Course: Introduction to Graph Algorithms

Professor: C Pandu Rangan

Department: Computer Science and Engineering

Institute: IISc

Week: 06

Lecture 25 Kruskal's Algorithm

Namaskara we are continuing our discussions on designing algorithms for finding minimum cost spanning tree. We have already seen an interesting algorithm by Prim and now we will begin our discussions on another algorithm by Kruskal, okay. This Kruskal's algorithm is a greedy algorithm designed somewhat directly and obviously based on the definition of a tree and the cost of the tree and so on, you want to build a tree whose cost is as low as possible. So the natural thinking is build the tree by including as many cheap edge as possible costlier edges should be ignored and avoided cheaper edges we should be able to accommodate and then build. So the tree is going to be built edge by edge we will consider cheaper edges first and costlier edges later, how do we achieve it first let us build a list L of all edges, sorted in the increasing order of weight and we will consider the edges of L one after another starting from the cheapest going to the costliest, in this way cheaper edges are considered earlier costlier edges are considered later. So as and when we consider an edge we will decide whether to include in the tree or throw it out, if it is not feasible to include we have to reject it okay.

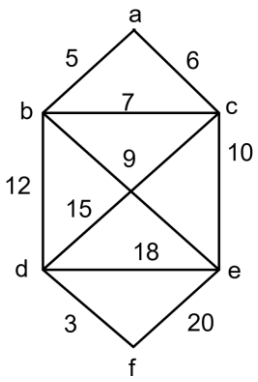
So the whole process can be expressed in a simple iterative algorithm and our rejection rule is also fairly direct okay. If an edge e forms a cycle with some of the edges that is already included. We cannot include the edge e the reason is it is forming a cycle with the edges that you have already selected and T is an acyclic structure. The set of edges in T will never form a cycle therefore I should not include e, if I include e a cycle is formed which is not possible which should be avoided therefore I reject e.

Hence I consider the edges in the increasing order of their weight if it is possible for me to include I will include it and if I have to reject I will throw that out and in this way I process the set of all edges and in the end I would have collected certain set of edges okay. We need to prove that those set of edges form a tree. Secondly, we have to prove that they are the edges of a minimum spanning tree, even if they form a tree if it forms a tree which is costlier, then we do not want such set of edges we want a set of edges that form the minimum cost spanning tree. So those things are to be proved. First let us take a look at a simple example to understand the way in which the algorithm works.

Here is the outline or pseudocode of the algorithm. G is a connected weighted undirected graph. The weight of an edge is denoted by w e, L is the list of edges in the increasing order of their weights. So this is a pre-processing step even before the algorithm starts whatever is the kind of input given to you first you have to build L consider all edges and sort them in the increasing order of their weights. and our algorithm is going to work on L, so L is the list of edges in the increasing order of their weight.

Now the output is a minimum spanning tree the T vertex set is same the edge set is A, A is the wedge set and this edge set is going to be selected by the algorithm. Initially A is empty, e is the first edge of L while not end of L you are going to consider one edge after another. Look at the step 2 while not end of L if e does not form a cycle with any subset of edges in A. right A already has some edges and e is not forming a cycle with any subset of edges, then A equal to A union e I am including e in A in this way A keeps growing edge by edge okay. And now e is next e L, so e is just a parameter that is used to run across L and e is the current and whether the current edge is forming a cycle or not I check if it is not forming a cycle I include.

If it is forming a cycle you see there is no action taken which means I am ignoring that edge and I am going to the next edge e equal to next e L that says go ahead and examine the next edge in this way one edge after another will be examined and if the test passes the edge will be included, if the test fails the edge is ignored and then we move to the next edge and so on. This process we will see later, collects n minus 1 edges and those n minus 1 edges will form the minimum spanning tree that we will see later but first let us see the way in which this algorithm works. This example also is very helpful in describing the data structures that are used in the implementation stage okay, so let us take a closer look at an example, so here is a graph,

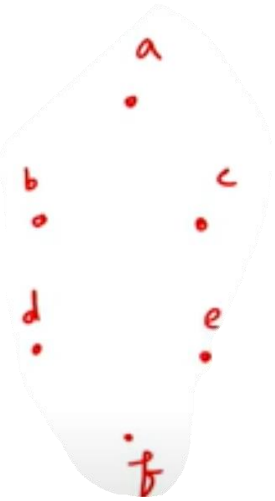


And this is a weighted graph this graph has got 6 vertices the edge weights are indicated there 5, 6, 7, 9, 15, 18 and various weights are indicated, okay. The sorted list of edges which is the cheapest edge the cheapest edge is df. Let us look at the example here is a graph it is a weighted graph undirected connected, each edge has got a weight and here is a sorted list of edge L it is given here, df the reason why df is given first is its weight is 3,

AB its weight is 5, you can see that I have arranged the edges in the increasing order of their weights ac is 6, bc is 7, and be is 9 and then ce is 10, bd is 12 and cd is 15, cd is 15 and de, this is e, de is 18, ef is 20 okay.

