CMPSC 250

Analysis of algorithms

Dr. Aravind Mohan

Allegheny College
Graph Traversals

- Traversing a graph means systematically following the edges of the graph so as to visit the vertices. Graph Traversal is also widely known as Graph Search.

- There are graph traversal algorithms such as Depth First Search and Breadth First Search algorithms that can be used to traverse through the graph from any given source vertex.
A depth-first search (DFS) explores a path all the way to a leaf before backtracking and exploring another path. Implemented using a stack. At any point of time, DFS keep track of all the visited nodes, final result, and the elements in the stack. An example will help understand:
DFS Graph Traversals - Example
Start with vertex 1, apply DFS technique
DFS Graph Traversals - Example

- Stack = [1]; Adjancency List [2, 5, 9]; Visited = [1]
- Stack = [1,2]; Adjancency List [1,4,8]; Visited = [1,2]
- Stack = [1,2,4]; Adjancency List [2,9]; Visited = [1,2,4]
- Stack = [1,2,4,9]; Adjancency List [1,4]; Visited = [1,2,4,9]
- Stack = [1,2,4]; Adjancency List [2,9]; Visited = [1,2,4,9]
- Stack = [1,2,8]; Adjancency List [2,3,5]; Visited = [1,2,4,9,8]
DFS Graph Traversals - Example

- Stack = [1,2,8,5]; Adjancency List [1,8]; Visited = [1,2,4,9,8,5]
- Stack = [1,2,8]; Adjancency List [2,3,5]; Visited = [1,2,4,9,8,5]
- Stack = [1,2,8,3]; Adjancency List [8,10]; Visited = [1,2,4,9,8,5,3]
- Stack = [1,2,8,3,10]; Adjancency List [3]; Visited = [1,2,4,9,8,5,3,10]
- Stack = [1,2,8,3]; Adjancency List [8,10]; Visited = [1,2,4,9,8,5,3,10]
- Stack = [1,2,8]; Adjancency List [2,3,5]; Visited = [1,2,4,9,8,5,3,10]
DFS Graph Traversals - Example

- Stack = [1,2,8]; Adjancency List [2,3,5]; Visited = [1,2,4,9,8,5,3,10]
- Stack = [1,2]; Adjancency List [1,4,8]; Visited = [1,2,4,9,8,5,3,10]
- Stack = [1]; Adjancency List [2,5,9]; Visited = [1,2,4,9,8,5,3,10]

Done, stack empty
DFS Graph Traversals - Algorithm

Procedure DFS(input: graph G)
begin
    Stack S;
    Integer s,x;
    while (G has an unvisited node) do
        s := an unvisited node;
        visit(v);
        push(v,S);
        While (S is not empty) do
            x := top(S);
            if (x has an unvisited neighbor y) then
                visit(y);
                push(y,S);
            else
                pop(S);
            endif
        endwhile
    endwhile
end
Chapter 04: Section 4.1 530 - 538
Questions
Graphs

Class activity: Post your solution in slack to get attendance points

1. Given the graph provided in Slide 04, generate the trace like shown in Page 533 shown in the textbook. Note the traces generated, picks up an arbitrary element from the adjacency list every time. Regularize your algorithm to pick the element in ascending order from the adjacency list.

2. The trace file is generated using the similar technique that we have discussed, the trace is just a different way of the solution representation.