Course: Introduction to Graph Algorithms

Professor: C Pandu Rangan

Department: Computer Science and Engineering

Institute: IISc

Week: 07

Lecture 29 DFS

Namaskara, We will continue our discussions on depth first search. Depth first search is a systematic exploration of a graph. Graph is a complex discrete structure and systematic exploration requires careful maneuvering of moving around the graph. We say visiting of the vertices and edges for our exploration. Initially no vertex is visited, no edge is visited, it is unexplored okay. So you start from some vertex okay, so here is the vanilla version where you can see you start from some vertex u, if it has a neighbor that is not visited you go there and continue the visit.

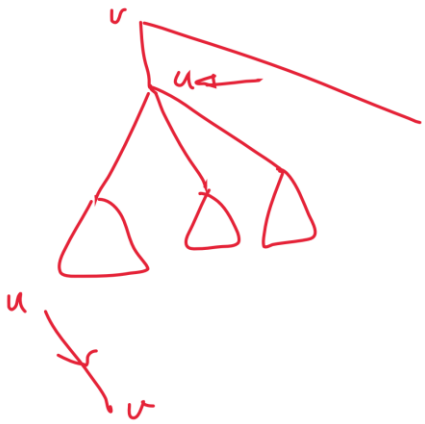
So you recursively you are proceeding, so you start from a vertex u and go to a neighbor v and from there you go to another neighbor v dash and in this way the exploration keeps building a path. okay,

when you are not able to proceed any further, that is suppose you are in a vertex such that it has no unvisited neighbors, all its neighbors are already visited, in that case you cannot proceed any further from here okay. Since you cannot proceed any further, you backtrack, that is all handled by a recursion. So in your exploration process if you go from one visited vertex to another unvisited vertex it defines tree edge.

If you go from a visited vertex to another visited vertex by an edge that is going to define a non-tree edge. So an edge takes you from one vertex to another vertex. From visited to unvisited it is tree edge, from visited to visited it is a non-tree edge. Now, the nodes of the graph could be in one of the three states during a depth first search, it might not been explored so far, that is not yet visited. All unvisited nodes will have the color white, in

our program we have an attribute called the color and if the color of a node is white then that means that node is not yet been visited.

Once a node is visited, you take one of the unvisited neighbor for example if it is u, you take one of the unvisited neighbor and proceed further and come back and take another unvisited neighbor and explore come back and so on. So whenever you are coming back to u and then take another option u is still alive we are still not finished with u it has still some unvisited neighbor. So as long as you have an unvisited neighbor you will be active or alive and you will use that unvisited neighbor to continue with your exploration, that period is called active period and during that active period we will have the color attribute set to grey. When you set something black, it is a done with if all neighbors of u are visited then the exploration at u is completed. The exploration should continue elsewhere. So what you do is that you backtrack to a vertex from which it has come and then continue your exploration from there. So at this point this is finished, you have a black color associated with you okay. So there are three colors we use to indicate the status of a node. Now a tree edge will have the following form you would have come from one vertex to another vertex, this is v, this is let us say from u you had come to v, u is called the parent of v.



The notation we use is, u equal to p of v, we can use another notation v dot p, v dot p stands for parent of v okay, this is another notation. Corman is using this kind of annotation which is typically used when you treat the items as objects. We have already discussed that we will have two timestamps discovery time and finish time when exactly a vertex v. is considered or explored for the first time and when all the explorations are over when it is a done with. So there are two timestamps v.d and v.f associated with.

So here is the full version in pseudocode, depth first search of a graph for each vertex u, u dot color is white and u dot p is nil, the parent is not there for a vertex and initially all vertices are unexplored, so you have u dot color is white. We have to keep track of the time at which a particular vertex is entered or visited and when a particular vertex is done with or finished. In order to keep track of that information we set a variable time and

keep updating the clock that ticks every time an event of interest occurs, so initially time is 0 and then for each u if u dot color equal to white you trigger a depth first search from that vertex okay. So what happens when you trigger a depth first search from a vertex? Time is time plus 1, u dot d is time the discovery time stamp is done here,, u dot color is grey, It has become active so far it was white, now it has become active u dot color is grey and for each v that is adjacent to u, if v dot color equal to white you go there to start exploring from there.

