Second Level Algorithms

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 06

Lecture 28

Welcome to the 28th lecture of second level algorithms course from the last couple of lectures we have been proving the correctness of push reliable algorithm in the last class we have bounded the number of relabel operations to be $2n^2$ and the number of saturating pushes to be at most $2mn$ in this lecture we will bound the number of non-saturating pushes okay so let's begin So, here is the key lemma that we will prove to bound the number of non-saturating pushes between two relabel operations. between two subsequent. There can be at most n non-saturating pushes.

Proof, till now we have not used the fact that our algorithm picks among all vertices having an excess flow the vertex with maximum height so that we will use here to bound the number of non-saturating pushes the push-reliable algorithm picks a highest height vertex among the vertices having positive axis to perform push or relabel operation in every iteration.

and after non-saturating push from a vertex v that vertex does not have any excess flow. Observe that after performing a non-saturating push from a vertex v that vertex we cease to have an excess flow. Because there are at most n vertices

More importantly, because v was the vertex of highest height, so not only v does not have an excess flow, also every vertex of height more than v, more than the height of v does not have an excess flow after a saturating push from v until a reliable operation is performed ok, because only in the reliable operation the height of any vertex changes and because we allow push only through downhill edge this is so, because only reliable operation increases or increase the height of any vertex and flow is pushed along downhill edges only. okay but the number of vertices is n so this implies that the number of non saturating push operations between any two subsequent reliable operations is at

most n. So, this proves the lemma and this gives the bound on the number of non-saturating pushes as a corollary.

So, we can derive the following corollary. The number of non-saturating pushes in any execution of the push reliable algorithm is at most to n cube. Proof, the total number of reliable operations

In any run of the push-relabel algorithm, it is at most to n squared, and we have seen that between two consecutive relabel operations, there can be at most n non-saturating pushes, and there can be at most n non-saturating pushes. Non-saturating pushes between any two subsequent relabel operations. Thus, the total number of non-saturating pushes is at most $2n^3$.

So, we have shown that the total number of relabel operations is at most $2n^2$, the total number of non-saturating pushes is at most $2n^3$, and the total number of saturating pushes is at most $2mn$. So, this shows that the total number of iterations of the push-relabel algorithm is less than or equal to the sum of all three operations because, in every iteration, we are performing either a relabel or a push operation. Hence, the push-relabel algorithm always terminates. We have already seen that if the push-relabel algorithm terminates then the output is a maximum s-t flow. Hence, the push-relabel algorithm is correct. That means, for every input instance, it correctly computes a maximum s-t flow. Good. Now, let us focus on the time complexity of the push-relabel algorithm. So, if we maintain the height information in an array of size n, then relabel takes $O(1)$ time. Okay, because we only need to increase the value of the correct index by 1, which is relabel.

If we maintain the flow values of all the edges in a 2D array of size $n \times n$, then push operation which involves changing the flow value of one edge takes $O(1)$. Now, in every iteration we have to quickly find out if there exist a vertex which has a positive axis.

and among all such vertices a vertex of maximum height. Actually there exists a data structure not very difficult to design which I give as a homework. Design a data structure that finds a vertex of maximum height among the vertices having positive axis in $O(1)$ amortized time.

Now because of this data structure we can find a vertex of maximum height among all vertices having a positive axis in $O(1)$ time and then we can also find and find a downhill of that vertex of course, if exists in $O(1)$ amortized time see then the subsequent push or reliable operations take $O(1)$ time. So, every iteration algorithm takes $O(1)$ amortized

time and the number of iterations is $O(n^3)$. Hence, the worst-case time complexity of push-relabel algorithm is $O(n^3)$. Kindly note that this time complexity is the worst case. This follows from the definition of amortized time complexity. Any sequence of k operations takes $O(k)$ time if the amortized time complexity of each operation is $O(1)$.

So, any sequence of $O(n^3)$ operations takes $O(n^3)$ time in the worst case if the amortized time complexity of every operation is $O(1)$. And we have seen the worst-case time complexity of push and relabel, both of them is $O(1)$. Hence, the worst-case time complexity of the push-relabel algorithm is $O(n^3)$. So, this finishes our discussion on the push-relabel algorithm, the pseudocode, its working example, the analysis, proof of correctness, and time complexity. From the next lecture onward, we will see many applications of this maximum flow problem in various other important domains, often seemingly unrelated at the surface. So, let us stop here. Thank you.