# CS C341 / IS C361 Data Structures & Algorithms

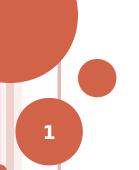
## **GRAPH ALGORITHMS**

**Graph Generation - Static vs. Dynamic Traversals** 

**Depth First Search** 

- Algorithm
- Properties and Time Complexity
- Applications

**Breadth First Search** 



## **GRAPH GENERATION**

- Graph generation can be static or dynamic :
  - If the graph on which an algorithm is to be applied is already available then it is static
  - If the graph can change during execution i.e. insertions / deletions may happen - then it is dynamic
- Examples
  - Traffic networks
  - Set of states in program execution
  - World Wide Web (web pages and hyperlinks)

## **GRAPH TRAVERSAL**

- Given a graph G, a traversal is a systematic procedure for exploring G by examining its vertices (and edges)
  - E.g. web spider / crawler
  - E.g. find operation in Unix/Linux
  - E.g. A broadcast in a network
- Depth First Search (DFS) in an undirected graph:
  - A traversal which explores one path completely before exploring another would be "depth" first.
  - "Backtrack" to explore the next path
    - i.e. consider a branch not taken in exploring the previous path

## DEPTH FIRST SEARCH

#### Outline:

- Start at a given "root" vertex v and recursively visit adjacent vertices;
- If you encounter an explored edge or an explored vertex then backtrack;
  - Keep track of explored edges and vertices

## **DEPTH FIRST SEARCH**

- OFS(G,v) // v is "visited"
  - for each vertex u that is adjacent to v such that (v,u) is "unexplored"
    - oif u is "not visited"
      - omark u as "visited"
      - label (v,u) as "discovery edge" (or as "tree edge")
      - OFS(G,u)
    - else
      - omark (v,u) as "back edge"

## DEPTH FIRST SEARCH - PROPERTIES

- Claim DFS1:
  - DFS(G,v) visits all vertices in the connected component of v
- Definition : Connected Component
  - A connected component G' = V',E' of a graph G=V,E, is such that
    - °V' is a subset of V
    - °E' is a subset of E
    - oand between any pair of vertices u and v in V', there is a path in E'.

## DEPTH FIRST SEARCH - PROPERTIES

#### • Claim DFS2:

- The discovery edges marked in DFS(G,v) form a "spanning tree" of the connected component of v.
  - The spanning tree is referred to as the "DFS tree"
  - The edges are referred to as "tree edges"
- Definition: Spanning Tree
  - Given a connected graph G = (V,E), a spanning tree T is a tree (V,E') such that E' is a subset of E.
    - Note that a tree is connected by definition.

## 04/14/14

## DEPTH FIRST SEARCH - IMPLEMENTATION

- Assume an adjacency lists representation for G.
- DFS(G,v) // v is "visited"
  - for each vertex u that is adjacent to v such that (v,u) is "unexplored"
    - o if u is "not visited"
      - mark u as "visited"
      - label (v,u) as "discovery edge" (or as "tree edge")

**Space Complexity:** 

Recursion /

backtracking

- DFS(G,u)
- else
- •O(d(w)) time per vértex w
- Total time:
- •Total time:
- $\sum_{w \text{ in } G'} d(w)$  where G' is the connected component of root
- i.e. O(m) where m = |E'| and G'=(V',E')

## **DFS**

#### • Theorem:

 A DFS traversal of G can be performed in O(|V|+|E|) time.

```
dft(G) { // DFS traversal - G may not be
  connected
  Let (V,E)=G.
  for j = 1 to |V| {
     if (j is "not visited") { mark j as "visited"; dfs(G,v); }
  }
}
```

### ALGORITHMS USING DFS

## Corollary:

- O(|V|+|E|) time DFS-based algorithms exist for the following problems:
  - Test whether G is connected
  - Find the connected components of G
  - Find a spanning forest of G
  - Find a path between two vertices of G, if it exists
  - Find a cycle in G if it exists

## BREADTH FIRST SEARCH (BFS)

- Traverse level-by-level
  - Discovery edges
  - Cross edges
  - BFS Tree Spanning Tree
- BFS outline
  - Start with a root vertex and at level 0
  - Maintain a FIFO queue
  - Insert all vertices at a level into the queue
     Discover edges from current level to next level
    - Discover edges from current level to flext lev
  - Traverse until queue is empty