ИТМО Кафедра Информатики и прокладной математики

Отчет по лабораторной работе №1 «Сортировки за $O(n^2)$ - сортировка пузырьком, выбором, вставками»

Выполнил: студент группы Р3117
Плюхин Дмитрий
Проверил: Симоненко З. Г.

2016 год

1. Задание

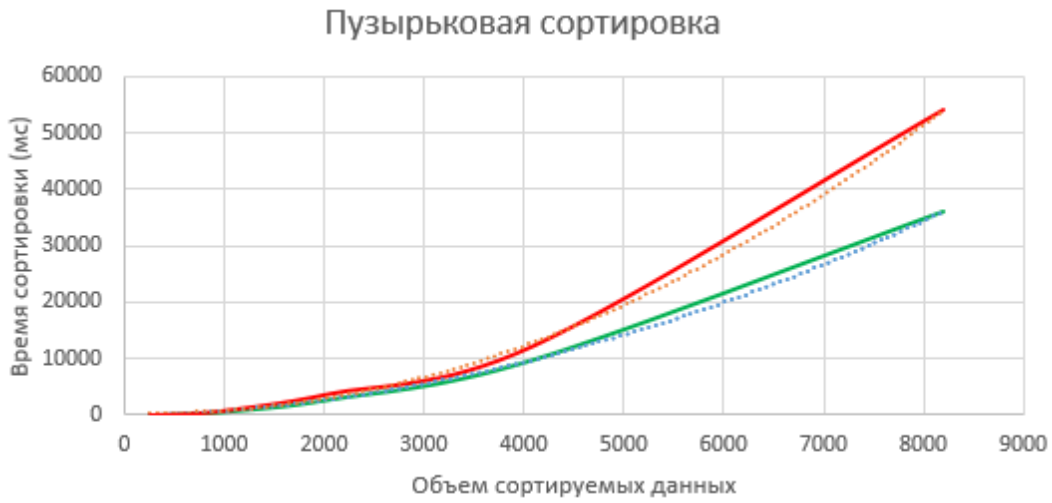
Реализовать на каком-либо языке программирования и сравнить между собой различные алгоритмы сортировки, работающие за время $O(n^2)$ – сортировку пузырьком, вставками, выбором.

2. Выполнение

Для выполнения работы был выбран язык программирования Python по причине того, что алгоритмы сортировок, реализуемых в работе, имеют небольшой объем и не очень сложны с технической точки зрения.

После реализации алгоритмов на языке программирования был проведен их запуск на различных исходных данных, в частности, на массивах разной длины. Была выполнена сортировка массивов с использованием трех алгоритмов (время сортировки усреднено для каждого массива), построены графики, отражающие зависимость времени работы каждого алгоритма от количества элементов в сортируемом массиве. Причем для пузырьковой сортировки и сортировки вставками представлены два графика. Это сделано для того, чтобы показать, что время работы этих алгоритмов зависит от того, был ли массив упорядочен изначально или нет (алгоритмы работают дольше на отсортированных массивах).

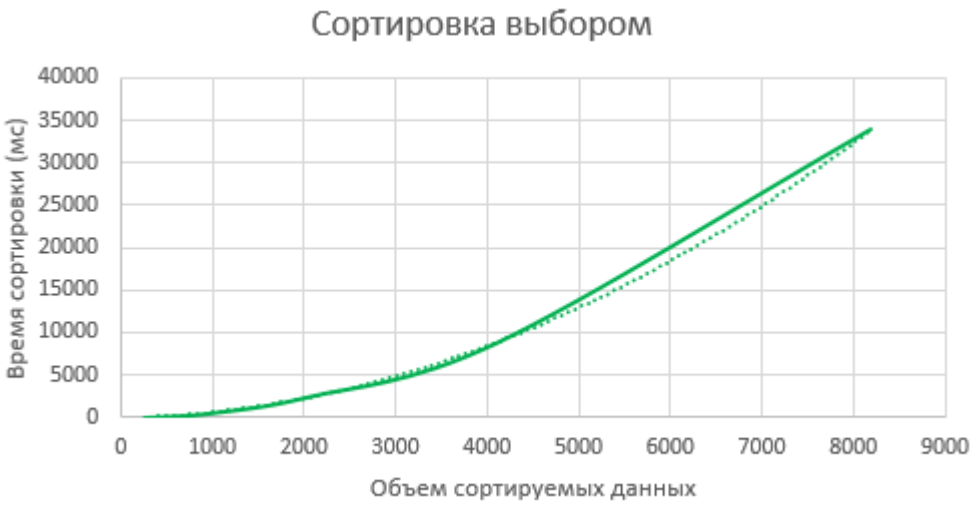
3. Результаты




```
import time
f = open('data8192.txt')
j=0
k = []

numberofelements = int(f.readline())
while j<numberofelements:
    i = int(f.readline())
    k.append(i)
    j=j+1

tit1 = time.time()
for n in reversed(range(len(k))):
        for m in range(1, n + 1):
            if k[m-1] > k[m]:
                k[m], k[m-1] = k[m-1], k[m]
tit2 = time.time()
```

