P1

(a) Since **Merge($L_1,\ L_2$)** takes $\mathrm{O}(\frac{n}{3}+\frac{n}{3})=\mathrm{O}(n)$ running time. The new array merged is of length $\frac{n}{3}+\frac{n}{3}=\frac{2n}{3}$. Thus **Merge**$(L_0,\ Merge(L_1,\ L_2))$ takes $O(\frac{n}{3}+\frac{2n}{3})+O(n)=O(n)$ running time.

(b) $T(n)=3T(\frac{n}{3})+\Theta(n)$

(c) According to the **Master Theorem**, $T(n)=3T(\frac{n}{3})+\Theta(n^1)$, $1=log_3 3$. Thus we have $T(n)=\Theta(n\ log\ n)=O(n\ log\ n)$

P2

Assume the DP-state $c[i,\ j]$ represent the LCS between the first i characters in S_1 and the first j characters in S_2. We have:

$$c[i,\ j]=\begin{cases} x=0 & if\ i=0\ or\ j=0 \\ c[i-1,\ j-1]+1 & if\ i,\ j>0\ and\ S_{1,i}=S_{2,j} \\ max(c[i,\ j-1],\ c[i-1,\ j]) & if\ i,\ j>0\ and\ S_{1,i}\neq S_{2,j} \end{cases}$$

The result is:

```
     j   0   1   2   3   4   5   6   7   8   9   10
i        S2  'i' 'n' 'e' 'q' 'u' 'a' 'l' 'i' 't' 'y'
0    S1  0   0   0   0   0   0   0   0   0   0   0
1    'q' 0   0   0   0   1   1   1   1   1   1   1
2    'u' 0   0   0   0   1   2   2   2   2   2   2
3    'a' 0   0   0   0   1   2   3   3   3   3   3
4    'n' 0   0   1   1   1   2   3   3   3   3   3
5    't' 0   0   1   1   1   2   3   3   3   4   4
6    'i' 0   1   1   1   1   2   3   3   4   4   4
7    't' 0   1   1   1   1   2   3   3   4   5   5
8    'y' 0   1   1   1   1   2   3   3   4   5   6
```

From the result, we can see that the **LCS** between S_1 and S_2 is `"quaity"`, whose length is 6.

P3

$\mathrm{O}(n\ log\ n)$ Algorithm:

We sort all the passenger's information with end_i ascending. According to their end_i, we can put the i_{th} person's information i in the set P_{end_i}. Note that we specially deal with passengers whose $start_i=end_i$. We do not put them in any set, but add their tip_i to S_{end_i}, which means we can obtain all their tips when we are available at end_i.

Formally speaking.
$P_i=\{x: end_x=i\ \land\ start_x\neq end_x\},\ S_i=\sum_{x\in\{x:end_x=i\ \land\ start_x=end_x\}}tip_x$

We defined the DP-state function f_i represent the maximum number of dollars we can get when we are at point i with no passenger on the taxi (There can be a passenger drop off at point i).

$$f_i=\begin{cases} 0 & if\ i=0 \\ max(f_{i-1},\ MAX_{p\in P_i}(f_{start_p}+(i-start_p)+tip_p)) & if\ i\neq 0 \end{cases}+S_i$$

Note that $MAX_{p\in P_i}$ means we choose the max value for the result of all p in the set P_i.

$\mathrm{O}(n)$ solution:

We can find that because the range of $start_i$ and end_i are within $1 \sim n$. So we do not need to sort in our previous solution. We can directly divide all the information of passengers in corresponding set.

P4

Algorithm:

First, for all the rectangles, we assumed that their length l_i are larger or equal to their width w_i. (If not, we can swap their length and width).

Sort all the rectangles with their **length** in ascending order.

We have the DP-state function F_i means the maximal number of rectangles ending with i_{th} rectangles.

$$F_i = \begin{cases} 0 & if\ i = 0 \\ 1 & if\ i = 1 \\ max_{0 \le j < i,\ w_j < w_i,\ l_j < l_i}\{F_j + 1\} & if\ i > 1 \end{cases}$$

And the final answer is $max\{F_i\}$

Proof:

For the first step we swap the length and width for $w_i > l_i$. It can be proved that if a solution have a rectangle with $w_i > l_i$. The rectangle contains i is k, the rectangle be contained by i is j. We have $l_k > w_i > l_j,\ w_k > l_i > w_j$.

Since $l_k > w_i > l_i > w_j,\ l_j < w_i < l_i < w_k$, we have $l_k > l_i > l_j,\ w_k > w_i > w_j$. So, if we do rotate, we cannot find a better solution.

And for the second step, we sorted their l_i in ascending order. When we perform DP later, we have consider all the F_j whose $l_j < l_i$. Thus the correctness can be ensured.

Complexity:

For the first part we need to check each rectangle with complexity $\mathrm{O}(n)$

For the next sorting part the complexity is $\mathrm{O}(n\ log\ n)$

And the DP part we need check all the previous j for each i. Thus the complexity is $\mathrm{O}(n^2)$

The total complexity is $\mathrm{O}(n) + \mathrm{O}(n\ log\ n) + \mathrm{O}(n^2) = \mathrm{O}(n^2)$