**Course: Introduction to Graph Algorithms**

**Professor: C Pandu Rangan**

**Department: Computer Science and Engineering**

**Institute: IISc**

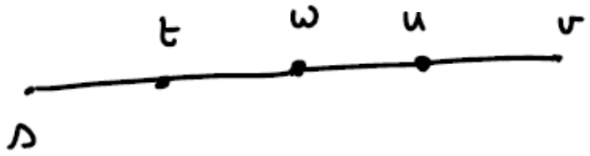
**Week: 04**

**Lecture 16 Dijkstra Algorithm**

Namaskara in this session we are going to take a look at one of the most elegant and most beautiful algorithms designed for shortest path problem, based on several interesting and subtle ideas. This is another special case where it is possible for us to solve the Bellman equation somewhat directly okay but the way in which we are going to handle the building of the solution is rather subtle okay. So this is Dijkstra's algorithm. So Dijkstra's algorithm for single source shortest path problem. So we have G which has a vertex set v, edge set E, weight function w, source s.

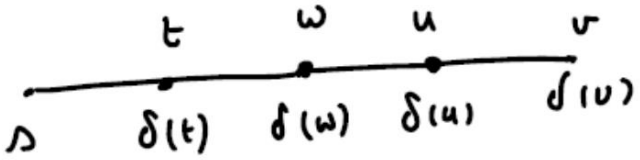
$$G(v, E, w, s)$$

One of the observations that is helpful in the building up of the intuition behind the algorithm is the following. If you look at the shortest path tree from s to v in the shortest path tree. You have a path and this path this path in the shortest path tree (SPT) is a shortest path from s to v, this is true for every vertex let us say u, w, t,




The same path the portion of the path from s to u is a shortest path from s to u, for the same path the portion of the path from s to w is a shortest path from s to w. In other words the shortest path from s to v contains the shortest path to s to u shortest path from s to, therefore if we plan to identify the vertices for which the shortest path is to be determined in the increasing order of their delta values, okay. Then we see that there is a natural way in which we can proceed.

So for example the delta t the shortest path distance here this will be delta w and this will be delta u and that will be less than delta v




Because we are going to work on a special class of graphs of course graph is arbitrary special kind of weight function where all weights are positive.

$$w(e) > 0$$

It is a very practical case since all weights are positive if these are the shortest path distances then delta t is delta w is less than delta u is less than delta v, okay because all edge weights are positive.

$$\delta(t) < \delta(w) < \delta(u) < \delta(v)$$

In other words we can see that if we know the shortest path distances for vertices whose shortest path weight values that is a delta t delta w delta these are all smaller than delta v. Then it is possible to probably find the vertex v and it is delta okay if we know all the vertices with the smaller delta values and their delta values then with a kind of an extension we should be able to obtain delta v. So, if we could somehow proceed in an order in the increasing order of the delta values the circular dependencies in the Bellman equation can be avoided and we should be able to smoothly construct the shortest paths and identify the shortest paths for vertices and we can complete our process.

We have to find out shortest path distance for n minus 1 vertices. There are n minus 1 vertices for which the shortest paths and their weights, the delta values are to be found. In the case of DAG there is a topological order and we were able to systematically use the delta value of the vertices which are before the given vertex in the topological order we could use that here. We do not have that kind of possibility here because we want to do, we want to find delta values in their increasing order. Also if you look at the Bellman equation delta v is equal to minimum of uv such that delta u plus weight of uv, so if the weight here all the weights are positive.

$$\delta(v) = \min_{(u,v)}\{\delta(u) + w(u,v)\}$$

Therefore the vertices with larger delta values are useless they may be neighbors to v but their weights, their delta values are not useful at all because we have to find a minimum, hence for v, we need only the delta values of u, delta u where delta u is less than delta v okay.

$$\delta(u) < \delta(v)$$

We require only such vertices under delta values. Even if we have the vertices with larger delta values they are not going to play any role in determining delta v. Therefore, for every vertex v we would like to have the delta values that are smaller than the delta value of the given vertex. So if I am trying to take, if I am trying to find the delta for v I must have the delta value of all vertices with a smaller delta values.

