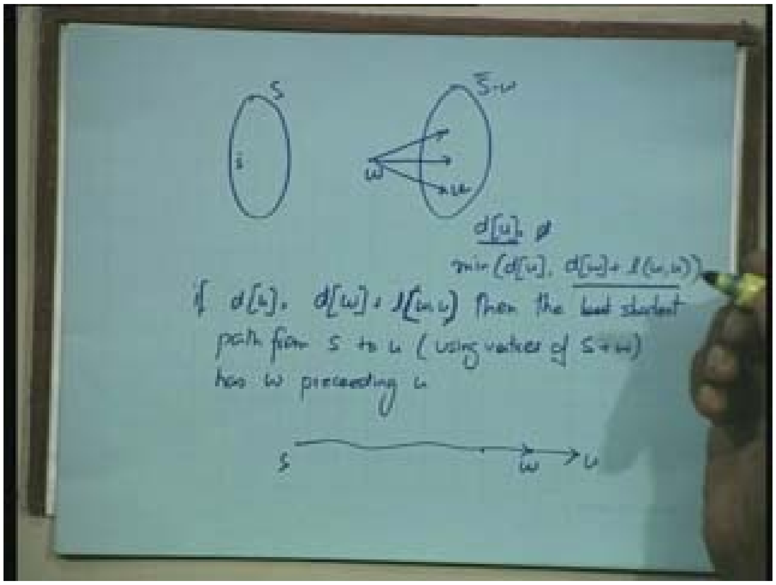
Data Structures and Algorithms
Dr. Naveen Garg
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi
Lecture – 36
Single Source Shortest Paths

We will continue our discussion on single source shortest path. Today we are going to see how to actually compute the path. In the last class we have seen how to compute the distance of any given vertex from the source, when all edges had non negative lengths. Today we are going to see how the path can be found and in second part we are going to see how to compute single source shortest path when edges might have negative lengths.

I am going to start from the picture. You have s, $\bar{s}$ and there was a vertex w. So the $\bar{s}$ should be $\bar{s} - w$. The vertex w is going to move from right to left and when this vertex is moving it causes the distance label of the adjacent vertices to be updated. If this were the vertex u, we said that d[u] is going to be min [d[u], d[w] +l [w, u]]. I have repeated this many times. Essentially what this means is that if d[u] is getting this (d[w] + l [w, u]) value then that means the best path that I am finding from s to u which includes only vertices of s and w. And it is going through w or w is the vertex preceding u on the best path from s to u.

(Refer Slide Time: 04:50)

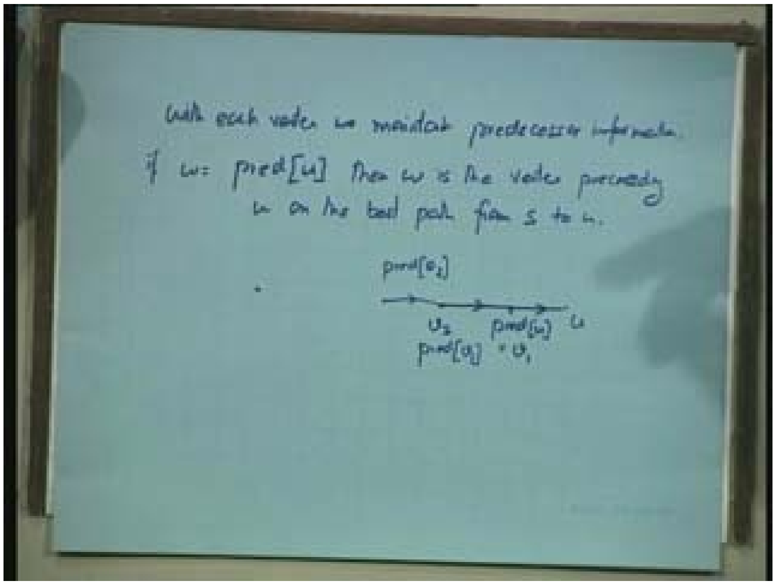


if d[u] = d[w] + l[w,u] then the best shortest path from s to u (using vertices of S+w) has w preceding u

