Course: Introduction to Graph Algorithms

Professor: C Pandu Rangan

Department: Computer Science and Engineering

Institute: IISc

Week: 05

Lecture 19 All Pair Shortest Path

Namaskara we will now begin our discussions on algorithms for all pairs of vertices, this is called all pairs shortest path problem. We have seen single source shortest path problems we have seen the algorithms by Bellman and Ford we have seen the algorithm by Dijkstra. We can naively repeat from every vertex as a source vertex when we do that it will be the complexity of n times the complexity of the single application. However, it would be nice to see whether we can have certain algorithms that are working directly from the input. There are several interesting ideas that have evolved when we are trying to build the solutions for the shortest path problem for all pairs. We are going to discuss about four basic algorithms for all pair shortest path problem then combining cleverly the single source shortest path problems by Dijkstra as well as Bellman and Ford a fifth algorithm will also be discussed.

We will start with a graph that is directed, it is weighted. We assume that it has got no negative or 0 cycles, we know that the shortest path problem can be meaningfully solved only if G has no negative or 0 cycle, so we will continue with that assumption.

For vertices u and v define delta uv as the weight of the shortest path from u to v, okay here the v is removed. It is defined delta uv as the weight of the shortest path from u to v, we define delta u to be 0 okay.

$$\delta(u, u) = 0$$

It is also useful to extend the weight function, weight function is defined only for edges Now we are going to extend the weight functions for all ordered pairs recall that E is a subset of V cross V.

$$E \sqsubseteq V \times V$$

Weight function is defined only for E but we will now extend a weight function in the following way these are all kind of obvious ways to extend it weight of uu is 0 for all u in V for all u that belongs to V, for every vertex we set w uu to be 0 and wuv is infinity if uv is not in E if there is no edge,we assume that the weight of the pair uv is infinity.

$$w(u,u) = 0 \;\; \forall\, u \in V$$

$$w(u,v) = \infty \text{ if } (u,v) \notin E$$

Otherwise it will be in E and the weight function is already defined for those pairs which are in E. If there is no path at all from u to v in the graph we are going to define delta uv to be infinity, delta uv is infinity if there is no path okay.

$$\delta(u,v) = \infty$$

Now we are interested in finding delta uv for all uv pairs, that is why it is called all pairs shortest path problem we are interested in finding the weight of the shortest paths for all pairs okay. Much like what we did in the single source shortest path problem, we have an implicit representation the predecessor array or the previous array which defines the vertex previous to a vertex v in the shortest path, using that as an implicit representation we could build the shortest path anytime that we want. The implicit representation provided a compact way of representing all paths. In the same we are going to derive certain implicit representation for all pairs shortest path problems. We may use that whenever we want and then we can build the shortest path explicitly okay. So what we are going to compute is only delta uv, the lengths our algorithms will focus on that with little more bookkeeping we will have an implicit representation for any path in that collection okay. That is a minor bookkeeping add-on and it can be co-computed as and when we are computing the delta uv the corresponding representation can also be built.

We will not focus on that we will focus only on the computation of the length which is the essential and the core part of the algorithm. Now we are interested in all pairs therefore there is a matrix of values are to be determined, that is the reason why it is convenient to set V as 1 to n, so our vertex set is 1 to n.

$$V = \{1,2,3,\cdots,n-1,n\}$$

The extended weight function can also be represented simply as a matrix W this is called the extended weight matrix w ij it is an n cross n square matrix, w ij is 0 if i equal to j that is how we have set and w ij is infinity if the pair does not belong to E, and wij is the weight of ij if ij belongs to E

$$W = \left[w_{ij}\right]_{n\times n},\; w_{ij} = 0,\; i = j$$

$$w_{ij} = \infty \text{ if } (i,j) \notin E,$$

$$w_{ij} = w(i,j) \text{ if } (i,j) \in E$$

there is a weight function available and that weight function assigns an integer to each edge put that edge. So you can imagine a matrix that has got zeros in the diagonal.

infinities in the place where there is no edge and weight of the edge where there is an edge, very simple to visualize the square matrix is called weight matrix.

