Second Level Algorithms

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 04

Lecture 18

Welcome to the 18th lecture of the second-level algorithms course. In the last class, we have seen the correctness of the Ford-Fulkerson method, and we have also seen that the runtime of the Ford-Fulkerson method is polynomial in the number of vertices, number of edges, and all the capacity values of the edges. So, that runtime is pseudopolynomial, but not polynomial. So, a natural question could be: is the runtime Can there be instances of the flow problem where the performance of the Ford-Fulkerson method can be really bad?

So, let us see a concrete example of an ST flow problem where the Ford-Fulkerson algorithm indeed takes time proportional to the sum of the capacities of all the edges. So, let us begin. An example of a max-flow instance where the Ford-Fulkerson method indeed makes iterations proportional to the sum of the capacity values. So here is an example: S to A is k, a big integer. So, suppose in the first iteration the Ford-Fulkerson method sends or chooses the residual that chooses the augmenting path S to A to B to T So, initially the value of the flow was 0. In this path, we can send one unit of flow, which is the minimum capacity of any edge in this augmenting path. So, we augment one unit of flow along this path, and the value of the flow becomes 1.

So, in the next iteration, if I draw the residual graph G_f. The forward edge S to A has T minus 1 amount of residual capacity left. Let us first draw the forward edges A to T. So, these are the forward edges.

Now, let's draw the backward edges. There is one unit of flow from S to A, so I have a backward edge from A to S with capacity 1. There is one unit of flow from A to B, so I have a backward edge from B to A. with capacity 1, and there was one unit of flow from B to T. Sorry, this was $K-1$. So, I have a backward edge from T to B with capacity 1.

So, in the next iteration, suppose the Ford-Fulkerson method chooses this augmenting path. S to B to A to T. Then again, we can send one unit of flow and the flow value becomes 2. So, what is the residual graph after the second iteration? Let us again draw the forward edges first.

So these are all the forward edges. Now draw the backward edges. And again, the algorithm chooses the S, H, B, T path. And again, it will send one unit of flow. You see, if this way the augmenting paths are chosen—in one iteration S, A, B, T, and in the next iteration S, B, A, T, and so on—then the number of iterations that the Ford-Fulkerson method takes in this case is k. Now, if the capacity value k is very large, then the number of iterations will also be very large. So, our analysis of the Ford-Fulkerson method was actually tight. Notice that the Ford-Fulkerson method does not specify which augmenting path to choose. If there exist multiple S-to-T paths, the Ford-Fulkerson method allows us to pick any S-to-T path.

The Edmonds-Karp algorithm—what we will do instead of picking any arbitrary S-T path in the residual graph—is we will pick an S-to-T path which uses the minimum number of edges, and we will see that this way of choosing the S-T augmenting path for flow augmentation ensures that the runtime of the algorithm is polynomial in input size—actually, it is not dependent on the capacity values at all. So, let us begin the Edmonds-Karp algorithm. Instead of picking an arbitrary S-T path in the residual graph, we pick an S-to-T path in the residual graph.

that uses the smallest number of edges in the residual graph. Everything else remains the same as the Ford-Fulkerson method. Let us first see the performance of the Edmonds-Karp algorithm in the example where the Ford-Fulkerson algorithm took many iterations.

So, this is the input graph, which is the same as the initial residual graph. Now, in this graph, I have to find an s-t path that uses the minimum number of edges. Such a path can be found using breadth-first search. There are two such paths: S-A-T and S-B-T. We can pick any one of them.

If there are more than one path that uses the minimum number of edges, we can pick any of the paths for flow augmentation in the Edmonds-Karp algorithm. So, let's pick S-A-T and augment as much flow as possible. So, we see we can send a key amount of flow along this S-A-T path because the minimum capacity of any edge in this path is K. So, the flow value here was 0.

Now, after the first iteration, the flow value jumps to capital K. Let us see how the residual graph looks after the first iteration. Let us first draw the forward edges. So, these are the forward edges. Forward edges are those edges of the input graph that have residual capacity. So, only these three edges have residual capacity.

The top edges are saturated; their flow value is the same as their capacity. So, they do not have any residual capacity. So, they are not forward edges. Those edges are not present in the residual graph. Now, let us draw the backward edges.

For every edge that carries some amount of flow, there is a backward edge corresponding to that edge in the opposite direction with a capacity value the same as the flow. Okay, so this is how the residual graph looks after the first iteration. So, in this residual graph, we will again search for an s-t path, which uses the minimum number of edges. There is actually only one path, s to b to t, and we will augment a flow of value k along this path. So, the flow value becomes 2k.

And then G is this. Let us again draw the forward edges first. This residual graph has only one forward edge from A to B. Now, let us draw the reverse edges. Now, in this residual graph, s and t are disconnected, and hence the flow value is the maximum s-t flow. Hence, the algorithm terminates. So, we see that in this instance, the Edmund-Karp algorithm takes only two iterations, whereas the Ford-Fulkerson method may take at most k iterations.

Indeed, now we claim that the running time of the Edmund-Karp algorithm is big O of m squared m. So, here is the important theorem. The time complexity of the Edmund-Karp algorithm is $O(m^2 n)$. where m is the number of edges and n is the number of vertices in the input graph. So, to prove this, let us introduce an important concept called a layered graph, which we will use in proving this theorem and in the design of another algorithm for finding a maximum s-t flow, called the Dinic algorithm.

So, let us see what that object is. So, that is called a layered graph. or layered residual graph, denoted by G_f and L layered. So, here in this graph, this is a subgraph of the residual graph. The set V of vertices is partitioned into

Sets called layers L_0, L_1, and so on, and L infinity where Li is the set of vertices for which the distance is in terms of the number of edges. Li is the set of vertices for which the distance, the set of vertices v for which the distance of v from s is i, for $i=0,1$, and so on. So, what is the distance?

That is the length of the shortest path. L infinity is the set of vertices that are unreachable from S. That is the Edmund Karp algorithm terminates when T belongs to L infinity for the first time.

So, the whole graph G_f Let us partition it into these layered sets. L_0 contains only S because only S is at distance 0 from S. These are $L_1, L_2, \ldots, L_i$. So, this is the vertex set. Now, let us see the edges.

There are various kinds of edges. One could be forward edges between two successive levels. Observe that I cannot have a forward edge from L_1 to L_3, say. So, forward edges can be from L_i to L_{i+1}. This is so because, for example, if I have an edge from L_1 to L_3,

then that means there is a vertex in L_3 whose distance is actually 2 from S, and so that vertex must not be in L_3; it must be in L_2. So, we have forward edges, and in the residual graph, I can also have backward edges or cross edges. Edges between vertices of one level. So, in this layered graph, we delete such edges. Observe that backward and cross edges are not used in any of the shortest paths in the current graph.

So, this is how the layered graph is defined, We will see in the next class how we use this concept of a layered graph or layered residual graph to bound the number of iterations of the Edmonds-Karp algorithm. So, let us stop here. Thank you.