Tsinghua University

Discrete Optimistic

Fall 2024

Homework 1

Lin Zejin May 27, 2025

- **Collaborators:** I finish this homework by myself.

Problem 1. By using contraction view, it suffices to show that the first step can actually be viewed as using blue rules sequentially.

For each node n_i, assume that v_i is the edge connected to n_i of minimal weight.

WLOG, we assume $\omega(n_i), 1 \leq i \geq |V|$ is increasing.

Then we prove that coloring v_i blue simultaneously is equivalent to applying blue rule to the cut $(\{n_1, \cdots, n_k\}, \{n_{k+1}, \cdots, n_{|V|}\})$ successively.

By induction. For applying blue rule to the cut $(\{n_1\}, \{n_2, \cdots, n_{|V|}\})$, v_1 turns blue since v_1 is of the minimal weight in all edges.

If we have colored $v_1, \cdots, v_{k-1}$, then for applying blue rule to the cut $(\{n_1, \cdots, n_k\}, \{n_{k+1}, \cdots, n_{|V|}\})$, v_k turns blue. Otherwise, if $v'_k = (a, b)$ turns blue, $a \in \{n_1, \cdots, n_k\}$ and $b = n_t \in \{n_{k+1}, \cdots, n_{|V|}\}$, then $\omega(v'_k) \geq \omega(v_t)$ by the primitive rule. Thus $\omega(v'_k) \geq \omega(v_t) \geq \omega(v_k)$ as $t \geq k$, which causes contradiction.

So actually we can apply blue rule one by one. Therefore, *Borůvka* algorithm is indeed applying blue rule constantly, and it terminates until there is only one blue component, *i.e.* forming a blue tree. As we proved in class, this tree is MST.

Problem 2. WLOG, we assume that $J_1, \cdots, J_n$ is exactly a possible schedule of minimal time cost.

If $f_i \leq f_{i+1}$, then swap J_i, J_{i+1}. It will not affect other jobs. However, the time when job J_i, J_{i+1} finished both will be changed. It is $\max\{t + p_{i+1} + p_i + f_i, t + p_{i+1} + f_{i+1}\}$, which is not longer than $\max\{t + p_i + p_{i+1} + f_{i+1}, t + p_i + f_i\} = t + p_i + p_{i+1} + f_{i+1}$ at first. So it is still a possible schedule of minimal completion time cost.

Using bumbling sort algorithm, we can rearrange $J_1, \cdots, J_n$ such that f_i is decreasing, and it is still a possible schedule of minimal completion time cost.

If $f_i = f_{i+1}$, the order will not affect the result obviously.

Since the schedule of minimal time cost must exist, it suffices to sort f_i to get a possible schedule of minimal completion time cost.

Problem 3.

$$\sum_{i=1}^{n} \omega_i C_i = \sum_{i=1}^{n} \omega_i \left(\sum_{j=1}^{i} t_j \right) = (\sum_{i=1}^{n} t_i)(\sum_{i=1}^{n} \omega_i) - \sum_{i=1}^{n} \omega_i \left(\sum_{j=i+1}^{n} t_j \right) \tag{3.1}$$

So it suffices to maximize $\sum_{i=1}^{n} \omega_i \left(\sum_{j=i+1}^{n} t_j \right)$.

WLOG, we assume that $(t_i, \omega_i), 1 \le i \le n$ has been a possible solution.

If $\frac{\omega_i}{t_i} < \frac{\omega_{i+1}}{t_{i+1}}$, $i < k$ then swap (ω_i, t_i) and (ω_k, t_k). It only affects $i \sim k$-th terms in the RHS of (3.1). Noticed that the difference

$$\left[\omega_i \left(\sum_{j=i+1}^{n} t_j \right) + \omega_{i+1} \left(\sum_{j=i+2}^{n} t_j \right) \right] - \left[\omega_{i+1} \left(t_i + \sum_{j=i+2}^{n} t_j \right) + \omega_i \left(\sum_{j=i+2}^{n} t_j \right) \right] = \omega_i t_{i+1} - \omega_{i+1} t_i$$

$$= t_i t_{i+1} \left(\frac{\omega_i}{t_i} - \frac{\omega_{i+1}}{t_{i+1}} \right) < 0$$

So (3.1) will be smaller as they swap, which contradicts that (3.1) has already taken the minimal value.

