Course: Introduction to Graph Algorithms

Professor: C Pandu Rangan

Department: Computer Science and Engineering

Institute: IISc

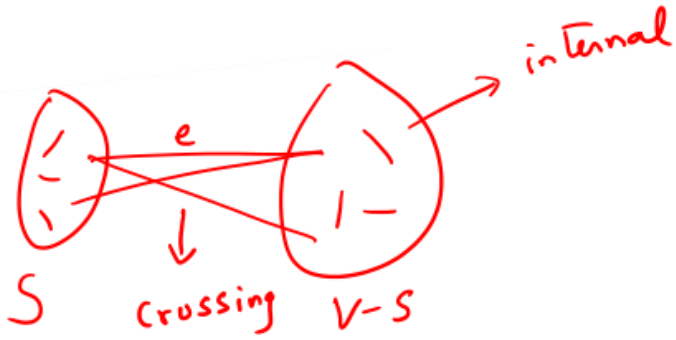
Week: 06

Lecture 24 Prims Algorithm P2

Namaskara, we continue our discussions on algorithms for finding minimum spanning tree, we have seen certain basic ideas and notations and we are in the process of designing and development of Prim's algorithm. So let us recall one of the important theorems that we have stated, let G be a graph, this is undirected connected.

$$G = (V, E)$$

We are going to partition the vertex set we are going to view the graph in the following manner S is a set of vertices V minus S is the remaining set of vertices. Now the edges of G are distributed either here or here these are called internal edges and there are some edges which could go this way these are called crossing edges okay, internal and these are crossing edges okay.



Now my spanning tree T is as the same vertex set V, I have not identified the full spanning tree I have identified only a small subset of edges, so A is a subset of edges of some spanning tree okay, so A is a set of edges, basically.

$$T = (V, A)$$

The cut respects A if no crossing edges in A in other words edges of A are all distributed either in S or in V minus S okay. So all edges of A are internal no crossing edges if that is the case then we say that the cut respects A. the cut respects A if this condition is satisfied. Suppose I have a set of edges of some spanning tree and I also have a cut respecting A then

there are several crossing edges find the edge e which has got the minimum weight crossing edge e is minimum weight crossing edge, e is a minimum weight crossing edge, there can be several minimum weight crossing edges you choose any one of them then I can add e to A and still I will have in my hand a subset of edges of some spanning tree. So A union e is subset of some spanning tree okay, this is still a subset.

$$A \cup \{e\} \sqsubseteq \text{some spanning tree}$$

So this is a very interesting result and this allows us to build A incrementally all spanning tree algorithms will attempt to build the edges of the spanning tree because vertex set there are n minus 1 edges to be identified, so we identify one edge after another and our algorithm will have n minus 1 stages where these n minus 1 edges would be added and we would complete the process okay. We need to first of all identify a cut that is respecting A, we will see how that could be done. So, if A is a subset of some spanning tree, e is a minimum weight crossing edge of a cut respecting A, then A union e is a subset of some spanning tree. So we start from empty set,

$$A = \emptyset$$

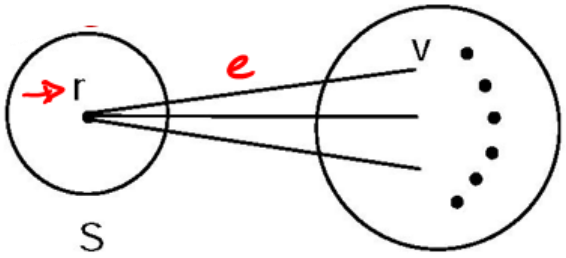
Empty set is clearly a subset of an edge set of some spanning tree, we find a cut respecting the empty set.

find the minimum weight crossing edge add to A. Now A has one edge for the new A again you find a cut respecting the new A find the minimum weight crossing edge add to A. Now A will have two edges this process you repeat n minus 1 times A will have n minus 1 edges and those n minus 1 edges will form the minimum spanning tree. okay so we are going to repeatedly do the following, alright. So which cut would respect A when A is empty, A is empty there is no edge in A any cut will respect A, right, because you take any cut it will have crossing edges none of the crossing edges are in A, A is after all empty. So you can take any cut whenever we have choice like this we have to choose a cut that is easy to compute easy to work with and so on. So we choose a cut we choose some arbitrary vertex it is called r the name r is given because that is going to form the root of the spanning tree. take a vertex r put that in S all other element V minus r this will be in V minus S. okay so S has got only one vertex all other.

