SC1007
Data Structures and Algorithms

Week 9: Graph Traverse
BFS & DFS

College of Engineering

School of Computer Science and Engineering

Instructor: Luu Anh Tuan

Email: anhtuan.luu@ntu.edu.sg

Office: #N4-02b-66

Overview

- Traversal of Graph
- Breadth-first Search
- Depth-first Search

Traversal of Graphs

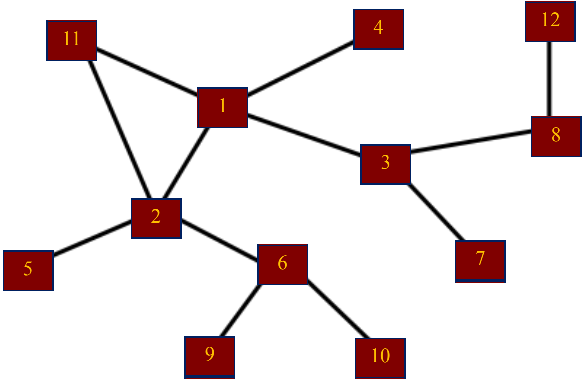
- To traverse a graph means to visit the vertices of the graph in some systematic order.

- The traversal problem: check all nodes once and only once

- To traverse a graph, we can apply:
    - Breadth-first Search
    - Depth-first Search

Breadth First Search (BFS)

- Work similar to level-order traversal of the trees
- BFS systematically explores the edges directly connected to a vertex before visiting vertices further away.

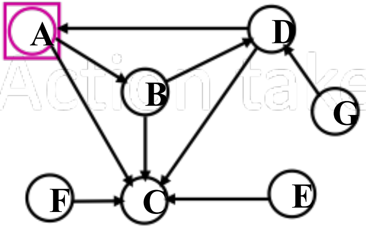


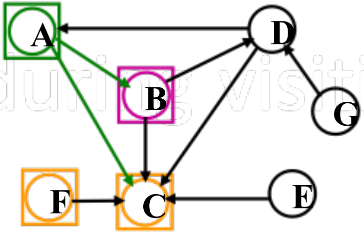
Breadth First Search (BFS)

```
typedef struct _linkedlist{
    ListNode *head;
    int size;
} LinkedList;
```

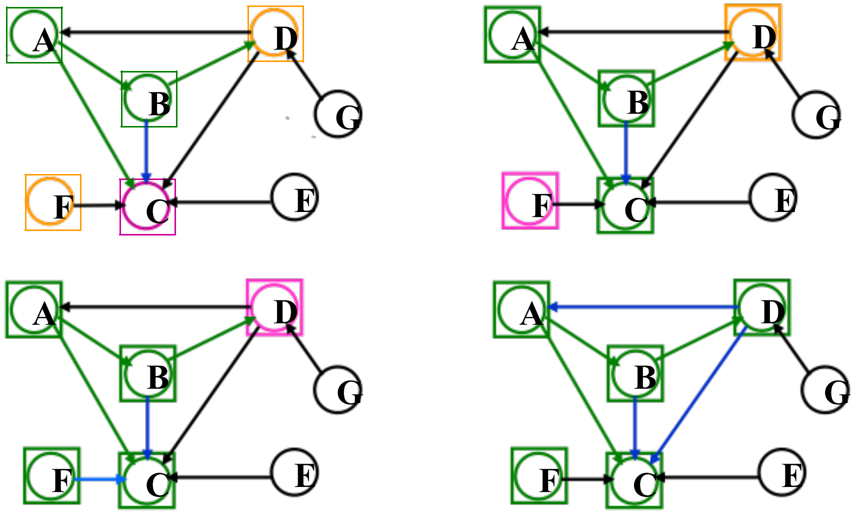
```
typedef ListNode QueueNode;
typedef struct _queue{
  int size;
  ListNode *head;
  ListNode *tail;
} Queue;
```

