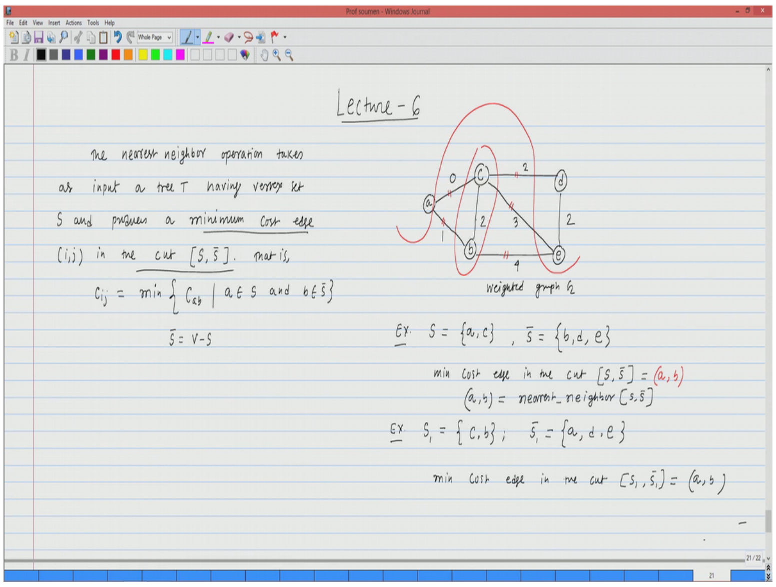
Graph Theory
Prof. Soumen Maity
Department of Mathematics
Indian Institute of Science Education and Research, Pune

Lecture - 06
Part 1
Minimum Spanning Trees (cont.)

Hello. This is my 6th lecture on Graph Theory. Today we will learn Prim's algorithm to find Minimum Spanning Tree in a weighted graph and also we learn up bending path algorithm to find maximum matching in by bipartite graph.

So, first we talk about Prim's algorithm to find minimum spanning tree in a weighted and connected graph. So, Prim's algorithm it starts with the vertex which is the tree at the beginning and then at every step it finds its nearest neighbor of the existing tree and includes that nearest neighbor in the existing tree. First we talk about what is this nearest neighbor operation.

(Refer Slide Time: 01:38)


Prof soumen - Windows Journal
File Edit View Insert Actions Tools Help
Whole Page
a
b
c
d
e
0
1
2
2
3
2
4
21/22
21

Lecture - 6

The nearest neighbor operation takes as input a tree T having vertex set S and produces a minimum cost edge (i,j) in the cut $[S, \bar{S}]$. That is,

$C_{ij} = \min \{ C_{ab} \mid a \in S \text{ and } b \in \bar{S} \}$

$\bar{S} = V - S$

weighted graph G

Ex. $S = \{a, c\}$, $\bar{S} = \{b, d, e\}$

min cost edge in the cut $[S, \bar{S}] = (a, b)$

$(a, b) =$ nearest_neighbor $[S, \bar{S}]$

Ex. $S_1 = \{c, b\}$; $\bar{S}_1 = \{a, d, e\}$

min cost edge in the cut $[S_1, \bar{S}_1] = (a, b)$

So, here is the operation; the nearest neighbor operations takes as input a tree T having vertex set S and produces minimum cost edge C i, j in the cut S, S bar. So, we do not know what is this minimum cost edge and what is this cut also I will explain in detail let me just finish this part that is C i, j is the cost associated with the edge ij C i, j is the minimum of Cab the cost of the edge a b where a belongs to S and b belongs to S complement. So, note that S complement is V minus S.

