ИТМО Кафедра Информатики и прокладной математики

Отчет по лабораторной работе №2 «Сравнение сортировки бинарными вставками и быстрой сортировки»

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

2016 год

1. Задание

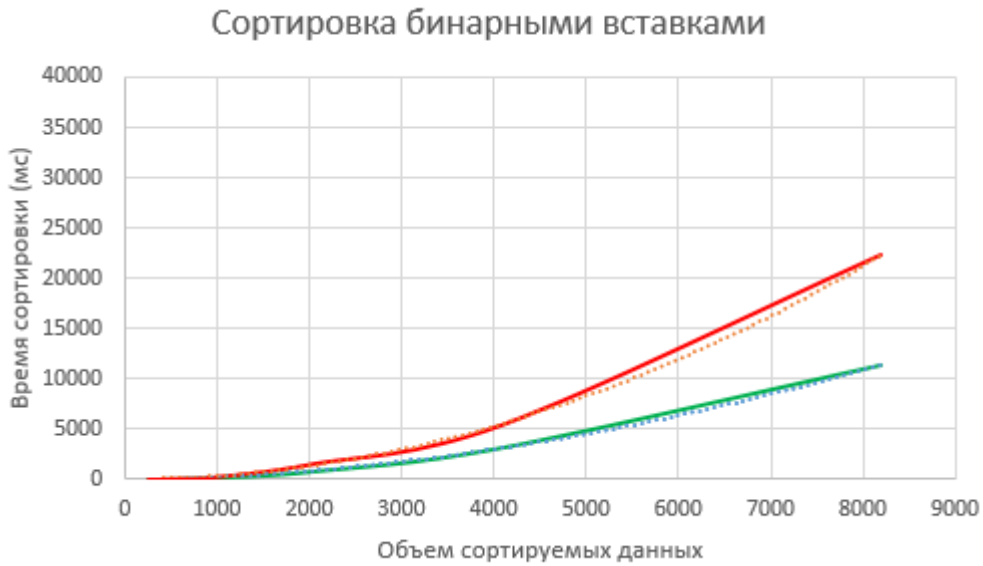
Реализовать на каком-либо языке программирования и сравнить между собой алгоритм сортировки бинарными вставками и алгоритм быстрой сортировки.

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 двух алгоритмов (время сортировки усреднено для каждого массива), построены графики, отражающие зависимость времени работы каждого алгоритма от количества элементов в сортируемом массиве. Для сортировки бинарными вставками представлено два графика в связи с тем, что время работы сортировки зависит от изначальной упорядоченности массива (работает дольше на отсортированном массиве). Быстрая сортировка представлена в рандомизированном варианте.

3. Результаты

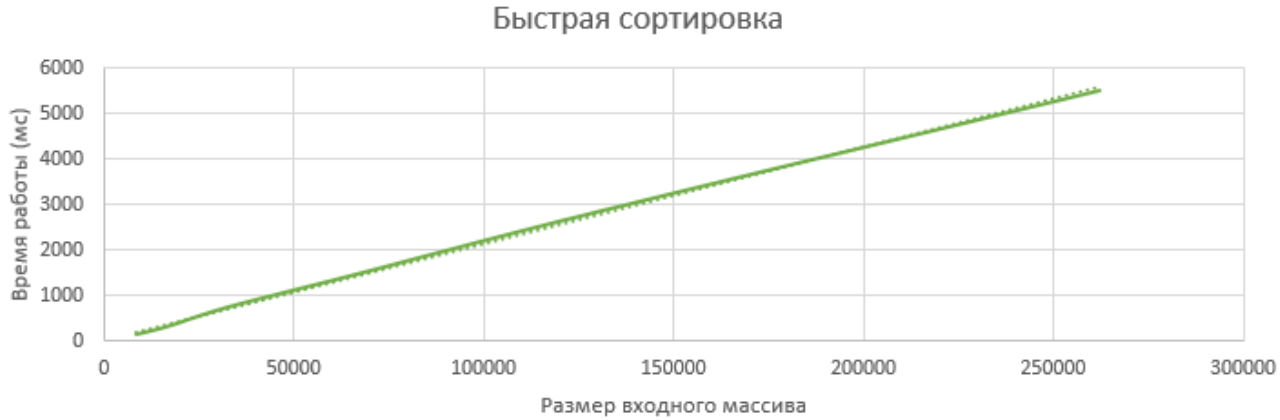



```python
import time
f = open('data4096.txt')
j=0
k = []

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)):
    if k[i-1]>k[i]:
        left = 0
```

```
            right = i - 1
            while True:
                mid = (left + right) // 2
                if k[mid]>k[i]:
                    right = mid - 1
                else:
                    left = mid + 1
                if left > right:
                    break
            key = k[i]
            for j in reversed(range(left+1,i+1)):
                k[j] = k[j-1]
            k[left] = key
    tit2=time.time()
    print(tit2-tit1)
```



