

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. To find a set  $G\{(m, a) : \forall m \in M \wedge \forall a \in A\}$ , where  $M$  is set of all morning routes and  $A$  is the set of all evening routes, we will use the following *strategy*:

1. Starting from the first morning route we try to find the *largest* afternoon route such that their sum is less than or equal to  $d$ .
2. If there is no such afternoon route such 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 2 (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 on. Hence, our 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 duration of the  $i^{th}$  morning trip and will return the best match (as described in our strategy) from the array  $A$  of size  $|A|$  which is the array of ordered Afternoon trips.

---

### Algorithm 1 Find-Pair

---

```

1: procedure FIND-PAIR(key, A, left, right)
2:   if left > right then return left
3:   mid  $\leftarrow \lfloor (\textit{right} + \textit{left}) / 2 \rfloor$ 
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 dividing the array in half for every iteration and using this algorithm we can complete our strategy on ordered arrays in  $O(n \log(n))$  time.

Let  $M$  be the array of all ordered morning trips and  $A$  be the array representing all ordered afternoon trips, our *strategy* will be represented by the following Pseudocode:

---

**Algorithm 2** Pseudocode of our greedy strategy

---

```

1: procedure FIND-ALL-PAIRS( $M, A$ )
2:    $i \leftarrow 0$ 
3:   while  $i < |A|$  do
4:     if  $M[i] \geq d$  then
5:       Pair  $M[i], A[0]$ 
6:       Delete  $A[0]$  from  $A$ 
7:       continue
8:      $key \leftarrow d - M[i]$ 
9:      $index \leftarrow \text{FIND-PAIR}(key, A, 0, |A|)$ 
10:    if  $A[index] \leq key$  then
11:      Pair  $M[i], A[index]$ 
12:      Delete  $A[index]$  from  $A$ 
13:    else
14:      if  $index = 0$  then
15:        Pair  $M[i], A[index]$ 
16:        Delete  $A[index]$  from  $A$ 
17:      else
18:        Pair  $M[i], A[index - 1]$ 
19:        Delete  $A[index - 1]$  from  $A$ 

```

---

## 1.2 Exchange Argument

Let  $r$  be all the route-pairs in a solution that are greater than  $d$  and let  $l_i$  be the length of the route-pair.  $p(G) = \sum_r (l_i - d)$  for our greedy solution  $G$ , similarly for an optimal Solution  $S$   $p(S) = \sum_r (l_i - d)$

If,  $G = S$  then clearly  $p(G) = p(S)$ .

Otherwise,  $G \neq S$ . So, there must be some 2 routes in  $G$  and  $S$ , such that the morning routes are the same but the paired afternoon routes are different. More formally, if  $(M_G(i), A_G(i)), (M_G(j), A_G(j))$  are 2 pairs in  $G$  and  $(M_S(k), A_S(k)), (M_S(l), A_S(l))$  are 2 pairs in  $S$ , then

$$M_G(i) = M_S(k) \ \& \ M_G(j) = M_S(l)$$

but,

$$A_G(i) \neq A_S(k) \ \& \ A_G(j) \neq A_S(l)$$

Let,  $M_{ik} = M_G(i) = M_S(k)$  and  $M_{jl} = M_G(j) = M_S(l)$  Now, consider the following scenarios in  $S$ :

$M_{ik} > M_{jl}$

Let  $S'$  denote a solution which would have selected the  $(M_{ik}, A_S(l))$  and  $(M_{jl}, A_S(k))$  pair contrary to  $S$ .  $S'$  is similar to  $G$  in a way that  $G$  would also have chosen the same pairing as  $S'$ . So, given the constraint on morning routes  $S'$  will choose the afternoon route first for  $M_{ik}$  and it will pair with  $A_S(l)$  if:

1.  $A_S(l) > A_S(k)$  **and**  $M_{ik} + A_S(l) \leq d$

In this scenario, it is trivial to see that  $M_{jl} + A_S(k) \leq d$  as both morning and afternoon trips are lesser than their counterparts i.e  $M_{jl} < M_{ik}$  and  $A_S(k) < A_S(l)$ . So, the choice of pair for  $S'$  does not yield any overtime. Hence,  $p(S') \leq p(S)$

2.  $A_S(l) < A_S(k)$  **and**  $M_{ik} + A_S(l) > d$

For the above constraints  $M_{jl} + A_S(k)$  can be less than, more than or equal to  $d$ , so let's analyze all the conditions: Here, if  $M_{jl} + A_S(k) > d$  it is trivial to see that  $p(S') = p(S)$  since the extra hours for both the selections will be  $2d - M_{ik} - M_{jl} - A_S(k) - A_S(l)$ .

If,  $M_{jl} + A_S(k) \leq d$  then it implies that  $M_{jl} + A_S(l) \leq d$  from the constraints.

$$p(S') = M_{ik} + A_S(l) - d$$

And

$$p(S) = M_{ik} + A_S(k) - d$$

And since  $A_S(l) < A_S(k)$  it is clear that  $p(S') < p(S)$

From the above analysis it is clear that  $p(S') \leq p(S)$  whenever the order is changed according to greedy strategy. Hence, our greedy strategy will provide an optimal solution.

### 1.3 Running Time Analysis

To sort get the routes in correct order for our greedy algorithm to process we will have to spend  $O(n \log n)$  time. The **Find-All-Pairs** procedure will iterate over all  $n$  elements in  $M$  and all the operations in these iterations are constant time except **Find-Pair**.