So a possible solution of minimal value must satisfy that $\frac{\omega_i}{t_i}$ is decreasing.

If $\frac{\omega_i}{t_i} = \frac{\omega_{i+1}}{t_{i+1}}$, (3.1) remains the same as they swap. So a possible solution of minimal value is a rearrangement that satisfy $\frac{\omega_i}{t_i}$ is decreasing.

So it suffices to sort $\frac{\omega_i}{t_i}$ decreasingly.

Problem 4. The problem can be reduced to the original case.

We can enumerate each job. Take the first job, and remove other jobs that are not compatible with the first job. Here we remove the interval of the first job, and hence get a line of jobs. Apply the original algorithm of interval scheduling to the rest of jobs and we get a number that maximize the numbers of compatible jobs in the case that the first job is taken. Compare such n numbers we can obtain the answer.

Problem 5. Use a algorithm similar to Dijkstra's algorithm.

Denote s as the place they start. We prove by induction to $|S|$ that every $d[u]$ is the minimal cost time from s to u.

Algorithm 1 Algorithm

1: $\begin{cases} d[x] \leftarrow f_{(s,x)}(0), \mathrm{Pred}[x] = s, & (s,x) \in E \\ d[x] \leftarrow +\infty, & \text{otherwise} \end{cases}$
2: $S \leftarrow \{s\}$
3: $d[s] \leftarrow 0$
4: **while** $S \neq V$ **do**
5: Choose $u \in \arg\min_{x \notin S} \{d[x]\}$.
6: Update $S \leftarrow S \cup \{u\}$.
7: **for** each $x \in V - S, (u,x) \in E$ **do**
8: **if** $d[u] + f_{(u,x)}(d[u]) < d[x]$ **then**
9: $d[x] \leftarrow d[u] + f_{(u,x)}(d[u])$
10: $\mathrm{Pred}[x] \leftarrow u$
11: **end if**
12: **end for**
13: **end while**

Proof. For $|S| = 1$, trivial. At each step in 6, let $v = \mathrm{Pred}[u]$. Then $d[u] = d[v] + f_{(v,u)}(d[v])$.

By the algorithm step 7, $d[u]$ is the minimal cost time from s to u as we restrict the graph to S.

If there is a path P that cost less than $d[u]$, then the path pass through points outside S. Let $\alpha \to \beta$ in this path that $\alpha \in S$, $\beta \notin S$. Then $\alpha \neq u$ and

$$\begin{aligned}\text{length}(P) &\geq \text{length}(P[s \to \beta]) \\ &= \text{length}(P[s \to \alpha]) + f_{\alpha\to\beta}(d[\alpha]) \qquad \geq d[\alpha] + f_{\alpha\to\beta}(d[\alpha]) \qquad \text{(by inductive hypothesis)} \\ &\geq d[\beta] \geq d[u] \qquad \text{(by step 5)}\end{aligned}$$

So $d[u]$ is the minimal cost time from s to u in G. □

Problem 6. Use Boruvka's Algorithm. After i-th WHILE iterations, with the contraction view, $|V_i| \leq \frac{|V|}{2^{i-1}}$, $|E_i| \leq |V_i| + 8 \in O(|V_i|)$.
Then the time complexity $\sum |E_i| \leq \sum O(\frac{|V|}{2^{i-1}}) \leq O(|V|)$

Problem 7. (a) Since all edge costs are distinct, applying blue rule or red rule is the unique operation at any time. Therefore, the final graph with a blue tree is unique. If MST is not unique, there exists a MST that is not the blue tree. WLOG we assume T is not the blue tree completely. Choose a blue edge $e \notin T$. Then for its corresponding fundamental cycle C, since any cutset contains at least two edges in C, $\exists e' \in C \cap T \quad s.t.\ e'$ and e are in the same cutset where we color e blue by applying blue rule $\Rightarrow \omega(e') \geq \omega(e)$. However, by the result in lecture, $\omega(e') \leq \omega(e) \Rightarrow \omega(e') = \omega(e) \Rightarrow$ contradiction! So MST can only be the blue tree.

