Lecture 17: Graph Algorithms: Matchings

Lecturer: *Sundar Vishwanathan*
Computer Science & Engineering Indian Institute of Technology, Bombay

In this lecture we reviewed old material. Mostly data structures. We noted that graphs will be represented by adjacency lists for this course.

We then defined a matching, given below.

1 Matching

Definition 1 *A* matching in a graph is a set of edges, no two of which share an end-point.

Our interest is the following problem:
Input: Graph G.
Output: Find a maximum cardinality matching in G.

We wish to design an algorithm (polynomial time) for this problem.

How do we begin our design? The only real design technique is induction, which you should try out first. We will pass this for the time being.

One way is to start with some matching and then see if you can improve this. For the greedy method, you would have tried some kind of local exchanges. They do not quite work here.

Another thing you can try is to consider what happens when you look at a union of your current matching M, and an optimal matching M_0. What does the resultant graph look like? I would advocate that you try answering the question as much as you can before reading ahead.

What can we say about the degree of each vertex? It has to be either zero or one or two. Since at most one edge of M and at most one edge of M_0 can be incident to any vertex. What then are the connected components of such a graph?

It is easy to see that they are either cycles or paths. In either case, the edges will alternate between M and M_0.

Given this, then we can say the cycles will have even length. What about the paths?

They can be characterised by which matching the first and last edges lie in.

It is possible that the path is a single edge which is in both M and M_0. We ignore such edges.

It is possible that the first is in M and the last is in M_0. It is also possible that both the first and last are in M_0.

It is not possible that both are in M. Why? The reason is that in this path if we interchange the edges in M and M_0 we get a matching which is larger in size than M_0 which is not possible. Make sure you understand this!

What other observations can you make? The crucial one is that if the matching is not optimal then there has to be one of the last type: edges at both the ends are in M_0. The reason is simple; in all other connected components, the number of edges from M and M_0 are equal. And if M is not optimal there has to be at least one connected component where M_0 has more edges than M.

Now you have all ingredients to design an algorithm. Can you do it?

The idea is to have an iterative algorithm. Start with an empty matching and at an iterative step increase the size if possible. If the matching is not maximal, we know that there is a path in the graph which starts and ends at vertices that do not have matched edges incident on them and in the path

matched and unmatched edges alternate. (Note that the edges in M_0 in the path we say must exist in the earlier discussion are not in M.) We find such a path and increase the size of matching. How? By a trick we have seen earlier: exchange the matched and unmatched edges on this path. Why does this increase the size of the matching?