So from u you are going to v. you are here this is a neighbor you had gone from there and then continuing from v therefore it is like this that means parent of v is u alright that is what is set here. Parent of v equal to u is set here because of this situation and you are continuing recursively when the for loop terminates what does it mean all adjacent neighbors have been explored. So from u all explorations are done, okay therefore time equal to time plus 1, finish time and then u dot color is black, you can see that once the for loop is completed we are indicating the fact that it is done with.

```
DFS (undirected graph).

DFS (G).
    For each u ∈ V
        u.color = White.
        u.p     = NIL;
    Time = 0;
    For each u ∈ V
        if (u.color == White)
            DFS (G,u).

DFS (G,u)
    Time = Time +1;
    u.d = Time;
    u.color = Gray;
    For each v ∈ Adj (u)
        if (v.color == White)
            v.p = u
            DFS (G,v)
    Time = Time +1;
    u.f = Time;
    u.color = Black;
```

So we time stamp it and record that in u dot f finish time and u dot color is made black. So this is the basic depth of search we call this as depth of search skeleton. The reason is we can add flush at various parts and make a complete solution for a problem okay.

So we are going to add more computational steps at various control points. It is a skeleton we will understand the skeleton and figure out what kind of computation can be inserted at which point alright. So here is the extended view of computation with the

depth first search, it is the same thing except that I have added where we can possibly enhance the DFS by adding more computational steps and what kind of computational steps can be added there, that also we are looking at. Time equal to time plus 1, u dot d is time, discovery time is set, this is the first time u is visited and this number 1 here indicated, that is a control point 1, at control point 1 you do whatever you have to do, suppose your graph algorithm a computational step requires that to do certain things when you enter a node for the first time, all those computations are to be placed here. So at control point 1 add all the steps that are processing, when u is encountered for the first time it is like the pre-order computation.

Now you set u dot color equal to gray because u has become active, the depth first search has arrived at u until the depth first search is staying with its neighbors and it will be gray. Now look at for each v which is an adjacent of u, this is a control point 2 include the computations to be done for every edge. If you want to do certain computation for all edges those computation and the logic related is to be fitted here this is control point 2. The current edge is uv. v is a neighbor of u, therefore the edge is uv and whatever you want to do for the edge uv is to be put here.

If v dot color equal to white, then uv is a tree edge because from one visited to unvisited you have gone. So the current edge is a tree edge this is control point number 3. At control point number 3 we include the computations to be done related to tree edges. We record the fact that it is a tree edge by setting v dot p equal to u and then we continue with the DFS G of v recursively we are computing at v and once that GFS v is done We have to include the computation to be done while backing from v to u you have done with v DFS v is done with and you are backtracking to u right. So there is a vertex u and from vertex u you have come to v, at v you have finished the exploration and you are coming back.

So that you find another neighbor and so on. So when you come back, this is where the control comes back DFS v is over so include the computation to be done while backing from v to u. So control point 4 is where you are going to add, it is not a tree edge v is visited, if v is unvisited you are going to do all of them. If v is visited, then if v equal to u dot p then uv is the second copy of the tree edge, in an undirected graph every edge will figure in two lists, okay one version will be, if you take an edge uv, v is a neighbor of u, u is a neighbor of v it is an undirected graph.

So, this edge will figure in two places, it will be in the adjacency list of u and it will be in the adjacency list of v. So, in one copy you have already made a tree edge, for the other copy this will be the parent. p of v, v is tree edge then v p of v is the second copy. p of v v is a tree edge vp of v is the second copy. So when you see that it is referring to the parent it is the second copy of the tree edge, otherwise it is a back edge, it is taking to an ancestor right.

So, whatever is to be done for back edge, you have to do here else uv is a back edge it takes to an ancestor already visited but it is like an ancestor. So all computation related to back edge is to be attached here control point number 5 okay, now all visit is complete time stamp, so u dot f is time and whatever you have done and you are leaving that no all computation related to, whatever you need to do while leaving u is to be done at this point. So you have a depth first search skeleton and in that skeleton there are various control points where you can add computational steps that would correspond to the way in which the control is positioned there okay.

If the control, at that control point you are exploring a back edge, back edge related computation is to be done there and so on. So there are 6 control points and we have also seen the properties of those control points. We need to add steps in those control points and then arrive at the complete solution to the problem. So the depth first search would do only exploration but in addition to the exploration we can build useful information about various nodes and vertices. We can add certain computational steps so that during the depth first search itself we solve certain problems.

We are going to see a canonical example of this and that will be in for finding the cut vertices.