Algorithm Design Assignment 5

Jianan Lin, 662024667, linj21@rpi.edu

Problem 7.6. There are n lights, n switches and m walls in a plane. One switch can control one light. A light is visible if the person can see it in the location of its switch, i.e. the line segment between them does not cross any wall. We know the start and end point of each wall. A plane is ergonomic if we can find a map between each light and each switch so that all the lights are visible. Design a polynomial algorithm with m and n to judge whether a plane is ergonomic.

Solution.

First we use a bipartite graph $G = (V, E)$. Let $V = X + Y$, in which X is the left side and Y is the right side of nodes. X includes all the switches and Y includes all the lights. We say there is an edge between $x_i \in X$ and $y_j \in Y$ if and only if light j is visible with switch i. So we know, a plane is ergonomic if and only if such a bipartite graph G has a perfect matching.

In this way, we divide our algorithm into two parts. The first part is to establish such a bipartite graph G. This part uses $O(mn^2)$ because we should test whether the line segment between each switch and each light (there are n^2 combinations) crosses each wall (there are m walls). So the total number of testing is mn^2.

The second part is to judge whether there is a perfect matching in this graph. There are different algorithms with polynomial time to achieve this. Here we choose Ford-Fulkerson algorithm. Although it is used to solve max-flow problem, we can also use this to judge whether there is a perfect matching in G.

First we add a source node s so that for any $x_i \in X$, there is a directed edge (s, x_i) with flow capacity 1. Then we add a destination/sink node t so that for any $y_i \in Y$, there is a directed edge (y_i, t) with flow capacity 1. Last, we set all the directed edge (x_i, y_j) has flow capacity 1 if and only if light j is visible with switch i. If there is no directed edge between two points, then obviously the flow capacity is 0.

We use Ford-Fulkerson algorithm to find the max $s - t$ flow and judge whether this flow equals n. If it is n, then there is a perfect matching in G and the plane is ergonomic. Otherwise it is not ergonomic. The time complextiy in this part is $O(n^3)$ because we know the time complexity of Ford-Fulkerson algorithm is $O(Ef)$ where E is the number of edges and f is the max flow. In this question, $E \leq n^2 + 2n$ and $f \leq n$, so it is $O(n^3)$. Therefore the time total complexity is $O(mn^2 + n^3)$.

Problem 7.19. There are n days and k doctors. Each doctor reports a set L_j containing in which day he/she is willing to work. The hospital needs p_i doctors in day i. so it needs to give a scheduling.

1. Write an algorithm with polynomial time in n and k. Our algorithm should output either k sets L'_j so that $L'_j \subseteq L_j$ and there are exactly p_i doctors in day i, or there is no such result.

2. We relax our restriction. We use a constant c so that $|L'_j - L_j| \leq c$. This time write an algorithm with the same requirement as (1).

Solution.

(1).

This question is rather similar to problem 7.6. Also we construct a directed graph G. First we have a source node s and a destination/sink node t. We add k nodes which denote each doctor and call the set of these nodes X so that for all $x_j \in X$, the flow capacity of directed edge $(s, x_j) = |L_j|$. Then we add n nodes which denote each day and call the set of these nodes Y so that for all $y_i \in Y$, the flow

capacity of directed edge $(y_i, t) = p_i$. Last for each $x_j \in X, y_i \in Y$, there is an edge (x_j, y_i) with flow capacity 1 if and only if $i \in L_j$.

Obviously, we can have a schedule which satisfies the requirement, if and only if the max $s - t$ flow of this graph is $f = \sum_{i=1}^{n} p_i$. The reason is that, the max flow is $f = \sum_{i=1}^{n} p_i$ if and only if any edge (y_i, t) has a flow p_i.

We divide our algorithm into two parts. First part: we use Ford-Fulkerson algorithm to find the largest flow (including the flow of each edge). The time complexity of this part is $O(n^2k^2)$ because we know the time complexity of Ford-Fulkerson is $O(Ef)$ where E is the number of edges and f is the max flow. Here, $E \leq nk + n + k$ and $f \leq nk$ because the flow capacity between any x_j and y_i is no more than 1. Therefore the time complexity is $O(n^2k^2)$.

Second part: If the max flow equals $f = \sum_{i=1}^{n} p_i$, then we can find such a result. We start from k empty sets $L'_j, j \in [1, k]$. For each x_j, y_i, if in our max flow, the flow of (x_j, y_i) is 1, then we add i into L'_j. Last we return $L'_1, ...L'_k$. Otherwise, if the max flow does not equal to $\sum_{i=1}^{n} p_i$, then we return "none" to denote there is no such result satisfying the requirement. The time complexity of this part is $O(nk)$ because we only need to check the flow of each (x_j, y_i).

So the total time complexity is $O(n^2k^2 + nk) = O(n^2k^2)$.

(2).

We modify the graph G in (1). We add k nodes and call the set of these nodes Z so that for all $z_i \in Z$, there is a directed edge (s, z_i) with flow capacity c. For each z_j, y_i, there is a directed edge (z_j, y_i) with flow capacity 1 if and only if $i \notin L_j$ (notice this is opposite to (x_j, y_i)).

In this time, we can have a schedule which satisfies the requirement, if and only if the max $s - t$ flow of this graph is $f = \sum_{i=1}^{n} p_i$. The reason is that, we have restricted that a doctor can work on the day he/she does not want, for at most c days in the graph.

So in the same way, we find the max flow with Ford-Fulkerson algorithm. If the max flow is $f = \sum_{i=1}^{n} p_i$, then there is a solution, otherwise there is no solution. To get a solution, we start with k empty sets $L'_j, j \in [1, k]$. For each $j \in [1, k], i \in [1, n]$, if in the max flow result, the flow of (x_j, y_i) is 1, then we add i into L'_j; if the flow of (z_j, y_i) is 1, then we also add i into L'_j. Finally, we return $L'_1, ..., L'_k$.

As for the time complexity, it is still $O(n^2k^2)$. For the first part, we have the number of edges $E \leq nk + 2k + n$ (notice if z_j can reach y_i then x_j cannot because a doctor cannot like and dislike day i at the same time) and the max flow $f \leq nk$. So the time complexity of first part is also $O(n^2k^2)$. For the second part, the time complexity is also $O(nk)$ because we need to check at most $2kn$ times.

So the total time complexity is $O(n^2k^2 + nk) = O(n^2k^2)$.