Grzegorz Szulik 7 7 7 7

..... ----------

(Name and Surname) (A, B, C, D)

Parameters:
liczba_prob = 7
iczba_dlugosci = 33

Exemplary report on Lab 1

Task 1./Zadanie 1.

1. Goal
   The aim of the task is to investigate the computational complexity of insertion and selection sort algorithms.

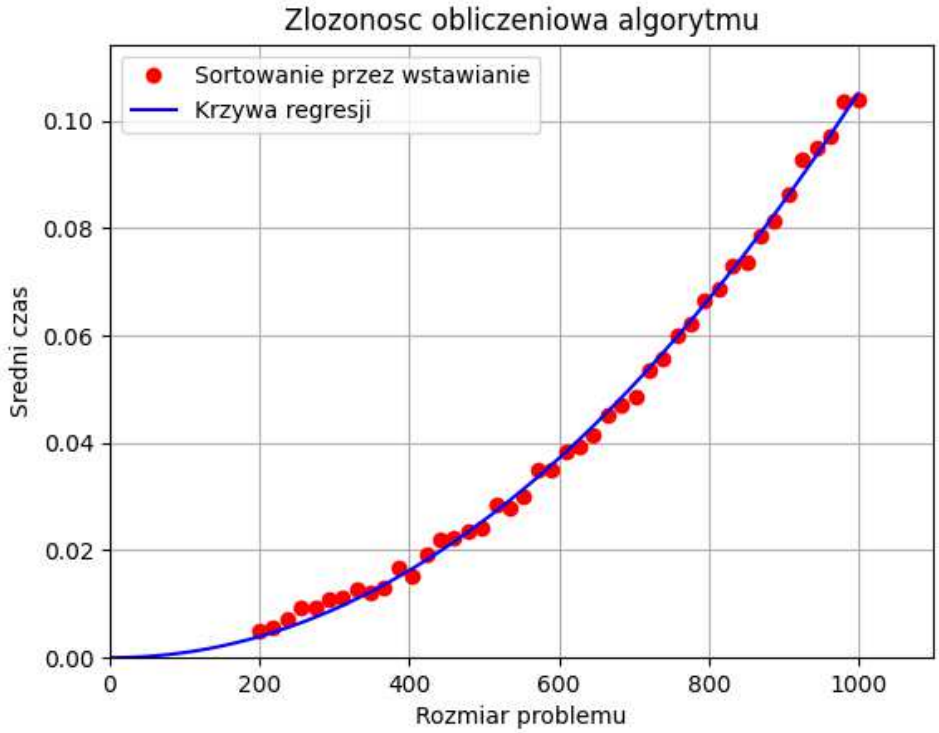
2. Methods
   Several classes developed in Python were used in the experiment. The relevant project was created and compiled in the Thonny environment on a laptop with an Intel Celeron CPU 1007U processor.

3. The experiment and the results
   The experiment began with fixing the minimal and maximal lengths of the lists. It was assumed that:
   - *min_dlugosc* = 200, for which the time of sorting is greater than 0.002 sec,
   - *n* = 1000, for which the time of sorting by choosing was approx. 0.2 sec.

The method *mierz_czas* was written – it allows to measure the time of sorting a list of a given length by one of two investigated methods. Here is the code of the methods:

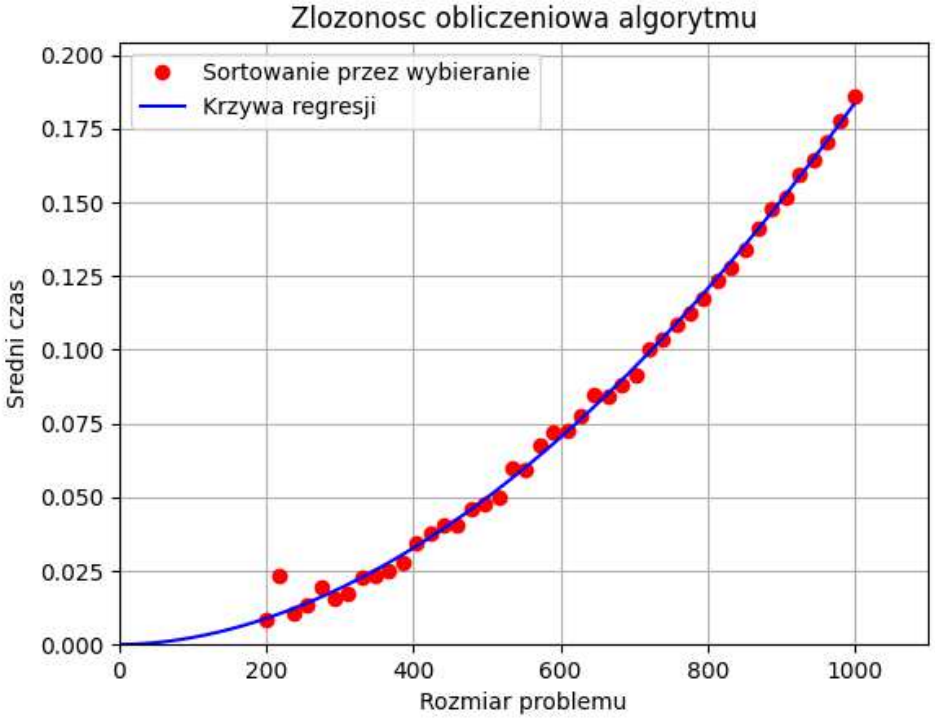
```
def mierz_czas(self, metoda, k=None):
    """Method measuring the time of sorting of random lists
           of the length k"""
    if k is None:
        k = self.dlugosc
    self.lista = []
    czas = 0.0
    for _ in range(self.liczba_prob):
        self.losuj(k)
        if metoda == 1:
            stoper = time.time()
            self.sortuj_przez_wstawianie(k)
            stoper = time.time() - stoper
        else:
            stoper = time.time()
            self.sortuj_przez_wybieranie(k)
            stoper = time.time() - stoper
        czas = czas + stoper
    return czas/self.liczba_prob
```