Let me write it down, if d[u] = d[w] + l [w, u] then the best path or the shortest path from s to u (using vertices of s + w) has w preceding u. Basically we are saying that the best path from s to u and then the vertex preceding u on that path is w. This is the best path from s to u, if this is how the updation is happening that is if d[u] was getting this (d[w] + l [w, u]) value. And this is not the best shortest path from s to u because this is under this qualification, using vertices only of s and w. Because w is going to left hand side and using only these vertices the best path from s to u will have w preceding u. This is the information we will maintain with vertex u. What is the vertex preceding it on the best path that I have found so far? The best path means the shortest path. With each vertex we maintain predecessor information. What do I mean by that? If w = pred [u] then w is the vertex preceding u on the best path from s to u. How am I going to use this information to compute the shortest path?

Suppose I have this predecessor information for each vertex, this is vertex u and I know that the vertex which precedes u and the best path from s to u is pred [u] and I have maintained this information. Let say this is a vertex v_1, I am going to look at predecessor of v_1 because this part of the path from s to v_1 has to be the shortest path from s to v_1. If this is the shortest path from s to u then this has to be the shortest path from s to v_1. The predecessor of v_1 would be the vertex preceding it, on the shortest path form s to v_1. So this vertex v_2 would be predecessor of v_1. This part of the path has to be the best path from s to v_2. I have the predecessor information for v_2 that tells me what is the vertex preceding v_2 on the best path from s to v_2. This I have information that is pred [v_2] and so on. I can keep doing this till I reach the source vertex.

(Refer Slide Time: 07:28)



With each vertex we maintain predecessor information.

If w = pred[u] then w is the vertex preceding u on the best path from s to u.

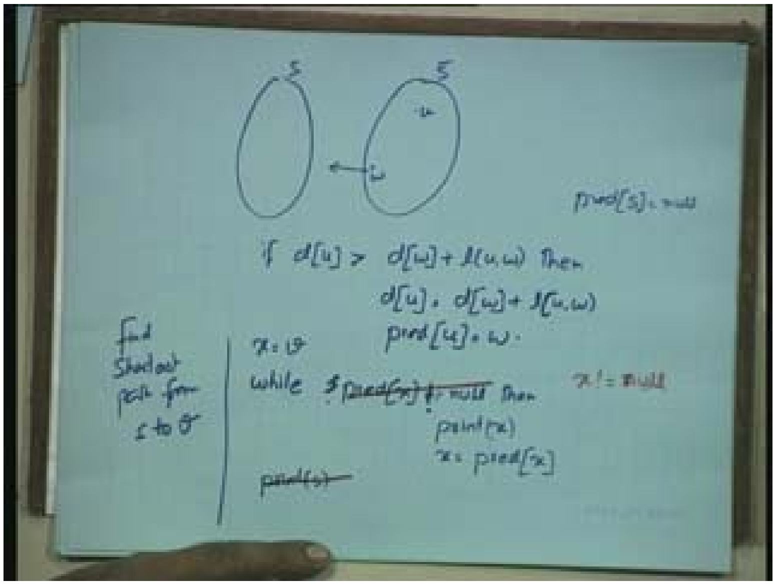
pred[v_2]
v_2
pred[u]
u
pred[v_1] = v_1

Once I have this information I can just keep tracing it back till I reach the source vertex and that will give me the path. How to we maintain this information? When ever we update the distance label, we have to keep track of this. We update the predecessor.

When do we update or how do we update? When we transfer a vertex from the side $\bar{s}$ to the side s, that is w is the vertex which is getting transferred. Then this is the vertex u and I am updating its information. How should I write it now? If d[u] is greater than d [w] + l [u, w] then d [u] is d[w] + l [u, w]. What is this saying? The d[w] + l [u, w] is the length of a better path. If the path you found now is better then what you already had then you update the information and pred [u] = w. That is the only modification we are doing. Then basically we are storing with each vertex, 2 pieces of information d and this predecessor information. And now using this predecessor information you can find the path and keep moving back up.

