> reversePost;

   public DepthFirstOrder(Digraph G)
   {
      reversePost = new Stack<Integer>();
      marked = new boolean[G.V()];
      for (int v = 0; v < G.V(); v++)
         if (!marked[v]) dfs(G, v);
   }

   private void dfs(Digraph G, int v)
   {
      marked[v] = true;
      for (int w : G.adj(v))
         if (!marked[w]) dfs(G, w);
      reversePost.push(v);
   }

   public Iterable<Integer> reversePost()
   {  return reversePost;  }
}
```

returns all vertices in "reverse DFS postorder"

Παράδειγμα με DFS



postorder

4 1 2 5 0 6 3

topological order

3 6 0 5 2 1 4

Σύνοψη

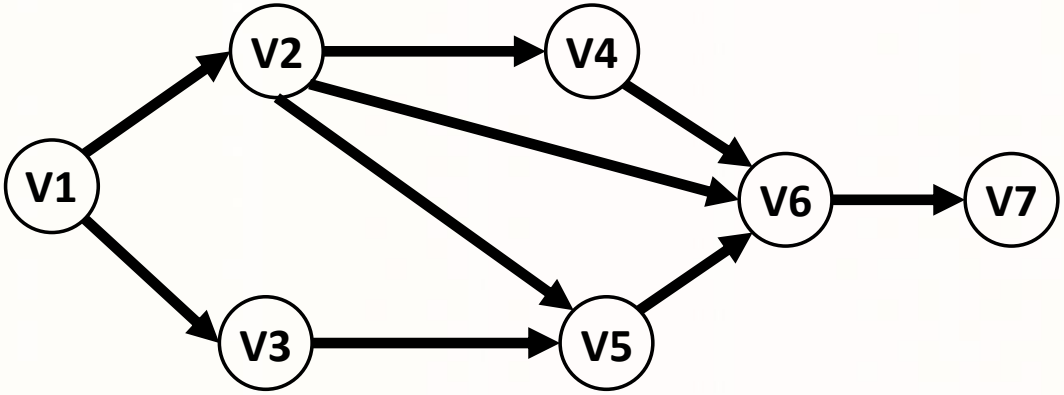
- Ο χρόνος εκτέλεσης της TopologicalSort είναι
  - Για υλοποίηση με λίστες γειτνίασης: $O(|V| + |E|)$
  - Για υλοποίηση με πίνακες γειτνίασης: $O(|V|^2)$

- Για αραιούς γράφους όπου $|E| \in O(|V|)$ η πολυπλοκότητα αυτή μεταφράζεται σε:
  - $O(|V|)$, για υλοποίηση με λίστες γειτνίασης, ενώ
  - για υλοποίηση με πίνακες γειτνίασης παραμένει $O(|V|^2)$.

- Για πυκνούς γράφους όπου $|E| \in O(|V|^2)$ η πολυπλοκότητα αυτή μεταφράζεται σε:
  - $O(|V|^2)$, για υλοποίηση με λίστες γειτνίασης, και
  - για υλοποίηση με πίνακες γειτνίασης παραμένει $O(|V|^2)$.

Γράφοι: Ασκήσεις

- Με δεδομένο εισόδου τον πιο κάτω κατευθυνόμενο γράφο, ποια επιλογή ΔΕΝ αποτελεί το αποτέλεσμα της τοπολογικής ταξινόμησης;



- Α. V1, V2, V4, V3, V5, V6, V7
- Β. V1, V2, V3, V5, V6, V4, V7
- Γ. V1, V2, V3, V5, V4, V6, V7
- Δ. V1, V3, V2, V5, V4, V6, V7

Παράδειγμα 1: Εφαρμογή Γράφων

- Η εταιρεία τηλεπικοινωνιών EPL-NET έχει ένα δίκτυο από σταθμούς που ενώνονται μεταξύ τους από n επικοινωνιακές γραμμές υψηλής ταχύτητας. Το τηλέφωνο του κάθε συνδρομητή της εταιρείας συνδέεται με το σταθμό της περιοχής του. Κάθε σταθμός συνδέεται με άλλους χρήστες αλλά και με άλλους σταθμούς. Οι ερευνητές της εταιρείας έχουν ανακαλύψει μία νέα τεχνολογία που επιτρέπει σε δύο χρήστες να βλέπουν και τρισδιάστατη εικόνα κατά τη διάρκεια της τηλεφωνικής τους επικοινωνίας. Για να είναι όμως η ποιότητα της εικόνας ικανοποιητική, πρέπει ο αριθμός των γραμμών που χρησιμοποιούνται για τη σύνδεση ανάμεσά στους δύο χρήστες (και ενδιάμεσους σταθμούς) να μην ξεπερνά τις 4 συνδέσεις.

- Να σχεδιάσετε ένα αποδοτικό αλγόριθμο (ψευδοκώδικα) ο οποίος με δεδομένο εισόδου την ταυτότητα κάποιου χρήστη u, να επιστρέφει (εκτύπωση) τη λίστα με όλους τους χρήστες που μπορεί να χρησιμοποιήσουν την καινούρια τεχνολογία με τον u.

Παράδειγμα 1: Εφαρμογή Γράφων

Το πρόβλημα μπορεί να μετατραπεί σε πρόβλημα διερεύνησης κατά πλάτος σε γράφο αρχίζοντας από τον χρήστη (κορυφή) u και μετά εκτελώντας την διερεύνηση μέχρι το βάθος 4.

```
BFS(vertex u, graph G){
  visited[|V|];
  queue Q;
  Q=MakeEmptyQueue();
  for each v in G
    visited[v]=False;
  visited[u]= True;
  Enqueue(u,Q, 0);
  while (!IsEmpty(Q)){
    (w, h) = Dequeue(Q);
    print(w);
    if(h+1>4) continue;
    for each v adjacent to w
    if (Visited[v]==False){
      Visited[v]=True;
     Enqueue(v,Q, h+1);
    }
}
```