CS577: Homework 5

Haruki Yamaguchi
hy@cs.wisc.edu

Keith Funkhouser
wfunkhouser@cs.wisc.edu

October 15th, 2015

1 Algorithm description

Input: n non-negative morning bus route lengths $a_1, a_2, \ldots, a_n$, n non-negative afternoon route lengths $p_1, p_2, \ldots, p_n$, and a non-negative d for the maximum non-overtime workday length.

Output: a set of pairings $((a_{i_1}, p_{j_1}), (a_{i_2}, p_{j_2}), \ldots, (a_{i_n}, p_{j_n}))$, where the kth pairing represents the two routes that the kth driver is assigned, such that the total overtime amount the city authority must pay is minimized.

One greedy algorithm which produces the minimum total overtime that that the city authority must pay is as follows:

Sort the n morning bus routes (henceforth a) in *increasing* order (let their sorted order be $a_1, a_2, \ldots, a_n$) and the n afternoon bus routes (henceforth p) in *decreasing* order (let their sorted order be $p_1, p_2, \ldots, p_n$). Then, pair each a_i with its corresponding p_i (such that the shortest morning route is paired with the longest afternoon route and the longest morning route is paired with the shortest afternoon route). Return this set of pairings.

2 Correctness

An exchange argument for correctness of the algorithm follows.

Let the *overtime of the kth bus driver* under the pairings in S be denoted $o(S, k) = \max(0, (a_k + p_k) - d)$ (i.e. we only pay overtime if $a_k + p_k$ exceeds d). The *total overtime of the pairings* S, denoted $t(S)$, is the sum of the overtimes for each pairing produced under S, i.e. $t(S) = \sum_k o(S, k)$.

Let G denote the pairings produced by the greedy algorithm above, and let S denote any optimal pairing. The proof of correctness follows by showing that $t(G) \le t(S)$.

If $G = S$, then clearly $t(G) = t(S)$. Otherwise, $S \ne G$. For simplicity, we will henceforth refer to the pairings with A fixed in sorted order, which we can do because the returned set of pairings is unordered, i.e. there is no difference between all of the drivers. Since $S \ne G$, there must exist at least one set of two routes $i, j \in [1, n]$, $i < j$ where p_i and p_j are swapped in S compared to G. In other words, we have:

$$\text{Under } G\text{: } \begin{array}{l} a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n \\ p_1, \ldots, p_j, \ldots, p_i, \ldots, p_n \end{array}$$

$$\text{Under } S\text{: } \begin{array}{l} a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n \\ \quad\cdots\ \ p_i\ \ \cdots\ \ p_j\ \ \cdots \end{array}$$

(*Note:* we have $1 \le i < j \le n$, even though the above could be interpreted as having $1 < i < j < n$.)

Consider the pairings S', which agree with S except that we swap these two routes in our pairing. Clearly, this is still a valid set of pairings.

We can show that the total overtime of S' is at most the total overtime of S, i.e. $t(S') \le t(S)$. Note first that swapping p_i and p_j in the ordering only can affect $o(S', i)$ and $o(S', j)$, since the contributed overtime from the other pairings will be unaffected. Thus, it suffices to show that $o(S', i) + o(S', j) \le o(S, i) + o(S, j)$ to prove that $t(S') \le t(S)$. We will consider four cases and show that, in each, $o(S', i) + o(S', j) \le o(S, i) + o(S, j)$.

Since we are swapping i and j in S' to the sorted order they appear in under G and G has p sorted in decreasing order, we have:

$$a_i \leq a_j \tag{1}$$

$$p_i \leq p_j \tag{2}$$

1. $o(S,i), o(S,j) = 0$: Since $a_i \leq a_j$ and $p_i \leq p_j$, we have:

   $$a_j + p_j \geq a_j + p_i \tag{3}$$

   $$a_j + p_j \geq a_i + p_j \tag{4}$$

   Furthermore, since $o(S,j) = 0$, we have $a_j + p_j \leq d$. This, in addition to (3) and (4), yields $o(S',i) + o(S',j) = \max(0, (a_i + p_j) - d) + \max(0, (a_j + p_i) - d) = 0 + 0 \leq o(S,i) + o(S,j)$.

