* **[Question 1]**

Sort set *A* and *B* in the decreasing order, and then let be the *i-th* element of set *A* so that be the greatest integer in the set *A*, and let be the *i-th* element of set *B* so that be the greatest integer in the set *B.* Then can receive the payoff on . This algorithm has runtime complexity since we need to sort the sets. However, if these two sets are sorted already, then then runtime complexity can be .

*Proof.*

Let’s prove its optimum by induction:

Base case: when *i = 1*, has the greatest payoff.

Inductive hypothesis: for any integer *k* such that , has the greatest payoff.

Inductive step: has the greatest payoff.

~~And we will prove the Inductive step by contradiction, so let’s assume that there is that has greater payoff, where is the~~ *~~i-th~~* ~~element of set~~ *~~A’,~~* ~~and is the~~ *~~i-th~~* ~~element of set~~ *~~B’~~* ~~and sets~~ *~~A’~~* ~~and~~ *~~B~~*~~’ denotes set~~ *~~A~~* ~~and~~ *~~B~~* ~~that are reordered in a way other than ours.~~

And we will prove the Inductive step by contradiction, so let’s assume that our algorithm is not the most optimal, which means, by swapping elements of our current sets at least once, we will have greater payoff.

For instance, if:

\*is our original (for )

then:

\*has the greater payoff.

However, since we know that and >, thus:

, which is greater than 1, which means > .

Now we have proven that each time we swap the payoff becomes less, however we assume that swapping makes payoff greater. That is a contradiction!

* **[Question 2]**

At each gas station, check is the gas going to be enough to reach the next gas station. If yes, then don’t stop, otherwise stop and get gas. The time complexity is *O*(*n*).

*Proof.*

We are going to prove the algorithm’s correctness and optimum by contradiction, which means we assume the algorithm is not optimal. Since our algorithm is to keep driving and only stop at a gas station if gas in the tank is not enough to reach the next station, thus the optimal solution must be stopping at the station earlier than that and reach the destination with less stops. However, that is not possible since if we stop earlier, then next time we have to drive longer in order to stop at the same place, or if the gas isn’t enough to reach the same place, then we have to stop earlier again and try to drive longer and reach the next same place next time. If we keep stopping earlier than the original, then we end up stopping one more time than the original, or the number of stops can be at best as same as the original if we catch up stopping at the same station.

* **[Question 3]**

Let there be two pointers: *p* and *p’*, and let *p* point to the front-most event of *S* and let *p’* point to the front-most event of *S’*, and then do the following until any of *p* or *p’* has reached the end of its sequence:

* If *p* and *p’* point to the same event, both move on to the next event of their sequences.
* If *p* and *p’* point to different events, then *p* moves to the next event of its sequence.

if any of *p* or *p’* has reached the end of its sequences:

* If either *p’* has reached the end first, or *p* and *p’* both have reached their ends at the same time, then return true because *S’* is a subsequence of *S*.
* Otherwise, if *p* has reached the end first, then return false because *S’* is not a subsequence of *S*.

The time complexity of this algorithm is *O*(*n+m*).

*Proof.*

We are going to prove it by Induction.

The base case:

* **[Question 4]**

Let

Initial state:

Truck t has the

Do it:

Step 1: