Homework 8

Name: 方嘉聪 ID: 2200017849

Problem 1. (Applications of Max-Flow Min-Cut).
Review the statement of max-flow min-cut theorem and prove the following two statements.

(a) Let $G = (L \cup R, E)$ be a unweighted bipartite graph. Then G has a L-perfect matching (a matching with size $|L|$) if and only if, for every set $X \subseteq L, X$ is connected to at least $|X|$ vertices in R. You must prove both directions.

Hint: Use the max-flow min-cut theorem.

(b) Let G be an unweighted directed graph and $s, t \in V$ be two distinct vertices. Then the maximum number of edge-disjoint s - t paths equals the minimum number of edges whose removal disconnects t from s (i.e., no directed path from s to t after the removal).

Hint: show how to decompose a flow of value k into k disjoint paths, and how to transform any set of k edge-disjoint paths into a flow of value k. ◀

Answer. (a) 必要性: 若 G 有 L-perfect matching, 那么对于任意 $X \subseteq L$, 取这一匹配对应的边, 则有 $|X|$ 条边连接 X 到 R, 满足条件.

充分性: 若对任意 $X \subseteq L$, X 连接到至少 $|X|$ 个 R 中的点. 考虑一个新的图 $G' = (V', E')$, 使得:

$$V' = V \cup \{s, t\}, \quad E' = E \cup \{(s, l) \mid l \in L\} \cup \{(r, t) \mid r \in R\}$$

其中, s, t 分别为源和汇, 且令每条边的的容量均为 1. 那么要证明 G 有 L-perfect matching, 等价于证明 G' 的最大流等于 $|L|$. 由 max-flow min-cut theorem, 等价于证明 G' 中的最小割 $C_{\min}$ 为 $|L|$.

考虑 G' 中一个大小为 $|L|$ 的流 f(存在性由已有条件保证), 注意到 $(\{s\}, L \cup R \cup \{t\})$ 为一个大小恰好为 $|L|$, 故 $C_{\min} \leq |L|$. 下面来证明, 对任意个割 $C = (S, T)$, 都有 $C \geq |L|$.

注意到割 C 可以拆分为三个部分

$$\{(s, l) \mid l \in L \cap T\} \cup \{(l, r) \mid l \in L \cap S \wedge r \in R \cap T\} \cup \{(r, t) \mid r \in R \cap S\}$$

记 $L_1 = L \cap S, L_2 = L \cap T, R_1 = R \cap T, R_2 = R \cap S$, 那么由题设, L_1 至少与 R 中 $|L_1|$ 个点相连, 而每条 $L_1 \to R \to t$ 路径至少有一条边在割 C 中 (要么横跨 $L_1 - R_1$, 要么横跨 $R_2 - t$), 那么:

$$|C| \geq |L_2| + |L_1| = |L|$$

故 $C_{\min} = |L|$, 从而 G 有 L-perfect matching. 证毕.

(b) 对任意一个图 $G = (V, E)$, 考虑如下一个新的图 $G' = (V', E')$, 使得:

$$\forall v \in V / \{s, t\}, \text{ add } v_{\text{in}}, v_{\text{out}} \in V', \text{ add } (v_{\text{in}}, v_{\text{out}}) \in E' \text{ with capacity 1;}$$
$$\forall e = (u, v) \in E, \text{ add } (u_{\text{out}}, v_{\text{in}}) \in E' \text{ with capacity 1.}$$

其中, 令 $s_{\text{out}} = s, t_{\text{in}} = t$. 那么 G' 中任意一个大小为 k 的流 f 对应于 G 中的 k 条边不交 $s - t$ 路径, 且反之亦然. 故 G 中边不交的 $s - t$ 路径数最大值等于 G' 中的最大流值, 即使 G 中 $s - t$ 不连通所需去除的最小边数. 证毕. ◁

Problem 2. (Trade Surplus).
Consider the following definition. We are given a set of n countries that are engaged in trade with one another. For each country i, we have the value s_i of its budget surplus; this number may be positive or negative, with a negative number indicating a deficit. For each pair of countries i, j, we have the total value e_{ij} of all exports from i to j; this number is always non-negative. We say that a subset S of the countries is *free-standing* if the sum of the budget surpluses of the countries in S, minus the total value of all exports from countries in S to countries not in S, is non-negative.

Give a **polynomial-time** algorithm that takes this data for a set of n countries, and decides whether it contains a non-empty free-standing subset that is not equal to the full set. Analyze its running time. ◀

Answer. 设图 $G = (V, E)$, 形式化的, 我们的目标为判断是否存在一个非空的集合 S, 使得:

$$\sum_{i \in S} s_i - \sum_{i \in S, j \notin S} e_{ij} \geq 0 \tag{1}$$

等价于优化问题:

$$\underset{S \neq \varnothing \wedge S \neq V}{\text{minimize}} \sum_{i \in S, j \notin S} e_{ij} - \sum_{i \in S} s_i \tag{2}$$

