Second Level Algorithms

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 10

Lecture 46

Welcome to the 46th lecture of the second-level algorithm course. In the last lecture, we started proving the correctness of the Edmond's-Blossom algorithm. So, let us finish that proof of correctness in this lecture. So, if you recall, the Edmond's-Blossom algorithm uses a subroutine which, given a matching M, either concludes whether it is a maximum cardinality matching or outputs another matching M' of the same cardinality as M and an M'-augmenting path. So, it is enough to show the proof of correctness of that subroutine.

So, the theorem, in particular, that we will prove is that the Edmond's-Blossom algorithm is correct. It is enough to prove that or prove the correctness of the subroutine which takes a matching M as input. and either outputs a matching M' with the cardinality of M' being the same as the cardinality of M and an M'-augmenting path or concludes that the given matching M is a maximum cardinality matching, okay. So, this we prove by induction on the cardinality of M. The procedure is recursive, as we have seen.

So, the base case is if the cardinality of M is 0, then every vertex is marked even at the beginning of the modified breadth-first search. Hence, if the graph G has any edge, then an M-augmenting path, which is any edge in this case, is returned by the algorithm. On the other hand, if the graph is an edgeless graph, then the current matching M, which is an empty matching, is clearly the maximum cardinality matching. Hence, if the input matching is an empty matching, then the subroutine outputs correctly. Now, the inductive step. Assume that the subroutine returns the correct output for all input matchings M with cardinality at most k. We will prove correctness for all input matchings.

of cardinality at most $k+1$. Now, let us recall what the algorithm was doing. The algorithm was performing a breadth-first search traversal—a modified breadth-first search traversal—and it can either find an augmenting path or a blossom, or it reports that the current matching M is the maximum cardinality matching. So, in all three cases, we

will show that the algorithm is correct. If an M-augmenting path is discovered during modified breadth-first search, then the algorithm or the subroutine is clearly correct.

This is so because if there exists an M-augmenting path, then the current matching M cannot be a maximum cardinality matching. This we have proved before. And in this case, the subroutine is supposed to return another matching—or a matching M' of the same cardinality as M—and an M'-augmenting path. So, in this case, the matching M' is the same as the matching M. So, this case is fine. Case 2: Suppose the algorithm discovers a blossom during modified breadth-first search. In this case, the subroutine replaces M with another matching M' of the same cardinality. So, this is not a blossom; this is, in general, a flower. To get rid of the stem of the flower, OK.

So, it gets rid of that steam of the flour. And it is left with the blossom, then it calls itself recursively on graph G/B with matching M/B where B is the blossom that the subroutine has discovered. If the subroutine returns an M/B augmenting path P in G/B, then we have seen in the last lecture that we can compute an M augmenting path Q in G from P in polynomial time. Hence, in this case the algorithm or the subroutine is correct.

On the other hand, if the subroutine returns that M/B is a maximum cardinality matching in G/B, which is correct by induction hypothesis. Then we have seen in the last lecture that M is also an M augmenting path—sorry, M is also a maximum cardinality matching in G. Hence, the subroutine is correct in this case also. Now, the third case: the subroutine returns that M is a maximum cardinality matching because the Q has become empty. So, in this case, let us see how the modified breadth-first search forest looks like.

It starts with nodes marked even. So, this is how a modified breadth-first search tree may look like. Now, what are the edges of the graph? There are 3 types of edges in the graph.

The first one: edges present in the modified breadth-first search forest. The second one are edges discovered during the modified breadth-first search, but because they are between even vertices and odd vertices, they were ignored. Edges discovered during modified breadth-first search. but ignored because they are between vertices already.

Marked even and odd. And the third type of edges are edges not discovered. During modified breadth-first search, OK. So, how come those edges are there, for example? If a connected component of the graph is perfectly matched, that means all the vertices are matched, then no edge in that connected component will be discovered during modified breadth-first search. So, in the next lecture, we will use these three types of edges to

construct a set of vertices which proves that the current matching M is a maximum cardinality matching, OK. So, let us stop here. Thank you.