Algorithm Homework 05

Qiu Yihang

May 2022

1 Running Time of Randomized Algorithm

1.1 Expected Running Time of the Best Algorithm for Π

Proof. Define $\mathbf{A} \triangleq \left\{ \overline{A} \mid \overline{A} \text{ is a randomized algorithm solving } \Pi. \right\}$ Note that $\mathcal{A} \subset \mathbf{A}$.

CASE 01. $\overline{A}$ is a deterministic algorithm, i.e. $\overline{A} \in \mathcal{A}$.

Obvious exists $\mathscr{A}$,

$$\forall A \in \mathcal{A}, \quad \mathscr{A}(A) = \begin{cases} 1, & A = \overline{A} \\ 0, & \text{otherwise} \end{cases} \quad \text{s.t. } \mathbb{E}\left[T(\overline{A}, x)\right] = T(\overline{A}, x) = \mathbb{E}_{A \sim \mathscr{A}}\left[T(A, x)\right].$$

CASE 02. $\overline{A} \notin \mathcal{A}$.

Assume the running time of $\overline{A}$ is only related to random variables $Y_1, Y_2, ...Y_{N_{\overline{A}}}$. In other words, if $Y_1, Y_2, ...Y_{N_{\overline{A}}}$ are given fixed values, $T(x, \overline{A})$ is deterministic for given x.

Let $\overline{A}(y_1, y_2, ...y_{N_{\overline{A}}})$ be the $\overline{A}$ when $Y_1 = y_1, Y_2 = y_2, ...Y_{N_{\overline{A}}} = y_{N_{\overline{A}}}$.

Obvious for any $\overline{A} \in \mathbf{A}$, $N_{\overline{A}} < \infty$. (Otherwise, $\overline{A}$ can not terminate in finite time, i.e. $\overline{A} \notin \mathbf{A}$.)

Then $\forall y_1, y_2, ...y_{N_{\overline{A}}}$, $\overline{A}(y_1, y_2, ...y_{N_{\overline{A}}}) \in \mathcal{A}$.

Thus, for a given input x,

$$\mathbb{E}\left[T(\overline{A}, x)\right] = \mathbb{E}_{Y_1, Y_2, ...Y_{N_{\overline{A}}}}\left[T\left(\overline{A}\left(Y_1, Y_2, ...Y_{N_{\overline{A}}}\right), x\right)\right].$$

We can construct a distribution $\mathscr{A}$,

$$\forall A \in \mathcal{A}, \quad \mathscr{A}(A) = \sum_{y_1, y_2, ...y_{N_{\overline{A}}} \text{ s.t. } \overline{A}(y_1, y_2, ...y_{N_{\overline{A}}}) = A} \mathbf{Pr}\left[Y_1 = y_1, Y_2 = y_2, ...Y_{N_{\overline{A}}} = y_{N_{\overline{A}}}\right].$$

s.t.

$$\begin{aligned} \mathbb{E}_{A \sim \mathscr{A}}\left[T(A, x)\right] &= \sum_{A \in \mathcal{A}} \mathscr{A}(A) T(A, x) \\ &= \sum_{y_1, y_2, ...y_{N_{\overline{A}}}} \mathbf{Pr}\left[Y_1 = y_1, Y_2 = y_2, ...Y_{N_{\overline{A}}} = y_{N_{\overline{A}}}\right] T(\overline{A}(y_1, y_2, ...y_{N_{\overline{A}}}), x) \end{aligned}$$

$$
\begin{aligned}
&= \mathbb{E}_{Y_1,Y_2,...Y_{N_{\overline{A}}}}\left[T\left(\overline{A}\left(y_1,y_2,...y_{N_{\overline{A}}}\right),x\right)\right] \\
&= \mathbb{E}\left[T(\overline{A},x)\right].
\end{aligned}
$$

In conclusion, for any $\overline{A}$, we can find a distribution $\mathscr{A}$ over $\mathcal{A}$ s.t. the expected running time of $\overline{A}$ on x is $T(\overline{A},x) = \mathbb{E}_{A\sim\mathscr{A}}\left[T(A,x)\right]$. ■

