Maximum Matching in Bipartite Graphs Name : Sasidhar Evuru

Abstract - The main aim of the project is to find the maximum matching inside a bipartite graph.

Problem Statement – Given an undirected graph find whether the given graph is a bipartite graph and if the given graph is bipartite the find the maximum matching in the graph. Print the number of maximum matching's and the time taken. If the number of vertices is <= 100 print the vertices in the increasing order with their matched vertex.

Implementation – The Algorithm which is implemented is a slight tweak of the algorithm which is discussed in the class.

1. Using BFS we will verify if the given graph is bipartite or not. If the graph is bipartite we proceed further. We set the inner and outer properties of the vertices while verifying for the bipartite in the BFS itself.
2. We will run the maximal matching initially in order to get the initial maximal matching which is a greedy way of allocation the matching.
3. After that we will call the find maximum matching where we will reset the properties of all the vertices.
4. We will iterate through the vertices list and queue the free outer nodes.
5. We will get the adjacency list of the vertices inside the queue and process the vertices only if the vertices are not seen before.
6. If we come across the augmenting path instead of going to start again the algorithm was tweaked to run till the current queue is emptied and we will go to the start of the while(true) loop only after the queue is finished and we found some augmenting oath while emptying the queue.
7. This will prevent the algorithm from going back to the start step every time a new augmenting path was found and helps running the algorithm effectively.

Functions implemented:

1. FindmaximalMatching: this function return the graph with maximal matching.
2. Isbipartite: returns whether the given graph is bipartite.
3. FindmaxMatching: finding the maximum matching after finding the maximal matching.
4. ProcessAugPath: found the augmenting path and we are processing the augmenting path.

Test results – The test cases are all tested with the given inputs and got correct output for all the cases. The following is the table with the given input list and the running time which is achieved by running this algorithm.

Input File	Output	Runtime in msec
matching-100	35	0
matching-1k	994	15
matching-10k	9924	125
matching-100k	99238	1109
p9-in-1m-10m	398263	19828

References:

1. Lecture slides for the implementation of the algorithm.