CSE 431/531 Homework 2 Spring 2022

Feng Wei 50428109

Problem 1.

(1a) Yes, the decision is safe.

Lemma 1. Let i be the job with the latest starting time. Then there is an optimum solution in which i is scheduled.

Proof. Let P be any optimum solution.

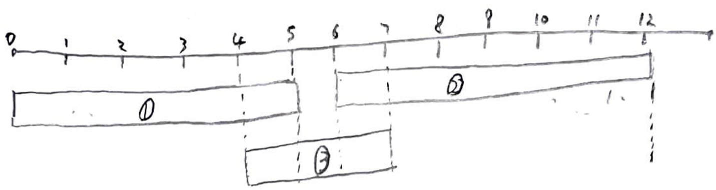
if $i \in P$, then we are done.

So, we assume $i \notin P$. Let i' be the latest staring job in P. Then we exchang i' with i and show it is an optimum solution $P' := P \setminus \{i'\} \cup \{i\}$. First, all jobs in $P \setminus \{i'\}$ finish at or before $S_{i'}$, since i' is the latest starting job in P. By our definition, $S_i \geq S_{i'}$, that means all job in $P \setminus \{i'\}$ finish at or before S_i.

Thus P' is a valid solution. $|P'| = |P|$.

In any case, there is an optimum solution contains i.

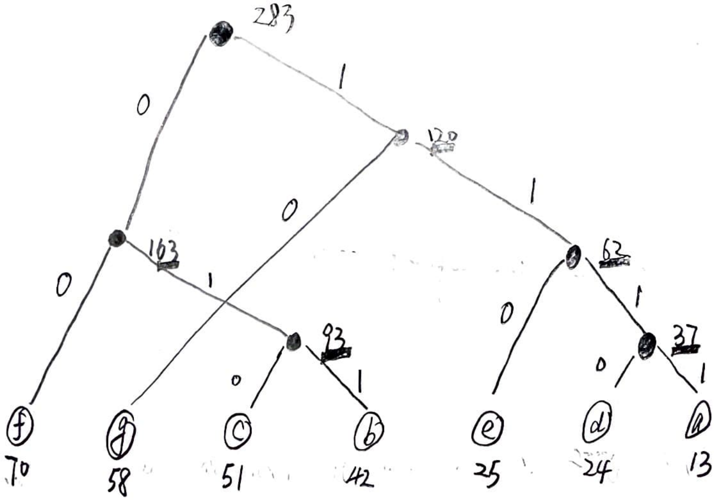
(1b) The answer is No



According to the decision, we will first remove job ② which is the longest, then remove job ①. As a result, we will only schedule job ③

But, the optimum solution should schedule job ① and ②.

problem 2.



f: 00

g: 10

c: 010

b: 011

e: 110

d: 1110

a: 1111

weighted length = $13 \times 4 + 24 \times 4 + 25 \times 3 + 42 \times 3 + 51 \times 3 + 58 \times 2 + 70 \times 2$

$= 758$

problem 3.

I use one minheap with size K to store the Kth largest number to the largest number in A[1⋯i]. Then the smallest number from the K biggest elements will be stored in the root.

```
minheap ← an empty heap with |S| = K
sum ← 0    the sum of the k biggest number
for i ← 1 to n do
      if i ≤ k then
            minheap.insert (A[i], A[i])
            sum ← sum + A[i]
            ~~bi ← sum~~
      else
            if A[i] ≥ minheap.get-min() then
                  sum ← sum - minheap.get-min() + A[i]
                  minheap.extract-min(), minheap.insert (A[i], A[i])
      bi ← sum
return b[k, ⋯ n]
```

$\boxed{\text{So } O(\log(k))}$

k is the size of the min heap

Total time

$O(n \log(k))$

Problem 4.

Strategy: Cover the left most point X_i, with interval $[X_i, X_i+1]$

Lemma: Let X_i be the left most point, then there exists an optimum solution which contains interval $[X_i, X_i+1]$

Proof: Let S be any optimum solution. If $[X_i, X_i+1] \in S$, then done.

We assume $[X_i, X_i+1] \notin S$.

Since X_i is covered by interval in S. Then there exists an interval in S that satisfies $[p, p+1]$ and $p < X_i < p+1 < X_i+1$.

Since X_i is the left most point, there is no point in interval $[p, X_i)$. Therefore, we replace $[p, p+1]$ with $[X_i, X_i+1]$ in S. And the updated solution S' is still an optimum solution. Since $|S'| = |S|$

After the Above step the reduced instance should be

$$X \leftarrow X \setminus \{x_j : x_j \in [x_i, x_i+1]\}$$

x_i is the left most point.

Time complexity:

First We need to find the left most point in $X = \{x_1, x_2 \cdots x_n\}$

For reduced instance, we also need to find the leftmost point.

To do that, we sort the points set X with an increasing order. Then the time complexity should be associated with the sorting method.

For example, if we use heap sort, then the time complexity will be $O(n\log(n))$

After sort the sets, the time complexity for removing the points x_i is $O(n)$

So the overall time complexity is $O(n\log(n))$