Homework 2 & 3

Chengyu Lin*

Problem 1

$$\begin{aligned} T(n) &= aT(\frac{n}{b}) + O(n^c) \\ &= a[aT(\frac{n}{b^2}) + O((\frac{n}{b})^c)] + O(n^c) \\ &= \cdots \\ &= a^k T(\frac{n}{b^k}) + \sum_{j=0}^{k-1} O(a^j (\frac{n}{b^j})^c) \\ &= a^k T(\frac{n}{b^k}) + O(n^c) \sum_{j=0}^{k-1} \frac{a^j}{b^{cj}} \end{aligned}$$

Here we assume that $b^k = n$, where $T(n) = a^k T(1) + O(n^c) \sum_{j=0}^{k-1} \frac{a^j}{b^{cj}}$. And $k = \log_b n$.

Case 1 $a = b^c$
So $T(n) = a^k T(1) + O(n^c) \sum_{j=0}^{k-1} \frac{a^j}{a^j} = O(n^c \log n)$.

Case 2 $a < b^c$
So $\sum_{j=0}^{k-1} \frac{a^j}{b^{cj}} \leq \sum_{j=0}^{\infty} \frac{a^j}{b^{cj}} = \frac{b^c}{b^c - a}$. Therefore $T(n) = O(n^c)$.

Case 3 $a > b^c$

$$\sum_{j=0}^{k-1} (\frac{a}{b^c})^j = \frac{b^c}{b^c - a} - \frac{a^k b^{-c(k-1)}}{b^c - a}$$

And $a^k = a^{\log_b n} = n^{\log_b a}$.
So $T(n) = n^{\log_b a}[T(1) - \frac{b^{c(k-1)}}{a - b^c}] + \frac{O(n^c)}{a - b^c} = O(n^{\log_b a})$.

Ex28, P334 (a) If all optimal solutions do not schedule jobs in increasing order of their deadlines.
Pick one schedule $\{a_m\}$ maximize k where $d_{a_i} \leq d_{a_{i+1}}$ for all $i+1 \leq k$, and $d_{a_k} > d_{a_{k+1}}$. Clearly $k < m$.
Then swap the order of job a_k and a_{k+1}. Given that $d_{a_k} > d_{a_{k+1}}$ we have

$$s + t_{a_k} + t_{a_{k+1}} - d_{a_{k+1}} \geq s + t_{a_{k+1}} - d_{a_{k+1}}$$

and

$$s + t_{a_k} + t_{a_{k+1}} - d_{a_{k+1}} \geq s + t_{a_k} + t_{a_{k+1}} - d_{a_k}$$

so

$$\max(s + t_{a_k} - d_{a_k}, s + t_{a_k} + t_{a_{k+1}} - d_{a_{k+1}}) \geq \max(s + t_{a_{k+1}} - d_{a_{k+1}}, s + t_{a_k} + t_{a_{k+1}} - d_{a_k})$$

Therefore such a change will not damage the result but given a new optimal solution with a larger parameter k, which contradicts our assumption.
So there will be an optimal solution which schedule jobs in increasing order of their deadlines.

*F1003028-5100309007

(b)
```
1. Sort the jobs in increasing order of their deadlines mark as d_1, d_2, ...
2. Denote A_i as the optimal solution ended with job i, and initially set them empty.
3. Enumerate the jobs increasingly from 1 as i:
    3.1 Calc s_i = d_i - t_i as the latest start time of job i
    3.2 Find the latest job k with f_k <= s_i
    3.3 If there is no such job set A_i = { i }
    3.4 Else set A_i = A_k union { i }
4. Enumerate all A_i and then choose the proper one as the answer
```

Ex29, P334 The algorithm:

Denote $f(i, S)$ as the minimal cost of steiner tree over S rooted at node i.

Here we just enumerate all $S \subseteq X$ in increasing order of their size. And the transition function is

$$f(i,S) = \min(\min_{i' \in V}(f(i',S) + w(i,i')), \min_{S' \subset S}(f(i,S') + f(i,S-S')))$$

Finally enumerate all node to take the minimal value of $f(i, X)$ which is the answer. And clearly the complexity is $O(n^2 4^k)$ since there are $n2^k$ functions to be calculated and to calc one of them we have to enumerate at most $n + 2^k$ items.