BATCH SCHEDULING

A. Solution

This problem can be solved using dynamic programming. Let C_i be the minimum total cost of all partitionings of jobs J_i, J_{i+1}, … , J_n into batches. Let $C_i(k)$ be the minimum total cost when the first batch is selected as $\{J_i, J_{i+1}, \ldots , J_{k-1}\}$. That is, $C_i(k) = C_k + (S + T_i + T_{i+1} + \ldots + T_{k-1}) * (F_i + F_{i+1} + \ldots + F_n)$.

Then we have that

$C_i = \min \{ C_i(k) \mid k = i+1, \ldots , n+1\}$ for $1 \le i \le n$,

and $C_{n+1} = 0$.

(a) O(n²) Time Algorithm

The time complexity of the above algorithm is $O(n^2)$.

(b) O(n) Time Algorithm

Investigating the property of $C_i(k)$, P. Bucker[1] showed that this problem can be solved in $O(n)$ time.

From $C_i(k) = C_k + (S + T_i + T_{i+1} + \ldots + T_{k-1}) * (F_i + F_{i+1} + \ldots + F_n)$, we have that for $i < k < l$,

$$C_i(k) \le C_i(l) \Leftrightarrow C_l - C_k + (T_k + T_{k+1} + \ldots + T_{l-1}) * (F_i + F_{i+1} + \ldots + F_n) \ge 0$$
$$\Leftrightarrow (C_k - C_l) / (T_k + T_{k+1} + \ldots + T_{l-1}) \le (F_i + F_{i+1} + \ldots + F_n)$$

Let $g(k,l) = (C_k - C_l) / (T_k + T_{k+1} + \ldots + T_{l-1})$ and $f(i) = (F_i + F_{i+1} + \ldots + O_n)$

Property 1: Assume that $g(k,l) \le f(i)$ for $1 \le i < k < l$. Then $C_i(k) \le C_i(l)$
Property 2: Assume $g(j,k) \le g(k,l)$ for some $1 \le j < k < l \le$ n. Then for each i with $1 \le i < j$, $C_i(j) \le C_i(k)$ or $C_i(l) \le C_i(k)$.

Property 2 implies that if $g(j,k) \le g(k,l)$ for $j < k < l$, C_k is not needed for computing F_i. Using this property, a linear time algorithm can be designed, which is given in the following.

Algorithm Batch

The algorithm calculates the values C_i for $i = n$ down to 1. It uses a queue-like list $Q = (i_r, i_{r-1}, \ldots , i_2, i_1)$ with tail i_r and head i_1 satisfying the following properties:

$i_r < i_{r-1} < \ldots < i_2 < i_1$ and

$g(i_r, i_{r-1}) > g(i_{r-1}, i_{r-2}) > \ldots\ldots > g(i_2, i_1)$ -------- (1)

When C_i is calculated,

1. // Using $f(i)$, remove unnecessary element at head of Q.

If $f(i) \ge g(i_2,i_1)$, delete i_1 from Q since for all $h \le i$, $f(h) \ge f(i) \ge g(i_2,i_1)$ and $C_h(i_2) \le C_h(i_1)$ by Property 1.

Continue this procedure until for some $t \geq 1$, $g(i_r, i_{r-1}) > g(i_{r-1}, i_{r-2}) > \ldots.. > g(i_{t+1}, i_t) > f(i)$.

Then by Property 1, $C_i(i_{v+1}) > C_i(i_v)$ for $v = t, \ldots, r-1$ or
$r = t$ and Q contains only i_t.
Therefore, $C_i(i_t)$ is equal to $\min\{C_i(k) \mid k = i+1, \ldots, n+1\}$.

2. // When inserting i at the tail of Q, maintain Q for the condition (1) to be satisfied.
If $g(i, i_r) \leq g(i_r, i_{r-1})$, delete i_r from Q by Property 2.
Continue this procedure until $g(i, i_v) > g(i_v, i_{v-1})$.
Append i as a new tail of Q.

Analysis

Each i is inserted into Q and deleted from Q at most once. In each insertion and deletion, it takes a constant time. Therefore the time complexity is $O(n)$.

B. Test Data Information and Grading

In total, 20 test cases are prepared and tested. Each test case is of 5 credits. Among them, 19 test cases are randomly generated so that overflow does not occur during computing *Fi*. The remaining 1 test case is that setup time and all processing times and cost factors are 1.

The test cases are mainly prepared to distinguish whether the competitors design an efficient algorithm or not. Among 20 test cases, algorithm by enumeration may solve for three ones within the given time limit, and an $O(n^2)$ time algorithm may solve for 14 test cases within the given time limit. If the competitors submit a correct $O(n)$ time algorithm, they will get 100 credits.