Alex Kwong

Scheduling Description

The goal of this problem was to determine if it were possible to schedule students in their desired courses given certain constraints.

To approach this problem, I read in the inputs of the graph and used an adjacency matrix to represent the graph. The graph was comprised of 4 layers where the first layer contained a single source node, the second layer contained the student nodes, the third layer contained the course nodes, and the fourth layer contained a single sink node. By doing so, the problem can be converted into a network flow problem which can be solved by using the Ford-Fulkerson algorithm.

The capacity between the sink node and a student node was n and the connections between a student node and course node was determined by the registration requests where the capacity was set to 1 which represents whether a course is chosen or not chosen. The capacity between a course and the sink node was set to the course's capacity which was kept track of using a dictionary. Once the adjacency matrix was finished being initialized which took $\Theta(V^2)$ time, Ford-Fulkerson was run and returned the max flow and the resulting graph. The implementation of the Ford-Fulkerson algorithm used runs in $\Theta(VE^2)$.

I initialized 2 Boolean variables named validenrollment and validcapacity and initialized them to true. Once the algorithm completed finishing, I checked if students were enrolled in exactly n classes, and if not, set validenrollement to false. I also checked the amount of students enrolled in each class, and if the number was above the class capacity, then I set validcapacity to false. Finally, I checked if validcapacity was true, validenrollment was true, and if max flow was equal to the number of students * n. If all conditions returned true, then "Yes" was printed, otherwise "No" was printed. This program overall runs in $\Theta(VE^2)$ time as the Ford-Fulkerson algorithm dominates the program's runtime.