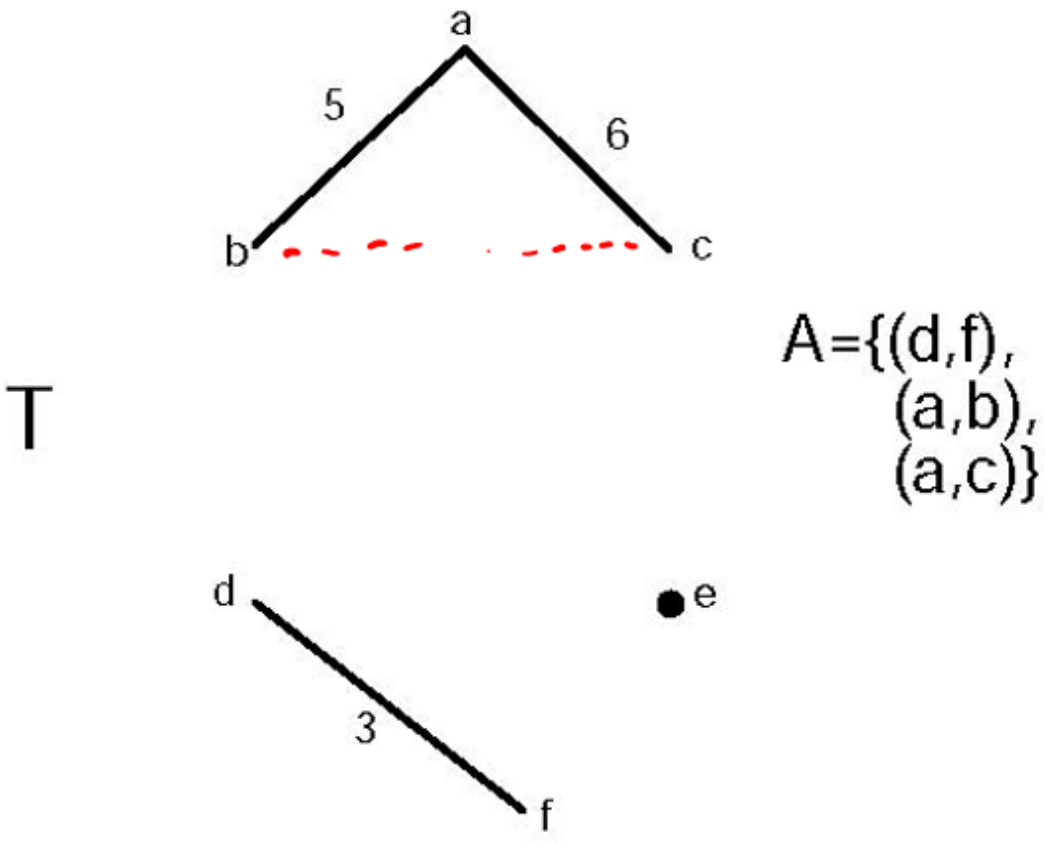
$$\{(d,f),\ (a,b),(a,c),(b,c),(b,e),\ (c,e),(b,d),(c,d),(d,e),(e,f)\}$$

So you can see that the edges are arranged in the increasing order of their weights. Now I am going to show the picture, I am going to execute the algorithm and show the development of this spanning tree. edge edges are added okay. So initially A is empty there are no edges so a, b, c, d, e and f these vertices are considered vertex set is same T has the same vertex set right now T has no edges, because A is empty, A is empty, the T with no edges is given by the picture with only the vertices marked no edges are there.