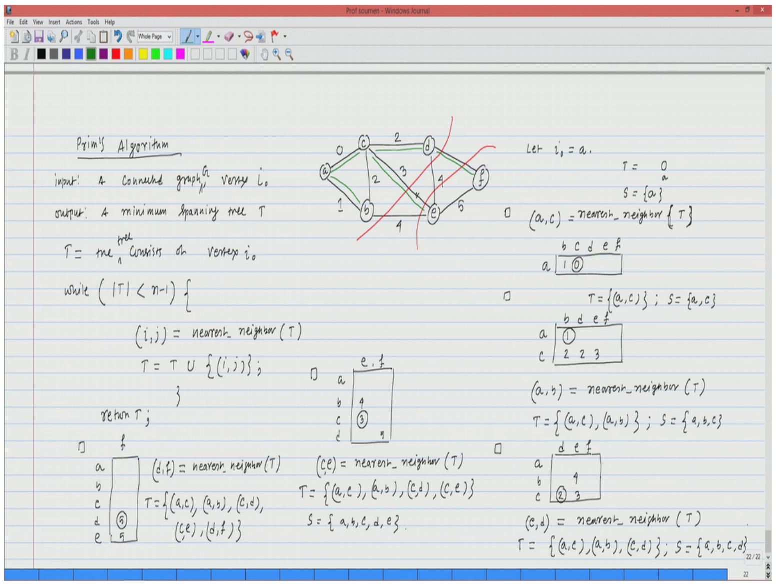
Let me give one example to illustrate this definition; I take this graph. So, these are the vertices of this graph and a and c are adjacent, a and b are adjacent, b and e are adjacent, c d adjacent, d e adjacent, c b adjacent, and c e are also adjacent. And this is a weighted graph weighted and connected graph, the weight of this edge ac is 0 this is weight one 2 3 2 4 2.

Now I will explain what is this minimum cost edge in the cut xy let me sorry x bar let me take S is equal to say a and c and, S complement is equal to b, d and e. Now what you do is that you what is the minimum cost edge in the cut. The minimum cost edge in the cut S S complement is obtained in this way. So, this may be this is S consist of a and c, this is one part and S complement consists of b d and e, this is another part. So, you are dividing the set of vertices into 2 parts basically S an S complement. Now the cut s s complement means cut s s complement consists of this edges the edge a b, b c, c e and c d. So, you see the weight of each of these edges and take the minimum. So, the minimum cost edge connecting in this cut s s complement is basically a b.

Similarly if you take say another S S equal to say for example, c b c b c b this is first example the second example then my S complement is equal to a, d and e right. So, I can now c b is one part let me see. Now, c b is one part and a, d and e this is another part this is S complement. So, the edges the cut the cut consists of this edges the cut consists of this edge this edge this edge and this edge show here the minimum cost edge in the cut let me call it say s 1 s 1 complement in the cut s 1 s 1 complement is now a b basically, this is this is again a b well.

Now we know what is what do you mean by the minimum cost edge in the cut s s complement. Now, we will write in this way the nearest neighbor the nearest for this example a, b is the output of the function or the operation nearest neighbor of s s complement or the other way of seeing writing this s s complement is you just write the set of vertices in the tree T is that that is the set of vertices of tree T is s.

(Refer Slide Time: 11:46)



Prim's Algorithm

input: A connected graph G, vertex i_0

output: A minimum spanning tree T

```
T = the tree consists of vertex $i_0$
while ( $|T| < n-1$ ) {
        $(i,j)$ = nearest_neighbor ( T )
        T = T ∪ { $(i,j)$ };
    }
return T;
```

Let $i_0 = a$.

$T = \emptyset$

$S = \{a\}$

(a,c) = nearest_neighbor { T }

	b	c	d	e	f
a	1	$\textcircled{0}$			

$T = \{(a,c)\}$; $S = \{a,c\}$

	b	d	e	f
a	$\textcircled{1}$			
c	2	2	3	

(a,b) = nearest_neighbor (T)

$T = \{(a,c),(a,b)\}$; $S = \{a,b,c\}$

	d	e	f
a			
b		4	
c	$\textcircled{2}$	3	

(c,d) = nearest_neighbor (T)

$T = \{(a,c),(a,b),(c,d)\}$; $S = \{a,b,c,d\}$

	e	f
a		
b	4	
c	$\textcircled{3}$	
d		4

(c,e) = nearest_neighbor (T)

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

$S = \{a,b,c,d,e\}$

	f
a	
b	
c	
d	$\textcircled{5}$
e	5

(d,f) = nearest_neighbor (T)

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

Next we talk about the Prim's algorithm. So, the input to this algorithm is a connected a connected graph if it is not connected you will get a minimum spanning forest not been a minimum spanning tree and the vertex any arbitrary vertex say i 0 and the output is a minimum spanning tree T.