But then how can I determine this order it is not possible for me to determine this order because it is the delta value that I have to compute. In topological sort, the topological ordering is independent of the shortest path weights, so I can work on the graph find a topological sort in a DAG and use it to go in a systematic way, in an order and then I could solve the single source shortest path problem. But here the order I have in my mind is depending on the values that I am yet to find. Therefore, the situation is a bit tricky and let us see how Dijkstra has resolved it okay. So we go incrementally.

This algorithm is going to have n minus 1 stages or phases or n minus 1 iterations you can say. In each iteration we identify one vertex whose delta value is determined, that is the shortest path weight is determined. We fix for one vertex in each iteration that is the reason why we have n minus 1 iteration because cardinality of v is n we know delta s equal to 0 we want to find delta v. for v not equal to s for all these vertices.

$$|v| = n, \qquad \delta(s) = 0$$

$$\delta(v) = ? \quad v \neq s$$

So there are n minus 1 vertices for which we have to find the delta v values okay.

So we will do n minus 1 iteration in each iteration we are going to find or determine the delta value of one vertex okay. We do the following we maintain two sets S, V minus S, S is set of all vertices for which delta values are known that is shortest path weights are known. Such vertices are maintained in a set S, this is not known. The other vertices I do not know the shortest path weights for these vertices, therefore, I will work in V minus S and find out one vertex for which the shortest path distance can be determined and then obviously I move that out to S because for that vertex I have found out the shortest path weight, so move that to S. So in each iteration one vertex will move from V minus S to S, so the high level plan is find v in V minus S for which delta v is computed unknown.

So you do some computation and for after that computation for one vertex you will know it is a delta v value. that vertex since the delta v is known our property is that S is set of all vertices for which a delta is known move v to S. So initially S, V minus S, s will have only the source vertex, V minus S will have V minus source vertex s, all vertices other

than the source vertex that is because I know delta s equal to 0. Since delta s equal to 0 this is known I start with this configuration this is my configuration starting point.

$$S = \{s\}, \text{ and } V - S = V - \{s\}$$

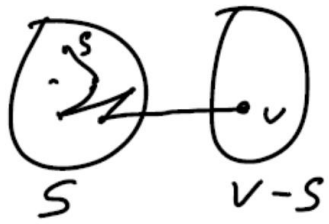
$$\delta(s) = 0$$

How do I find what should I do with vertices in V minus s so that I identify this v, after identifying the v, I can move that V to S.

So S will become larger and larger in each iteration it will be larger by 1 vertex. so after n minus 1 iteration all vertices will be moved here V minus S will be empty and the job is done for all vertices I have found out the shortest path okay. In order to determine a vertex for which the delta v is to be computed I have to maintain some information with respect to each vertex in V minus s okay. So for each vertex v in V minus S we maintain some info that helps to determine a vertex. for which the shortest path weight can be computed using this information I should be able to do that.

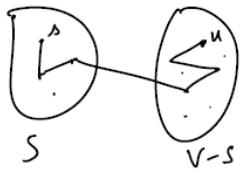
$$v \in V - S$$

So what information is to be maintained for each vertex, so this is where the intuition which I have discussed before you can see that the vertex the shortest path from S to V goes through the vertices and if I know the shortest paths for all of them okay then as an extension of that vertex the last vertex I can determine the shortest path for V as well. So we plan in the following manner okay call a path from s to v in V minus S, a special path if all vertices other than v is, all vertices are in S, it is called a special path, a special path may or may not exist. For example if this is the vertices in S and this is a vertex V if a path is like this this is a special path




Because other than the vertex v all vertices in the path are in S this is S this is V minus S this is a special path okay if there is a vertex for which a path from S to let us say u if it is like this is a path but this is not a special path because it has got several vertices in V minus S also in that path it should not be like that. A special path is a path okay which is passing only through the vertices of S which means it is passing only through the vertices for which the shortest path weights are known, like the path pv in the shortest path tree, it is one such path that is what we are aiming to build. There are several vertices for some the special path may be there for some special path may not be there and for some of

them more than one special path might be there and what we are going to maintain is the weight of the shortest special path okay.