(b) Yes, the proof is below:

Proof. For each MST T in G, choose $\epsilon > 0$ small enough such that $\forall e \in E, [\omega(e) - \epsilon, \omega(e)) \cup (\omega(e), \omega(e) + \epsilon]$ will not contains any of $\omega(e), e \in E$.

Take some changes whose rule set is:

(1) If $T = \{e_1, e_2, \cdots, e_t\}$ where $\omega(e_1) \geq \omega(e_2) \geq \cdots \geq \omega(e_t)$, replace $\omega(e_i)$ with $\omega(e_i) - \frac{i}{t}\epsilon$. Then after the changes, $\omega(e_1) > \omega(e_2) > \cdot > \omega(e_t)$.

(2) After (1), if $\exists \omega(e_1) = \omega(e_2) = \cdots = \omega(e_k)$, where $e_i \in E$ distinct and k is the largest number, then we replace $\omega(e_i)$ with $\omega(e_i) + \frac{i}{k}\epsilon$.

Step (1)(2) makes sure that after the change, there are no edges that have the same cost. So in this case the MST is unique. Now we prove this unique MST is exactly the original one, T.

First, after (1), $\omega(e), e \in T$ will not be same as the cost of other edges, so they will never change after (2). Therefore, the cost of T decreases $\sum_{i=1}^{t} \frac{i}{t}$. Moerover, the cost of any tree in G has less decreased or even increased. So T is still the MST.

Finally, we prove that apply Kruskal's Algorithm to the changed graph produces a same valid execution of Kruskal's Algorithm on the original G.

That's because, the change will not change the ordinal relationship between edges with distinct cost. So the effect of the change on the order is only to rearrange the edges with the same weight. So the effect of the change on the order is only to rearrange the edges with the same weight. Equivalently, the order of edges in Kruskal's Algorithm on the changed graph is still a valid order in Kruskal's Algorithm on the original graph. Since Kruskal's Algorithm returns the same result if the order and the edges are determined, they produces the same MST, T. □

(c) The output graph H have no cycle of 3 or 4 edges. Or equivalenty, it has no triangles and squares. That's because, if it has a cycle of k edges, $k \leq 4$, then assume e is the last edge we put into the graph. Then e is the longest edge so $l < kl_e < 4l_e$. Let l be the length of the cycle. Then $3l_e < l - l_e$. But $l < 4l_e \Rightarrow$ contradiction!

Since H has no triangles, it has to be a bipartite graph (V_1, V_2) where all edges connect points in V_1 and V_2. Let $c_v, v \in V_1$ be the degree of v and S_v be the set of $v' \in V_2$ connected to v. Since there is no squares in H, $|S_{v_1} \cap S_{v_2}| \leq 1$, $\forall v_1, v_2 \in V_1, v_1 \neq v_2$.

Therefore, each pair $\{u_1, u_2\} \subset V_2$ is contained by one S_v at most. S_v contains $\dfrac{c_v(c_v-1)}{2}$ such a pair So we have

$$\sum_{v \in V_1} \frac{c_v(c_v-1)}{2} \leq \frac{|V_2|(|V_2|-1)}{2}$$

Then by Cauchy's inequallity,

$$\sum_{v \in V_1} c_v \leq |V_1| + \sqrt{|V_1| \cdot \sum_{v \in V_1} (c_v - 1)^2} \leq n + \sqrt{|V_1| \cdot |V_2|(|V_2|-1)} \leq n + \sqrt{\frac{n^2}{4}(|V_2|-1)} \leq 2025 n^{\frac{3}{2}}$$

So the number of edges $f(n) \sum_{v \in V_1} c_v \leq 2025 n^{\frac{3}{2}}$. Hence $\lim_{n \to \infty} \dfrac{f(n)}{n^2} = 0$

Problem 8. Since it contains no cycles, we can determine the root at first.

Also because it contains no cycle, each edge in minimum-cost arborescence is some cheap edges, otherwise one can replace it with the cheap edge, it stil has no directed cycles and each node $v \neq r$ has exactly one incoming edge, hence remains a arborescence.

So this arborescence A contains all cheap edges, so it should be minimum-cost arborescence as we proved in the lecture.