How? Suppose I wanted to find the shortest path from s to a vertex x. while! pred [x] equals null. I will set the predecessor of my source to null. Because the source vertex has no predecessor. Initially let us say x = v, v is the vertex for which I am finding. Find the shortest path from s to v? Initially x = v, now I will go through this loop, while pred [x]! = null. Print it directly that is print [x]. (Hindi Conversation) Initially v, (Hindi Conversation). So in this manner you can also print out the path, I have printed out the vertices of the path once you have the vertices in the path you can also figure out the edges of the path if you want to print out the edges. One suggestion is instead of having this condition I can also right x! = null. (Hindi Conversation) All of you can figure this out.

(Refer Slide Time: 12:36)



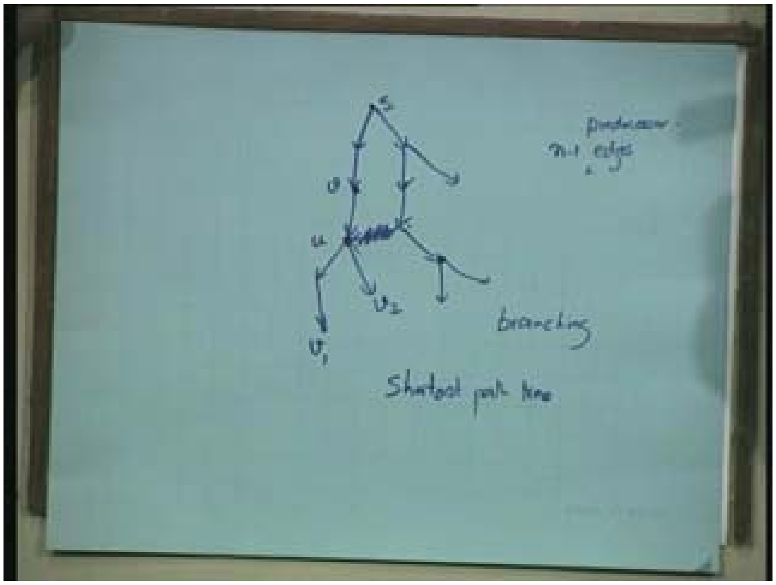
```
if d[u] > d[w] + l(u,w) then
        d[u] = d[w] + l[u,w)
        pred[u] = w.
x = v
while ~~pred[x] ≠ null~~ then        x != null
        print(x)
        x = pred[x]
~~pred(s)~~
```

What we have done using this procedure? We have computed the shortest path from s to every vertex in the graph not just one vertex but every vertex in the graph. Let us draw all the shortest path and let see what is the sub graph that we get. (Hindi Conversation)

Let say this is the shortest path from s to vertex v_1. (Hindi Conversation) Except for root vertex that is the source vertex. (Hindi Conversation) n-1. Do you understand what edges I am taking about? I can also call them as the predecessor edges. (Hindi Conversation) This is the predecessor vertex. If I am looking at the vertex u and v is the predecessor vertex, we have kept track of the predecessor vertex information. Bur we could also have thought of this (u v) path as the predecessor edge. The edge connecting the predecessor vertex to this vertex is the predecessor edge. (Hindi Conversation)

How many predecessor edges we have? The n-1 predecessor edges. (Hindi Conversation) I can reach every vertex because if I take any vertex I can take its predecessor edge go to the previous vertex, go to the predecessor vertex and I said I will eventually reach s. Is it clear that I will reach s in this process. Because the whole point is that this guy got its length from here, which means we must have been able to reach this vertex in the first place. And this guy got its length from here and so on. (Refer Slide Time: 16:20-16:30) We will eventually hit the root. So what can we say about this sub graph? If I ignore or forget the directions of the edges then it is a connected sub graph with n-1 edges. It will be a tree that is a spanning tree. But if I bring back the direction, note that in a directed graph there is really no notion of the spanning tree.

