Assignment 2

Erik Norlin, CID: norliner

September 10, 2023

Problem 3

a)

This algorithm needs to run in cubic time because we have n^2 pairs of houses (including duplicates and pairs with themselves). In addition to this we have to compute the distances between every pair of houses. The longest distance to compute is from H_1 to H_n which is an iteration of $n-1$ times. The number of computations must therefore be $\approx n^3$ at most which $\in O(n^3)$.

Pseudo code for this algorithm

```
D = [0] # n*n array to contain computed distances
for i = 1,...,n
    for j = 1,...,n
        if i < j
            for k = 1,...,j-1
                D[i,j] += x_k # x_k is the distance from H_k to H_(k+1)
            D[j,i] = D[i,j] # Making array symmetric
```

The number of computations in this algorithm is $\approx \frac{n^3}{2}$, $\frac{1}{2}$ since we only compute the upper diagonal of this array. Thus, $\frac{n^3}{2} \in O(n^3)$.

b)

A more clever algorithm in this case that runs faster than cubic time is to compute the distance between two houses only once. In the naive algorithm we computed the distance between two houses from scratch for every pair. However, we can utilize the distances already computed. This would result in a quadratic running time since there are n^2 number of pairs.

Pseudo code for this algorithm

```
D = [0] # n*n array to contain computed distances
for i = 1,...,n
    for j = 1,...,n-1
        if i < j
            D[i,j+1] = D[i,j] + x_j # x_j is the distance from H_j to H_(j+1)
            D[j,i] = D[i,j+1] # Making array symmetric
```

This algorithm make $\approx \frac{n^2}{2}$ computations which $\in O(n^2)$.