DATA STRUCTURE
Part-17

GRAPH TRAVERSAL:-

In this we are going to talk about graph traversals in general and we are going to look at two very important algorithms. The minimize spanning tree algorithm and the shortest path algorithm so let at look at first what is the meaning of the graph traversals. We have already seen about tree traversals, the same definition goes for the graph traversals so from one vertex, we have to list out all vertices that can be reached in a graph G. so it is basically set of nodes to expand and each node has a flag indicating whether that particular node has been visited or not. And as usual there are two types of transversals, Depth First Traversals and Breadth First Traversals. We are looking at the first in detail we are now going to look at something different called as spanning tree. So a connected sub graph that connects include all vertices of the original connected graph is called the spanning tree and this sub graph will be a tree because we will not have a edge which has connect that is not be an edge connecting node to an end another node if that node has been already visited so sub graph is actually a tree it original graph has n vertices the spanning tree will have a n vertices and only n-1 edges. That is the basic differences in a normal graph you can have any more than n-1 edges in fact in the completely connected graph you can have many more number of edges so basically this sub graph does not have a circle and that's why results in a tree so basically in the spanning tree we are trying to find out the minimum number of edges that will still keep it connected so if you have n vertices you can see here n vertices, the spanning tree will have N-1 edges so here you have 1,2,3,4,5 nodes and here you have 1 2 3 4 5 6 edges but here you look at this edge doesn't have to be there for it to be connected right o that's the basic. Now we go to the spanning tree, a spanning tree is tree which connects all the nodes in the graph but it has no circles that we have already seen. Now we are going to look at the minimum spanning tree. The minimum spanning tree is the spanning tree with minimum weight so when you have two edges connecting node that you will tend to connect that edge which is has the minimum cost if we look at this you have 1 2 3 4 5 nodes and as by the definition spanning tree will have four edges. And the question is which of the 4 edges to connect such that I mean to select certain that still remains the spanning tree so if you look here between these two nodes have only one lines you connect then between two have one edge connect so now rest of the two nodes you have from here you can connect we look at this you cannot have this connection you have then from here you have so this is the minimum weight spanning tree for this instead of connecting like this we could

also selected this 20 that will not be minimum and instead of coming here from this could have come from here which would have be the minimum so basically we define the minimum spanning tree. Minimum spanning tree suppose you have a network with redundant connections, what we want to do is find the minimum wiring that connects the entire network and this is the application of the minimum spanning tree typical for the network it could be a power network could be computer network to have a idea is in connection select the connect which have the minimum distance or minimum wiring required so this is the typical for a network of hubs that you have the typical example you have A B C D E F G H and you have the distances could be in kilometer and could be whatever it is. This is what the interpretation is so if you look here this is a graph and this is the spanning tree and you find the minimum amount of cable which is required is 241 if you have done this you would have much more this is the typical application for the minimum spanning tree so for the finding out the minimum spanning tree we called a famous algorithm called the prims algorithm so algorithm goes as follows all nodes are initially unselected, unselected means un selected so you start with the node v and mark it as selected. Now for the each node of graph, find the edge with minimum weight that connects the unselected nodes with the selected node so you start with the node and see what all the connection node and you select one which has the minimum also that that edge which has the minimum weight should be connected to an unselected node or a unvisited node so this unselected node is now marked as selected and you keep going till you have no more unselected node in the, in the representation. So prim's algorithm is a greedy algorithm and now what's the meaning for the greedy algorithm from the local condition which you select the best that's we got the greedy algorithm so we look at the long term algorithm in the short term decision that you make so prim's algorithm is a greedy algorithm for finding a minimum spanning tree i.e., a collection of edges that connects all the nodes, you don't look for this the minimum to select the minimum at the particular local condition that is for this node which is the minimum that's what we connect that's why we called as greedy algorithm so again let's look at the algorithm starts with the list which is the representation so start with a list of all nodes and an empty tree. As long as the node list is non-empty repeatedly that is all the nodes in the graph has not been selected you have to keep doing this algorithm so you select one node from the list with the shortest edge connecting it to an node in a tree add that node and edge to the tree, and remove the node from the list. So you start with the node in the list then you put it in the tree then you start selecting the one with the as we already told you minimum distances of the minimum value and which is connecting to the unselected node. The node that is in the list and the node along with the edge is and then put it into the tree so you finally have