As I said that this algorithm start with single term tree I mean the tree consists of 1 vertex. So, the T is initially this is the minimum spanning tree initially the T is the tree consists of the tree consists of vertex i 0 and this T is basically the set of 3 edges. So, if the graph G is having n vertices then tree will have n minus 1 edges, because we know that a tree with a tree with n vertices has n minus 1 edges. So, that is why we have this while loop while this cardinality of tree that is T is the set of 3 edges when this is less than n minus 1 you do the following.

What you do is that you find out the nearest neighbor of T the current T tree the nearest neighbor of the current tree. So, see be before I wrote here s s complement. So, here I wrote T the only difference is that I mean once you know T you know s s complement because S is the set of vertices of the tree T and then you update your tree. So, T union ij I will explain this algorithm of course, and when you are outside when you have n minus 1 edges in your tree you are sort of done you got this minimum spanning tree and you return you are outside of this while loop and you return T.

So, now I will explain this algorithm using an example the same example ac b d e and f. These are the vertices again ac adjacent, a b are adjacent, c b are adjacent, c d adjacent, be adjacent, d e adjacent c f sorry e f and d f there adjacent and also c and e are adjacent, now I write down their widths 0 2 5 5 4 3 4 2 1.

Let now I illustrate this algorithm using this example let i 0. So, the input to this algorithm gives a tree sorry gives graph weighted graph and the vertex i 0 let that I naught b equal to a the vertex a.

Now, at this moment your tree consists of only 1 vertex that is the vertex a that is all this is your tree, and what you have to do is that you have to find the nearest neighbor of the S at this moment is equal to a it consists of only 1 vertex that is a and you find the nearest neighbor of s s complement. So, the nearest neighbor nearest neighbor of this T at this movement the T is consist of only 1 vertex a. So, that is equal to, this S and it is nearest neighbor is c. So, the output to this thing to this operation is ac.

Another way of illustrating this one is that you make it small table like this one. So, this part is sorry, this part is a; that means, the set S and this S complement. So, S S complement consist of b c d e and f and you can see that a and b there is an edge and weight is 0 a and c there is an edge and the weight of it is 0 a is not adjacent to def. So, you can keep this blank. So, this is the smallest one. So, the nearest neighbor is ac this is another way of, computing the nearest neighbor you can see from the figure also.

This is at the first iteration and at the second iteration you have now your tree is consists of your tree is consists of 1 edge ac. So, your S is consists of 2 vertices a and c and T consists of 1 edge that is a, c. So, again you find the nearest neighbor of T, one way of finding that is that you know a and c in one part and the remaining vertices are in the other part that can be also represented in this way in the form of a table. So, ac and this is the elements in S and in S complement you have b d e and f you can check that a and b are adjacent and that is weight 1 def are not adjacent to a c is adjacent to b and the weight is 2 similarly here it is 2 and 3. So, the minimum 1 is this one. So, a b is the output of the nearest neighbor operation at this stage nearest neighbor of the current tree.