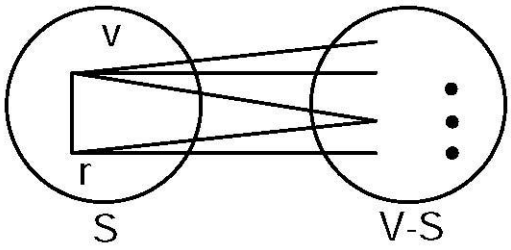
So what are the crossing edges crossing edges are all the adjacent vertices and edges incident on r so from r lot of edges they go to V minus S they are all crossing edges. So how to find the minimum weight crossing edge simple look at the adjacency list of r each edge has a weight find the minimum weight. let us say rv is that edge with minimum weight this is the edge e with minimum weight, therefore I am going to add rv to A right I am going to add rv to A. Now I have a new A, I have to get a cut that is respecting this A, I had this cut which is respecting A

$$A = A \cup \{r, v\} = \emptyset \cup \{r, v\} = \{(r, v)\}$$

How to get a new cut respecting new A, should I simply start with such for a new cut or, can I modify the existing cut that would respect A, the new A. So I have a cut respecting old A, can I modify that cut to get a cut respecting new A, that is the question it is very simple.

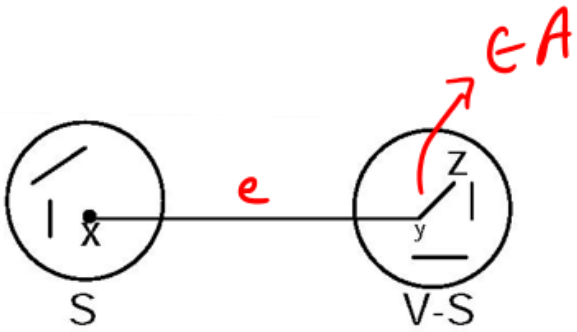


Right now I have included this e which is a crossing edge in A, I want all the edges of A as internal edges. So what is a natural solution move v to S, when you move v to S this what happens you can see that rv has become an internal edge, this cut respects new A what is A, A is rv and that rv is an internal edge now there is no crossing edge in A so this is a cut respecting A.

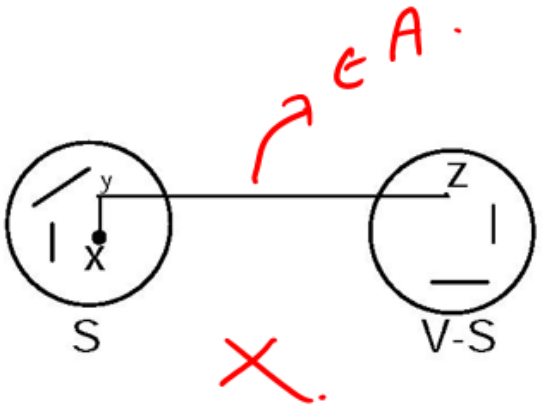


Therefore when you are adding an edge to A, it is a crossing edge when you are adding that edge the other end of the edge can be moved, from V minus S you move to S. then the crossing edge becomes an internal edge as shown in this picture however there is a minor catch here okay. For example the minimum weight crossing edge is e, so xy is the edge However the edges of A are internal to S and V minus S.

Suppose I have this edge belong to A that is an internal edge, that is in A. Now when you move y to S, what happens I have yz but this belongs to A, yz was an internal edge. but when you move y, why we want to move y because xy is the minimum weight crossing edge we had a plan to move the vertex in V minus S to S

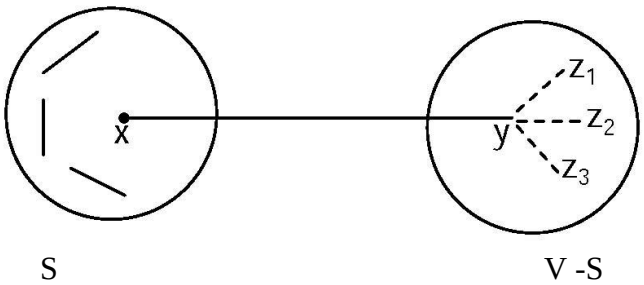


But when we move what happens is an internal edge becomes a crossing edge and this belongs to A, this does not respect A.