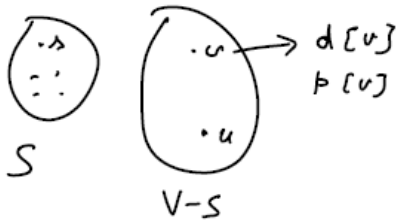



So let dv be the weight of shortest special path from s to v. this is for each v belonging to V minus S we maintain this information. How we generate this information we will see but assume that I have d of v the weight of the shortest special path from S to V is available these numbers assume that these numbers are available and not only this for every, if there is a shortest special path I also maintain the previous vertex in S for V because the special path will pass through vertices in S and then it comes to V minus S. the last edge is called crossing edge why it is called the crossing edge that edge goes from S to V minus S the special path is going to be like this. then in one jump it will come to v this is s, s is in S, v is in V minus S, so from S to V minus S with one crossing edge it will reach that is the property of special path not all paths special paths okay.


This vertex this vertex in s that is called the previous p of v is the vertex in S that is previous to v, that is previous to v. This vertex let us say if this is v dash if this is v dash then p of v equal to that v dash in this picture okay.

$$p(v) = \acute{v}$$

v dash is here and so for each vertex we maintain just 2 pieces of information d of v and p of v, we are still to explain how these things are computed. Let us assume that this is available for the time being right that is we are in one intermediate stage where for some vertices the shortest path distance is known they are all in S. s is there and few other vertices are there and for V minus S for every vertex v you have dv is available and pv is available for every vertex the corresponding values are available okay these two are available.



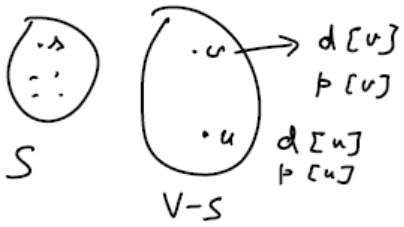

The first claim is, find the vertex v find a vertex v in V minus S with minimum d value. that means d of v is less than or equal to d of v dash for all v dash belonging to V minus S okay.

$$d[v] \leq d[\acute{v}] \; \forall \; \acute{v} \; \in V - S$$

Yeah you can even use the notation u because v dash is used here let me not confuse that.

$$d[v] \leq d[u] \; \forall \; u \; \in V - S$$

So look at V minus S vertices each vertex has got a dv value find that vertex which has got a minimum dv value. So what is the property? The property is that the dv will be less than or equal to du for any u if it is u this has got du and pu and so on




Any vertex its d value will be larger this is just finding the minimum okay. The size of V minus S is maximum n minus 1 in fact V minus S will shrink in every iteration we are going to identify a vertex in V minus S for which the delta value is known once the delta value is known we move to S. because S is collecting all the vertices for which the delta values are known therefore we will move that out that means V minus S will shrink by 1 S will grow by 1. in each iteration V minus S will keep becoming smaller S will keep becoming larger by one vertex okay. So find a vertex v with the minimum this is easy to do just to scan the vertices in V minus S under d values find a minimum our claim is d of v is equal to delta v. okay

$$d[v] = \delta(v)$$

That means I have identified a vertex for which delta v is known therefore because of this move v to S. what is the next thing you have to do simply move v to S.

