**Group 22**: Divyanshu Paliwal

Radhika Simant

Aditya Mahajan

Shweta Nair

**LP5 Level1 Report**

**Algorithm:**

We have used the algorithm discussed in class for bipartite-matching. The algorithm initially gave wrong results for higher inputs. To overcome this issue, we came up with a parallel processing approach. In this we have made a few changes in the existing *processAugmentingPath()* method. The idea is to check if we have any existing frozen node in the path from the free inner node vertex from which we start the augmenting process, till its parent. If we find such a path, we don’t have to process that path. Only if we don’t find a frozen node, then we go on to process the augmenting path. While traversing the augmenting path, we keep on freezing the nodes.

Also we check for the count of free inner nodes. If the count of the free inner nodes is zero, that means we were not able to reach the free inner node through the free outer node in process, we break from the loop and the program terminates.

The results obtained after running the sample inputs on the level 1 are as below:

* Sample input 1: bip1.txt

Output: **35**

Time taken: 2 msec

* Sample input 2: bip2.txt

Output: **994**

Time taken: 37 msec

* Sample input 3: bip3.txt

Output: **9924**

Time taken: 167 msec

* Sample input 4: bip4.txt

Output: **99238**

Time taken: 4866 msec