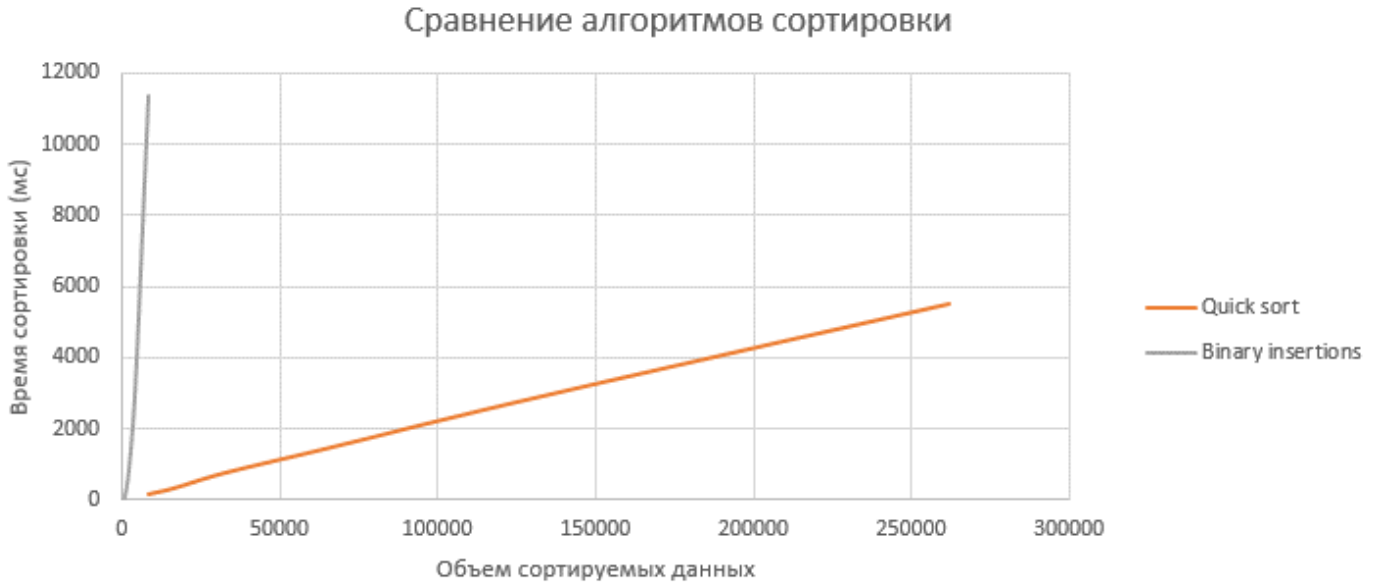

```
#QuickSort
import random;
import time
def Partition(a,p,r):
    x = a[r]
    i = p - 1
    j = p
    while (j < r):
        if a[j] <= x:
            i+=1
            a[i],a[j] = a[j],a[i]
        j+=1
    a[i+1],a[r] = a[r],a[i+1]
    return i+1

def RandomPartition(a,p,r):
    x = random.randrange(p,r+1)
    a[x],a[r] = a[r],a[x]
    i = Partition(a,p,r)
    return i

def RandomQuickSort(a,p,r):
    if (p<r):
        i = RandomPartition(a,p,r)
        QuickSort(a,p,i)
        QuickSort(a,i+1,r)
```

4. Анализ

Время работы (мс)		Размер массива	
Сортировка бинарными вставками	Быстрая сортировка	Сортировка бинарными вставками	Быстрая сортировка
9	140	256	8192
36	312	512	16384
184	750	1024	32768
805	1453	2048	65536
3176	2875	4096	131072
11385	5531	8192	262144



Известно, что сортировка бинарными вставками обладает наилучшим временем работы среди всех сортировок, работающих за квадратичное время. Однако при ее сравнении с быстрой сортировкой становится очевидна ее неэффективность при упорядочивании больших объемов данных. Тем не менее, и у быстрой сортировки есть свои недостатки, такие как увеличение глубины рекурсии при подаче на вход уже упорядоченного массива и неэффективность на массивах с большим числом одинаковых элементов. Для устранения всех этих недостатков используют различные варианты процедуры Partition.