Halil Onur Çeçen
161044057

CSE 321 HW4

Part 1

In this part i started iterating over hotels to see the next hotel is viable or not. To evaluate its fitness i compared next hotel with the hotel after that repeatedly until i couldn't get a better penalty (hotel being far from optimum distance). After evaluating its fitness i increased total distance and total penalty with their respective current values.

To evaluate minimum penalty trip i used 2 loops nested both being capable of looping n times. This makes time complexity of **$O(n^2)$**.

Part 2

In this part i first i used two while controls but in total they makes one loop over given string being its time complexity **O(n)**. To check if given string reconstructable or not i checked strings first characters repeatedly until i find a match with True value in dictionary.

Part 3

In this part i took a variable number of arguments as input since we dont know how many lists we are going to merge together. If only one argument given returned that list and if multiple lists have been given dived them and again called the same method with that values. After first and secon halves merged i merged them assuming they're two sorted lists.

Time Complexity of algorithm is **T(k) = 2T(k/2) + 2n => T(k.n) = O(k) + O(n)**

Part 4

In this algorithm first i created given relations as a map for each person with relatives being key and his/her relatives being its values in a list. This way i would Access those relatives quickly. Using relation map that i created filled invitee list with respect to relative counts. I also added all relatives of a person if that persons relative count matches with requirements. But this would result in added relatives being not knowing other people enough, so i looped through invitee list repeatedly to extract people that doesnt match requirements until no change has been done. Also while looping through invitee list i extracted people that isnt in the party from the relation map so to make relation map invitee based.

The running time of the algorithm is T(NxR) = O(n*n*r) + O(n*r) + O(n*n*r) = **$O(n^2 * r)$**

(r being maximum number of relations for a person)

Halil Onur Çeçen
161044057

Part 5

In this part first i seperated equality and inequality in a map with keys False/True. After that i created an undirected graph with n vertices to represent their equal or inequal values as distinct nodes being different than others meanwhile connected ones being equal with each other. So if i have an inequality constraint i couldnt be able to find a path between those 2 edges. So i inserted True values as edges and after that checked every Fail value to find a path. If there is a path it means we can not satisfy given constraints.

Time complexity of algorithm is **T(n) = O(n + m) (m being constraint count)**

(Assuming path finding algortihm is O(1) since graph path finding is not our question.)