We are going to introduce a notation that is going to be very helpful in designing our algorithm. The set of all paths from i to j with utmost l edges. that is number of edges is less than or equal to l number of edges is less than or equal to l is what we mean by at most. So maximum number of edges that can be present in that path is l collect all those kind of set all those kinds of paths. that set is Pl colon i, j we read it that as Pl i, j

$$P[l\colon i, j]$$

but remember the colon, that colon is separating l which is representing a different parameter i and j are vertices of the graph, l stands for the upper bound on the number of edges.

The weight of the shortest path in Pl ij is denoted by delta sub less than or equal to l ij.

$$\delta_{\leq l}(i, j)$$

So less than or equal to l occurs as a subscript to indicate the fact that we are considering the shortest path in Pl ij okay it is not the overall shortest path it is shortest among the paths that has got maximum l edges, l is of course greater than or equal to 0 okay. We will collect and organize all these values in the form of a matrix, so delta less than or equal to l ij will be collected in a matrix called Dl matrix. and the Dl matrix entities are denoted by dl ij which is nothing but delta less than or equal to lij.

$$D^{l} = [\delta_{\leq l}(i, j)] = \left[d_{ij}^{l}\right]$$

So this is an n by n matrix of values since any path in G may contain at most n minus 1 edges. Remember path means no vertex is repeated and no edge is repeated. So source to destination you have to go through distinct vertices, distinct edges maximum you can have n minus 1 edges therefore P n minus 1 uv is going to collect all the paths.

$$P[n-1\colon u, v]$$

Any path, because any path is going to have less than or equal to n minus 1 edges therefore all paths are collected in P n minus 1 uv. Therefore the minimum in that is the shortest path that is the reason why we are interested in dij n minus 1 you see why we are interested in d this actually represent the length of the shortest path. So d n minus 1 ij is equal to delta less than or equal to n minus 1 ij which is equal to delta ij, because it is the shortest among all paths okay.

$$d_{ij}^{(n-1)} = \delta_{\leq l}(i, j) = \delta(i, j)$$

So, the observation that Pn minus 1 uv is set of all paths is the critical observation that is allowing us to set our goal.

$$P[n-1: u, v]$$

Therefore we have to compute the matrix denoted by Dn minus 1 we can go incrementally right that is the strategy the plan is the reason why we have we can step through.

We define D0 matrix in the following way what is the definition of D0, less than or equal to 0 edges. less than or equal to 0 edges the number of edges is always 0 or more less than or equal to 0 means it is just 0 edges.

$$D^0 = [d_{ij}^0]$$

How can you go from a vertex i to j with 0 edges there is no path of that type. Therefore when i is not equal to j d0 ij is infinity no path exist with 0 edges. We have defined dii the delta uu is 0 right that is what we have defined so we are defining D0 ij is 0 if i equal to j in other words this D matrix is 0 along the diagonal infinity everywhere else. Okay

$$d_{ij}^0 = 0 \text{ if } i = j$$

$$= \infty \text{ if } i \neq j$$

So 0 0 0 infinity, infinity, infinity it is going to be a matrix like this, infinity everywhere 0 along the diagonal okay.

$$\begin{matrix} 0 & \infty & \infty \\ \infty & 0 & \infty \\ \infty & \infty & 0 \end{matrix}$$

So let us see how we can go about building dl from d l minus1. Here is the Bellman equations that we have seen already in the case of the single source shortest path rephrased in the context of all pairs shortest path. So this Bellman equation that is valid for all pairs take any pair of vertices i and j, We prove that dlij is greater than or equal to dl minus 1ik plus wkj.

$$d_{ij}^l \geq \left(d_{ik}^{l-1} + w_{kj}\right) \forall k$$

We are showing this and we are going to show that there is one particular k for which the equality holds good.

