Q1 ① Algorithm which maximize my payoff: I want to use Greedy Algorithm
First, we should sort set A from largest to smallest that $a_1 \geq a_2 \geq \cdots \geq a_n$ and sort set B that $b_1 \geq b_2 \geq \cdots \geq b_n$. Finally, we return $\prod_{i=1}^{n} a_i^{b_i}$.

② Prove the algorithm maximize payoff:
I want to use a contradiction to prove this algorithm.
First, we assume that the maximize payoff cannot work out with this algorithm
Assume we already get a maximize payoff, which is a_1 and b_x, a_x and b_1, obviously. $a_1 \geq a_x$ and $b_1 \geq b_x$

$$\text{Payoff (maximize 1)} = \prod a_i^{b_i} = a_1^{b_x} a_x^{b_1}$$

Assume another maximize payoff, which is a_1 and b_1, a_x and b_x

$$\text{Payoff (maximize 2)} = \prod a_i^{b_i} = a_1^{b_1} a_x^{b_x}$$

Then $$\frac{\text{Payoff (maximize 1)}}{\text{Payoff (maximize 2)}} = \left(\frac{a_1}{a_x}\right)^{b_x - b_1}$$

We already know a_1 and b_1 is the largest in each set, the $a_1 > a_x$ and $b_1 > b_x$, the $\left(\frac{a_1}{a_x}\right)^{b_x-b_1} < 1$

Obviously, maximize payoff 1 is less than maximize payoff 2

This contradicts our assumption that payoff 1 is maximize payoff.

③ Complexity of this Algorithm.
(a) set A and set B are already sorted in nondecreasing order, time complexity is $O(n)$
(b) set A and B are not sorted, the time complexity is $O(n\log n)$, because we have to use more time to sort A and B.

Q2: I want to use greedy algorithm.

Pseudo-code:

```
i = 1
travel_distance = distance[i]
while (di+1 != di) {
        travel_distance = Distance[i]
        while ((di+2 is not empty) & (d ≥ travel_distance))
        {       travel_distance = travel_distance + distance[i]
                temp_destination = di+1
                i = i+1
        }
        set.add(temp_destination).
        }
return set
```

Proof: We use contradiction to prove greedy algorithm is optimal solution for this problem.

Assume greedy algorithm is not optimal, then there are an optimal solution exist. Assume solution set of greedy algorithm is $\{G\}$, solution set of optimal solution is $\{O\}$. As we all know, number of elements in $\{G\}$ must more the elements in $\{O\}$

① If the first element in $\{G\}$ which is G_1 is equal to O_1, then from start point to G_1 equal to start point to O_1, the route before G_1 is same to the route before O_1 at least, if $O_2 = G_2$, then route before G_2 same as route before O_2 ($S \to G_2 = S \to O_2$) and so on. If all elements in $\{G\}$ equal to all elements in $\{O\}$, then $\{G\}$ is optimal solution, greedy algorithm is optimal.

② If the first element in $\{G\}$ is not equal to first element in $\{O\}$, then delete O_1, swith O_1 to G_1 in $\{O\}$, $\{O\}$ is still optimal, because from O_2 to last element O_n is optimal.

No.
DATE / /

Now, the problem reduced to whether from O_1 to Santa Monica is optimal or not
If {O} is optimal, then {O} minus G_1 is also optimal.
The reason is that if there are a set {S} is optimal from G_1 to Santa Monica, then {S} + G_1 smaller than {O}, but that's impossible, because {O} is optimal, thus {G} and {O} is identical.
Thus, {G} is optimal set and greedy algorithm is optimal

Time complexity: This greedy algorithm has 2 while loops

```
while{
    while{
        i=i+1
    }
}
```

Assume there are n gas stations and n-1 distance between gas stations
In the worst case, inner while loops run n-1 times. Outer while loop iterator at most n times.
If we want to stop this algorithm, takes O(n)

Q3. Algorithm: We can use two pointers refer to two strings, move from left to right. If two pointers refer to different characters, then the pointer pointing to S moves to the right. When the pointer pointing to S' moves to the end of string S' (which matches all characters of S' exactly), then S' is a substring of S, and if the pointer pointing to S' is not pointing to the end of S' and does not traverse all characters of S', then S' is not a subsequence of S.

Pseudo-code:

```
if S' is empty:
return True // the empty sequence is also a subsequence of the sequence
if length of S' is greater than length of S:
return False

i, j = 0, 0
while i < length of S:
    if S[i] == S'[j]:
        i, j = i+1, j+1   // move two pointers two the right
    else:
        i = i + 1   // move pointer of S' to the right

if j == length of S':
    return True
else:
    return False.
```

Time complexity:

The time to traverse S + The time to traverse S'

$O(m) + O(n) = O(m+n)$

Q4: First, assume we have n packages (1, 2, 3, ..., n) and assume the weight of each package x is W_x.
The sequence of truck T is a non-decreasing sequence, because the (i+1)th truck cannot be placed packages before sending off the ith truck (According to definition of Greedy Algorithm)
We use $T_n = m$ to represents that n trucks sending off m packages.

I want to use a contradiction to prove greedy algorithm for this problem must be optimal.
First, we assume that greedy algorithm is not optimal solution.
thus, the optimal solution should be T'_n (represents how many trucks we have to use in minimum using optimal solution).
Thus, $T'_n < T_n$.
Now, we have to consider a point, when do we switch trucks.
we can assume that we switch trucks at the Sth truck.

$$\begin{cases} T_x = T'_x, & \text{if } i < S \text{ (Before switch truck)} \\ T_S \mathrel{!}= T'_S, & \text{if } i = S \text{ (switch truck)} \end{cases}$$

① The optimal method switch trucks before Greedy Algorithm.
$T'_S = T_S + C$ (constant)
The original T'_S become $T'_S - C$, thus Truck T'_S cannot exceed the weight limit. (we reduced C packages than optimal solution)
Truck $T'_S - C$ cannot exceed weight limit, because this truck $T'_S - C$ has less package than it did in greedy algorithm.

We can assume another solution, which should be T''_n

$$\begin{cases} T''_x = T_x, & x <= S \\ T''_S = T'_x, & x > S \end{cases}$$

T''_n should be a correct solution, because T'_S and $T'_S - C$ cannot exceed weight limit.

No.
DATE / /

When we do not change the number of trucks, so we create an optimal solution matching Tx greater than $T'x$, which contradists definition of $T'x$. As s approaches n infinitely, the number of trucks decreases accordingly, which contradicts the definition of $T'x$.

② $Tx = T'x + c$, the greedy algorithm switch trucks before the optimal solution.
When the truck is overpacked, greedy algorithm will switch truck.
Since $Tx = T'x$, then greedy algorithm and optimal method will switch truck at the same time.
We have already assumed that optimal method is better than greedy algorithm, but these two solutions are identical, so it's a contradiction.

Thus, greedy algorithm is optimal method of this problem.