(Refer Slide Time: 18:57)



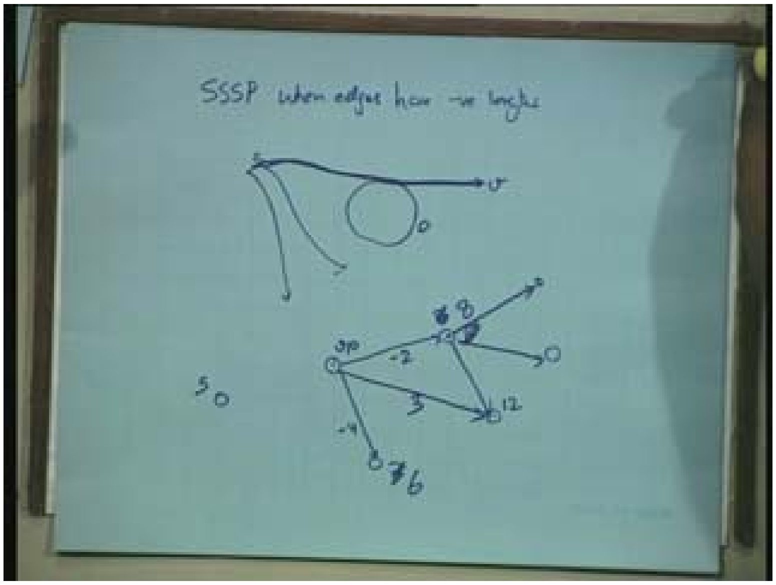
There is another name give to it. It is called a branching, the same thing. It is basically a sub graph with n-1 edges such that I can reach from a specific root vertex to every vertex. Such a thing is called a branching. It is not very important, you do not need to remember this step, but you understand what the idea is. With these edges I can go from the source vertex to every other vertex. In the context of shortest path this is called the shortest path tree. Because if I just look at this tree, these set of edges then it is very nice. If I want to find out the shortest path from s to this vertex which is next to v_2, what do I have to do?

I just follow the unique path that is there in this tree from s to this vertex, or conversely I go to the predecessors and then to the root. That is the path, which is the shortest path from s to this vertex which is next to v_2. Similarly the shortest path from s to this vertex is this path and so on. All the shortest paths are getting captured here by just these n-1 edges. These n-1 edges capture for us in a second manner, all the shortest path from the source to all the various vertices. (Hindi Conversation) This is called the shortest path and we can compute this tree in the same time as required by Dijkstras algorithm. We have not spent any additional time.

You also know how to find out the shortest path. What remains is what we had mentioned in the last class if edges have negative lengths. How can you then find out the shortest path? Once again we are talking of single source shortest path. When edges have negative lengths. What I mean by this is, not all edges have negative lengths, some have negative some have positive. But we are permitting edges to have all kind of negatives and positives. Once again we have a source and we are trying to find out the shortest path from this source to all the vertices of the graph. I am going to assume that there is no negative cycle because recall if there is a negative cycle then this thing is meaningless.

There is no negative cycle in the graph. What can you say about the shortest path from s to v? Can it be a non simple path? What is a simple path? It does not repeat any vertex, no cycle on the path. Can this be a path like this? Why not? This cycle cannot have negative length which means if I were to go in the straight line above the circle, it would be shorter. Because this cycle has positive length. It will only be shorter to go directly like this. The cycle could have the also zero length in which case it will not be shorter but this will be a path of same length.

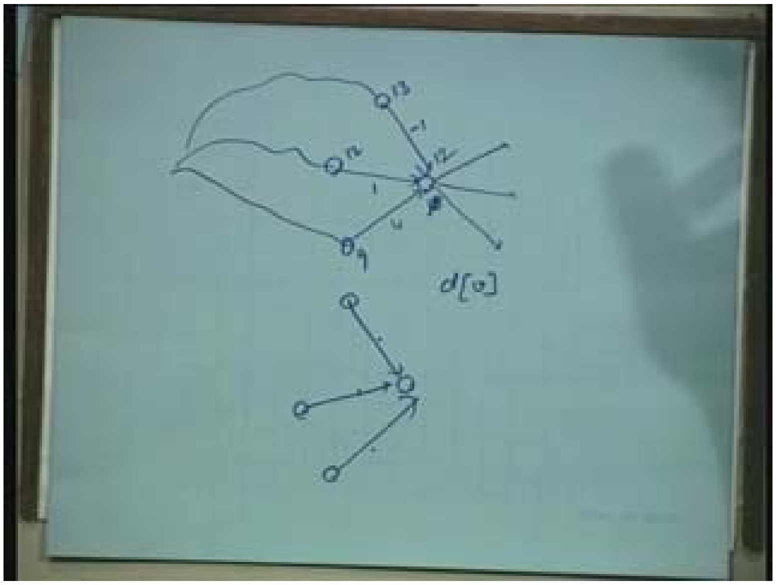
(Refer Slide Time: 25:02)