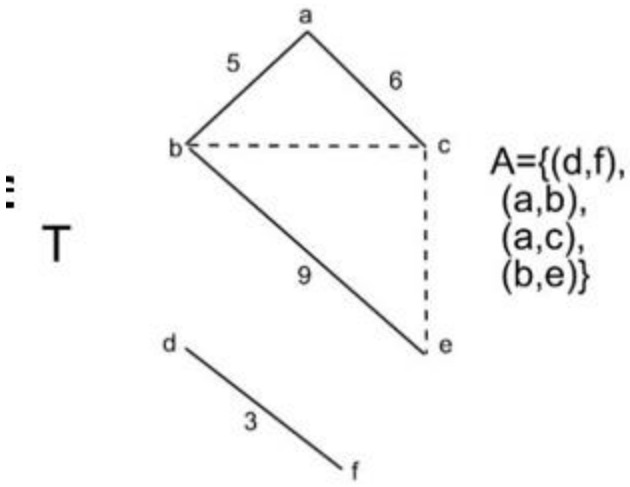
So consider the first edge, df so I draw that df, here df is not forming a cycle with the edges in A, in fact A has no edge at all right now A has only df alright. The next one is ab look where ab is ab is here ab is not forming a cycle with df now A has two edges. A now has two edges the third edge is ac let us add ac if you add ac still no cycle is formed you can see that now A has three edges the new edge that is added is not creating any cycle okay it is remaining a cyclic therefore we have executed three steps of the algorithm and we have collected the 3 edges. Now consider the fourth edge bc what about bc when you add bc what happens look at the picture here bc when you add I have filled that as a dotted line because if you add bc a cycle is formed a, c, b is a cycle Therefore I should not include bc if I include bc the edges already in A there are 3 edges that are already in A.

Now I am adding one more edge unfortunately this is forming a cycle. Therefore though this edge is cheap I cannot include this, that creates a cycle I am looking for a tree, tree is an acyclic structure therefore I should not include this edge or ignore that and go to the next edge. The next edge is be, so let me show that in the picture here I have shown the development



So you can see that 3 edges are included this what I have drawn already and then shown to you and then I have ignored bc, I have ignored bc because it forms a cycle okay. Now let us proceed with our example. whatever we have done so far is shown in the picture here.

Three edges are in A, there is one ignored edge, all of them are shown in this picture okay. The dotted lines are showing the ignored edges, the next edge is be, but when I add be, it is not a problem you can see that I can add be without creating any cycle, be does not create a cycle remember I am saying that it is not forming a cycle by showing you the picture right but in the algorithm the computer has to determine it has to logically decide it does not have a human interface like me and you and show well it is forming a cycle and it is not forming a cycle and so on. We have to algorithmically determine it that is a challenge we will see that later but right now to show which edge is included and which edge is ignored I have shown the dotted lines and when I have drawn the dotted line you can see even the cycle that is formed.

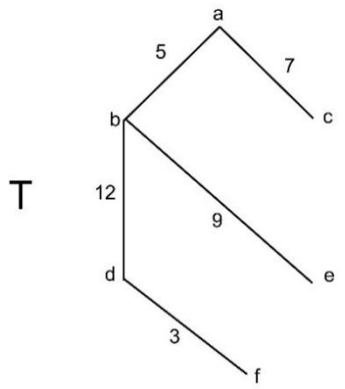


you could see the reason why I have ignored that particular edge. So I include be and in this way we can continue and then see the edges that are included and the edges that are ignored okay, what I would do is I will go back to the picture where the list is available and I have drawn the then I will come back to this.

So be is considered, be is considered, ce will be ignored because ce is forming a cycle right when you include ce, ebace it forms a cycle. so that edge you cannot include bd, bd can be very well added because it does not form a cycle so let me move d little bit to the left and right here d, bd i can include, what about cd, now cd should also be ignored because cd is closing cdbac that cycle is formed therefore that should also be ignored de, de should also be ignored because de is forming a cycle dbe, dbe is forming a cycle, so this should also be ignored. and ef, ef should also be ignored because it is forming a cycle. So each edge is

considered and if it is possible for us to include we will include it, if it forms a cycle it should not be included and it is rejected, in this way the entire list is processed. of processing L you have certain edges considered, certain edges ignored.

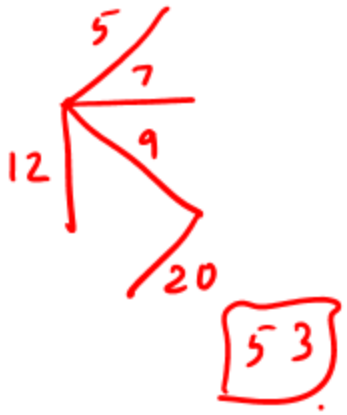
The key question is, the set of edges that you have considered does it form a tree, the set of edges do they construct a minimum spanning tree, these are the two questions that we have to answer. So, let us look at the final output the final output is drawn here in this picture.



This picture shows only the considered edges all the ignored edges are removed from the picture this is a tree okay. Let me continue with the example. This is a tree and this has got only the edges that are considered or included in A and all the ignored edges are not shown in the picture. This is a tree. It is an acyclic structure.

What about the weight of the tree? What is the weight of the tree? It is sum of the weights of the edges in the tree. So you have 3. plus 5, 3 plus 5 here I have shown the addition 3 plus 5 plus 7 plus 9 plus 12, these 5 edges are included and their weights are added and you get 36. If you consider any other tree it will be more expensive. okay I can show that just I will demonstrate it right in the picture that I have considered.

