ALGO 01 : PROJECT

PART 1 :

The algorithm we are using is very simple. We start by calculating the average position of the houses, then we calculate the sigma of the distribution of the houses.

Once this is done, we split the list of houses in 4 different lists:

- ***positiveMajorLlist***: contains all houses that are positioned between our starting position (0) and the value (average + 0.96*sigma) that defines the **upper** limit where we'll find most of our houses.

- ***negativeMajorLlist***: contains all houses that are positioned between our starting position (0) and the value (average - 0.96*sigma) that defines the **lower** limit where we'll find most of our houses.

- ***positiveRemainingLlist***: contains all the remaining values that are not in the previous lists and that are **positives**.

- ***negativeRemainingLlist***: contains all the remaining values that are not in the previous lists and that are **negatives**.

We then do our choice regarding the length of each **MajorLlist** or the further value contained in them. We always go through both the **MajorLlists** before the **RemainingLists**, this operation is simply used to define in what order we are going through the lists.

This way, we don't loose time changing directions during the snowplow cleaning. Going back and forth is taking a lot of time and we should always go in the same direction as long as we're not going too far away from the other houses (that's why we calculate the sigma and the average value).

This algorithm doesn't have nested loops and the loops only depends on the size of the list.

The algorithm contains 5 for loops, it has a running time of 5n.

It also uses 3 times Python's sort, which has a complexity of O(n log(n)) - (According to Wikipedia, Python uses Timsort).

So the Algorithm has a running time of : 5n + 3(n log(n))

We can simplify this and say that the algorithm has a time complexity of O(n log(n)).

Therefore, it is a quasilinear algorithm, which is better than polynomial.

PART 2 :

CHOSEN HEURISTIC: Welsh-Powell algorithm

HOW DOES IT WORK: We sort each node by its degree (number of neighbors). Then we go through this list and attribute a color for each node that is not a neighbor with one of the previous colored nodes (the first node – the one with the biggest degree – automatically get the first color). Once we finished going through the list, if all the nodes hasn’t be colored we start again the previous process with a different color.