HW 4

Yunhao Lin

October 9, 2018

1. In order to prove S is feasible if and only if $\forall t \leq n$, the number of deliveries in S due within t days is no more than t. We need to prove that if S is feasible, then for $\rightarrow \forall t \leq n$ the number of deliveries in S due within t dyas is no more than t. And also for $\forall t \leq n$ the number of deliveries in S due with in t days is no more than $t \rightarrow S$ is feasible.

   - We will use induction to prove that for every $t \leq n$, the the number of deliveries in S due within t days is no more than t, then S is feasible.
     Claim: For every t is smaller or equal to n, the number of deliveries due within t days is no more than t, then S is feasible.
     Base case: Zero customer is not meaningful to the process, therefore we choose 1 as base case. There is only one customer which means, $n = 1$. Since we know that only one package can be delivered in one day. So the number of package that can be delivered in one day is no bigger than 1, which means the there is a way to deliver the one package in time. Our base case holds.
     Inductive Hypothesis: $P(n)$:Assume for $\forall t \leq n$, the number of number of deliveries in S due within t days is no more than t, then S is feasible.
     Induction Step: We need to prove for $\forall t \leq n + 1$, the number of deliveries in S due within t days is no more than t, then S is feasible. Since P(n) works, then the number of deliveries in S due within t days is no more than t, and we can deliver at most t packages, for n +1, we have one extra package to deliver. And that package can be delivered on the extra day, since every t smaller than n+1, then the compared with P(n) there is one extra day to deliver the n+1 package. So, there is still a way to manage deliver all the package, which mean P(n+1) holds and S is feasible.

     Therefore, $P(n) \rightarrow P(n + 1)$. By induction, for every t smaller or equal to n, the number of deliveries is no more than t, then S is feasible.

   - If S is feasible, by definition, there is a way to order the deliveries in S so that each one is made on time. Assume S has n packages, as we know that only one package can be delivered in each day. So the least time to deliver n packages would be n days. Let's assume $\exists t > n$, then the number of deliveries in S due within t days is no more than t. So every day untill t there can be a package delivered, so in total the maximum packages delivered could be t packages. And by assumption, $t > n$, which contradicts the statement that deliveries in S due within t days cannot be more than t. By contradiction, $\forall t \leq n$, the number of deliveries in S due within t days is no more than t. So no matter what n number it is for $t \leq n$, there is always a way to deliver as more as n number of packages, which means each package is delivered and it's feasible.f S is feasible, then for $\rightarrow \forall t \leq n$ the number of deliveries in S due within t dyas is no more than t.

2. In order to maximize the profit. First we sort the due date t_i in descending order. Therefore at 0 index is going to be the last due date. We then traverse through the due date in order from last to first. At each iteration, add all the p_i corresponding to the day of the largest due date. Dequeue from the priority queue to get the highest profit available at that day. By doing this, we guarantee that each day we choose is going to give us the largest profit untill we reach the end of the package list.

Algorithm 1 Find maximum

```
procedure MAXPROFIT
    Initialize $q$ = priority queue
    Quick Sort the List of $t_i$ in descending order to be $L$
    $k \leftarrow$ largest due date
    while k is not equal to 0 do
        Create a list of all packages due at $k$                    ▷ Get the due date from $L$
        Add every $p_i$ at $k_{th}$ day to $q$
        Get highest priority $\beta$ from $q$ to gain highest profit on that day
                                                    ▷ if $q$ is empty, there is no delivery
        Mark dilivery of the package corresponding to $\beta$ at $k_{th}$ day
        $k \leftarrow k - 1$                                 ▷ Update to the next date in $L$
```

Proof Of Correctness:
We will use induction to prove that algorithm would give us the maximized profit.
Claim: The algorithm above would arrange deliveries delivers the largest profitable package at each day.
Base case: Our base case is when k is maximum k, the largest due date is k. Therefore a list of package is created at line 6 and add to q, the priority queue is added with all the package that has a due date of k. When we get highest priority from q, the largest number is returned so that is the most profitable job we need to deliver at k days. Then we just find the corresponding package to diliver. Therefore at k_th day we deliver the most profitable package. Our base case holds.
Inductive Hypothesis: Algorithem would produce the most profitable delivery at $k = x$, where x is less than the maximum of possible k.
Inductive Step: We need to prove that algorithm would get the correct delivery at day $k = x - 1$. After $k = x$ days and the package delivered. There is two possible situations at $k = x - 1$:

- First, the queue after the $k = x$ delivery is not empty. Then we first add all the package price due at $k = x - 1$ to the queue, therefore the queue contains everything that is due after $x - 1$, which means these are all valid package to deliver. Then algorithm find the maximum price we can do at $x - 1$, then at $x - 1_{th}$ day, the package that can get us most profit is marked to be dilivered at this date.
- Second, the queue after the $k = x$ is empty. Then as above, algorithm would find the package that can give us the greatest profit at $k = (x - 1)$ day. Then the correct package would be marked to b dilivered at $k = (x - 1)$ day.

Therefore, the P(k) holds then P(k-1) holds. The algorithm would end when the date is 0, by induction, every day before k would deliver the most profitable package at the day. Therefore in total, that is the maximum profit we can get. And each package to be delivered is marked at the correct day to be delivered.
The algorithm's complexity would be $O(klog_k)$, since we are doing k times every time the complexity is log_k. Our algorithm would detemine the correct order to deliver package to gain max profit.