CS5820 HW7

Renhao Lu, NetID: rl839

**1. Scheduling interviews – Part II**

Reduce to maximum flow problem:

**Step 1**: From the input of scheduling interviews problem, create input for maximum flow problem

Create a directed graph ,

, Where , ,

.

The edge cost:

**Step 2**: Show the output of maximum flow problem give us the output of scheduling interviews problem

If the maximum flow < n, then there don’t exist a schedule that satisfies all the restrictions. If the maximum flow = n, then we will consider all edges . If the flow of any is 1, then we assign the candidate to recruiter at time slot .

**Step 3**: Running time

For , , hence . , , hence . The sum of all edges capacity out of is . Hence the running time for Ford-Fulkerson Algorithm is

Converting interview scheduling problem input to maximum flow problem: creating the nodes is . Creating the edges is . Creating the edge capacity is also .

Converting maximum flow output to interview scheduling: we need to go through all edge to find the edges that has flow value of one. This would take

In summary, the total running time is

**Step 4**: correctness

We will prove that there exists a valid full schedule there exists a max flow of value n.

Proving “”

Assume in the valid full schedule, each candidate is matched to the recruiter at the time . Then we can set the flow of all as 1, set flow of as 1 for matched and, and 0 for all other unmatched . Then we set the flow all according to flow conservation constrains. Then the flow will be valid, because (1, capacity constrains) each edges in will have flow of 1, each edges in will have flow of either 0 or 1, and each edge in will have a flow that is smaller than their capacity because the schedule is full. (2, flow conservation constrains) because each candidate has been matched to one recruiter in one time slot, the inflow and outflow of each and will be equal. We also set the outflow of according to the flow conservation constrain, so for this constrain is also satisfied. And obviously the flow in this graph is n.

Proving “”

Assume we got the maximum flow n, and we assign the candidates and time slots according to the maximum flow result as we stated before. We need to prove this schedule is valid. We will discuss the restrictions one by one: (1) each candidate must be matched to a recruiter at , where and . Because each has capacity of 1, there must be a flow of value 1 through each . Meanwhile, because the definition of edges , and must be right. (2) Each recruiter can interview at most 1 candidate in each time slot, and at most in total. Because each has a capacity of 1, each recruiter won’t have more than one interview in one time slot. Also, because each has a capacity of , each recruiter won’t have more than interviews either. In summary, the output of maximum flow problem can be converted to a valid full schedule output.

**2. Disrupting an Enemy's Railway Network**

**(a) give an algorithm that find that maximizes**

We can reduce this problem to minimum cut problem

**Step 1**: convert the input of the problem to the input of minimum cut problem

Create a directed graph , where and . The capacity of edges is if , and the capacity of , is . Using node as start node .

**Step 2**: convert the output of minimum cut problem to the output of disrupting railway network problem.

Note is the output of the minimum cut algorithm. If then , we don’t destroy any edge. Otherwise, the output for disrupting railway network is .

**Step 3**: Running time

Converting input: adding a new node t and new edges will take

Ford-Fulkerson algorithm: The total edge number is , and . So the running time for FF algorithm is

Converting output: Using BFS to find the minimum s-t cut take O(m), and converting the minimum s-t cut to the output set F take at most O(m)

In summary, the total running time is .

**Step 4**: Correctness

(1) The output of the reduction is a valid solution for the problem

The output set is all edges from to , hence destroying all edges in will disconnect and . We note set , , then is the terminals that are disconnected (), and is the terminals that are still connected. Hence, . Hence, the desired , and .

(2) No other solution is better than

Assume , then we note set is all node that are reachable from after destroy , and . We can tell that is another cut of . Hence, . But because is the minimum cut of , , hence . cannot be better than .

**(b) undirected**

We can still solve this problem by reducing to minimum cut problem.

**Step 1**: convert input

Build another directed graph . ,

The edge capacity:

**Step 2**: convert output

Note the output minimum cut is . If , then , we don’t destroy any edge. Otherwise, .