Second Level Algorithms

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 06

Lecture 26

Welcome to the twenty-second lecture of the second-level algorithms course. In the last class, we finished the analysis of the Dinic's algorithm. We have seen that the runtime of the Dinic's algorithm is $O(mn^2)$. In this lecture, let us see the dry run of all three algorithms for computing a maximum flow in a graph using some small examples to ensure that we have understood all the algorithms in a crystal-clear way. So, let us begin.

First, let's begin with the Ford-Fulkerson method. So, this is the input graph, which is the same as the residual graph in the first iteration because the flow value is 0. I find any s-to-t path and send as much flow as I can along that path. So, let us take maybe this path. S, C, A, D, T. So, this is the beginning.

Then, what does the residual graph look like in the first iteration? Let us write down the flow values. How much flow can I send along this path? I can send the minimum capacity of any edge along this path, which is 2. So, here I send 2 units of flow along every edge.

So now, how does the residual graph look? Let's first draw the forward edges. So, these are the forward edges. Let's draw the backward edges. So, this is the residual graph after the first iteration, and again, let us find the s to t path.

Suppose now we find we use this path s to a to b to t and send 3 units of flow. So, what is the flow value now? Let us write the flow, write down the flow value s to We can send three units of flow. So, in this graph, I am showing the flow values, okay.

So now, let us draw the residual graph in the next iteration. So, let us draw the forward edges first. S to A, I have a forward edge of capacity 2; B to T, I have a forward edge of capacity 7. Then, I have the other edges. So, B to A is a backward edge.

So this is how the residual graph looks like in the second iteration, at the beginning of the second iteration. Let us see if there is any path from S to T. Yes, there is a path. So let us send, let us pick maybe this path and send as much flow as we can, which is 2. So, let us draw the flow graph. So, S to A, I am now sending 2 units of flow.

Before that, 3 units of flow were there. So, the total is 5. This is how the flow graph now looks like. Now let us draw the residual graph for the next iteration. So here there was a back edge also of capacity 3.

So A2 is a back edge of capacity 3, capacity 5. So this is how the residual graph looks like in the next iteration. Let us see if there is a path from S to T. Yes, there is. So let us send as many units of flow as we can, which is 1. Now, let us draw the residual graph again for the next iteration.

Now is there any s to t path? Now there is no s to t path in the residual graph. So the flow, so this is the maximum flow. So, this is how the Ford Fulkerson method works and here as you can see although the input graph may not have antiparallel or parallel edges, the residual graphs can have antiparallel edges. So, now let us move on to Edmund Karp algorithm and see a dry run of Edmund Karp.

So let us again begin with an example. Let us take the same example. that is the input graph which is the same as the residual graph in the first iteration. I need to find a shortest s to t path. So, I can pick s to a to t path and send as much amount of flow as I can.

So, the flow graph looks like how does the residual graph look in the next iteration. Again, I need to find the shortest S to T path and augment along that path. You see the distance of T from S remains the same. We have proved that it cannot drop.

It has remained the same. In G_f^0, the distance of T from S is 3 hops. In G_f^1, it is also 3 hops. I can send one unit of flow. So, let's send it.

What is the residual graph? Okay, I find the shortest s to t path, which is this: s to a to d to t. Again, it has only three edges, and I send two units of flow. Let us again draw the residual graph of the next iteration. So, this is how the residual graph looks. Again, I find the shortest s to t path in this residual graph. s to c to a to t. Now, you see that the distance of t from s has increased from 3 to 4. As we have already proved, distances cannot drop; they can either increase or remain the same. You can send as much flow as possible. So, this is how I can send one unit of flow. How does the flow look? I can say not one, but two units of flow. Correct. Let us send two units of flow. So, this should be 3, and this should be 5. As we can see, with respect to this flow graph, if I draw the residual graph, there is no s to t path, and hence this is the maximum flow.

Okay, so now let us see the execution of Dinic's algorithm. So, let us again consider the same input graph. In the Linux algorithm, we need to construct the layered graph. So, S is in L_0, then A and C are distance 1 away from S. So, they are in L_1. B and D are distance 2 away, they are in L_2 and T is in L_3.

Layered graph has only forward edges, so S to A edge. This is the residual graph also in the first iteration. we ignore the cross edges. So, A to C edge is between because both the endpoints are in the same layer, we do not keep those edges in the layered graph. So this is how the layered graph look like.

Now what I will do? I will find the blocking flow. find a path we can use the naive algorithm also naive greedy algorithm to find a blocking flow. So, let us take any s to t path all are shortest paths. So, may be s to a to b to t and so the flow value becomes

I can send at most 3 units of flow. So, let us send 3 units of flow in all these edges. This makes the A to B edge saturated. Let us look for another path from S to T where every edge is not saturated. There indeed exists such a path: S to A to D to T.

And the maximum amount that I can send is 2 units along this path. So, I make this flow on S to A 5, this is 2, this is 2. Now, I am again looking for an S to T path where all edges are not saturated. So, S to C to D to T, and I can send one unit of flow. So, how does the flow graph look like? Or let us draw the residual graph—that will be easier.

G_f^1 S A S to A is saturated. So, I have a reverse edge. A to B is saturated. This is how the residual graph looks like after the first iteration. Now, I need to draw the layered residual graph.

So, $G_f^L(1)$, S, you see distance of A has increased. Actually, A seems unreachable. this a to c was on the other direction sorry the direction of this edge was from c to a yeah so now you see the distance of c a has increased from 1 to 2 only c is at distance 1 A is at distance 2 and B and D, not B, D is at distance 3. and t is at distance 4 and b is at distance 5 and actually we can delete the all the levels beyond the level where t belongs.

Now let us draw the edges. Only forward edges I will draw. S to C there is an edge of capacity 6, C to A edge of capacity 2, A to D is an edge of capacity 2 and D to T is an edge of capacity 9. So, there is only one path in this layered graph from S to T and I can send 2 units of flow along this path and that will be the maximum flow as we have seen because currently we are sending 6 units of flow after first iteration and in the second iteration I will send 2 more units of flow from S to T resulting in a total of 8 units of flow from S to T. which is the maximum.

So, we see that the number of iterations in Dinic's algorithm is substantially smaller than the number of iterations in the other two algorithms. So, let us stop here. Thank you.