the minimum spanning tree and so again what we normally use a priority queue i.e. the priority queue is used to list the number of nodes in order of minimum so the first node will have the lowest of the minimum distances so a min heap to store the edges and select the shortest edges. So this is how we will have the step 1 marking the node A as selected now from you have 2, 10 and 1. 3 edges connecting to B,C,D according to the prim's algorithm you will select the one with the least which is one and in addition it should not be it should not be selected and has not been selected so the selection you do is 1D so the edges you select this is now this will to be marked as selected and so that's how you do that so that's the one selection you do and so now you have marked D also as selected and this is the edge w and now what you have to do from D and A you see which is the one which has got the least minimum so from A will have 2, 10. 2 you have you should consider this because D has already its 2 an node already been selected and from D you have 6 so you have 2 and 10, 6 so you select this 2. That's the one you select so you have now this now you have A B D. now from A you have 10. From B you have 5. And D you have 6. The minimum is 5 and it is connecting you have the D edges also that will be 2 already selected nodes so now you have to select this edge so that's the minimum to 2 this is your tree and you have 4 nodes so you have 3 edges that is n-1. So all the vertices are marked as selected so we find the minimum spanning and so this is the minimum spanning tree. Please note these edges are not selected. And so for this prim's algorithm the minimum spanning tree algorithm by prim. You have a graph G, you have a weight function and you have a source and for the each vertex in the graph G set the key of u to∞. Set parent of u to nil. And then set the key of source vertex to zero. Enqueue all vertices to Q. while Q is not empty this is equivalent to that lists. Extract vertex u from Q , u is the vertex with the lowest key that is in Q. for each adjacent vertex of u do, if (v is still in Q) and (weight function(u,v) < key of v) then set u to be parent of v, update v's key to equal weight function (u,v). this is just find the weight and to build the spanning tree and that's the first algorithm so in the first algorithm what we have done is we have selected what is called a minimum spanning tree in the minimum spanning tree what we have done is we have define what the spanning tree. The spanning tree is the tree which contains the all the nodes in the graph and it is connected such that all the nodes are taken care of but it does not have a cycle. That's the spanning tree, the minimum spanning tree is 1 in which those nodes are selected where selecting the node you take the condition that from every node you will select 1 that has the minimum that's how we do the selection of the next edge so the prim's algorithm are the uses was the algorithm that was found that has been written to find out the minimum spanning tree and that was using the greedy algorithm. The greedy algorithm is that given the state of the all the node you are trying to find out an

edge which is got the minimum value and which connects to an unconnected node those two conditions are satisfied with that the condition are with have to be satisfied or that it should have the minimum value the edge should have the minimum value and the edge should connects from the node already in the spanning tree with an node which is not in the spanning tree that's how you select and then you build the tree. And you continue to do this till no more nodes are selected that's the prim algorithm we also said that the minimum spanning tree has the number of applications one application that we saw in this lecture was with the actual node. First the network has that you have we have number of connection between the network of. You would have to select the one which gives the minimum cable for that purpose you use the minimum spanning tree. The same is true that you want to travel ling with the typical case of the problem you want, of course we assuming the travelling that not comes to the source that's the different in the issue. So here we are looking at sending people to different nodes you have to select the minimum spanning tree for the doing. The next application is the electrical application where we have power connection that is how to power station with electrical cable again there will have use the minimum spanning tree as the typical application of the minimum spanning tree of the power required, I mean the power cable requirement.

