Florian COMTE

Paris street processing

Hands on 1

My program takes 3 command line argument
args[0] is a string telling either we are using the std implementation of lists and maps or the naive ones
args[0] = “std” or args[0] = “custom”

args[1] is another string telling either we are using a weighted graph or not
args[1] = “weighted” or args[1] = “notWeighted”

args[2] is the number of points to read from the file (all elts if null or greater than number of elements in file)

The default values if no argument is given are (“std”, “notWeighted”, Integer.MAX_VALUE)

Here are the objects used for data structure depending on the first 2 parameters:
std notWeighted : an ArrayList for the collection of Nodes and an Arraylist for the collection of a node’s neighbors
custom notWeighted : a SingleLinkedList for the collection of Nodes and a SignleLinkedList for the collection of a node’s neighbors
std weighted : an ArrayList for the collection of Nodes and a TreeMap for the collection of a node’s neighbors (key = neighbourNode, value = distance)
custom Weighted : an SinglelinkedList for the collection of Nodes and a ListMap for the collection of a node’s neighbors (key = neighbourNode, value = distance)

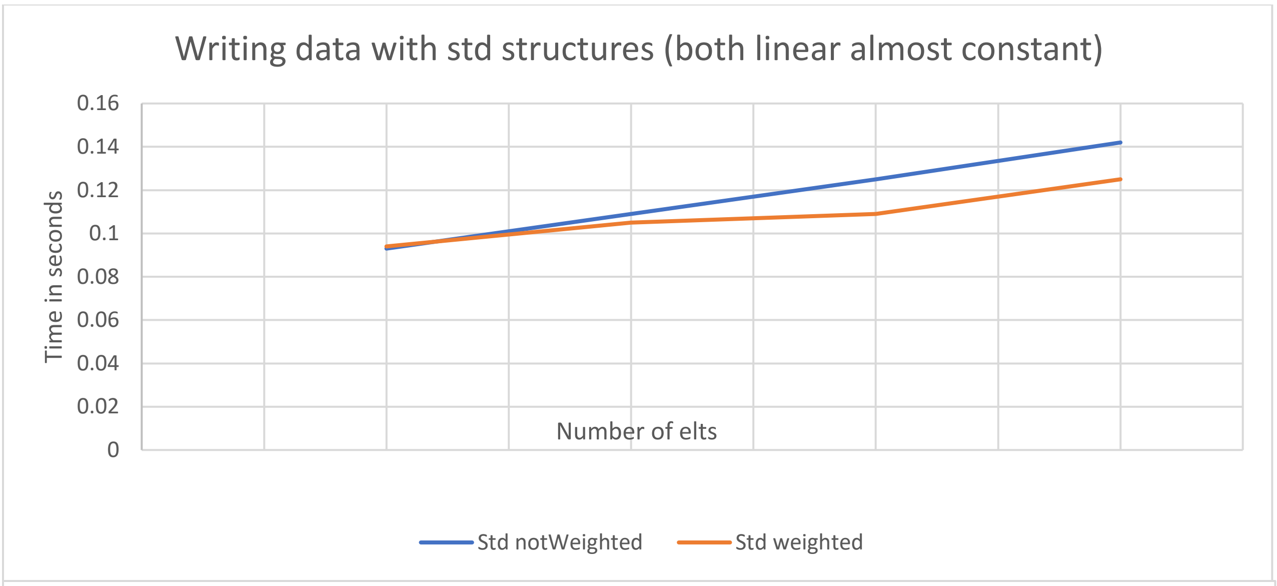
Here are the times it took in seconds to write the data structure:

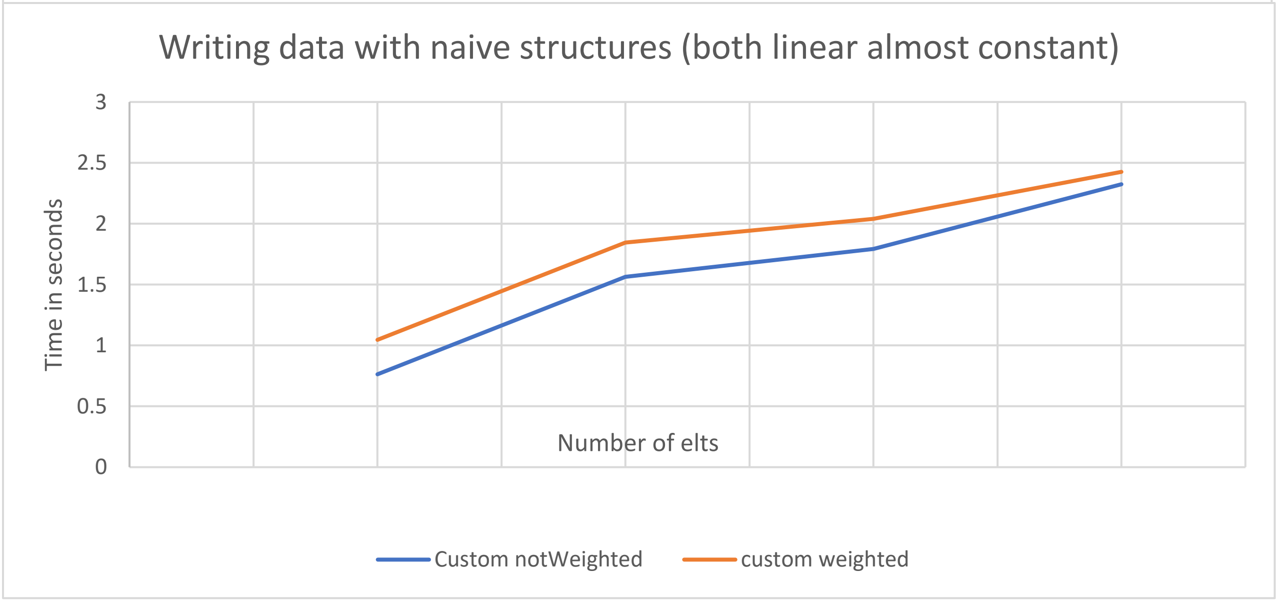
	2 500 elts	5 000 elts	7 500 elts	10 000 elts
Std notWeighted	0.093	0.109	0.125	0.142
Custom notWeighted	0.763	1.565	1.792	2.324
Std weighted	0.094	0.105	0.109	0.125
custom weighted	1.046	1.846	2.039	2.426

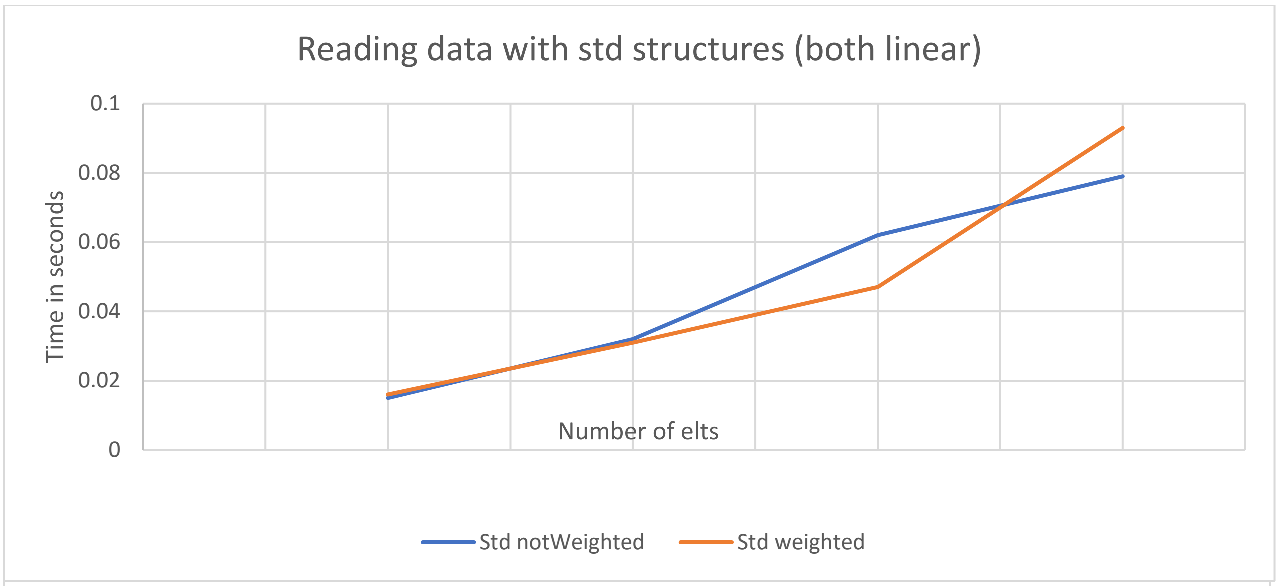
Here are the times it took in seconds to plot the points of Paris (i.e. to read the data structure):