-
  - A queue is used to monitor whic
  - Action taken during visiting v_i de



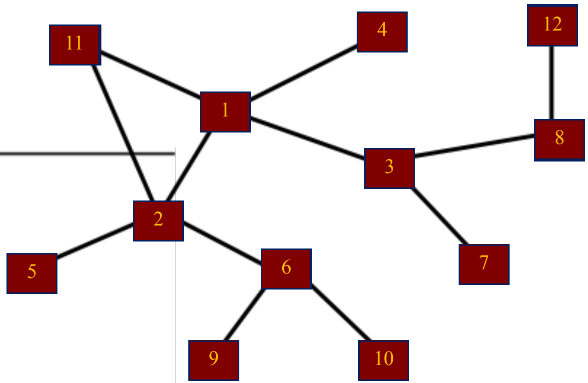


Breadth First Search (BFS)



BFS Algorithm

```
function BFS(Graph G, Vertex v)
    create a Queue, Q
    enqueue v into Q
    mark v as visited
    while Q is not empty do
        dequeue a vertex denoted as w
        for each unvisited vertex u adjacent to w do
            mark u as visited
            enqueue u into Q
        end for
    end while
end function
```



Breadth First Search (BFS)

- If a vertex has several unmarked neighbours, it would be equally correct to visit them in any order.
- If the shortest path from s to any vertex v is defined as the path with the minimum number of edges, then BFS finds the shortest paths from s to all vertices reachable from s.
- The tree built by BFS is called the **breadth first spanning tree** (when graph G is connected).

Applications of BFS

- Finding all connected components in a graph

- Finding all vertices within one connected component

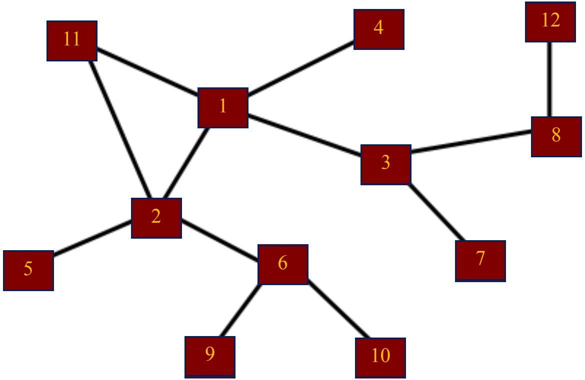
- Finding the shortest path between two vertices

Time Complexity of BFS

- Each edge is processed once in the while loop for a total cost of $\Theta(|E|)$
- Each vertex is queued and dequeued once for a total cost of $\Theta(|V|)$
- The worst-case time complexity for BFS is
  - $\Theta(|V| + |E|)$ if graph is represented by adjacency lists
  - $\Theta(|V|2)$ if graph is represented by an adjacency matrix
    - each vertex takes $\Theta(|V|)$ to scan for its neighbours

Depth First Search (DFS)

- Work similar to preorder traversal of the trees
- DFS systematically explores along a path from vertex v as deeply into the graph as possible before backing up.



```
struct _listnode
{
   int item;
   struct _listnode *next;
} ListNode;
```

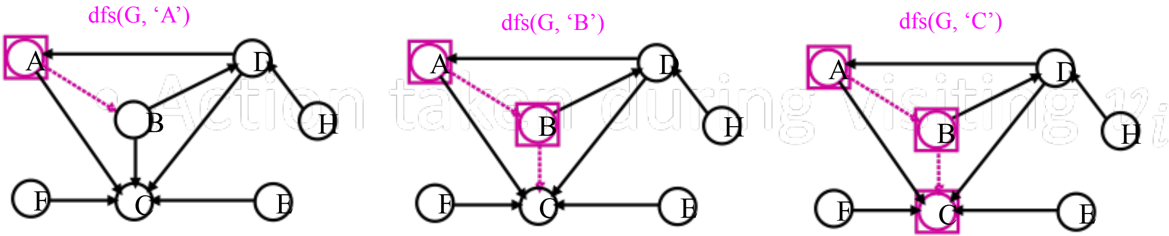
```
typedef struct _linkedlist{
    ListNode *head;
    int size;
} LinkedList;
```

```
typedef ListNode StackNode;

typedef LinkedList Stack;
```

Depth First Search (DFS)