Therefore, the expected running time of the best algorithm of Π is

$$
\begin{aligned}
T_{\texttt{best}} &= \min_{\texttt{randomized algorithm } \overline{A} \text{ solving } \Pi} \max_{x\in\mathcal{X}} \mathbb{E}\left[T(\overline{A},x)\right] \\
&= \min_{\overline{A}\in\mathbf{A}} \max_{x\in\mathcal{X}} \mathbb{E}\left[T(\overline{A},x)\right] = \min_{\texttt{distribution } \mathscr{A} \texttt{ over } \mathcal{A}} \max_{x\in\mathcal{X}} \mathbb{E}_{A\sim\mathscr{A}}\left[T(A,x)\right] \quad ■
\end{aligned}
$$

1.2 Yao's Minimax Principle

Proof. Consider a game with two player, $\mathcal{P}_A$ and $\mathcal{P}_X$.

Player $\mathcal{P}_A$ can determine the algorithm A while player $\mathcal{P}_X$ can determine the input X. Let $\mathcal{P}_X$ be the row player and $\mathcal{P}_A$ be the column player. When $\mathcal{P}_A$ pick an algorithm $A\in\mathcal{A}$ and $\mathcal{P}_X$ pick an input $X\in\mathcal{X}$, both of them can receive a payoff of $T(A,X)$.

Then strategies of $\mathcal{P}_A$ and $\mathcal{P}_X$ are actually distribution $\mathscr{A}$ over $\mathcal{A}$ and distribution $\mathscr{X}$ over $\mathcal{X}$.

The goal of $\mathcal{P}_X$ is to maximize the expected payoff of $\mathcal{P}_X$, i.e.

$$
\begin{aligned}
\max_{\texttt{distribution } \mathscr{X} \texttt{ over } \mathcal{X}} \min_{\texttt{distribution } \mathscr{A} \texttt{ over } \mathcal{A}} \sum_{a\in\mathcal{A},x\in\mathcal{X}} T(a,x)\mathscr{X}(x)\mathscr{A}(a) &= \max_{\texttt{distribution } \mathscr{X} \texttt{ over } \mathcal{X}} \min_{a\in\mathcal{A}} \sum_{x\in\mathcal{X}} T(a,x)\mathscr{X}(x) \\
&= \max_{\texttt{distribution } \mathscr{X} \texttt{ over } \mathcal{X}} \min_{a\in\mathcal{A}} \mathbb{E}_{X\sim\mathscr{X}}\left[T(a,X)\right]
\end{aligned}
$$

The goal of $\mathcal{P}_A$ is to minimize the expected payoff of $\mathcal{P}_X$, i.e.

$$
\begin{aligned}
\min_{\texttt{distribution } \mathscr{A} \texttt{ over } \mathcal{A}} \max_{\texttt{distribution } \mathscr{X} \texttt{ over } \mathcal{X}} \sum_{a\in\mathcal{A},x\in\mathcal{X}} T(a,x)\mathscr{X}(x)\mathscr{A}(a) &= \min_{\texttt{distribution } \mathscr{A} \texttt{ over } \mathcal{A}} \max_{x\in\mathcal{X}} \sum_{a\in\mathcal{A}} T(a,x)\mathscr{A}(a) \\
&= \min_{\texttt{distribution } \mathscr{A} \texttt{ over } \mathcal{A}} \max_{x\in\mathcal{X}} \mathbb{E}_{A\sim\mathscr{A}}\left[T(A,x)\right]
\end{aligned}
$$

By **Von Neumann's Minimax Theorem**, we have

$$
\begin{aligned}
&\max_{\texttt{distribution } \mathscr{X} \texttt{ over } \mathcal{X}} \min_{\texttt{distribution } \mathscr{A} \texttt{ over } \mathcal{A}} \sum_{a\in\mathcal{A},x\in\mathcal{X}} T(a,x)\mathscr{X}(x)\mathscr{A}(a) \\
=&\min_{\texttt{distribution } \mathscr{A} \texttt{ over } \mathcal{A}} \max_{\texttt{distribution } \mathscr{X} \texttt{ over } \mathcal{X}} \sum_{a\in\mathcal{A},x\in\mathcal{X}} T(a,x)\mathscr{X}(x)\mathscr{A}(a)
\end{aligned}
$$

i.e.

$$
\max_{\texttt{distribution } \mathscr{X} \texttt{ over } \mathcal{X}} \min_{a\in\mathcal{A}} \mathbb{E}_{X\sim\mathscr{X}}\left[T(a,X)\right] = \min_{\texttt{distribution } \mathscr{A} \texttt{ over } \mathcal{A}} \max_{x\in\mathcal{X}} \mathbb{E}_{A\sim\mathscr{A}}\left[T(A,x)\right] \quad ■
$$

1.3 Locating Problem

Solution. In any deterministic algorithm, we probe $A[i]$ in a fixed order of i.