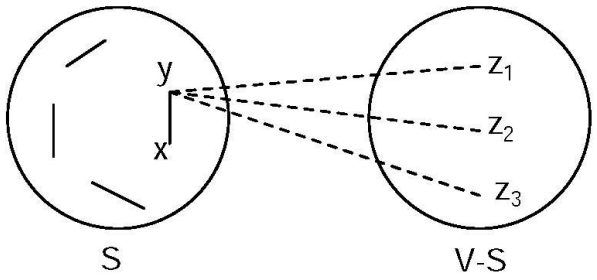


Therefore moving will not work unless you are a bit more careful okay. In general moving should work but if the other end has got an edge in A, then the moving is not going to work it will bring a crossing edge which is in A such a cut is not a respecting cut.

How to fix the situation, very simple. Make sure that all edges of A are in S, if all edges of A are in S, internal to S, then you can simply move absolutely no problem there is nothing like this yz being creating a crossing edge in A that does not happen right. New crossing edges are created but none of them will be in A. Therefore, the idea is make sure that no edge of A is in V minus S, right no edge is in V minus S, no edge of A is in. So now you can move y, no problem at all when you move y, yz1, yz2, yz3 they will all become crossing edges that is okay. They are not in A because all edges of A are here in S this is S this is V minus S that is the fix.



Therefore we will ensure that the edges of A are all in S so when you find a crossing edge simply move the end vertex in V minus S to S, that will make that crossing edge also an internal edge and everything is done peacefully okay. Therefore, there is a new crossing edge created but they are not in A, this picture clearly shows that. Therefore the new cut respects A and has all edges of A internal to S no issues at all okay.



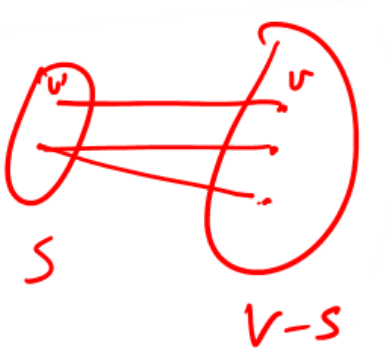
Now you have updated the cut, the cut is going to have more crossing edges already it had some crossing edge. Among that crossing edge you have found out the minimum weight, then you have moved one vertex from V minus S to S like this see earlier y was internal when you move y what happens yz1, yz2, yz3 they all become new crossing edges. So more crossing edges are added, so you have to find the minimum, once the minimum is lost and some more elements are added, if you want to find a minimum you have to examine all of them okay. You do not know the minimum is lost, the edge which is defining the minimum cost is already moved and some new edges have become crossing edges. So, the new cut has got some old crossing edges, some new crossing edges and you have to find the overall minimum. okay and finding that overall minimum is going to take order e time that is because there could be order e crossing edges possible.

Therefore, you can update easily a cut but after updating the cut finding the minimum weight crossing edge has become order e, order m process right. So the naive algorithm would be n minus 1 times, it is a very simple naive algorithm no clever method you move the vertex now you have new crossing edges you already have the old crossing edge that and this put together lot of crossing edges, again go through all of them find a minimum weight crossing edge. maximum there can be m edges, so this cost can be maximum order m, n minus 1 times you are doing this therefore the complexity is n minus 1 times m this is order nm.

$$(n-1).m$$

$$O(nm)$$

This is not a clever solution, it is a naive solution, so here is a clever move done by Prim okay. You want to find a minimum weight crossing edge. So let us maintain 2 pieces of information with respect to every v in V minus S okay so here you have S, here you have V minus S



for every v, find the minimum weight crossing edge coming out of v again for this one the minimum weight crossing edge, again for this one the minimum weight crossing edge, maintain only that information then the overall minimum weight crossing edge will be minimum among these edges because from each edge from each vertex I am maintaining the minimum weight crossing edge, only that information I maintain. So how do I maintain that it is very simple p of v right, if it is v it is v dash p of v equal to v dash.

$$p(v) = v'$$

I maintain an array called p array right in that p array I maintain the other end of the edge and weight of v, p of v this I maintain in c of v.

$$c(v) = w(v, p(v))$$

Weight of that edge, minimum weight crossing edge crossing edge weight that is why c of v, the notation c of v says from v which edge is having the minimum weight crossing edge and what is its weight, the weight is maintained in c of v, p of v is the other end of the minimum weight crossing edge, natural information. Therefore I have to find out the vertex with minimum c value, vertex with minimum c v value that is what we are going to do right.

