Subproblems

Define $T(i, t_1, t_2, t_3)$ be the **minimum additional waiting time** spent by customer $i, i+1, \ldots, n$, given that server 1 have accumulated service time t_1. Same goes for server 2 and 3. The output we want is $T(1, 0, 0, 0)$, that is, the minimum additional waiting time spent by customer from 1 to n (i.e. all waiting time), given that all server have no accumulated time (i.e. all server free from start).

Recursive Formula

When all customer are assigned, there is no additional waiting time. So:

$$T(n+1, t_1, t_2, t_3) = 0 \qquad \forall t_1, t_2, t_3$$

and if customer from i to n total waiting time is the waiting time for customer i plus the waiting time for customer from $i+1$ to n, picking the minimum of customer i choosing different server:

$$T(i, t_1, t_2, t_3) = \min \begin{cases} t_1 + T(i+1, t_1 + \ell_i, t_2, t_3) \\ t_2 + T(i+1, t_1, t_2 + \ell_i, t_3) \\ t_3 + T(i+1, t_1, t_2, t_3 + \ell_i) \end{cases}$$

Suppose the customer comes in order from 1 to n, then it should be obvious that waiting time for customer i is t_j if assigned to server j, and the corresponding server j will have its service time increased by ℓ_i.

Evaluation Order

Since $T(i, t_1, t_2, t_3)$ depends on bigger i, we need to evaluate i from n to 1, in **decreasing order**. For a given i, we need consider all triples (t_1, t_2, t_3) such that $t_1 + t_2 + t_3 = \sum_{j=1}^{i-1} \ell_j$ as all service time in server equals to the time required by all customers that arrived so far.

Algorithms

Optimal value

Declare 4D array $T[n+1, nL, nL, nL]$ to be all 0. (as the service time for a server could be nL given that L is maximum possible service time for single customer).

For $i : n \to 1$

- For (t_1, t_2, t_3) such that $t_1 + t_2 + t_3 = \sum_{j=1}^{i-1} \ell_j$
  - Set $T[i, t_1, t_2, t_3] = \min \begin{cases} t_1 + T[i+1, t_1 + \ell_i, t_2, t_3] \\ t_2 + T[i+1, t_1, t_2 + \ell_i, t_3] \\ t_3 + T[i+1, t_1, t_2, t_3 + \ell_i] \end{cases}$

Return $T[1, 0, 0, 0]$

Optimal solution

Define Reconstruct(i, t_1, t_2, t_3)

- If $i > n$, return.
- If $T[i, t_1, t_2, t_3] = t_1 + T[i+1, t_1 + \ell_i, t_2, t_3]$
    - Output $(i, 1)$
    - Reconstruct$(i+1, t_1 + \ell, t_2, t_3)$
- Else if $T[i, t_1, t_2, t_3] = t_2 + T[i+1, t_1, t_2 + \ell_i, t_3]$
    - Output $(i, 2)$
    - Reconstruct$(i+1, t_1, t_2 + \ell, t_3)$
- Else
    - Output $(i, 3)$
    - Reconstruct$(i+1, t_1, t_2, t_3 + \ell)$

Call Reconstruct$(1, 0, 0, 0)$

Complexity

Notice that although we have three parameters for t_1, t_2, t_3. The constraint $t_1 + t_2 + t_3 = \sum_{j=1}^{i-1} \ell_j \leq nL$ makes the there are only $O(n^2L^2)$ possible configuration to choose. Given that we also iteration through i, the total time complexity is $O(n^3L^2)$. For space complexity, if we only consider the optimal value, then we only need to save previous layer, so to have a space complexity of $O(n^2L^2)$, if we care about optimal solution, then we could still use Hirschberg divide-and-conquer algorithm to reduce the complexity to $O(n^2L^2)$.