Second Level Algorithms

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 04

Lecture 19

Welcome to the 19th lecture of second level algorithms course in the last lecture we have seen Edmond Karp algorithm in this lecture we will analyze that algorithm using the concept of layered graphs okay so let's begin So let us understand the layered graph with a more concrete example to ensure that we have all understood the layered graph clearly. So suppose let us take a residual graph. So, suppose this is a residual graph and there are capacities. I am not putting the capacity values because layered graph is an unweighted graph.

We ignore the capacity so that we do not need. The first task is to partition the vertices based on the distance from S. So, L_0 contains only S which is at zero hop away from S. Let's find out the vertices which is one hop away from S that is A and C. So, this constitutes L_1. Let us find out the vertices which is two hops away from S. They must be one hop away from some vertex in L_1.

So, that is B and D. T is the only vertex which is 3 hop away from S. So this is L_3. Now let us draw the edges but we will not incorporate or not use the backward edges or cross edges. We will only put edges which go from L_i to L_{i+1}. okay so I put the edge s to a but I do not put the edge from a to s similarly I put the edge from s to c I put the edge from a to b but i do not put the edge from b to a because that edge goes from L_2 to L_1 that is a backward edge. For the same reason I do not put the edge from B to C because that edge goes from L_2 to L_1. I put the edge from C to D and again I do not put the edge from D to B because that edge is a cross edge that edges that edge does not go from L_i to L_{i+1} and then I put the edge from b to t. So, this if this is the residual graph G_f, this is the corresponding layered graph. A crucial observation about the layered graph is the following.

for every vertex V in capital V minus s, every shortest in terms of number of edges used from S to B in the residual graph G_f is also present in the layered graph. Why this is the case? So, for that let V be any vertex other than S if V is unreachable from S then the statement is vacuously true otherwise Let us assume that V belongs to L_i for some integer i that is the distance of V from s is I in G_f.

Distance means again the number of edges used. Now we see that any shortest path from S to B cannot use any back edge or cross edge; otherwise, its length will be more than I. Observe that any shortest s-to-v path in G_f cannot use any back edge or cross edge; otherwise, the length of the shortest path will be more than i, okay.

So, the layered G_f^L succinctly represents all shortest paths from s in the residual graph G_f, okay. So, now using this layered graph, let us bound the number of iterations of the Edmund-Karp algorithm. So, suppose the Edmund-Karp algorithm makes key iterations for an input instance. Let δ_i be the distance of t from s after i iterations for $i=0,1,\ldots,k$. So, delta 0 is the distance from s to t. Again, distance is the number of edges used. in the beginning of the algorithm. So, and δ_k is infinity.

So, here is the crucial that δ_0 is less than or equal to δ_1, less than or equal to δ_2, and so on up to less than or equal to δ_{k-1}, which is strictly less than δ_k, which is infinity. The last inequality is obvious from the algorithm: if the algorithm makes k iterations, that means after $k-1$ iterations, t was still reachable from s, and hence δ_{k-1} must be less than infinity. And because the algorithm has made k iterations, after k iterations, t was unreachable from s. So, δ_k must be infinity. But what about the intermediate ones?

So, proof. So, take $i\in\{0,1,\ldots,k-2\}$, and to prove that δ_i is less than or equal to δ_{i+1}. So, what is δ_i? Delta i is the distance from s to t after i iterations.

So, consider the layered residual graph after i iterations. So, this is G_f^L after i iterations. This is how it looks like: s must be in L_0. and suppose δ_i is lowercase l, then t must be in l lowercase l, s is in l0, and these intermediate layers, there could be some layers after L also, and this graph has only forward edges. Now, what does the algorithm do? It finds an s to t path in this layered graph. Every s to t path in this layered graph must be a shortest path. Here is an easy observation; let us note it down.

Every s to v path in the layered graph is a shortest path. So, throughout the discussion of the maximum flow problem, the shortest path always means a path with the minimum number of edges. So, if I pick a path from s to t in an iteration and send as much flow as

possible, then you see at least one edge of this s to t path is not there in G_f because that edge becomes saturated. So, let us give a name to this path.

Let the algorithm use the s to t path P for flow augmentation. And since every flow augmentation in the Ford-Fulkerson method in general and the Edmonds-Karp algorithm in particular, saturates at least one edge in G_f, at least one edge from $G_f^L(I)$ is not available in the residual graph. in the next iteration. Now, if the next iteration it has two choices either there exist another s to t path in the layered graph G which avoids that saturated edge in that case the $\delta_i$value remains same. So, if there exists another s to t path in $G_f^L(I)$ after removing the saturated edges, then the distance of T from S in the next iteration remains the same δ_{i+1} is same as δ_i. Otherwise, every shortest path shortest s to t path in the residual graph in I plus 1th iteration must use at least one back edge or cross edge with respect to $G_f^L(I)$ but in this case then or in this case δ_{i+1} if it uses any cross edge or backward edge with respect any path from s to t cannot reach t is with l edges, it has to use more than l edges is greater than l which is same as δ_i. So, this proves that in this case δ_{i+1} is strictly more than delta i and the other case that remains same. So, this concludes the proof of the plane that this distance of t from s is a non decreasing function not only that from this proof we can conclude this important corollary that because every flow augmentation makes at least one edge in the layered graph saturated.

So, at least one forward edge is removed in the next iteration, or at least one edge of the layered graph is not available in the residual graph of the next iteration. So, the number of edges in the layered graph is at most m. So, the number of iterations where the distance of t from s in the corresponding residual graph remains the same is at most the number of edges in the layered graph, which okay. And also, another observation is if delta i is less than infinity—it is finite—then the maximum distance of t from s can be at most n minus 1, less than or equal to. So, δ_i—this distance value— can go up to $n-1$, and in every m iterations, it must increase. It cannot remain stagnant for $m+1$ consecutive iterations.

So, if I look at any first $m+1$ iterations, the value of the distance is at least 1. In the next m plus 1 iterations, the value of the distance is at least 2. So, this proves the number of iterations of the Edmonds-Karp algorithm is $O((n-1)(m+1))$, which is $O(mn)$. And in every iteration, we have seen we need to just run the BFS and the other operations.

It takes $O(m+1)$ time. So, the time complexity of Edmonds-Karp is $O(mn)$ times m plus n, which is polynomial and actually strongly polynomial, as it does not depend on the value of the input numbers. So, let us stop here. Thank you.