Let us consider another spanning tree okay. This is also a tree connected acyclic structure. These edges which are considered are 5, 7, 12, 9 and 20. This also a tree but what about its cost? Now 12 plus 12, 24, 33 and the cost is 53. This is a spanning tree but its cost is 53 we have got already a spanning tree with cost 36 okay.

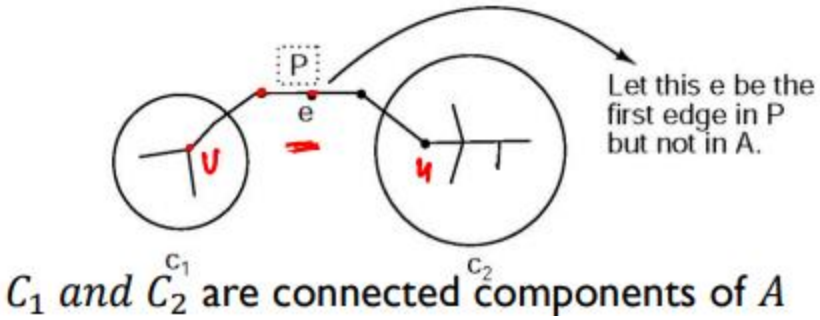


We have to prove that it is indeed the cheapest there is no spanning tree with a cost smaller than that can be constructed that way our algorithms correctness is established, okay. This is an example to show how other spanning trees are costlier than what we have constructed.

So let us turn our attention towards the proof okay we first prove that the set of edges in A form a tree first of all it is an acyclic structure, that is clear because whenever an edge is about to form a cycle we are ignoring it therefore all the edges that are included they do not form a cycle at all, therefore A is an acyclic structure. But what is the definition of a tree? A tree is a connected acyclic structure, it is a connected acyclic graph. TA, T equal to vertex set A, this is acyclic is it connected once we prove that it is connected it is a connected acyclic graph therefore it is a tree, we would have proved that. So the proof is by contradiction, suppose T is not connected then it will have several disconnected components.

TA right, T with edge set A will have several components, each one is acyclic therefore it may be like this there will be one component like this there will be another component like this and one more component. The edges of A will form cluster of acyclic structures, so basically A will be a forest if it is not forming a tree, it will be a forest like this a collection of tree. What we are going to prove is, it is connected we have assumed that it is disconnected and we are going to arrive at a contradiction. We are going to arrive at a contradiction. Now the original graph is connected.

These are the component C1 and C2 are connected components generated by A but since the original graph is connected take any vertex here v and take another vertex u, v and u are connected by a path in the original graph.



That path has got several edges all edges will not be in A, if all edges are in A. that means the entire path is in A that means C1 and C2 are not disconnected component, right. Therefore not all edges of the path are in A, there is one edge at least that is not in A, which is available in the path P, that is what I have shown in the picture let e be the first edge that is not in A. okay it is the vertices of e they are not at all in e.

So for this edge one vertex is here and another vertex is outside but for e both vertices are outside the connected component let e be the first edge right it is not forming a cycle with any of the structure it is completely outside. right because it is completely outside it is not going to form a cycle with any of the edges in A. So at the time when e was considered you would have got a subset of A I am saying that it is not forming a cycle with the full set A but consider the execution of the algorithm when e was considered only a part of A would have been considered e is not forming a cycle even with full A. So it is not going to form a cycle with the part of A that was developed up to that point when e was taken for consideration. So when e was considered by the algorithm you have some subset of A, but e will not form a cycle with them. So algorithm would have included e that means e would be in A but then here we are taking a e outside A this is a contradiction right. So if A has different component then there will be an edge A which the algorithm would have included but not shown in A. This is a contradiction therefore A cannot contain several components, that means A is a single component. which means A is connected, that is A is connected acyclic graph, therefore A is in fact forming a tree, T equal to VA is indeed a tree.

$$T = (V, A)$$

okay so therefore A will have n minus 1 edges ,where n is number of vertices this is a big V, n is the number of vertices and this concludes the first part of the proof. We are still to prove that it is minimum spanning tree we have shown that it is a spanning tree. It is a tree it is working on all vertices, therefore it is a spanning tree, our algorithm has collected certain set of edges they form a spanning tree. But do they form a minimum spanning tree that is the question we are interested in, if it is proved that it is forming a minimum spanning tree then we have an algorithm in our hand and then we can go about implementing it and then look into the challenges and complexities of executing that algorithm okay. So that is what we are going to look at in our next session.