2. $o(S,i) = 0$, $o(S,j) \geq 0$:
   Consider these four subcases:

   - If $a_j + p_i > d$, then we have:

     $$o(S',j) = o(S,j) - (p_j - p_i) \tag{5}$$

     (Proof: $o(S',j) = a_j + p_i - d = a_j + p_j - d - (p_j - p_i) = o(S,j) - (p_j - p_i)$)

     – If $a_i + p_j > d$, then we have:

     $$o(S',i) \leq p_j - p_i \tag{6}$$

     (Proof: $o(S',i) = a_i + p_j - d = a_i + p_i - d + (p_j - p_i)$. But $o(S,i) = 0 \Rightarrow a_i + p_i - d \leq 0 \Rightarrow o(S',i) \leq p_j - p_i$)
     By (5) and (6), $o(S',i) + o(S',j) \leq o(S,j) = o(S,i) + o(S,j)$.

     – If $a_i + p_j \leq d$, then we have:
     $o(S',i) = 0$, and $o(S',i) + o(S',j) = 0 + o(S,j) - (p_j - p_i) \leq o(S,j) = o(S,i) + o(S,j)$.

   - If $a_j + p_i \leq d$, then we have:

     $$o(S',j) = 0 \tag{7}$$

     – If $a_i + p_j > d$, then we have:

     $$o(S',i) \leq o(S,j) \tag{8}$$

     (Proof: $o(S',i) = a_i + p_j - d$. But, by (4), $o(S,j) = a_j + p_j - d \geq a_i + p_j - d = o(S',i)$)
     By (7) and (8), $o(S',i) + o(S',j) = o(S',i) + 0 \leq o(S,j) = o(S,i) + o(S,j)$)

     – If $a_i + p_j \leq d$, , then we have:

     $$o(S',i) = 0 \tag{9}$$

     By (7) and (9), $o(S',i) + o(S',j) = 0 + 0 = 0 \leq o(S,i) + o(S,j)$.

3. $o(S,i), o(S,j) > 0$:
   The cases are identical to the above except:

   - If $a_j + p_i > d$, then (5) still holds.

     – If $a_i + p_j > d$, then we have:

     $$o(S',i) = o(S,i) + (p_j - p_i) \tag{10}$$

     (Proof: $o(S',i) = a_i + p_j - d = a_i + p_i - d + (p_j - p_i) = o(S,i) + (p_j - p_i)$) By (5) and (10), $o(S',i) + o(S',j) = o(S,j) - (p_j - p_i) + o(S,i) + (p_j - p_i) = o(S,i) + o(S,j)$.

   - If $a_j + p_i \leq d$, then we have:

     $$o(S',j) = 0 \tag{11}$$

– If $a_i + p_j > d$, then we have:

$$o(S', i) = o(S, j) - (a_j - a_i) \tag{12}$$

(Proof: $o(S', i) = a_i + p_j - d = a_j + b_j - d - (a_j - a_i) = o(S, j) - (a_j - a_i)$)
By (11) and (12), $o(S', i) + o(S', j) = o(S', i) + 0 = o(S, j) - (a_j - a_i)$. By (1), $o(S, j) - (a_j - a_i) \leq o(S, j) < o(S, i) + o(S, j)$.

If $S' = G$, we are done. Otherwise, S' is closer to G than S is, since it has fewer inversions in P to perform with respect to the greedy pairings. Repeating this swapping procedure will ultimately produce G, and since at each step we did not increase the total overtime, it follows that $t(G) \leq t(S)$ for any schedule S. Thus, the pairings produced by the greedy algorithm constitute the optimal solution.

3 Runtime

Sorting the arrays takes $O(n \log n)$ time, and pairing off the routes sequentially can be done in $O(n)$ time by simply traversing the sorted arrays once. Thus, the overall runtime of the algorithm is $O(n \log n)$.