The worst case is that x will be probed in the last place of the order. Note that if the first $(n-1)$ probing does not give x, we already know the only index not visited yet is the index of x.

Thus, the worst running time for a deterministic algorithm is $n-1$. ■

To improve the performance, we introduce randomization. Instead of probing $A[i]$ in a fixed order of i, we uniformly pick an unvisited i, i.e. $A[i]$ has not been probed yet, and probe $A[i]$ to see if $A[i]=x$. Note that after at most $(n-1)$ probings, we can determine the index of x.

For any fixed x, the expected time cost of the randomized algorithm is

$$T(n)=\frac{1}{n}\left(\sum_{k=1}^{n-1}k+(n-1)\right)=\frac{1}{n}\left[\left(\sum_{k=1}^{n}k\right)-1\right]=\frac{n+1}{2}-\frac{1}{n}.$$ ■

1.4 The Lower Bound of Running Time on Locating Problem

Proof. By **1.1** and **1.2** (Yao's Minimax Principle), we know

$$\begin{aligned}
T_{\texttt{best}} &= \min_{\texttt{distribution } \mathscr{A} \texttt{ over } \mathcal{A}} \max_{x\in\mathcal{X}} \mathbb{E}_{A\sim\mathscr{A}}\left[T(A,x)\right] \\
&= \max_{\texttt{distribution } \mathscr{X} \texttt{ over } \mathcal{X}} \min_{a\in\mathcal{A}} \mathbb{E}_{X\sim\mathscr{X}}\left[T(a,X)\right] \\
&= \max_{\texttt{distribution } \mathscr{X} \texttt{ over } \mathcal{X}} \left(\sum_{k=1}^{n-1} k\cdot\mathscr{X}(i_k)+(n-1)\cdot\mathscr{X}(i_n)\right)_{\texttt{where } \{i_1,i_2,\dots i_n\}=\{1,2,\dots n\},\mathscr{X}(i_1)\geq\mathscr{X}(i_2)\geq\dots\mathscr{X}(i_n)} \\
&= \sum_{k=1}^{n-1} k\cdot\frac{1}{n}+(n-1)\frac{1}{n} \\
&= \frac{n+1}{2}-\frac{1}{n}
\end{aligned}$$

Thus, any randomized algorithm for the problem in **1.3** costs at least $\left(\frac{n+1}{2}-\frac{1}{n}\right)$ in expectation in the worst case. ■

Our algorithm in **1.3** also matches this lower bound. ■

2 Perfect Matching

2.1 Solution by Max-Flow

Solution. We can convert the problem into a max-flow problem as follows.

First consider $|V_1|$ and $|V_2|$. If $|V_1| \neq |V_2|$, obvious there does not exist a perfect matching.

When $|V_1| = |V_2|$, construct a graph $G' = (V', E', \texttt{capacity})$.

Construct source vertex s and sink vertex t. Then $V' = V1 \cup V_2 \cup \{s, t\}$.

Preserve all edges in E with capacity $+\infty$. Add edges from s to all $u \in V_1$ with capacity 1. Add edges from all $v \in V_2$ to t with capacity 1.

Then the original problem is equivalent to computing the max-flow on the graph G'. If the max-flow is exactly $|V_2|$ (which is also $|V_1|$), there exists a perfect matching. ■

2.2 Hall's Condition

Assumption. $|V_1| = |V_2|$.

Proof. Define $N_M(S) \triangleq \{u \in V_2 \mid \{u, v\} \in M \text{ for some } v \in S\}$.

<u>Proof of Necessity.</u> When graph G contains a perfect matching M, for any $v \in V_1$, exists exactly one edge in M, i.e. exists exactly one neighbor in V_2. Thus, $\forall S \subset V_1,\ |N_M(S)| = |S|$.

We know $M \subset E \Longrightarrow \forall S \subset V_1,\ N_M(S) \subset N(S)$, i.e. $\forall S \subset V_1,\ |N(S)| \geq |N_M(S)| = |S|$. □

<u>Proof of Sufficiency.</u> Consider the min cut on the graph G'.