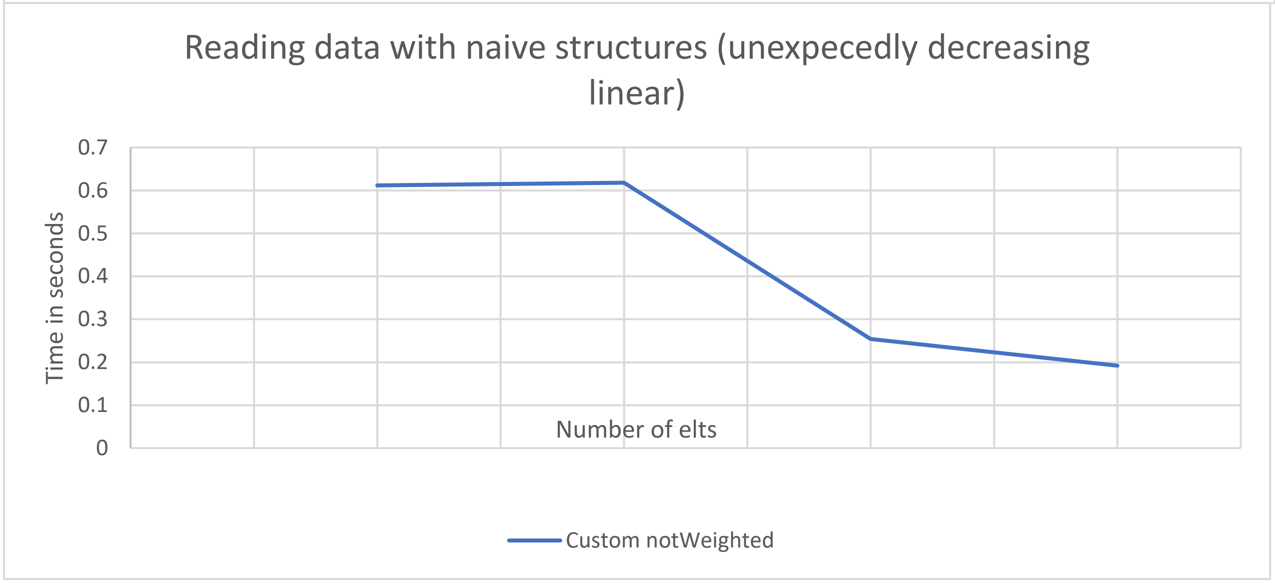
	2 500 elts	5 000 elts	7 500 elts	10 000 elts
Std notWeighted	0.015	0.032	0.062	0.079
Custom notWeighted	0.612	0.618	0.254	0.192
Std weighted	0.016	0.031	0.047	0.093
custom weighted	//	//	//	//

Note: we can’t try to plot the map with the naïve ListMap class because the keyset function is not implemented


Writing data with std structures (both linear almost constant)
Time in seconds
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
Number of elts
Std notWeighted
Std weighted


Writing data with naive structures (both linear almost constant)
Time in seconds
0
0.5
1
1.5
2
2.5
3
Number of elts
Custom notWeighted
custom weighted





How could you improve the algorithmic complexities ? You propably want to replace the SingleLinked list by something else.

We could replace lists by arrays to improve the reading function algorithmic complexity

If you were to use a datastructure based on arrays, how could you make it "grow" ?

To make it grow by 1 element for example, we would have to copy it into another one of size n+1 and add our element at last indice.

What would be the algorithmic complexity of adding a new element ?

It would be linear because the bigger the array is, the more it has to copy before adding the element

What kind of more efficient search would it enable ? How ?

It would enable a search by id without looping all over the array to find the wanted element.