There always exists a shortest path which is the simple path. This could also be the shortest path if this cycle had zero length. But the point is there will always be a simple path also we can just restrict our attention in finding a simple path. We will only be interested in finding simple path. The algorithm is very simple actually, it is a gossiping kind of algorithm. What is that mean? Think of each of this vertex as a person and this person talks to 3 people. (Hindi Conversation) Whatever information this guys gets, he promptly communicates it to his 3 mates. What they do is, they intern communicate this information to their mates and so on. (Hindi Conversation) What is the information that is being gossiped about? Length of the shortest path from the source.

What this guy is going to tell these people is that the shortest path that I have seen so for from the source has length 10. So he will communicate this to all this 3. And what will this guy do? (Hindi Conversation) Now I have found a path of length 8. (Hindi Conversation) What is the shortest path it had found so for from the source? May be that information was 9, so now it will quickly change its thing to 8 because it had found a path of length 8 now. And may be (Hindi Conversation) that will call one round (Hindi Conversation) from its in neighbor (Hindi Conversation)

(Refer Slide Time: 29:39)

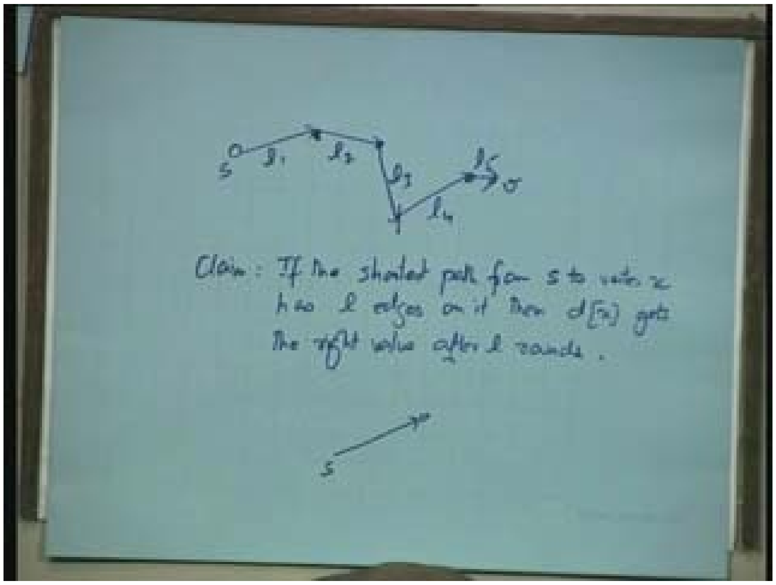


I have found a path of length 9 and this is 4th and here you have a path of length 13. (Hindi Conversation) I have found a path of length 12 and this is 1 and you have the path of length 13. (Hindi Conversation) I have found a path of length 13 this is minus one so you have a path of length 12. What is this guy going to do? It is going to take the minimum of these 3 and if it had some original, earlier information. It is going to take the minimum all of them and put that down as the new path length. Let say original information was more than 12 then that will become path length and this is the information that it would communicate in the next round. (Hindi Conversation)

One round is inner round (Hindi Conversation) ever one talks to their neighbors and sends the information. And then each node also gathers the information from its neighbors and updates its value. That is the end of the round after it has updated its value. In inner round, each guy talks to its neighbors. (Hindi Conversation) First transfer then updation. Let us say the round starts with the transfer then when you have got information from all your neighbors, you update. All the nodes do the same thing simultaneously. A node will get information only from its inmates. (Hindi Conversation) You have these vertices, you have to implement this thing. What is the information we are maintaining with each node, once again we call it a distance label. With the node v, I am maintaining this distance label.

How will you implement one round? I take a vertex, there is no notion of actual transmitting any more. This was just to show you the idea. What does this vertex do? It look at its in adjacent vertices. (Hindi Conversation) propositional to its degree. The total time (Hindi Conversation) number of edges. This is how you will implement one round. (Hindi Conversation) zero on the source and infinity for all others (Hindi Conversation) If in a round nothing changes then we can stop.