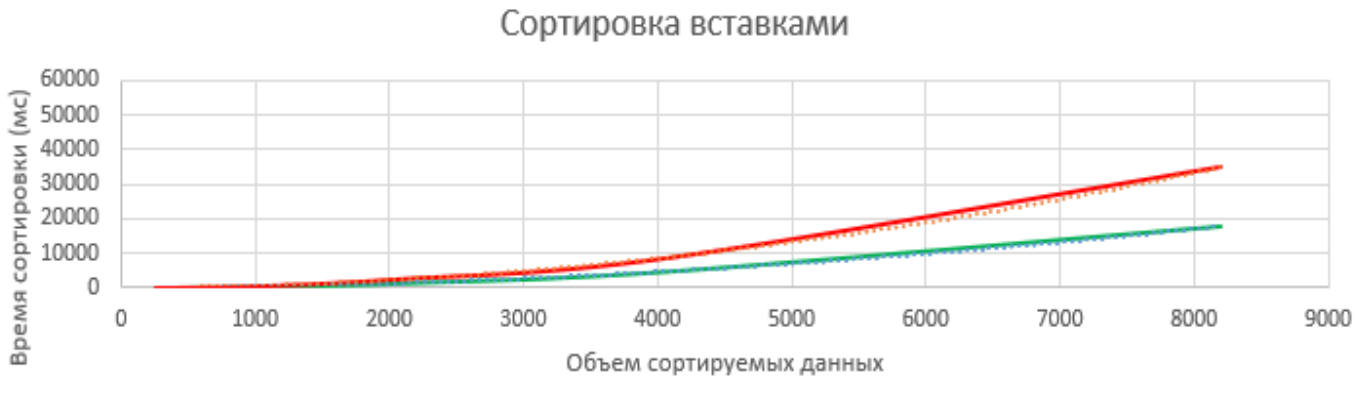



```python
import time
f = open('data8192.txt')
j=0
k = []

numberofelements = int(f.readline())
while j<numberofelements:
    i = int(f.readline())
    k.append(i)
    j=j+1

tit1 = time.time()
for n in range(len(k)-1):
    minimum = n
    index = n + 1
    while index < len(k):
        if k[index] < k[minimum]:
            minimum = index
        index+=1
    t = k[n]
    k[n] = k[minimum]
    k[minimum] = t


tit2 = time.time()
```



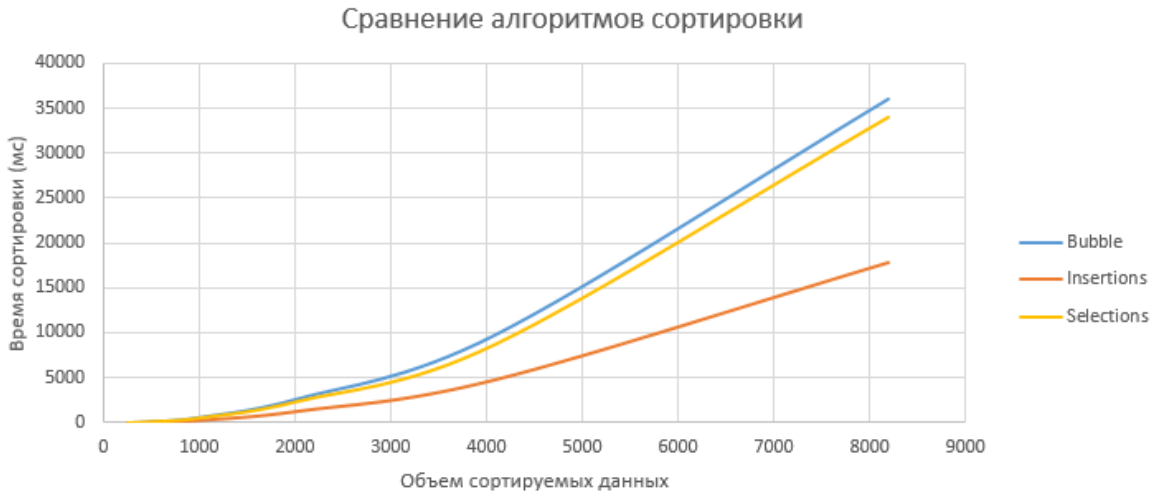
```python
import time
f = open('data8192.txt')
j=0
k = []

numberofelements = int(f.readline())
while j<numberofelements:
    i = int(f.readline())
    k.append(i)
    j=j+1

tit1=time.time()
for i in range(1,len(k)):
    key = k[i]
    j = i - 1
    while j >= 0 and k[j] > key:
        k[j+1] = k[j]
        j = j-1
    k[j+1] = key
tit2=time.time()
```

4. Анализ

Время работы (мс)			Размер массива
Сортировка пузырьком	Сортировка выбором	Сортировка вставками	
34	26	14	256
138	142	64	512
652	586	298	1024
2742	2441	1317	2048
9816	8750	4808	4096
35977	33922	17805	8192



Так, если сравнивать между собой эти три сортировки, то наиболее быстрой является сортировка пузырьком, однако время ее работы в определенной степени зависит от входных данных. Сортировка вставками работает медленнее, чем сортировка пузырьком, и также обладает тем недостатком, что работает дольше на массивах, упорядоченных заранее определенным образом. Наиболее оптимальным вариантом является сортировка выбором – время ее работы не зависит от того, как был упорядочен массив перед началом сортировки.