Second Level Algorithms

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 05

Lecture 24

Welcome to the 24th lecture of second level algorithm course. In the last lecture we have started the study of push to level algorithm. The high level idea is to allow some excess flow at various vertices and in every iteration push the excess flow to its neighboring vertex. Now, as it is obvious that if we do not push the excess flow systematically the flow may end up circulating in the network. So, we will see how using the notion of height of the vertices we can make the pushing of excess flow thereby ensuring that we are making progress towards making the preflow into a flow. So, let us begin. So, we introduce a height function for every vertex which is a natural number including 0. The push-reliable algorithm maintains the following invariants. It's an iterative algorithm.

the first one is height of s the source vertex is always n what was n let us recall it is the number of vertices the second one is height of sink is always 0 and the third one is whenever we have a edge from u to v in the residual graph G_f, the height of v must be at least $h(u)-1$. For every edge u v in the edge set of the residual graph, height of v must be greater than equal to $h(u)-1$. That means if I look at from height

Even if I have an edge going out of u, then either the edge can go in the same level—the other endpoint of the edge could be at the same level—or it could be some level higher. But if it is down, it can be down but not down by more than one level. So, if there is an edge from u to v, the height of v should be at least the $h(u)-1$. In particular, I cannot have an edge from u to some vertex whose height is more than 1 less than the height of u. Such kinds of edges are not allowed, okay? So, these invariants we will maintain, okay?

So, again, it is an iterative algorithm, and so let us see the initialization. And let us recall what the invariant was. The invariant we will maintain is these three invariants. But before that, let us see why ensuring these three invariants is enough—that in the

corresponding residual graph, there is no path from S to T. So, let us first prove that. Here is a claim.

If invariants 1, 2, and 3 hold then there is no path from s to t in the residual graph G_f proof. It is a proof by contradiction. So, if possible, let us assume that there is a path.

let us give it a name to the path it is a sequence of vertices $s, u_1, u_2, \ldots, u_k$ ok which is equal to t from s to t in G_f. So, if possible let us assume there is a path from s to t and because it is a simple path, the length of the path k is less than equal to $n-1$, where n is the number of vertices. Now, because of the invariants. 1, 2 and 3 hold. Height of S is n. What could be the minimum height of u_1?

Because there is a edge from S to u_1, the minimum height of u_1 is $n-1$. So, height of u_1 is at least $n-1$ because the edges can either go up or be at the same level or if it goes down it can goes down by only one level. So, for this reason height of $u_1$is at least $n-1$. Height of u_2 similarly is at least $n-2$ because there is an edge from u_1 to u_2. Hence, height of u_k is greater than equal to $n-k$, but k is at most $n-1$. So, this is greater than equal to $n-(n-1)$ which is equal to 1, but height of u_k because u_k is t is height of t which is 0 from our second invariant. So, on the one hand, height of u_k is at least 1, on the other hand, height of u_k is 0.

So, this is a contradiction. Okay. So, this shows that if we maintain invariants 1, 2, and 3, then there is no path from s to t. So, if the flow f is a flow, then that flow must be an optimal maximum s to t flow. Because we are maintaining the optimality condition that there is no path from s to t in the residual graph G_f. Let us see the push-relabel algorithm.

It is an iterative algorithm. So, we will initialize the pre-flow in such a way that the invariants hold, and then in every iteration, we work towards making the pre-flow a flow. So, initialization. For flow augmentation-based algorithms, we initialize the flow to zero for every edge. But that initialization to zero flow violates the invariant.

How? First, before initializing the flow, we have to say what we are initializing the height function to. So, initialize the height of s to be n. This is required by the first condition of the invariant, and the height of every other vertex including t is 0. So, this initialization of heights ensures that invariants 1 and 2 are satisfied, but the third invariant is not satisfied by this sort of initialization. The third invariant fails to hold for every edge going out from S, ok, because in G_f. All the edges which are going out of S in G have full residual

capacity. Hence, those edges are present in the residual graph, but these edges—the height of the source is n, but the height of the other endpoint of all these edges is 0.

