Problem 1: **Road Trip**:

a) Because we want to live in fewer hotels, we have to go farther as possible. Then we will check the max distance of the hotel we can reach in the daytime and keep repeating until we reach the final goal. In this condition, we must assume that the distance between hotels is no more than the distance I can go, or I can't reach the destination just by driving in the daytime.

if the max distance of driving in the daytime is "d", and |xi-xi-1|<=d

then the pseudocode would be:

```
for hotel_index in (0, n)
    for i in (hotel_index, n):
        if xi>d:
            pick hotel xi-1
            break
        if xi ==d:
            pick hotel xi
            break
        if xi <d:
            continue
```

b) The running time is n*n. however, the run time depends on distance conditions. The best case would be if the last hotel is matching the max riding distance, then the running time is 1. If we always compare hotels, with the rest of the hotels, the worst case is 1+1+1….1 so the average running time is theta(n).

Problem 2: CLRS 16-1-2 Activity Selection Last-to-Start Greedy Criteria

a) if we started with the last activity, then we check the shortest time from the end day of the last activity and keep repeating the same procedure.

b) It is an optimal solution. If it is not, the shortest time from the preview day has a shorter time than this shortest time activity. It is a logical paradox.

Problem 3: Activity Selection Last-to-Start Implementation

Last-to-start: Description:

The last activity is ensured, so we need to check back from the last activity and ensure more activities are attended. We assume the last activity is n, the start day of the n-1 activity should be shortest to the end day of the n activity. then repeating the procedures until all possible activities are attended.

Last-to-start: Pseudocode:

1: sort the end days of all activities, from early days to late days

2: find the last activities n

3: find the shortest activities from the last activities, the “start of the activity” to “the end of the activities n” should be the shortest path from all activities, then record this activity.

4: replace the last activities n as activities n-1

5: repeat step 2

Last-to-start: Theoretical running time and explanation

The run time has 3 loops. 1 loop for sorting the activities' order. 1 loop for searching all activities. 1 loop compares activities for the shortest path. so the running time is n^2+ n^2, so the final runtime is theta(n^2)