(Refer Slide Time: 35:44)



Claim: If the shortest path from s to vertex x has l edges on it then d[x] gets the right value after l rounds.

(Hindi Conversation) from the source we will have to prove that (Hindi Conversation) let see (Hindi Conversation) There is a difference between length and number of edges on the shortest path. This is the shortest path using edges length (Hindi Conversation) n-1 shortest path (Hindi Conversation) A shortest path from s to v then what is the shortest path from s to this vertex which is between $l_1 and l_2$. That property continues to hold true. (Hindi Conversation)

That was independent of whether the edge lengths were positive or not. Shortest path from s to this vertex will be this they can not be a shorter path because if there were the shorter path then this could not have been the shorter path from s to v. Similarly shortest path from s to this would have been $l_1 + l_2 + l_3$ and so on.

So the claim we are going to make is, if the shortest path from s to vertex x has l edges on it then d[x] gets the right value which means the value of the shortest path after l rounds. (Hindi Conversation) After 5 rounds this vertex v will have the correct length of the shortest path. Why? This claim is true we will prove the claim by induction on l. l equals one (Hindi Conversation) is this true? (Hindi Conversation)

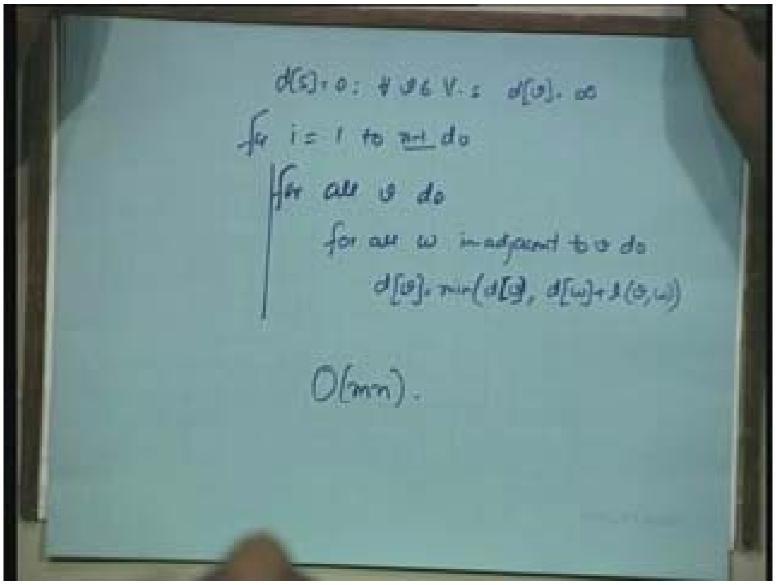
When we find a better path because if there is no better path, we will never be changing the value later (Hindi Conversation) value $l_1 + l_2 + l_3 + l_4 + l_5$ (Hindi Conversation) after five steps then it will not change beyond that. (Hindi Conversation) You take the value of the in adjacent vertices which has not been updated in that round. There is a vertex whose value has been updated right now and it has vertex you reach the vertex after that you will consider new value or old value for confusion a new value or old value. Let us forget about new values or the old values.

Let me write down the code (Hindi Conversation) for all v do (Hindi Conversation) for all w in adjacent to v do. The d[v] = min [d[v], d[w] +l [v, w]] (Hindi Conversation) for i =1 to (Hindi Conversation) n-1 do (Hindi Conversation) you will try to reduce the running time a bit. The worst case will remain the same. (Hindi Conversation) d[s] equals zero, for all v in v-s, d[v] equals infinity. (Hindi Conversation)

For i=1 to n-1 do is just to count the number of time we have to repeat the process.(Hindi Conversation) in adjacent vertices (Hindi Conversation) Why did you put down only n-1 and no more because I argued in this claim that (Hindi Conversation) What is the maximum number of edges that there can be on any shortest path. We have argued that on any shortest path there can be no more than n-1 edges because there will always be a shortest path which is simple. Does not repeat, so at most n-1 edges on it and so I need to repeat it only n-1 edges and no more. (Hindi Conversation) Should we do the correctness for this? Yes let us do the correctness, why is it correct? Why have we computed the shortest path? Have you computed the shortest path?