而后判断最小值是否小于等于 0 即可. 考虑如下的图 $G' = (V', E')$, 使得 (M 是个足够大的常数):

$$V' = V \cup \{s, t\}, \quad \forall v \in V, \text{ add } (s, v), (v, t) \text{ to } E' \text{ with capacity } M, M - s_v$$

那么由最大流最小割定理, 优化问题 [2] 等价于在 G' 中找到一个最大流 (设为 $f_{\max}$), 则 $f_{\max} - |V|M$ 即为问题(2)的最小值. 故只需判断是否满足

$$f_{\max} - |V|M \leq 0 \tag{3}$$

即可, 若成立则令 S. 注意到, 当问题(2)的最小值恰为 0 时, 上述算法可能会得到一个退化的解 $S = \varnothing$. 为了避免这一问题, 对任意 $v \in V$, 将 s_v 改为 $s_v + \varepsilon$(ε 为一个较小的正常数), 这么做相当于将限制条件(3)放松到了

$$f_{\max} - |V|M \leq \varepsilon \tag{4}$$

得到解 S 后, 在验证是否满足条件(3), 若否则说明 *free-standing* 不包含 v. 遍历所有 v 直至找到合法解或者说明解不存在. 算法时间复杂度为 $O(n|V'|^2|E'|) = O(n \cdot (n+2)^2 \cdot (n^2+2n)) = O(n^5)$. ◁

Problem 3. (Flow Disconnecting with Multiple Terminals).
Suppose we are given a directed network $G = (V, E)$ with a root node r, and a set of *terminals* $T \subseteq V$. We'd like to disconnect many terminals from r, while cutting relatively few edges.

We make this trade-off precise as follows. For a set of edges $F \subseteq E$, let $q(F)$ denote the number of nodes $v \in T$ such that there is no $r - v$ path in the subgraph $(V, E - F)$. Give a polynomial-time algorithm to find a set F of edges that maximizes the quantity $q(F) - |F|$. (Note that setting F equal to the empty set is an option). Analyze the running time of your proposed algorithm. ◀

Answer. 对 $\forall e \in E$, 令 $c_e = 1$, 同时添加一个汇点 t, 对 $\forall v \in T$, 添加边 (v,t), 容量为 1, 设这样新的到的图为 $G' = (V', E')$. 考虑 G' 上的最小割问题, 我们来证明最小化割的问题等价于最大化 $q(F) - |F|$.

考虑 G' 的一个割 (S_1, S_2), 设 $T_1 = S_1 \cap T, T_2 = S_2 \cap T$, 取 $F = \{(u,v) \mid (u,v) \in E \wedge u \in S_1 \wedge v \in S_2\}$. 那么由割的定义, $q(F) \leq |T_2|$. 若存在点 $t \in T_1$, 使得不存在路径 $r - t$, 则我们可以将 S_1 分成两部分

$$S_1 = S_1' \cup S_1'', \text{ 其中 } \forall v \in S_1', \text{存在 } r - v \text{ 路径}, \forall v \in S_1'', \text{不存在 } r - v \text{ 路径}.$$

考虑一个新的割 $(S_1', S_1'' \cup T_2)$, 注意到 S_1', S_1'' 间没有边相连, 故这个新的割的大小小于原来的割. 故当取到最小割时, 有 $q(F) = |T_2|$. 下面计算 (S_1, S_2) 的大小:

$$|F| + |T_1| = |F| - |T_2| + |T| \geq |F| - q(F) + |T|$$

当恰为最小割时取等. 注意到 $|T|$ 是一个给定常量, 故最小割问题等价于最大化 $q(F) - |F|$. 从而我们只需在 G' 上求解最小割问题即可. 时间复杂度为最小割问题的复杂度 $O(|V'|^2|E'|)$. ◁

Problem 4. (Actress/Actor Chain Game).
Some of your friends invent a game: you start with a set X of n actresses and a set Y of n actors, and two players P_0 and P_1. P_0 names an actress $x_1 \in X, P_1$ names an actor y_1 who has appeared in a movie with x_1, P_0 names an actress x_2 who has appeared in a movie with y_1, and so on. Thus, P_0 and P_1 collectively generate a sequence $x_1, y_1, x_2, y_2, \ldots$ such that each actor/actress in the sequence has co-starred with the actress/actor immediately preceding. A player $P_i (i = 0, 1)$ loses when it is P_i 's turn to move, and he/she cannot name a member of his/her set who hasn't been named before.

Suppose you are given a specific pair of such sets X and Y, with complete information on who has appeared in a movie with whom. A strategy for P_i, in our setting, is an algorithm that takes a current sequence $x_1, y_1, x_2, y_2, \ldots$ and generates a legal next move for P_i (assuming it's P_i 's turn to move). Give a polynomial-time algorithm that decides which of the two players can force a win, in a particular instance of this game. Analyze the running time of your proposed algorithm. ◀

Answer. 我们来证明如下的论断: **若 X, Y 存在完美匹配, 则 P_1(后手) 必胜, 否则 P_0(先手) 必胜.**
注: 这里的匹配关系指 $x \in X, y \in Y$ 且 x 与 y 在一部电影中出现过.