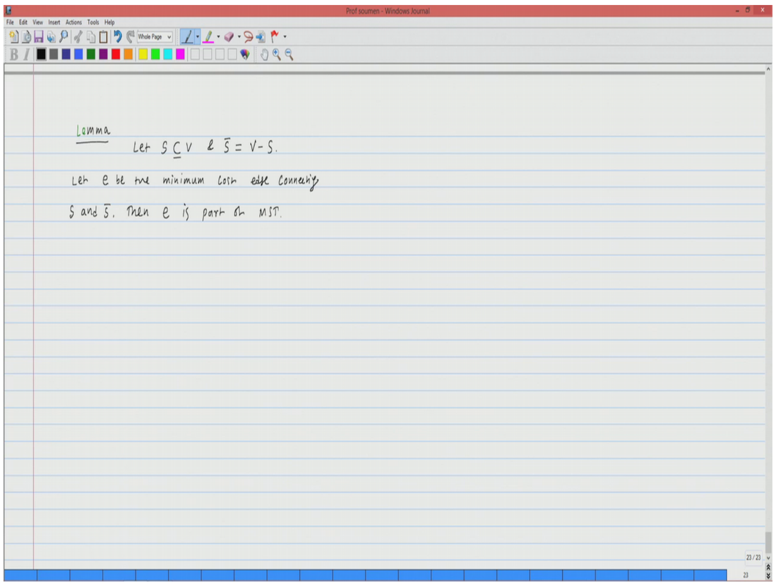
Now your T you update T then you update your T T consists of 2 edges now, one is ac and the other one is a a b right and then S consists of 3 vertices now a b and c a b and c right. So, at the third iteration what you do is that again you construct this table a b c are the vertices in S and in S complement you have the vertices d e and f, a is not adjacent to b e and if, b is adjacent to e, b is adjacent to e and the weight is 4. Similarly c is adjacent to d and the weight is 2 and c adjacent to e and the weight is 3 and the minimum one is this one. So, the output of nearest neighbor here c is c d right c d is the output of nearest neighbor operation with the current tree T.

So, one more iteration, now you are you update your T. So, T is consists of the tree T consists of 3 edges now that one is a,c a,b and c,d and of course, your S is then S is a b c d right abcd. So, at the fourth iteration what you do is that you make the table again a b c d. So, a b c d is in one part is in one part and S complement is this. If you write it in the form of table then what you get is that you have e and f e and f in the other side and you can see that the minimum one is c e definitely this is this is the minimum one here. So, in the form of table it is 4 here, it is 3 here and it is 5 here. So, the minimum one is 3 and c, e is the output of the nearest neighbor of the current tree T and, you update your T your current tree is consists of 4 edges ac a b c d and c e.

Since there are 6 edges we need one more sorry 6 vertices one more edges is required and that is the last iteration. So, the table for this one is that, now S consists of everything like a b c d and e and S complement is f. We construct this table again, this side you have a b c d e and you can check that this quantity is 5 and this is 5 you can pick any one of them. So, the output of the nearest neighbor is d f of which is the nearest neighbor of this tree. So, you update your T finally, and this is the final T is consist of ac a b c d c e and d f.

We can also we can show that a 3 edges here. So, the 3 edges are ac a b c d c e and bf and that is the minimum spanning tree. We have learnt how to find minimum spanning tree using Prim's algorithm it starts with 1 vertex and at the first step it will find the nearest neighbor of that initial vertex and once you get the nearest neighbor, now your tree is consist of 2 vertices and again you expand that tree at every step until all the vertices are included or there are n minus 1 edges.

(Refer Slide Time: 28:44)



Lemma

Let $S \subseteq V$ & $\bar{S} = V - S$.

Let e be the minimum cost edge connecting S and $\bar{S}$. Then e is part of MST.

Now, we talk about 1 algorith not algorithm one result which is sort of explain why this Prim’s algorithm is giving the correct answer let s b a subset of the vertex v and of course, the S complement is V minus S, let e b the minimum cost edge connecting S and S complement then this result says that e is part of MST well.

So, S is any subset of the vertex set V and S complement is um another part and e is the minimum cost edge connecting S and S complement then e must be part of MST. So, this lemma sort of gives a proof that the Prim’s algorithm is correct.

So, we will look at the proof of this lemma in the second part of this lecture.

Thank you very much.