Is it clear that after round 5 this vertex v is going to get a distance label of $l_1 + l_2 + l_3 + l_4 + l_5$. It is going to get a distance label of this or smaller (Hindi Conversation) I am assuming this is the shortest path. After one round this guy has to get a label of l_1 that is completely clear, after 2nd round this guy is going to get a label of $l_1 + l_2$ or less. This guy is going to get a label of $l_1 + l_2 + l_3$ after 3 rounds or something even lesser.

(Refer Slide Time: 41:49)



```
$d[s]=0;\ \forall v \in V-s\ \ d[v]=\infty$
for i = 1 to n-1 do
    for all v do
        for all w adjacent to v do
            $d[v]=\min(d[v], d[w]+l(v,w))$

$O(mn)$.
```

After 4 rounds this is going to get a label of $l_1+l_2+l_3+l_4$ or some things lesser. This guy is going to get a label of $l_1+l_2+l_3+l_4+l_5$ after 5 rounds or something lesser. So for correctness we need to argue that this guy is not going to anything lesser than $l_1+l_2+l_3+l_4+l_5$. If you have argued those then we have that it gets exactly the distance label (Refer Slide Time: 42:00-43:17)

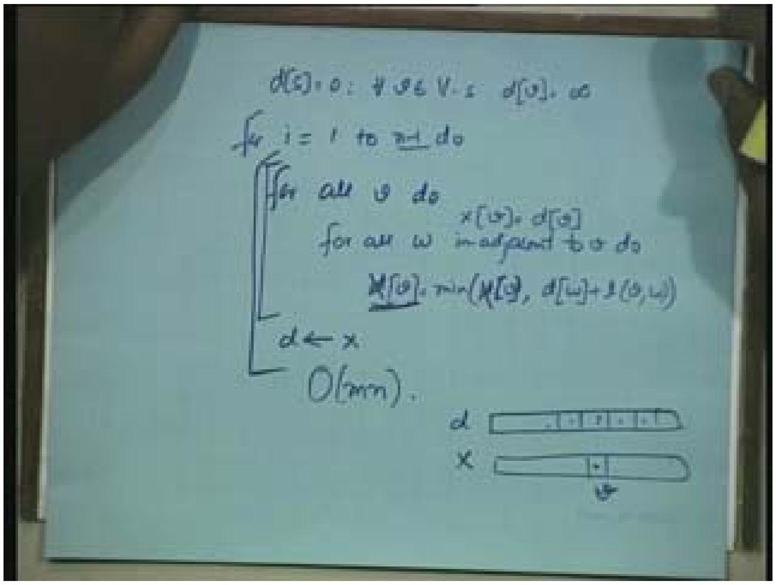
We have to argue that a vertex cannot get a distance label which is lesser than the length of the path from the source to that vertex. I give a vertex of a certain distance label, it is because I have found a path of that length from the source to that vertex that I have to convince you. And that follow from the way we are doing things. That will be a proof by induction. What should we induct on?

What is the claim we are trying to make? Let us write that down. If a vertex v gets a certain distance label d in round r, then we have found a path of length d from s to v. This is what we are trying to argue. Basically what I am saying is that if a vertex is getting some distance label it is not that (Hindi Conversation) Why is this claim useful?(Hindi Conversation) $l_1+l_2+l_3+l_4+l_5$ l(Hindi Conversation) This is not a shortest path. Contradiction Why is it true? (Hindi Conversation) Distance label d need not be an integer it is just the length of the path. (Hindi Conversation) Suppose this statement is true till round r-1. Then in round r, if this vertex v is getting as a certain distance label d because either that was the distance label on it or because one of its neighbors, so this vertex getting a distance label d (Hindi Conversation) It could have been updated in the same round and you are worried about that aspect. Then we will have do a double induction kind of a thing.