Since $(\{s\}, V_1 \cup V_2 \cup \{t\})$ is a cut with $\texttt{capacity}((\{s\}, V_1 \cup V_2 \cup \{t\})) = \texttt{degree}(s) = |V_1|$, we know the min cut is no larger than $|V_1|$. Now we prove that the min cut is exactly $|V_1|$ by contradiction.

Assume exists a cut $\texttt{cut}^* \triangleq (\{s\} \cup A_2 \cup B_2, \{s\} \cup A_1 \cup B_1)$ s.t. $\texttt{capacity}(\texttt{cut}^*) < |V_1|$, where $A_1 \cap A_2 = \varnothing, A_1 \cup A_2 = V_1, B_1 \cap B_2 = \varnothing, B_1 \cup B_2 = V_2$.

Obvious, $\forall e \in \texttt{cut}^*,\ e \notin E$. Otherwise, $\texttt{capacity}(\texttt{cut}^*) \geq +\infty > |V_1|$.

Thus, any edge in $\texttt{cut}^*$ is either from s to some $v \in V_1$ or from some $u \in V_2$ to t. Then we know there are no edges between A_2 and B_1. Also, $\texttt{capacity}(\texttt{cut}^*) = |A_1| + |B_2| < |V_1| = |V_2|$.

This yields $|B_2| - |A_2| = |B_2| - |V_1| + |A_1| < |V_1| - |V_1| = 0 \Longrightarrow |B_2| < |A_2|$.

There are no edges between A_2 and $B_1 \Longrightarrow N(A_2) \subset V_2 \setminus B_1 = B_2$, i.e. $|N(A_2)| \leq |B_2| < |A_2|$.

Thus, exists $S = A_2 \subset V_1$ s.t. $|N(S)| < |S|$. **<u>Contradiction</u>** to $\forall S \subset V_1, |N(S)| \geq |S|$.

Therefore, the min cut on graph G' is exactly $|V_1|$. By **Max-flow Min-cut Thm.**, we know the max flow on graph G' is $|V_1|$. By **2.1**, we know exists a perfect matching.

In conclusion, graph G contains a perfect matching **<u>iff.</u>** $\forall S \subset V_1,\ |N(S)| \geq |S|$. ■

3 Debt Network

Proof. Inspired by the process of **Fold-Fulkerson** Algorithm, we design the following algorithm to remove cycles from the debt network to convert it into one with $(n-1)$ edges.

1. Initialization: $G^{(0)} = G,\ t \leftarrow 0$.
2. Regard $G^{(t)} = (V, E^{(t)}, w^{(t)})$ as an undirected graph. Try to find a cycle C_t on $G^{(t)}$.
   If cannot find such cycle, jump to step 6.
3. Find the edge $e^* = \{u^*, v^*\} \in C_t$ with minimum weight. Let $w^* = w^{(t)}(u^*, v^*)$.
   Let the direction of C_t be the same as the direction of e^*, i.e. $C : v_0 = u^* \to v_1 = v^* \to v_2 \to ... \to v_k = u$.
4. Update the weight of all edges in C_t. $G^{(t+1)} \leftarrow G^{(t)} = (V, E^{(t)}, w^{(t)})$.
   For each edge e in the cycle C_t,
   - If $e = \{v_i, v_{i+1}\}$ for some i, $w^{(t+1)}(v_i, v_{i+1}) \leftarrow w^{(t)}(v_i, v_{i+1}) - w^*$.
     If $w^{(t+1)}(v_i, v_{i+1})$ is 0 after the update, remove edge $\{v_i, v_{i+1}\}$ from $G^{(t+1)}$.
   - If $e = \{v_{i+1}, v_i\}$ for some i, $w^{(t+1)}(v_{i+1}, v_i) \leftarrow w^{(t)}(v_{i+1}, v_i) + w^*$.
5. $t \leftarrow t + 1$. Jump to step 2.
   Repeat step 2-5 on the updated graph $G^{(t)}$ (here t is already incremented).
6. Suppose $t = T$ when the algorithm arrives at step 6.
   We know $G^{(T)}$ contains no cycle when regarded as an undirected graph.
   Obvious there are at most $(n-1)$ edges in $G^{(T)}$.
   Then $G^{(T)}$ provides how debts can be settled with at most $(n-1)$ person-to-person payments, i.e. u pays v money with $w(u, v)$ amount if $\{u, v\} \in E^{(T)}$.

