LP5

G16: Ian Laurain & Vasudev Ravindran

CS6301: IADSA

**Introduction:**

This project required the student to implement algorithms to solve three graph algorithms to find the maximum cardinality matching in bipartite and general graphs. The project was divided into three levels. For level one, the student needed to implement the algorithm to find the maximum cardinality matching in non-weighted bipartite graphs. Level two required the student to find the maximum cardinality matching in general graphs. Level three required the student to find the maximum weight matchings in bipartite graphs.

**Methodology:**

Level-1:

The level-1 implementation closely followed the pseudocode for the algorithm to find the maximum cardinality matching in a bipartite graph, introduced in class. This algorithm, as implemented by strictly following the pseudocode, worked quite well for the provides bipartite test cases 1-3 (bip1.txt, bip2.txt, & bip3.txt). However, for the fourth test case (bip4.txt) the runtime took about 3-5 minutes to complete, depending on the machine it was run on. This was far to slow. An optimization was necessary. There was a hint in the assignment description document that an optimization would be necessary, but that a reduction to the max-flow problem was not allowed, Therefore the student looked for ways to optimize the runtime of the program. This optimization required a slight change to the provided pseudocode to the algorithm. Instead of maintaining one queue, two queues were maintained. One queue contained the free outer nodes, while the other queue handled the actual augmenting path logic of the. Basically, all the free outer nodes were processed in one iteration. These lead to drastic improvement in the runtime of the fourth test case (from 3-5 minutes, down to less than 1 second).

**Analysis:**

Please see readme.txt file for analysis of runtime results and an explanation of how to run the submitted programs.