Санкт-Петербургский Национальный Исследовательский Университет
Информационных Технологий, Механики и Оптики

Факультет инфокоммуникационных технологий

Лабораторная работа №3

Выполнили:

Бабаев Р.С.

Садовая А.Р.

Абдулов И.А.

Проверил:

Мусаев А.А.

Санкт-Петербург,

2022

Оглавление

Введение

Данная работа представляет собой отчет о выполненных заданиях:

1. Написать программу для сортировки массива пузырьком и сравнить ее эффективность с методом sort().
2. Придумать алгоритмы, имеющие сложность O(3n), O(nlog(n)), O(n!), O(n^3), O(3log(n))
3. Построить зависимость между количеством элементов и количеством шагов для алгоритмов со сложностями O(1), O(log(n)), O(n^2), O(2^n). Сравнить сложности алгоритмов.

1 Задание №1

Сортировка пузырьком заключается в том, что элемент сравнивается со следующим и, если он оказывается больше, то они меняются местами. Таким образом каждый элемент сравнивается с элементами, которые стоят «выше» него, и останавливается, как только следующий встречающийся элемент оказывается больше или равен текущему. Также можно пользоваться просто методом sort()

```
from random import randint
array=[]
n=10
for i in range(n):
    array.append(randint(1,100))
print(array)
b=array
for i in range(len(array)-1):
    for j in range(len(array)-i-1):
        if array[j]>array[j+1]:
            array[j], array[j+1]=array[j+1], array[j]
print('Пузырьком ', array)
b.sort()
print('Через sort() ', b)
```

Рисунок 1.1 – Код программы для сортировки пузырьком и через метод sort()

```
[21, 13, 56, 52, 10, 82, 42, 42, 58, 14]
Пузырьком  [10, 13, 14, 21, 42, 42, 52, 56, 58, 82]
Через sort()  [10, 13, 14, 21, 42, 42, 52, 56, 58, 82]
```

Рисунок 1.2 – Пример работы программы

Как мы видим, результат получен одинаковый. Посмотрим, чем тогда отличаются два этих способа.

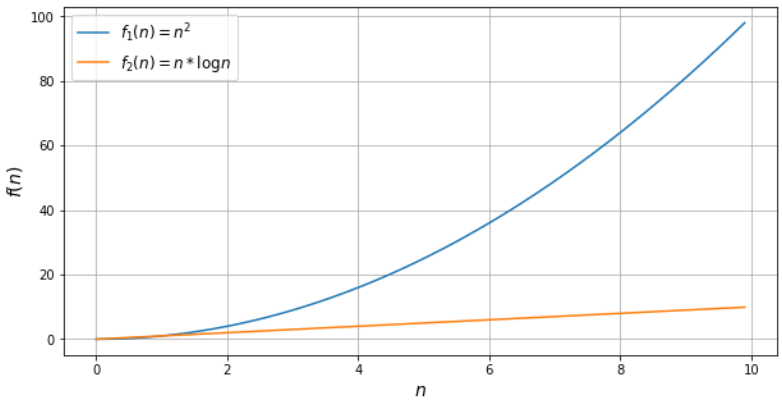

Рисунок 1.3 – Сравнение эффективности двух способов сортировки

Вывод:

Сортировка пузырьком имеет большую сложность, чем сортировка через метод sort(). Кроме того, чем больше элементов в массиве, тем больше различие в сложности исполняемого алгоритма. Однако оба этих способа подходят для сортировки массива, просто один более оптимизированный.