SHORTEST PATH ALGORITHM:-

Now when you talk about shortest path algorithm now still remember in the minimum spanning tree what we did was we had a graph converted into the tree basically in the sense we assume that the minimum spanning tree did not have the cycle. When you start with the source no question now coming back to the source but often you have a other problem that you need to apply the graphs to that is the shortest path algorithm. the shortest path algorithm is given a source node and it is destination node we want to find the shortest path between the source and the destination the source that we have used and the destination is given as v. so when we talk about the values here between the nodes they could be the past, they could be the distance, they could be the travel time, they could be the hop in the case of mobile networks could be a hop. The shortest path algorithm is typical way in which mobile nodes use which of these station to use and how many hops they are and all those issue are typical of rooting algorithm uses the shortest path now let us go to an example of the shortest path we will see before that please note that it is up to you give whatever number you wasn't here and interpret those numbers so we will see about the shortest path algorithm. now let us go to the shortest path algorithm and the shortest path problem is defined as the shortest you have find the shortest path between the source and the destination here as u and v. so now let us look at the examples suppose you were

given a network and wanted to find the optimal path that a message should travel between two specific computers. Then use the shortest path algorithm. They are variety of algorithm to find the shortest path from start to end. And there is backtracking –style of solution. Start with the list priority queue and then select the shortest path in the list and if node i that is you start with the node and then you keep on going till the next step you find the neighbor has the shortest path from there you find the neighbor which has the shortest path and you end the algorithm and you have reached the node that you was that is the destination node that is if you not reached node you expand that path and each neighbor and then add the resulting path to the list. This solution is inefficient due to the redundancy because the λ nodes that you already a number of times and you have to eliminate that and so that the optimum algorithm. So you may end up otherwise you may end up storing the same sub path multiple times and you have to potentially generate and test every person will path whose length is optimal .

DIJKSTRA ALGORITHM:-

What you go is the famous dijkstra algorithm. The Dijkstra algorithm does the greedy way again just like in the previous algorithm in the prim's algorithm we used the greedy algorithm approach. Here also we approach the greedy in addition to using the greedy approach we also use the dynamic programming approach to achieve the, to achieve what we want and this has an order of n2 where n is the number of nodes. Now what is greedy I have already explained what is greedy is now the dynamic programming approach. A dynamic program approach if we have the path which we have seen already we store it and we do not try to calculate the path again that is basically what we seen you store it you store it when you want it just look it up that is basically the concept of dynamic programming. So in other word this maintains a table of shortest distance so far from start to each node so if you have started with the start node and you have got 5 parts all the 5 are stored it, stored there in less distances so that when you want to go back as it expands, it updates the distance so that no sub-optimal sub path is never used and never work. It updated in such a way is never used. We have many types of shortest path of algorithm. We are looking at single source shortest path algorithm that is you start with the single source and try to find the shortest path to all other nodes so that's called the single source shortest path algorithm and sometimes you may have to find every node find the shortest path to every other node then that's called the all source, all node shortest path algorithm and the dijkstra's source shortest path algorithm. We find the shortest path greedily by updating distance to all other nodes so that is the dijkstra's source shortest path algorithm. Now let us look at the dijkstra algorithm. We assume that all weight of edge greater than zero. We cannot do the algorithm if you assume. So you just give a