So find the vertex with minimum c value. That is, what is minimum it is a vertex v such that c of e is less than or equal to c of u for all u belonging to V minus S.

$$c(v) \leq c(u) \forall u \in V - S$$

This V is defining the minimum weight crossing edge from each vertex, you have a minimum weight crossing edge from that vertex. Overall minimum is having c of e minimum and v and its partner v and p of e that edge is the minimum weight crossing edge, that minimum weight is c of v. Now I have found the time taken for this is cardinality of V minus S

$$|V - S|$$

Because for each v belonging to V minus S, I have this information only I compare c of v and finding the minimum value.

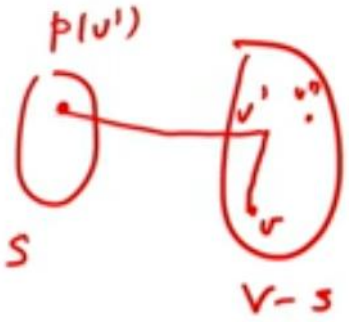
$$v \in V - S$$

Therefore in number of steps proportional to V minus S, V minus S is less than or equal to n minus 1 right.

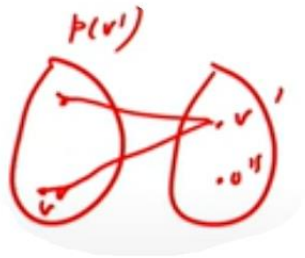
$$v \in V - S \leq n - 1$$

Therefore, minimum weight crossing edge is found out in time less than or equal to n minus 1, very clever idea okay. In fact V minus S is a shrinking set first the cardinality will be n minus 1, then V minus S becomes smaller because you are moving vertex out from V minus S. So V minus S will become smaller and smaller and smaller. Therefore the cost of finding the minimum weight crossing edge becomes smaller and smaller, as you advance it becomes easier to find minimum weight crossing edge okay.

Therefore the idea is we have to maintain for each vertex in v, 2 pieces of information p of v the other end of that and which edge is the minimum weight crossing edge. Now when S changes the minimum weight crossing edge information may change. okay so let us take a simple picture to understand the situation. I have S, here I have V minus S V is over here v dash is its neighbor there is another vertex v double dash v double dash is not a neighbor v dash is a neighbor v double dash is not a neighbor okay. v dash has got the minimum weight neighbor this is pv dash pv dash is the minimum weight neighbor for v dash.



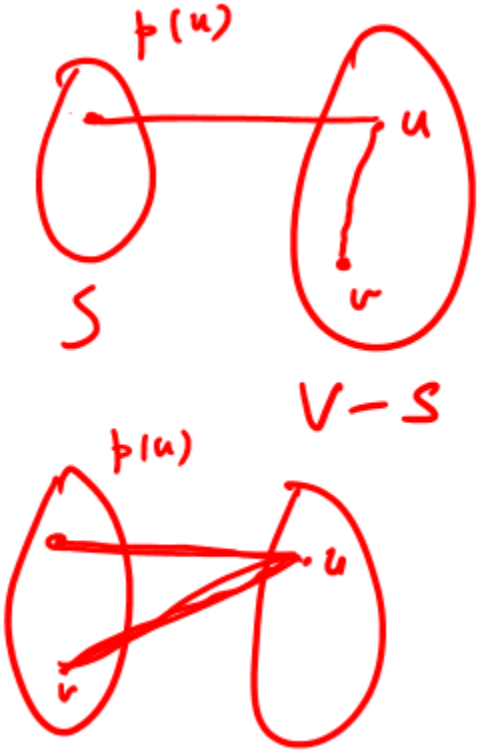
Now v is moved. In the next step v is moved so how will the picture look like I have v dash v is also here v dash has got pv dash this is a new neighbor v so this is a new crossing edge that has come into picture with respect to v dash and with respect to v double dash nothing has happened okay, with respect to v double dash you have moved only v. so v double dash and the information related to that can remain as it is okay but is this a new neighbor? Is this a new minimum weight crossing edge? I already have a minimum weight crossing edge from v, that is v pv dash.



