Second Level Algorithms

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 10

Lecture 47

Welcome to the 47th lecture of second level algorithm course. In the last two lectures we have been doing the proof of correctness of Edmond's algorithm. So, let us finish that proof of correctness in this lecture. We were in the third case when because the empty because the queue became empty during the modified breadth first search traversal the subroutine returned that the current matching M is a maximum cardinality matching. We were proving we were in the process of proving that the step is correct for that we have the edge set of the graph into 3 sets. The first one are visible in the depth in the modified breadth first search forest that we have built. Here is a schematic diagram of that modified breadth first search forest. These are type 1 edge type 2 edges are which were discovered during the modified best first search traversal, but they were ignored because they were between an even level and an odd level. vertices marked between even and odd.

So, these are type 2 and type 3 are they were not discovered at all. So, these are type 1. Now, we consider the set let us call it O of vertices marked odd. A crucial observation

Is that the vertices or every vertex? Marked even has neighbor marked odd only. Let E_1 be the subset of vertices marked even, then because every vertex in E_1 has neighbors marked odd, then every vertex becomes an isolated vertex in the induced graph $G[V \setminus O]$. So, if I remove all the vertices marked odd, then all the vertices marked even become isolated vertices.

Hence, the number of odd components in the induced graph $V \setminus O$ is at least the number of vertices in the even component. Now, what is the size of the matching? The size of the matching is in this depth, in this modified breadth-first search forest, the number of matching edges is the number of vertices which are marked odd.

So, this is the cardinality of O plus there are type 3 edges which are edges not discovered during the modified breadth-first search traversal, as we have discussed in the last class. All the vertices and the corresponding edges that were not discovered in this modified breadth-first search traversal should be perfectly matched. So, the cardinality of M is the number of vertices marked odd plus the other vertices, which is the cardinality of V. So, what is the cardinality of the remaining vertices? Here, the the cardinality of the remaining vertices is the cardinality of V minus the cardinality of $E_1 \cup O$ because all the vertices in the modified BFS traversal are marked either even or odd. So, this is the number of vertices not discovered. During the modified BFS traversal, all of them should be perfectly matched. So, the number of matching edges on these vertices is half of them. So, this is the cardinality of M. Let us simplify it.

Let us take half common. So, this is the cardinality of the matching M. Now, we use this inequality 1. So, from 1, what we get is The cardinality of M is at least. Which can be written as

OK. On the other hand, for any subset of the vertices, this right-hand side is a lower bound. So, from the other direction of That we have already proved for any matching M', we have the cardinality of M is at most half the cardinality of V.

Minus the number of odd components in $V \setminus O$ minus the cardinality of O. Actually, let us write it for S because this is true for every subset of vertices. So, in particular, This is less than or equal to for the particular subset S being O, OK. The cardinality of any matching is at most the cardinality of the matching M. Hence, M is a maximum cardinality matching in G. This concludes the proof of correctness.

of the subroutine and thus the proof of correctness of the Edmond's blossom algorithm, OK. So, this proof also shows the following. That for a maximum cardinality matching M of G, the cardinality of M is equal to the Tutte-Berge bound, which is equal to the minimum over all subsets of V this. So, this proves the Tutte-Berge theorem, OK. Analyze the runtime of the algorithm. So, in the Edmond's Blossom algorithm, we were calling a subroutine which either returns an M-augmenting path or it returns that the current matching M is a maximum cardinality matching. If we get an augmenting path, we use that augmenting path to get another matching of size 1 more than the current matching.

and hence in every iteration the size of the current matching increases by 1 that is the important observation. Observe that the size of the matching increases by 1 in every iteration of the Edmund Blossom algorithm. Hence the number of iterations is at most $n/2$. Now, let us see what is the time complexity of the subroutine which either finds an augmenting path or reports that the current matching is a maximum cardinality matching. performs a modified breadth for search. that takes $O(m+n)$ time. The number of the depth of recursion or recursion is at most the cardinality of the matching M because whenever we perform a recursive call we are passing the matching M/B where B is the blossom as a parameter and the cardinality of M/B is strictly less than cardinality of M. now the cardinality of m can be at most $n/2$ and hence the number of recursive call is at most n by 2. Hence the time complexity of the subroutine is $O((m+n)\times n)$. This subroutine is called at most $n/2$ times as we have argued before.

Hence, the time complexity of Edmond's' Blossom algorithm is $O((m+n)\times n^2)$. So, this finishes our time complexity analysis of Edmond's' Blossom algorithm. To wrap up the Edmond's' Blossom algorithm, let us see an overview or the pseudocode of Edmond's' Blossom algorithm. So, it returns with an empty matching.

Now, let us perform a do-while loop. Call the subroutine—let us give it a name—call it augment or call this subroutine. If it returns a matching M' and an augmenting path, M' augmenting path P, then replace M with M' symmetric difference P; otherwise, return M as a maximum cardinality matching of G. So, that is it.

So, make it an infinite loop. So, this overall algorithm is as simple, and in the next class, we will see the pseudocode of the augment subroutine, which takes the graph G and a matching M as input and either finds a matching M' of cardinality same as M and an M' augmenting path P or returns or correctly concludes that M is the maximum cardinality matching. So, let us stop here. Thank you.