ALGO 1 : Project

NICOLAS

1 PART 1 : THE SNOWPLOW PROBLEM

A snowplow must clean the snow in front of 1000 houses. The position of each house is represented by a float number (not necessary positive). Initially, the snowplow is in 0. The traveling time of the snowplow is equivalent to the distance that it went through since the beginning of the day. If there are several houses at the same point, it does not take a longer time to clean the snow (the snow is cleaned as soon as the snowplow has passed). However, since the positions of the houses are represented as floats, this should not happen often.

The snowplow must minimize the average waiting time before the snow is cleaned in front of the houses, **from the point of view of the houses**.

A polynomial algorithm based on the positions of the houses must be determined to establish a strategy. If it is not polynomial (if it is exponential for example) we saw in class that it probably could not run.

Conventions : You must write a function that takes has input the positions of the 1000 houses, and outputs the positions of the houses, sorted in the same order as they are cleaned.
Name of the function : parcours(list)
returns : list of sorted houses in the relevant order.
test : I generate random initial positions of houses with a normal distribution centered in 0, with **numpy.random.normal(0,1000,1000)** , and then I statistically evaluate the performance of your algorithm.

I test your solution and compare it to several benchmarks. Your solution should give a waiting time that is smaller to 90 percents of the time that we would obtain with the greedy solution, consisting to going to the closest house at each step.

Important : You must add a document to your project explaining why your algorithm runs in polynomial time. A pdf file is preferred. You don't have to be extremely formal, but clear enough to show that you understand why your solution is polynomial.

2 PART 2 : YOUR CHOICE

Option 1 : Pick a heuristic of your choice for the coloring problem and implement this heursitic on a reasonable instance (small enough that would run quickly on a normal machine). For instance you can use a graph with around 50 vertices and 100 edges or so. It is required that you provide images illustrating your heursitc, for instance generated with graphviz.

Option 2 : Write a program that build the line graph of a graph and explain the connection between the Hamiltonian path problem, the Eulerian path problem, and line graphs, with your words and images.

3 CREDITS

The project is optional and may provide 2 additional credit. The time given is until sunday October 6th.