$$d_{ij}^l = \left(d_{ik}^{l-1} + w_{kj}\right)$$

So all the values are greater than or equal to the right hand side and it is actually equal to one of them. And therefore these two together would imply this,

$$d_{ij}^{l} = \min_{k}\{d_{ik}^{l-1} + w_{kj}\} \text{ --1}$$

This is very similar to the Bellman equation that you have seen in the context of single source shortest path. So this is what we are going to prove okay dl ij is greater than or equal to dl minus 1 ik plus w kj and dl ij is exactly equal to one of these values.   So if Dl minus 1 values are available each dl ij can be computed by using n addition and n minus 1 comparisons as shown in the equation above. So look at this equation 1, if you look at the equation 1 you have to find the minimum of n numbers, so how do you find each number? dl minus 1ik plus wkj one addition, so if dl matrix is available that means Dl minus1 matrix is available means d l minus 1 ik is available wkj is the weight matrix entity these are available. Since these values are available for each k find the sum okay there are n values, so n additions, you get n numbers, compare them and find a minimum you get the ij entity for dl ij okay.

So you have to use n additions and n minus 1 comparisons for one number of Dl.   okay. So Dl matrix is to be computed by computing dlij for each of them you are using n addition and n minus 1 comparisons there are n square numbers to be computed therefore Dl can be computed from Dl minus1 in n cube time okay. So you have D0 from D0 you compute D1 from D1 you compute D2 and from D2 you compute D3 in this way you compute up to Dn minus 1. Each hop is n cube D0 to D1 n cube D1 to D2 another n cube so n cube plus n cube plus n cube, it comes to order n power 4.

$$D^{l-1} = d_{ij}^{l}$$

$$D^{0} \rightarrow D^{1} \rightarrow D^{2} \rightarrow D^{3} \dots \rightarrow D^{n-1} \rightarrow n^{3}$$

$$O(n^{4})$$

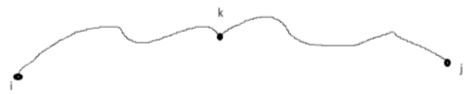
Therefore if we establish the Bellman equation it gives you the first direct and simple algorithm with complexity order n power 4. So Bellman equations are the key for our first simple and direct algorithm. Again to recall this is the Bellman equation.

$$d_{ij}^{l} = \min_{k}\{d_{ik}^{l-1} + w_{kj}\}$$

This relates the entries of the matrix Dl minus 1 to the entries of the matrix Dl, so we are able to compute systematically D0 to D1, D1 to D2 and so on in n power 4 steps. How do you prove this? what is that we have to prove? We have to first prove that dl ij is greater than or equal to dl minus 1 ik plus weight of kj.

$$d_{ij}^{l} \geq \left(d_{ik}^{l-1} + w_{kj}\right)$$

So it involves ik and kj, kj is the last edge okay. So you have the following picture is helpful i to k j okay, i to k and j.



Now the whole path the whole path has l edges this is the last edge okay less than or equal to l edges therefore this path will have less than or equal to l minus 1 edges path is going to have less than or equal to l minus 1 edges. So if you consider the shortest path the shortest path assume that this is the shortest path if this is the shortest path its length if you call that as let us say P. weight of P, delta less than or equal to l ij delta less than or equal to ij which is equal to weight of the path P is actually the shortest path, so weight of the path P from i to j this is weight the path P from i to k plus weight of kj up to this you add and then kj weight of kj. This is some path this will be greater than or equal to delta l minus 1 delta means we have to D means we put on the top delta means we put in the bottom okay. So I first have to be consistent with this, this is what we want to show is d greater than or equal ij plus weight of kj, this is what we are denoting by dl ij, this is what we are denoting by dl minus 1 ik, so what we have shown is this dl ij is greater than or equal to dl minus 1ik

$$\delta_{\le l}(i,j) = w\big(P(i,j)\big) = w\big(P(i,k)\big) + w(k,j)$$

$$\ge \delta_{\le l}(i,j) + w(k,j)$$

$$\delta_{\le l}(i,j) = d_{ik}^{l-1}$$

Okay so straight forward common sense argument alright the straight forward common sense argument that is because this is going to have less than or equal to l minus 1 edges. This is some path with less than or equal to l minus 1 edges its weight is going to be greater than or equal to this therefore you derive this. okay.