Next, the method *check_complexity* was run with the following parameters of the class *Sorting*: *n = 1000*, *lprob = 7*, *ldugosci = 44* oraz *najkrotsza* = 200.



Plot 1. Dependence between the time of sorting of list by inserting and the length of the list

Empirical computational complexity was: $n^{2,039}$, what is close to the theoretical value n^2. Similar experiment was done for sorting by choosing.



Plot 2. Dependence between the time of sorting of list by choosing and the length of the list

Here the obtained comuptational complexity was $n^{1.881}$, what was surprising for the author of this report due to the lack of the optimistic time for sorting by choosing (in case of sorting by inserting the optimistic time is n). However, one can notive that the average times for sorting by inserting are almost twice smaller than the obtained times for sorting by choosing.

4. Conclusions

The experiment helped to estimate the computational complexity of the algorithms of sorting by inserting and by choosing. It occurs that the experimental complexity is close to the theoretical complexity $O(n^2)$.

Task 2. / Zadanie 2.

1. Goal

The aim of the the task was to compare two methods of sorting the lists – by inserting and by choosing.

2. Methods

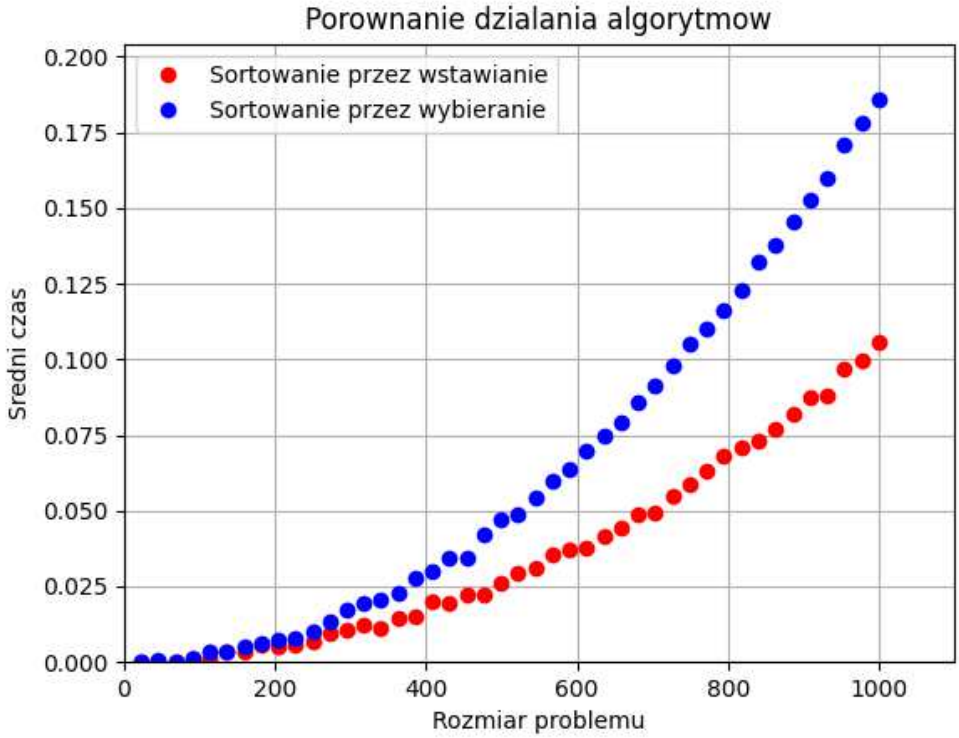
In the experiment two classes written in Python were used. The relevant project was created and compiled in the Thonny environment on a laptop with an Intel Celeron CPU 1007U processor.

3. The experiment and the results

The maximal length of the list is the same as in the task 1. The method *mierz_czas* from the task 1 was used here for both methods of sorting. Compiling the method *porownaj_metody* for the object of the class *Sortowanie* with the parameters *n* = 1000, *lprob* = 7, *ldlugosci* = 44 it was possible to obtain the graph presented at the next page.

4. Conclusions

In the experiment it occured that the sorting by inserting is twice faster than sorting by choosing – the difference between the times is small for short lists, but it increases with the increase of the length of the list.



Plot 3. Comparison of two methods of sorting
(red points – sorting by inserting,
blue points – sorting by choosing)