(1) 若 X, Y 存在完美匹配, 那么每次 P_1 只需要选择 P_0 选择的人对应的匹配的人即可, 显然 P_1 必胜.

(2) 若 P_1 必胜, 我们来证明 X, Y 存在完美匹配. 假设 X, Y 不存在完美匹配, 设 $|X| = |Y| = n$, 最大匹配为 $\{(x_{k_i}, y_{j_i})\}_{1 \leq i \leq t}$, 其中 $t < n$. 那么只要 P_0 第一步选择任意一个 $x \in X / \{x_{k_i}\}_{1 \leq i \leq t}$, 那么

- 若 P_1 选择了 $y \in Y / \{y_{j_i}\}_{1 \leq i \leq t}$, 则 (x, y) 构成了一个新的匹配, 这与最大匹配的假设矛盾.
- 若 P_1 选择了任意一个 $y_{j_i} \in \{y_{j_i}\}_{1 \leq i \leq t}$, 那么 P_0 只需选择对应匹配的 x_{k_i} 即可, 使用这个策略那么 P_0 必胜, 这与 P_1 必胜的条件矛盾.

故若 P_1 必胜, 则 X, Y 存在完美匹配.

故我们只需要判断 X, Y 是否存在完美匹配即可. 时间复杂度为 $O(|V| \cdot |E|) = O(n^3)$(匈牙利算法). 如果使用 Hopcroft-Karp 算法, 时间复杂度为 $O(\sqrt{|V|} \cdot |E|)$. ◁

Problem 5. (Job Scheduling of Multi-Processors).
Suppose you're managing a collection of processors and must schedule a sequence of jobs over time. The jobs have the following characteristics. Each job j has an arrival time a_j when it is first available for processing, a length ℓ_j which indicates how much processing time it needs, and a deadline d_j by which it must be finished. (We'll assume $0 < \ell_j \leq d_j - a_j$). Each job can be run on any of the processors, but only on one at a time; it can also be pre-empted and resumed from where it left off (possibly after a delay) on another processor.

Moreover, the collection of processors is not entirely static either: you have an overall pool of k possible processors; but for each processor i, there is an interval of time $[t_i, t'_i]$ during which it is available; it is unavailable at all other times.

Given all this data about job requirements and processor availability, you'd like to decide whether the jobs can all be completed or not. Give a polynomial-time algorithm that either produces a schedule completing all jobs by their deadlines, or reports (correctly) that no such schedule exists. You may assume that all the parameters associated with the problem are integers.

Example. Suppose we have two jobs J_1 and J_2. J_1 arrives at time 0, is due at time 4, and has length 3. J_2 arrives at time 1, is due at time 3, and has length 2. We also have two processors P_1 and P_2. P_1 is available between times 0 and 4; P_2 is available between times 2 and 3. In this case, there is a schedule that gets both jobs done:

- At time 0 , we start job J_1 on processor P_1.
- At time 1 , we pre-empt J_1 to start J_2 on P_1.
- At time 2 , we resume J_1 on P_2. (J_2 continues processing on P_1.)
- At time 3, J_2 completes by its deadline. P_2 ceases to be available, so we move J_1 back to P_1 to finish its remaining one unit of processing there.
- At time 4, J_1 completes its processing on P_1.

Notice that there is no solution that does not involve pre-emption and moving of jobs. ◀

Answer. 我们考虑构造如下的网络图 $G = (V, E)$:

- 添加源点汇点 s, t, 对任意一个可能时刻 t, 任务 J, 处理器 P, 都添加一个对应节点.
- 对于任意一个任务 J_i, 添加边 (s, J_i) 且 $c(s, J_i) = l_{J_i}$, 代表任务 J_i 需要 l_{J_i} 的处理器时间. 对任意时刻 $t_j \in [a_{J_i}, d_{J_i}]$, 添加边 (J_i, t_j) 且 $c(J_i, t_j) = 1$, 代表任务 J_i 在时刻 t_j 只能在一个处理器上运行.
- 对任意一个处理器 P_j, 添加边 (P_j, t) 且 $c(P_j, t) = \infty$. 对任意时刻 $t_k \in [t_{P_j}, t'_{P_j}]$, 添加边 (t_k, P_j) 且 $c(t_k, P_j) = 1$, 代表处理器 P_j 在时刻 t_k 只能运行一个任务.

那么在图 G 上的任意一个整数流对应着一个任务的调度方案: 对所有的时刻 t, 若满足 $(J, t) = 1$ 且 $(t, P) = 1$, 则说明这一时刻将任务 J 分配给处理器 P. 而流的大小即所有任务进行的处理器时间之和, 故只需运行最大流算法, 并判断 $|f| = \sum l_{J_i}$ 是否满足. 是则输出对应的调度方案, 否则输出无解.

设有 n 个任务, p 个处理器, T 个时刻, 则 $|V| = n + p + T, |E| = O((n + p)T)$, 故使用 Dinic 算法的时间复杂度为 $O(|V|^2|E|) = O((n + p + T)^2(n + p)T) = O(n^3p^3T^3)$. ◁