Now if all values are strictly smaller right, we may not be able to use this formula there is one value for which the equality is attained in other words the minimum among all of them okay will be so we consider, again the shortest path alright from i to j assume that this is the shortest path that means it is a length is delta ij assume that it has got less than or equal to l edges. So this part will have less than or equal to l minus 1 edges, this part will have less than or equal to l minus 1 edges and we claim that this is the shortest path. is the shortest path from i to k okay. So right now the situation is delta l ij is equal to weight of the path P from i to k plus the weight of kj.

$$\delta_{ik}^{l} = w\big(P(i,k)\big) + w(k,j)$$

$$= \delta_{ik}^{l-1} + w(k,j)$$

My claim is this is the shortest path with less than or equal to l minus 1 edges. If this is the shortest path then this will be equal to delta l minus 1 ik plus wkj.   We would have proved that if this is the shortest path suppose this is not the shortest path we arrive at a contradiction okay. If this is not the shortest path we are going to arrive at the contradiction. We claim that this is the shortest path so I write like this weight of this is this I write because I claim this is the shortest path the portion of P from i to k is the shortest if this is not the shortest path. If so i to k, if i to k is not the shortest path let Q be a path this is the path Q, Q less than or equal to l minus 1 edges and weight of Q equal to delta less than or equal to l minus 1 ik

$$w(Q) = \delta_{\leq l-1}(i,k)$$

This is the shortest path I have written its length. Therefore Q plus kj is a walk Q plus kj is a walk and notice that wQ is less than weight of Pik

$$w(Q) < w\big(P(i,k)\big)$$

This is the because P is not the shortest path Q is the shortest path which is shorter than that Q plus kj is a walk with the weight wQ plus weight of kj.

$$Q + (k,j) = w(Q) + w(k,j)$$

Recall if a graph has got no negative cycle no zero cycle every walk will have a path with a shorter or equal weight, so let P dash be a path which is a sub walk of Q such that is less than or equal to wQ sub walk of Q plus kj. wP dash is less than or equal to wQ plus wkj but this is less than wPj Pik plus wkj which is equal to wP which is equal to delta lij.

$$w(P') \leq w(Q) + w(k,j)$$

$$< w\big(P(i,k)\big) + w(k,j)$$

$$= w(P) = \delta_{ij}^{l}$$

So I have a path that is smaller than the weight of the shortest path, this is a contradiction this is impossible. You cannot have a path whose weight is smaller than the shortest path. Here is a path, which is a shortest and the smallest path therefore such a Q cannot exist okay, this implies Q cannot exist and this implies wPik is the shortest path since wPik is the shortest path I can write like this, this justifies the Bellman equation.   Since Bellman equation is justified this leads to an n power 4 algorithm. Here is the first n power 4 algorithm summarizing V is the vertex at 1 to n, G has no negative or 0 cycles, W is the extended weight matrix. D0 is defined as discussed earlier all I am going to do is that for l equals 1 to n minus 1, compute Dl using Dl minus 1 as discussed in the Billman equation. returned Dn minus 1

APSP-1 $(G = (V, E), W)$

\\$V = \{1,2,\cdots,n\}$

\\G has no negative or zero cycles

\\W is the weight matrix representing extended weight function.

$D^0 = \left[d_{ij}^0\right]$ where

$d_{ij}^0 = 0$ if $i = j$

$= \infty$ if $(i \neq j)$

For $l = 1$ to $n - 1$

Compute D^l using D^{l-1} as shown in Eq. 1

Return D^{n-1}

That is what we wanted and the complexity of this is order n power 4 this is order n power 4 algorithm based on Bellman equations for bounded paths.

$$O(n^4)$$

Bounded paths means paths with the number of edges upper bound by a constant they are called bounded paths. We considered the paths which are bound by length l and then we derived this algorithm okay. The complexity of this algorithm is order n power 4. Thank you and we will continue our discussions about this algorithm and improving the same in our next session.