Let us not worry about that, I will modified the code so that every thing gets updated simultaneously. (Hindi Conversation) But let us just assume this vertex u, this d[u] was the label in round r-1. So now we can apply induction hypothesis which means that we found a path of length d[u] from s to u. That was the induction hypothesis saying. (Hindi Conversation) And this plus this means that I have found a path from s to v of length exactly d[u] +l [u, v] which is all d (Hindi Conversation) that is complete (Hindi Conversation)

There is a valid doubt, you guys are worried a little bit (Hindi Conversation) look at the algorithm but let us modify the thing so that we do not do that at all (Hindi Conversation) for all v do (Hindi Conversation) x [v] equals d[v], x[v] equal to min[x[v] , d[w]+ l[v,w]](Hindi Conversation) In fact that will be faster than this one because you are kind of doing a little bit more work in each round. For this perhaps the correctness is more apparent so it is okay if you do it this way. But even that it is correct.

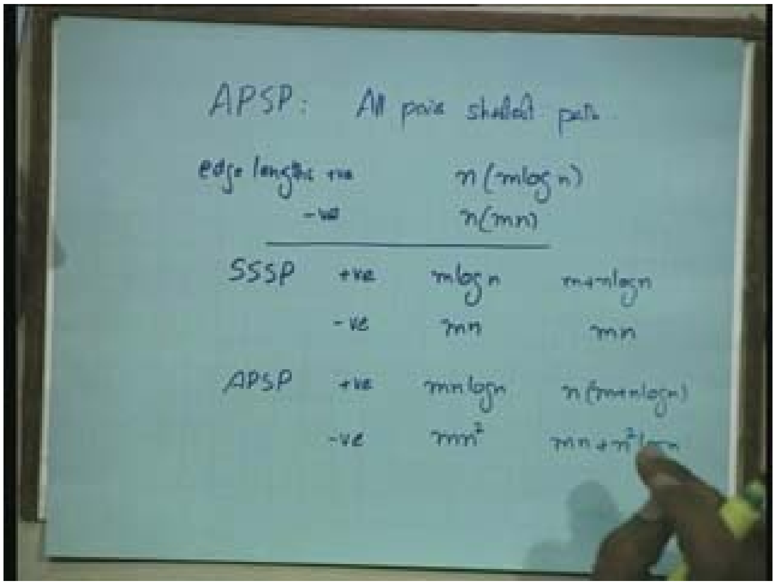
(Refer Slide Time: 52:12)



```
$d[s] = 0 ; \forall v \in V - s \quad d[v] = \infty$
for $i = 1$ to $n-1$ do
    for all $v$ do
        $x[v] = d[v]$
        for all $w$ adjacent to $v$ do
            $x[v] = \min(x[v], d[w] + l(v,w))$
    $d \leftarrow x$
$O(mn)$.
```

The last thing I want to say very briefly is APSP, this is All Pair Shortest Path. (Hindi Conversation) all pair, every pair of vertices (Hindi Conversation) n (mlog n) time (Hindi Conversation) n (m, n) (Hindi Conversation) these are not the best possible, there are algorithm which can do better than this. You will learn about these algorithms in the algorithms scores. But let me just tell you what are the better bound nodes.

(Refer Slide Time: 56:29)



APSP: All pair shortest path

edge lengths +ve	$n(m \log n)$	
-ve	$n(mn)$	

SSSP	+ve	$m \log n$	$m+n \log n$
	-ve	mn	mn
APSP	+ve	$mn \log n$	$n(m+n \log n)$
	-ve	mn^2	$mn+n^2 \log n$

So for single source shortest paths with positive edges length (Hindi Conversation) m logn, the best bound known is m+n log n (Hindi Conversation) m could be as larger n^2 in which case this will be n^2 +nlog n could be n^2 and this will be n^2 log n. So factor log n more. This is single source shortest path when edge lengths are positive. When edge lengths can be negative, when I write negative it does not mean that every thing is negative. Please remember this, it just means that in fact every thing is negative (Hindi Conversation) best possible be mn. You can not (Hindi Conversation) All Pair Shortest Path (Hindi Conversation) mn logn. Best possible (Hindi Conversation) n (m+ n logn) (Hindi Conversation) best possible is mn+ n^2 log n, basically same as this one. Best known, no these are not the lower bounds these are only the best known. (Hindi Conversation) So with that we will end today lecture on shortest path. In fact that also brings us to the end of this course. We are not going to take up any other topic, this is all we have to discuss.