Algorithmic complexity and graphs

Antoine Duplaa, Hugo Mangematin, Adrien Pilet

October 2022

Contents

1 The Snowplow Problem

1.1 Introduction

1.2 Exercise

1) Propose a $n \in \mathbb{N}$ (it does not need to be large n = 10 is fine) and a configuration of the houses where the optimal order of cleaning is not obtained by either :

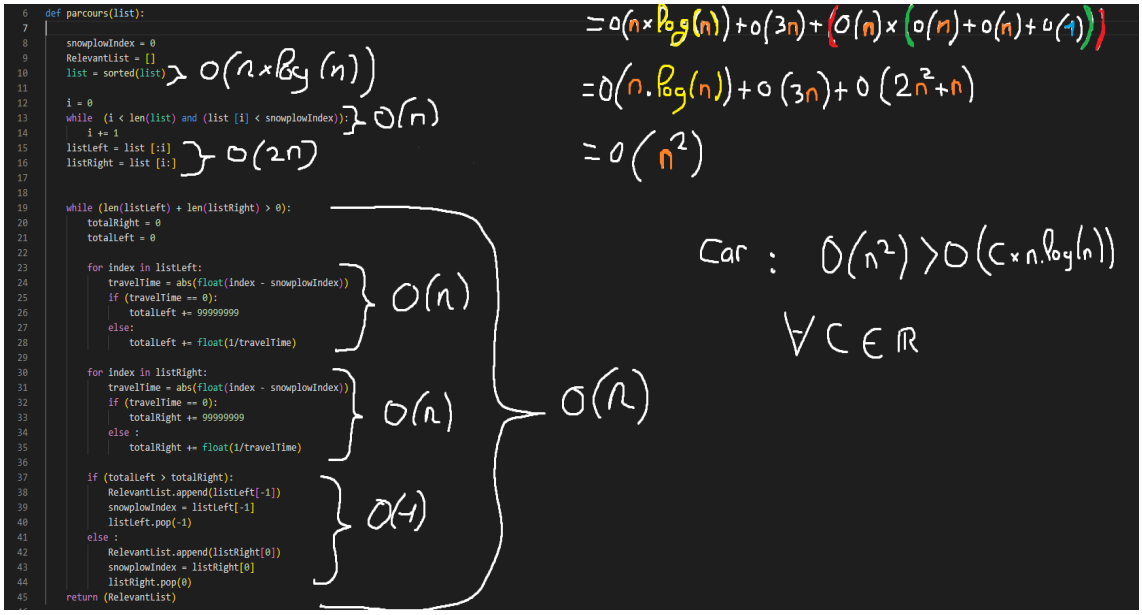
- sorting the houses positions and cleaning them in that order.
- going to the closest house at each time step.

n = 10

List = [-8, -4, -1, -1, 5, 5, 5, 5, 5, 5]

2) See Exercise1/snowplowProblem.py

3)



2 Choosing Binoms in a Company (matching)

2.1 Exercise

3 Complexities

3.1 Function 1

```
1       def function_1(n: int) -> None:
2       temp_list = list()
3       for i in range(n**2):
4           temp = 0
5           for j in range(i):
6               temp += j
```

```
            temp_list.append(temp)
        sum(temp_list)

```

The complexity of the first loop is in $O(n^2)$ (because we iterate n^2 times) and in the second one the number of iterations is the rank of the previous one. The the complexity in the loop is in $O(1)$. The sum function as a complexity of $O(n)$. The total number of iteration in this algorithm is a sum which can be written as:

$$\left(\sum_{i=0} i = \frac{a(a+1)}{2}\right) + n$$

In our case $a = n^2$:

$$\frac{n^2(n^2+1)}{2} + n$$

We can reduce it in:

$$\frac{n^4+n^2}{2} + n$$

The complexity of function 1 is $O(n^4) + O(n) = O(n^4)$

3.1.1 Function 2

```

def function_2(n: int) -> None:
    print(n)
    for i in range(n):
        temp_list = [j+i for j in range(n)]
        shuffle(temp_list)
        max(temp_list)
```

The complexity of the print is $O(1)$. The main loop is in $O(1)$. In this loop there is another one iterating on n. The two functions shuffle and max are in $O(n)$ according to the python doc. The overall complexity is:

$$O(1) + O(n)\left(O(n) + O(n) + O(n)\right) = O(n^2)$$

Figure 1: Transformation of a graph containing no Eulerian path into an linear graph with an Hamltonian path {(1,4), (4,3), (2,3), (1,3), (1,2)}.

4 line graphs, Hamiltonian paths and Eulerian paths

4.1 Explanation

We present theses problems as follow:

The Hamilton Path problem:
Input: A graph G.
Output: true, iff there is a path that travel through each vertex only once, false otherwise.

The Hamilton Path problem:
Input: A graph G.
Output: true, iff there is a path that travel through each edges only once, false otherwise.

A linear graph is a graph in which we invert vertex and edges. We can construct a linear graph by creating an vertex named (u, v) for each edges (u, v) and and edges betwin vertex that contain at least one common number in their name. Every graph that contain an Eulerian path can be transformed into an linear graph containing an Hamiltonian path because edges and vertex are inverted. But graphs that contain no Eulerian path can be transformed into a linear graph with a Hamiltonian path as follow:

4.2 Exemples

Figure 2: Transformation of a graph containing an Eulerian path {4, 1, 2, 3, 1} into an linear graph with an Hamltonian path {(1,4), (1,2), (2,3), (1,3)}.

4.3 Complexities of the two algorithms

To find an Hamiltonian cycle in a graph is considered "harder" because the problem belongs to the NP-hard class and can't be computed in a polynomial time, only in an exponential time. To find if an Eulerian path is contained in a graph belongs to the P class which can be computed in a polynomial time. Furthermore Eulerian path can't be reduced to an Hamiltonian. Thus Hamiltonian is "harder" than Eulerian.