Now we prove the correctness of the algorithm above, i.e. to prove that for any person, his or her total payment according to $G^{(T)}$ is exactly the same as the total payment according to G.

Define $\mathtt{pay}_t(u) \triangleq \sum_{v:\{u,v\}\in E^{(t)}} w(u, v) - \sum_{v:\{v,u\}\in E^{(t)}} w(v, u)$. The first term is the money u owes other people and the second term is the money u should receive from other people.

Then we only need to prove that $\forall u \in V, \mathtt{pay}_0(u) = \mathtt{pay}_T(u)$.

By the process of our algorithm, the change of $\mathtt{pay}_\cdot(\cdot)$ only happens in step 4.

Obvious for $u \notin C_t$, $\mathtt{pay}_{t+1}(u) = \mathtt{pay}_{t+1}(u)$.

For $u \in C_t$,

CASE 01. In C_t, the two edges adjacent to u are both from u to others, i.e. $\{u, u_1\}, \{u, u_2\}$.

Obvious exactly one of these two edges is in the inverse direction of C_t. Thus,

$$\begin{aligned}\mathtt{pay}_{t+1}(u) &= \mathtt{pay}_t(u) - w^{(t)}(u, u_1) - w^{(t)}(u, u_2) + w^{(t+1)}(u, u_1) + w^{(t+1)}(u, u_2) \\ &= \mathtt{pay}_t(u) - w^* + w^* = \mathtt{pay}_t(u).\end{aligned}$$

CASE 02. In C_t, the two edges adjacent to u are both from other vertices to u.

Similar to **CASE 01**, $\texttt{pay}_{t+1}(u) = \texttt{pay}_t(u) - w^* + w^* = \texttt{pay}_t(u)$.

CASE 03. Exists $u_1, u_2 \in V$ s.t. $\{u, u_1\}, \{u_2, u\} \in C_t$.

1) When $\{u, u_1\}$ and $\{u_2, u\}$ are in the same direction as C_t's.

$$\begin{aligned}\texttt{pay}_{t+1}(u) &= \texttt{pay}_t(u) - w^{(t)}(u, u_1) + w^{(t)}(u_2, u) + w^{(t+1)}(u, u_1) - w^{(t+1)}(u_2, u) \\ &= \texttt{pay}_t(u) + \left(w^{(t+1)}(u, u_1) - w^{(t)}(u, u_1)\right) - \left(w^{(t+1)}(u_2, u) - w^{(t)}(u_2, u)\right) \\ &= \texttt{pay}_t(u) - w^* + w^* = \texttt{pay}_t(u).\end{aligned}$$

2) When $\{u, u_1\}$ and $\{u_2, u\}$ are in the inverse direction of C_t's.

$$\begin{aligned}\texttt{pay}_{t+1}(u) &= \texttt{pay}_t(u) - w^{(t)}(u, u_1) + w^{(t)}(u_2, u) + w^{(t+1)}(u, u_1) - w^{(t+1)}(u_2, u) \\ &= \texttt{pay}_t(u) + \left(w^{(t+1)}(u, u_1) - w^{(t)}(u, u_1)\right) - \left(w^{(t+1)}(u_2, u) - w^{(t)}(u_2, u)\right) \\ &= \texttt{pay}_t(u) + w^* - w^* = \texttt{pay}_t(u).\end{aligned}$$

In conclusion, $\forall t \in \{0, 1, ..., T-1\}, \forall u \in V,\ \texttt{pay}_{t+1}(u) = \texttt{pay}_t(u)$.

Thus, $\forall u \in V,\ \texttt{pay}_0(u) = \texttt{pay}_1(u) = ... = \texttt{pay}_T(u)$.

Therefore, our algorithm is correct. $\square$

In other words, all debts can be settled with at most $(n-1)$ person-to-person payments. $\blacksquare$

4 Rating and Feedback

The completion of this homework takes me three days, about 18 hours in total. Still, writing a formal solution is the most time-consuming part.

The ratings of each problem is as follows.

Problem	Rating
1.1	4
1.2	3
1.3	2
1.4	2
2.1	2
2.2	3
3	4

Table 1: Ratings.

This time I finish all problems on my own.