Let us focus on the algorithmic aspect up to this stage yes some more things to do but let us see why for this particular vertex the delta v is actually dv okay, so the proof is a very

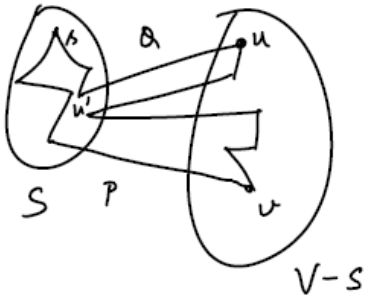
elegant proof and the proof is by contradiction. okay so let us look at a picture S, V minus S, I have a vertex v assume that this v has the minimum d value okay so from S I have a path okay this path you call it as P okay let P be the special path from s to v with weight of P equal to this is a special path its weight This is a path from s to v and this paths weight is dv. My claim is this is the shortest path. We claim P is a shortest path. We claim that P is a shortest path.

$$w(P) = d[v]$$

Suppose this is not the shortest path. Let us consider a shortest path from s to v if not let Q be a shortest path from s to v okay. P is not the shortest path and Q is a shortest path and weight of Q will be less than weight of P which is equal to d of v.

$$w[Q] = w[P] = d[v]$$

So how will the Q look like it would not be a special path why it would not be a special path because d of v is the minimum weight special path this is a path whose cost is smaller than this therefore that would not be a special path and Q is like this it may have some vertices then it may go out and it may come again and all it does not matter. So this is how Q may look like okay




The Q is starting from s it goes and then reaches because the path has to reach v it must have one crossing edge it cannot be the path Q cannot be completely contained in S that is not possible.

Because v is in V minus S the Q has to cross it has to go out of S at some point it may come again and it may go that does not matter but it has to go out at least once. right it must have one crossing edge at least. Let u u dash, u u dash be the let u dash u be the first crossing edge in Q. Q must have one crossing edge let u dash u be the first crossing edge. Now I am going to write Q from s to v as Q from s to u plus Q from s to u plus Q from u to v.

$$Q[s, v] = Q[s, u] + Q[s, v]$$

I am breaking the path into two parts I am breaking at u at this vertex u I am breaking the path Q from s to u Q from u to v, I write like this. This is a special path ,why this is a special path this is the first crossing edge before that all the edges are in S that means all the vertices are in S therefore the part of Q from s to u is a special path that may not be the minimum weight special path but that is a special path this is a special path from s to u okay. So weight of Q s to v is same as weight of Q from s to u plus weight of Q from s to v.

$$w(Q[s,v]) = w(Q[s,u]) + w(Q[s,v])$$

It is a path that is split into two parts this part any path will have positive weight because all edge weights are positive therefore this will be strictly greater than wQsu

$$w(Q[s,v]) = w(Q[s,u]) + w(Q[s,v])$$

$$> w(Q[s,u])$$

$$\geq d[u]$$

because I am dropping a positive term this part I have dropped so I have cut off. this part will have a smaller weight than the whole the whole path will have a bigger weight one part will have a smaller weight.

And this is greater than or equal to d of u because what du has is the minimum weight special path right weight of minimum weight special path the shortest special path. this is some special path su is some special path, du is the weight of the shortest special path therefore that will be smaller than this one. But what about v, v has a property that that has got the smallest d value among all vertices in V minus S, u is in V minus S, v is in V minus S okay this is by the property of v, d of v is equal to weight of P okay this that means weight of Q. is greater than weight of P

$$w(Q[s,v]) = w(Q[s,u]) + w(Q[s,v])$$

$$> w(Q[s,u])$$

$$\geq d[u]$$

$$= w(P)$$

$$w(Q) > w(P)$$

This is a contradiction. Because we have assumed that Q is the shortest path look at here weight of Q is less than weight of P. We have assumed that P is not the shortest path and Q is the shortest path therefore it is shorter than P that is what we have started with. But now we have got it this is a contradiction, this is a contradiction to the property of Q,

because we have assumed that Q is the shortest path and is a contradiction which we have already assumed. Hence such a Q cannot exist hence P is the shortest path. it is not only a shortest special path, it is also shortest path in the entire graph. Thus dv is in fact delta v hence we can move v from V minus S to S.

$$d[v] = \delta(v)$$

Okay we have discussed one major and important part of the algorithm we will continue our discussions on the other aspect of this algorithm in our next session thank you.