October 12, 2015

CS 577 - Homework 5
Sejal Chauhan, Vinothkumar Siddharth, Mihir Shete

1 Graded written problem

Input: In a city there are n bus drivers. There are also n morning bus routes and n afternoon bus routes, each with various lengths. Each driver is assigned one morning route and one evening route. For any driver, if his total route length for a day exceeds d, he has to be paid overtime for every hour after the first d hours at a fixed rate per hour.

Output: Assign one morning route and one evening route to each bus driver so that the total overtime amount that the city authority has to pay is minimized.

1.1 Algorithm

Our greedy algorithm begins by sorting the n morning bus routes in reverse order (Longest route first) and the n afternoon bus routes in order. Then on these *ordered* routes we apply the following strategy:

1. Starting from the first morning route we try to find the *largest* afternoon route such that there sum is less than or equal to d.
2. If there is no such afternoon route so that the sum of the morning and afternoon route is less than or equal to d then we will pair the smallest available afternoon route with the given morning route.
3. Rule *2.* Also implies that if morning route is greater than d we will pair it with the smallest available afternoon route.

Consider the following morning and evening routes ordered as per our algorithm:

$Morning : \{7, 4, 3, 1\}$ $Afternoon : \{2, 3, 6, 9\}$ $d : 8$

In the above example we will start from the largest morning route 7, we can see that the smallest available afternoon route and the largest morning route add to a value greater than d, since there is no available route in the afternoon with which we can pair the largest route in the morning to get the total duration less than d we will pair 7 and 3 (i.e the smallest available afternoon route). Similarly, we will pair (4, 3), (3, 6) and (1, 9) according to our greedy algorithm.

If we do time complexity analysis of the *strategy* of pairing the routes we can see that in the worst case we will have to compare the 1^{st} route in the morning to n routes in the afternoon, the 2^{nd} route in the morning to $n-1$ routes in the afternoon and so out time complexity will be $O(n^2)$. To better the running time of our strategy we will use the following algorithm (a variant of binary search). This algorithm will take key as one of the inputs where $key = d - m_i$, m_i is the route duratio of the i^{th} morning trip and will return the best match (as decribed in our strategy) from the array A of size $|A|$ which is the array of oredered Afternoon trips.

Algorithm 1 Find-Pair

```
1: procedure FIND-PAIR(key, A, left, right)
2:     if left ≥ right then return left
3:     mid ← ⌊(right + left)/2⌋
4:     if key<A[mid] then
5:         return FIND-PAIR(key, A, left, mid − 1)
6:     else if key>A[mid] then
7:         return FIND-PAIR(key, A, mid + 1, right)
8:     else if key = A[k] then
9:         return k
```

If $key = d - m_i$ is present in the array A then **Find-Pair** will return an index k such that $A[k] = key$ and we can directly pair this trip of length $A[k]$ with trip m_i. if key is not present and $A[k] < key$ then we will also pair k with m_i, but if $A[k] > m_i$ then we will pair m_1 with $A[k-1]$ if $k>0$. The runtime of **Find-Pair** is $log(n)$ because we are divind the array in half for every iteration and using this algorithm we can complete our strategy on ordered arrays in $O(nlog(n))$ time.