minimum edge distance. So this is the source now you want to find the shortest path all nodes in this. That are one, so V1, V2, V3 V4 and the best. So now if you look at this V1 has the path to 1 to 10 has got path to 2, 5. If it does not have path we put the value to infinity so if it does not have path to 3 and does not have direct path to 4 both have put as infinity. Now out of this from the source the best that you have is 2 which have 5. So V2 is best alternative here. So that's what written here. Now let us go the step 2. Step 2 what we are going to do is recalculate the path to all other nodes. So what we have done here is we have recalculated the path. We find the shortest path to node 0. Node 4 is selected. Now let's go to step 3, we have to recalculate the path to all other nodes what you do here is V0 to V1 so there are two paths now. One path and other is direct path and this is another path 8 is the lowest path still remains the lowest. V2 to V2 0 to 2 is 2 only 5. You don't have any path that you can choose. If you have one more path that is 12 is the larger so you still remain at 5. Now this is interesting, if you look here you had gone from V0 to V2 to V3 that gave you 14. But now if you look at you see another path 5 0 2 4 and 3 you have 5+2 = 7 + 6 is 13. So you choose that 13 and here. Now minimum path now here is V1 so you don't choose V2. So you choose V1. V1 is over. Now you have to find the shortest path to node 0. Node 1 is selected. Now go to the next we have to find the path. Now next look at the final column. V1 to V1 still remain the same 8. 0 to 1, 0 to 2 is 5. 0 to 3 again undergoes to change if you look at the 3, you can go the 0 to 5 to 2 to 4 and 4 to 3 that was the path you got. Instead of that now you can go to 0 to 2, 2 to 1, and 1 to 3. If you look 5+ 3= 8 +1 = 9. So it is changed again. 7 and you choose 3. So what happened now is path is look like this. 0 to 2 is this path you choose V4. 0 2 4 so you choose V4. 7 is the lowest. Than you choose the V1. For going to V1 choose the path 0 2 and 5. So you choose this path 5 8 path. Ok. Now V3 we choose 0 2 1 is 3 this is the path that's how you know the path. You do the dijkstra algorithm. How we get all shortest paths to each node. So you got shortest path. 2 is was 5 to 1 it is 8. To 3 it is 0 2 4 3 and that's it. So those are the one 4 it is 5 and 0 2 4. So those are the path you got. So this is another representation what we did there. From node V0 to other nodes. This is to tell you that (0,2) to you went then you went to V1, (0,2) you went V1. (0,2) you directly went. (0,2,4) you went. Here you went (0,2,4,3). Here you went (0,2,1,3) that's the path. We have already seen the path in the last example if you remember. This was the path. For example, I take the three examples 0 first shortest path was 0, 2 and then 3. Next was (0,2,4,3) and next you found is (0,2,1,3) is the shortest that is what we told here. And then we have (0,2,4) here. So now these are the path. From V0 to V1 it is (0.2). V0 to V2 is (0,2). V0 to V3 is (0,2,1,3) and V0 to V4 is (0,2,4). So these are the paths. Now look at the algorithm seen in example now let us look at the algorithm you mark the source node as selected remember that we are assuming

that there is 1 node is not necessary then you have to repeat these types of algorithm so mark the source node to selected. If the source node there is no direct connection to any node then you initialize that to be infinity and initialize all distances to infinite, source node distance to 0. Mark source node the current node now there is still an unselected node how you select the node only when you find the shortest path. Expand on current node. Update distance for neighbors of current node. Find an unselected node with smallest distance, and make it current node and mark this node selected. So that's what we are going to do. Please note that it is not a direct path it can be rooted through other nodes that's why the algorithm is little bit different what you have done before so this is what called as dijkstra called pseudo code it a graph, it's a weight, its the source for each vertex v in V[G]. You just put the distance to infinity and previously v to undefined and s is the undefined and Q is V[G]. So this is the algorithm what we are doing is updating the path every time so you get the lowest. Despite the speed of modern computers, some of these problems still intractable. Intractability you have the famous traveling salesperson problem (TSP). Given a map of cities and roads, what is the shortest path that visits each city exactly once and then returns to the starting city? This is called the travelling salesperson problem and considered to be NP-hard problem. But there is no known polynomial time algorithm for finding an optimal solution. Significantly research has gone finding efficient into actually finding efficient solution to NP hard problems like TSP but there has been not be a solution. Since there are (n-1)! Possible tours, generate-and-test will take a long time on any non-trivial graph. For example here are some example given

5! = 120

10! = 3,628,800

20! = 2,432,902,008,176,640,000

And dynamic programming can avoid some redundancy, but still O(n!) as the type.