So, the edge is going from height n to height 0; hence, this invariant is failing unless n equals 1, ok. So, this invariant, in general, is not holding. So, to tackle this problem, what we will do is initialize the flow values of all the edges going out of S to be full.

So that the edges going out of S do not have any residual capacity and thus they are not in G_f. So, we initialize flow value not to all zero—it is the capacity of the edge if e leaves S and zero otherwise. Now, let us see whether the invariants are satisfied or not. Of course, the first invariant and second invariant are satisfied because the first invariant says the height of s should be n, which is what we are setting.

The second invariant says the height of t should be 0, which is what we are saying. The third invariant says that edges can go only upward or stay at the same level, and if an edge goes downward, it can go down by only one height. So now, you see in G_f, the edges involving S—all those edges are going into S. They are going from level 0 (or height 0) to height n, which is allowed under invariant 3. For every other edge, they are going from height 0 to height 0, which is also allowed under invariant 3. So, this particular initialization of preflow guarantees that the three invariants hold true.

This initialization of preflow satisfies all three invariants, okay. So now, let us describe what we do in every iteration. We introduce the height function to systematize the push operation. In the push operation, we only push flow from a high vertex to a low vertex—that is what we mean by systematic push.

So, we push excess flow from a high vertex to a low vertex only. This is what we mean by systematizing the pushing of excess flow using the height function. So now, let us write down the pseudocode of the push-relabel algorithm. First is initialization: $h(s)$ is n, $h(v)=0$ for all $v \in V \setminus \{s\}$, $f_e = c_e$ for every edge e leaving s and f equal to 0 for every other. So, this is the initialization. Now, we will run an iterative procedure until f is a flow. So, while f is not a flow. That means what?

That is, there exists a vertex $v \in V \setminus \{s, t\}$. Such that there is an excess flow at v, $\alpha_f(v) \geq 0$. In this case, we choose a vertex v which is at the highest. Intuitively, it makes sense because we will allow push only from a high vertex to a low vertex, but in the analysis, we will see why we are choosing a vertex v which has excess flow and among all such vertices, it is the vertex with maximum height.

So, choose among all vertices having a max, having an excess, a vertex v of maximum height, okay. Now, we have to check whether there exists a neighbor of v whose height is 1 less than the height of v. If there is an outgoing edge vw in the residual graph such that

$h(v)=h(w)+1$, that means vw edge is a downward edge. And for downward edges, the height difference can be 1 only. If there is such an edge, then we perform a push operation. Otherwise, that means all the neighbors of v are either at the same level of v or above; then we increment $h(v)$, and that operation is called relabel.

This operation is called relabel, and that is it—that is the end of the while loop. Then return f. Let us see the push operation. Which is called from the while loop: choose an edge vw in the residual graph such that the $h(v)=h(w)+1$. The existence of such an edge is guaranteed in the checking of this if condition. So, whenever we call push, there it is guaranteed that there is an edge vw which is going downhill. Okay, now next we have to find how much we can push. There are two bottlenecks: one is how much excess we have at v—that much you can push; the other bottleneck is the capacity of the edge along which we are pushing. So the minimum of the two is the maximum excess flow at v which we can push. So let δ equal to the minimum of $\alpha_f(v)$ and the residual or the capacity of vw. Okay, and then we push delta unit of flow along that path, along the edge vw. What does push mean? Pushing delta unit of flow—recall that vw is an edge in the residual graph.

So, if it is a forward edge, then we increase the flow value of the forward edge by vw by delta, and if it is a backward edge, then we reduce the flow value of the forward edge wv by delta. That is what we mean by pushing. So this is the push-relabel algorithm. It is not clear why it is doing what it is doing, how it is making progress, or whether it will terminate at all. So these questions we will answer in the next lecture. So let us stop here.

Thank you.