Second Level Algorithms

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 09

Lecture 41

Welcome to the 41st lecture of the second-level algorithms course. In the last lecture, we started seeing the Karger-Stein algorithm. In this lecture, we will do the error analysis of that algorithm. So, let us begin. This is the theorem that we will prove today. For any min-cut F, the probability that one run of the Karger-Stein algorithm outputs F is $\Omega(\frac{1}{\log n})$ proof. Let us draw the recursion tree of the Karger-Stein algorithm. The root node has n, as it was called with n vertices. Then the recursive call is made with roughly $\frac{n}{\sqrt{2}}$ vertices and so on.

The leaf nodes have at most 2 vertices. The height of the recursion tree is $\log_{\sqrt{2}} n$. The crucial observation is that the Karger-Stein algorithm outputs F if no edge of F is picked for contraction at this root node and at least one of its children does not touch F or does not pick any edge from F for contraction.

So, let us write we observe that the Karger Stein algorithm outputs F if and only if no edge from F is contracted within the first t iterations and at least one of the two recursive calls succeed. So, actually this will be easier if we instead of if we insist of outputting a min cut instead of outputting a particular min cut F. So, this we will now prove the theorem by recursion on the height of the recursion tree.

We prove the following claim. Let P_d be the minimum probability of success for any recursion tree at least d or at most d not at least at most d. So, with this P_d we claim that $P_d \geq \frac{1}{d+2}$ for every $d \geq 0$ ok.

So obviously, P_0 is 1 and hence the claim holds for d equal to 0. So, the base case is fine. Now, the inductive step. So, inductively let us assume that $P_{d-1} \geq \frac{1}{d+1}$. So, let us assume this for $d-1$ and we will prove for d. So, what is P_d?

This is the probability that it outputs the a min cut f and first to output the min cut f 2 things needs to happen. First no h from f should be contracted within the first 3 iterations that probability is by the choice of t is at least half and the second condition is that at least one of the two recursive calls must succeed. So, the probability what is the probability that at least one of the two recursive calls succeed? It is 1 minus the probability that both the recursive calls fail.

So, the probability that both the recursive calls fails is $1-p$ whole square because they are independent. Let us simplify it. Now, this is an increasing function of P_{d-1}. So, this expression is greater than equal to if I replace P_{d-1} with $\frac{1}{d+1}$. So, this is greater than equal to $\frac{1}{d+1}$ minus to this. This is since $x-x^2/2$ is an increasing function for x in between 0 and 1. So, this is now greater than equal to $\frac{1}{d+1}$ minus times $d+2$, but this is $\frac{(d+1)}{(d+1)(d+2)}$ which is our desired bound.

So, this proves the claim. Now, recall the height of the recursion tree is log of n base root 2. So, the success probability of one run of Karger-Stein algorithm is at least this bound, which is what we have to show.

So, this finishes our description of Karger-Stein algorithm. The overall runtime of Karger-Stein algorithm is $O(n^2 \log^2 n)$. Contrast this with the overall runtime of Karger's algorithm, which is n^4, OK. Start our next topic, which is a very beautiful algorithm for finding a maximum matching in a general graph. We have seen that we can find a maximum matching in a bipartite graph using a maximum flow algorithm, but how about a general graph? So, let us recall the problem statement: input is an arbitrary unweighted graph G, which need not be a bipartite one, and we have to output a matching M of G, OK. A matching is, recall, a set of edges sharing no endpoint. Not any matching—maximum cardinality matching, OK. So, for that we need to introduce a concept of an M-augmenting path, where M is a matching.

Let M be any matching in G. Then, what is an M-augmenting path? It is a path, say $u_0, u_1, u_2, \ldots, u_k$ in G, such that both the endpoints u_0 and u_k are unmatched in M, and the second one is the alternate edges belong to matching and toggle between one from matching and one not from matching. That means, if you look at the edges from u_{2i} and u_{2i+1}, then they are not part of the matching. So, first observe that the condition that u0 and uk must be unmatched vertices means this path length must be odd, that means k must be an odd number. So, this is k/2 flow. So, these edges alternate: the first edge, the third edge, the fifth edge, and so on, they should not be from the matching, and the other edges should be from the matching.

OK. That means, pictorially, if I draw the path u_0, u_1, u_2, So, the first edge must be a non-matching edge, the third edge must be a non-matching edge, and so on. So, the last edge, the other edges must be matching edges. So, these are matching edges.

Whereas the other edges are non-matching edges, ok. What is the significance of such a path? The significance of such a path lies in the fact that if we get hold of such a path, we can use that path to get another matching of size more than 1. How? So, using M, you define another matching $M^{'}$ where from M you remove the edges from the path P. So, let us call this the path $P \cap M$, the matching edges in the path that you remove, and include the non-matching edges from the path into the matching. Then you see this is a valid matching because the endpoints of the path u_0 and u_k are non-matching edges, and hence no two endpoints of the edges in $M^{'}$ share an endpoint. So, $M^{'}$ is also a matching. Moreover, more importantly, the cardinality of $M^{'}$ is the same as the cardinality of M plus So, its cardinality is 1 more than the cardinality of M. We are able to get hold of another matching of size 1 more than the size of the matching we started with, ok. So, if there exists an M-augmenting path P, then we can use P to construct another matching of size cardinality M plus 1. In particular, existence of an m augmenting path shows that m is not a maximum matching. Is the converse also true? We will show this we will show in the next lecture that if there is no m augmenting path, then m must be a maximum cardinality matching.

That means existence of an M augmenting path characterizes whether M is a maximum matching or not. So, if M is not a maximum matching then there must exist an M augmenting path all we have to do we have to find an M augmenting path efficiently computationally efficiently. And then once we get hold of the M augmenting path we can use that path to get another matching of size 1 more than the current matching M and then we repeat this procedure. And, that is the high level idea of the algorithm for computing maximum cardinality matching in a general graph.

So, in the next lecture we will prove first we will prove this claim that an maximum and matching M is a maximum cardinality matching if and only if if and only if there is no m augmenting path p and then once we prove this theorem we will use this to design our polynomial time algorithm for computing a maximum cardinality matching ok. So, let us stop here. Thank you.