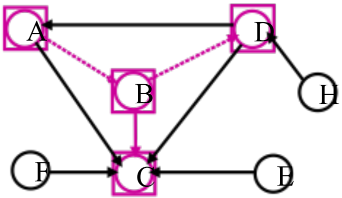
-
  - A stack is used to monitor which



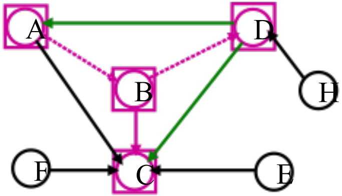
- Action taken during visiting v_i, d

Depth First Search (DFS)

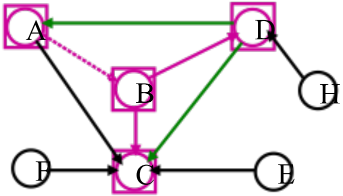
Back to dfs(G, ‘B’) then dfs(G, ‘D’)



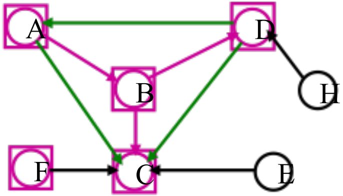
dfs(G, ‘D’)



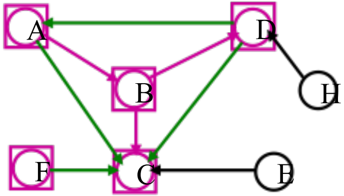
Back to dfs(G, ‘B’)



Back to dfs(G, ‘A’) then dfs(G, ‘F’)



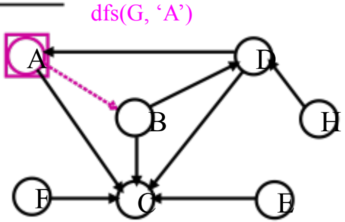
Back to dfs(G, ‘A’)



This directed graph is **not** strongly connected

DFS Algorithm

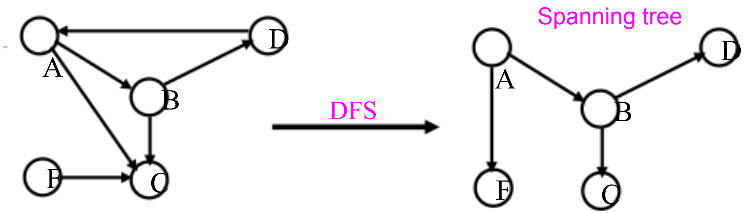
```
function DFS(Graph G, Vertex v)
    create a Stack, S
    push v into S
    mark v as visited
    while S is not empty do
        peek the stack and denote the vertex as w
        if no unvisited vertices are adjacent to w then
            pop a vertex from S
        else
            push an unvisited vertex u adjacent to w
            mark u as visited
        end if
    end while
end function
```



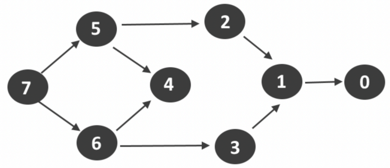
Depth First Search (DFS)

- If a vertex has several neighbours it would be equally correct to go through them in any order.

- If the graph is strongly connected, the tree T, constructed by the DFS algorithm is a spanning tree, i.e., a set of |V|-1 edges that connect all vertices of the graph. T is called the **depth first search tree**.



Applications of DFS



Topological Sort : 7 6 5 4 3 2 1 0

- Topological Sorting
- Finding connected components
- Finding articulation points (cut vertices) of the graph
- Finding strongly connected components
- Solving puzzles

Time Complexity of DFS

- The DFS algorithm visits each node exactly once; every edge is traversed once in forward direction (exploring) and once in backward direction (backtracking).

- Using adjacency-lists, time complexity of DFS is $\Theta(|V| + |E|)$.

Summary

- Two elementary algorithms for graph traversal
  - Breadth-first search (BFS): Use queue – Finding shortest paths
  - Depth-first search (DFS): Use stack – solving some puzzles
- Time complexity of BFS or DFS:
  - Using adjacency lists: Θ (|V| + |E|)
  - Using adjacency matrix: Θ (|V|2)

1	2 3 4 6
2	1 3 4
3	1 2 4 5 6
4	2 1 3 6 7
5	3 7 7
6	4 1 3
7	5 4 5