Now a new crossing edge has been introduced from v, is this the potential minimum weight? So how do I check? I just check this, is the weight of vu is less than weight of u pu right, if this is the case, pu is updated to v. right, this becomes the new partner that is because this crossing edge is now cheaper, alright.

$$w(v,u) < w(u,p(u))$$

So let me draw another picture, let me erase this, this is I will ask the following question, I have S, I have V minus S, I have v, I have u, I have p of u, pu is an internal edge here.

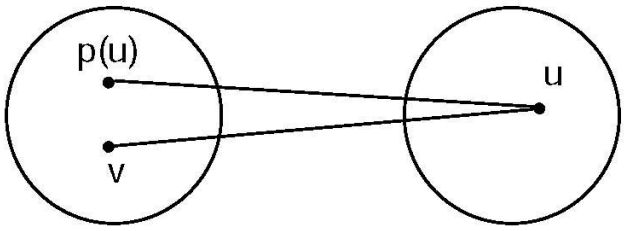


Now you have moved v, when you moved v, i have this is v, this is p of u. if weight of vu, if this weight is smaller than weight of u p of u if this weight is smaller than this weight. then now then you can easily see that this is the minimum weight crossing edge from u. Therefore, I update p of u to v and I will also update cost of u to weight of uv

$$w(v,u) < w\big(u,p(u)\big), p(u) = v, c(u) = w(u.v)$$

Because this is now so you check there is a new crossing edge that has come is this crossing edge minimum weight crossing edge leaving u, therefore you do this check and take these two action so for each u, this is what we have to do okay this completes the description of the algorithm.

So you start with A as an empty set S is an arbitrary vertex called root V minus S is V minus r to begin with cost of v is infinity parent of v is null because you do not have any vertex associated and no edge so cost will be infinity. If for each u in V minus S if ru is in the edge p of u is r, c of u is w ru because that is the only crossing edge. And that is the minimum weight crossing edge and the minimum weight crossing edge is determined like this so this is the way, you would initialize. After initializing we repeatedly do the following we find the cost we find a vertex with minimum cost value move v to S and vpv to A because this that the minimum weight crossing edge I want to add to A and I have to update cu and pu. With respect to the new cut and this is the code in detail same steps all we have to do is that look at the logic in this step that would clarify the point for each u in V minus yes if uv belongs to E and weight of uv is less than right Cu, less than Cu currently u has some weight associated with it and wuv is smaller than this therefore p of u equal to v C of u equal to wuv. okay so we have to repeatedly do this while V minus S is nonempty. Once the while loop is completed you can return p of v for all V this is an implicit representation of the tree the edges are vpv, vpv are the set of edges, vpv is the set of edges



High level Pseudocode:

$A = \emptyset, S = \{r\}, V - S = V - \{r\}$

For all $v \in V$,

$p(v) =$ NULL

$c(v) = \infty$

For each $u \in V - S$

If $(r, u) \in E$

$p(u) = r;$

$c(u) = w(r, u)$

While $S \neq V$

Find a vertex v in $V - S$ with minimum $c(v)$ value

Move v to S and add $(v, p(v))$ to A

Update $c(u)$ and $p(u)$ values for all $u \in V - S$

For all $u \in V - S$

If $(u, r) \in E$

$p(u) = r$

$c(u) = w(u, r)$

while ($V - S$ is non-empty)

1) Find $v \in V - S \ni$

$C(V) \leq C(u) \forall u \in V - S$

2) Move $v\ to\ S$

3) For each $u \in V - S$

if $((u, v) \in E$ and $w(u, v) < C(v))$

$p(u) = v$

$C(u) = w(u, v)$

Return $\big(p(v)\big)$

\\ $T = (V, A)$ is MST

\\$A = \big\{(v, p(v)),\ v \in V - \{r\}\big\}$

Therefore this is the minimum spanning tree very nice algorithm very simple algorithm that gives the set of all edges of some spanning tree. We are building incrementally and at every stage there is a natural way in which we could construct the or identify the minimum weight crossing edge and complete the process of building the spanning tree okay. This completes the description of the algorithm, we will discuss about its complexity and implementation in our subsequent discussions, thank you.