Drew Goldman

Scheduling- Description

The Scheduling problem requires a relatively simple reduction of the problem into the maximum flow problem. Indeed, the solution is as simple as taking all the students and courses and creating a graph, represented as an adjacency matrix, where each student and each class receives its own row in the graph. The edges are subsequently filled in under three categories. The first of which is from the source node (artificially created) to the student nodes, where the capacity of each edge = n, the number of classes each student is to be enrolled in. The second of which is from the student nodes to the class nodes, in which case every edge request has a capacity of 1: either the student can take the course or he/she cannot. The third of which is from the courses to the sink node (artificially created), where each edge capacity represents the total capacity for each course. Once this adjacency matrix is created, after indexing each node with a unique number and setting up the array based on the values of the edges, the algorithm simply runs Ford Fulkerson on the graph. If the maximum flow outputted by Ford Fulkerson equals s*n, the number of students * the number of classes each student is to be enrolled in, then the schedule can indeed be created and the algorithm prints "Yes;" else, the algorithm prints "No." **The runtime of this algorithm is $O(E*V^3)$** because the runtime of breadFirstSearch() is $O(V^2)$ and breadthFirstSearch is called a total of E*V times within Ford Fulkerson for each test case *t*.

No pseudocode required, as the code is available to the grader via Gradescope.