Name: Kerry Zheng Student ID: 28794346

FIT2004: Prac 6

Task 1: Salesman's Dilemma

Let N be the number of houses in the town.

In my algorithm, I read the data from the file, storing the profits from each house into a list with a 0 at the start of that list, let's call data list. This has time and space complexity of O(N).

Then by using dynamic programming, I create a memo list that is the exact same as the data list. Let i be the house I am currently checking and k be the user input of the amount of neighbouring houses on one side of i, and j be the house that is the closest non-neighbour of i. Keeping track of the house with the largest profit that is not the neighbour of the house (initialising this to be the first element of memo list) I am currently checking, I iteratively loop through all the houses. I iteratively loop through the houses until I found a valid j on the left of i such that $i - j > k$. Then I check if the profit of j is bigger than the variable with the largest profit that is not the neighbour of the previous house. If it is, we add the profit of j to the memo list at index I, to memoise it. Then we update the largest profit variable to be equal to the profit at j, otherwise we add the largest profit variable to the memo list at index i. Because this is a linear scan of the memo list, it has O(N) time complexity and space complexity.

In order to fetch the optimal solution, we backtrack it. To do that, we first do a linear scan to find the max profit in the memo list. Once we have done that, we then keep track of the max profit and add the house number to a list. To find the other houses, we subtract the max profit from the corresponding house profit in the data list, let's call backtrack max profit, and then move left in the memo list till we find a number that matches backtrack max profit and then do the same. Then, because we require an ascending solution, we iteratively pop and append the descending solution into a list. This has time and space complexity of O(N).

Task 2: Playlist

Let N be the number of songs, and T be the time to reach the destination

Firstly, we record the number of songs. Then we put all the song durations into a list with a 0 at the start. This has time and space complexity of O(N).

To memoise the playlist length, I initialise a memo table of size $(N + 1)$ by $(T + 1)$, with False values for each cell. Then I iteratively loop from 0 to n, looking at the song duration of the song list at index n. If the song duration is less than or equal to T, then we set the row we are currently on and column that corresponds to the song duration to be true. Then we iteratively loop from 0 to T, and if the previous row of the column we are currently searching is true, then we set the row we are currently on and the column to be true, and also the column number plus the song duration at the current row we are on to be true as well if that number is less than or equal to the playlist length. This has time and space complexity of O(NT).

Then, to find our optimal solution we backtrack our memoisation. First of all we check if the bottom right corner cell is true. If it's false, then that means we cannot construct a playlist length that exactly matches T. Otherwise, we iteratively check each row of that column until the previous row and that specific column we are currently on is false. If it is, that means the row we are currently on corresponds to a song that is in the playlist. Then we subtract the column number by the corresponding song length of the row we are currently on to the backtrack playlist length. Record the song duration along with the id which we determine by the row number we are currently on to the solution list. Again, because we require an ascending solution, we iteratively pop and append the descending solution into a list. This has time complexity of O(N) and space complexity of O(NT).