2 Задание №2

```
mas1 = [0]*10
mas2 = [0]*10
mas3 = [0]*10
```

Рисунок 2.1 – Алгоритм со сложностью O(3n) – создание трёх пустых массивов

```
mas1 = [-3, 4, 2, 0]
mas2 = sorted(mas1)
print(mas1)
print(mas2)
for i in mas1:
  l, r = 0, len(mas2)-1
  while l < r:
    mid = (l+r)//2
    if mas2[mid] < i:
      l = mid + 1
    else:
      r = mid
  print(l, end=' ')
```

Рисунок 2.2 – Алгоритм со сложностью O(n(log(n))) – поиск индексов элементов массива в отсортированном массиве

```
m1 = [1, 3, -2]
def rec(a, m1):
  if len(a) == len(m1):
    print(a)
    return
  for i in m1:
    if i not in a:
      rec(a+[i], m1)
rec([], m1)
```

Рисунок 2.3 – Алгоритм со сложностью O(n!) – количество перестановок

```
for i in range(n):
  for j in range(n):
    for z in range(n):
      b[z] += a[i]-m[j]
```

Рисунок 2.4 – Алгоритм со сложностью $O(n^3)$ – 3 цикла

```
from random import randint
m1.sort()
print(m1)
for i in range(3):
  elem = randint(-4, 4)
  l, r = 0, len(m1)-1
  while l < r:
    mid = (l+r)//2
    if m1[mid] < elem:
      l = mid + 1
    else:
      r = mid
  if elem == m1[l]:
    print(elem, l)
  else:
    print('no', elem, 'in the list')
```

Рисунок 2.5 – Алгоритм со сложностью O(3(log(n))) – 3 бинарных поиска

3 Задание №3

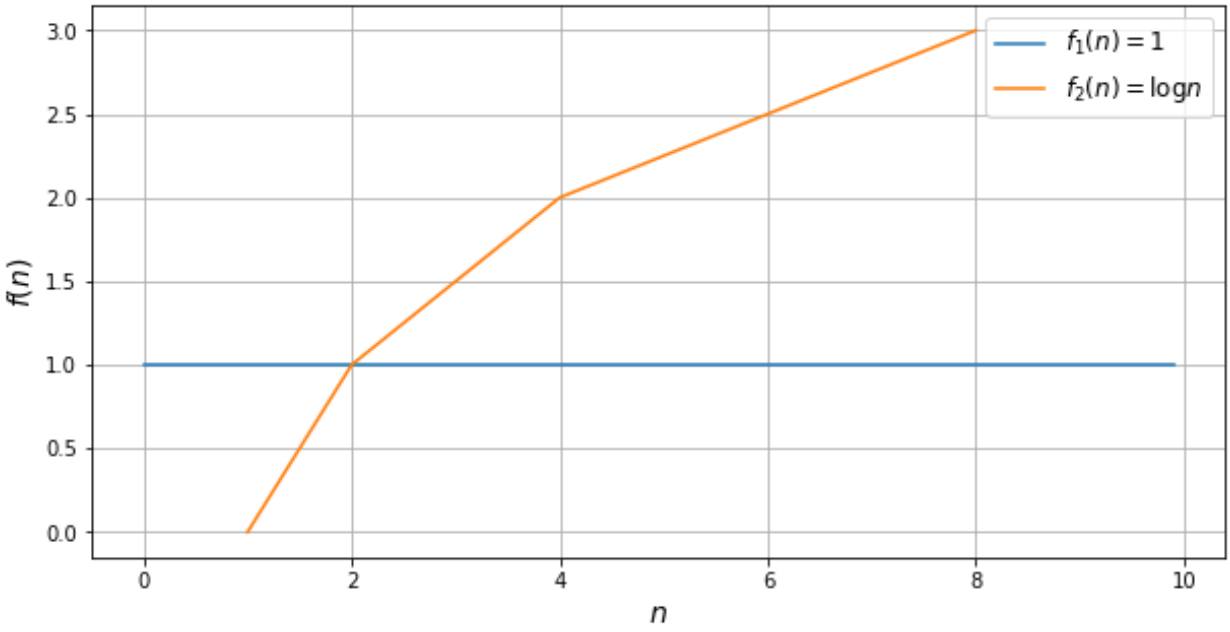

Рисунок 3.1 – Зависимость между количеством элементов и количеством шагов для алгоритмов сложности O(1) и O(log(n))d

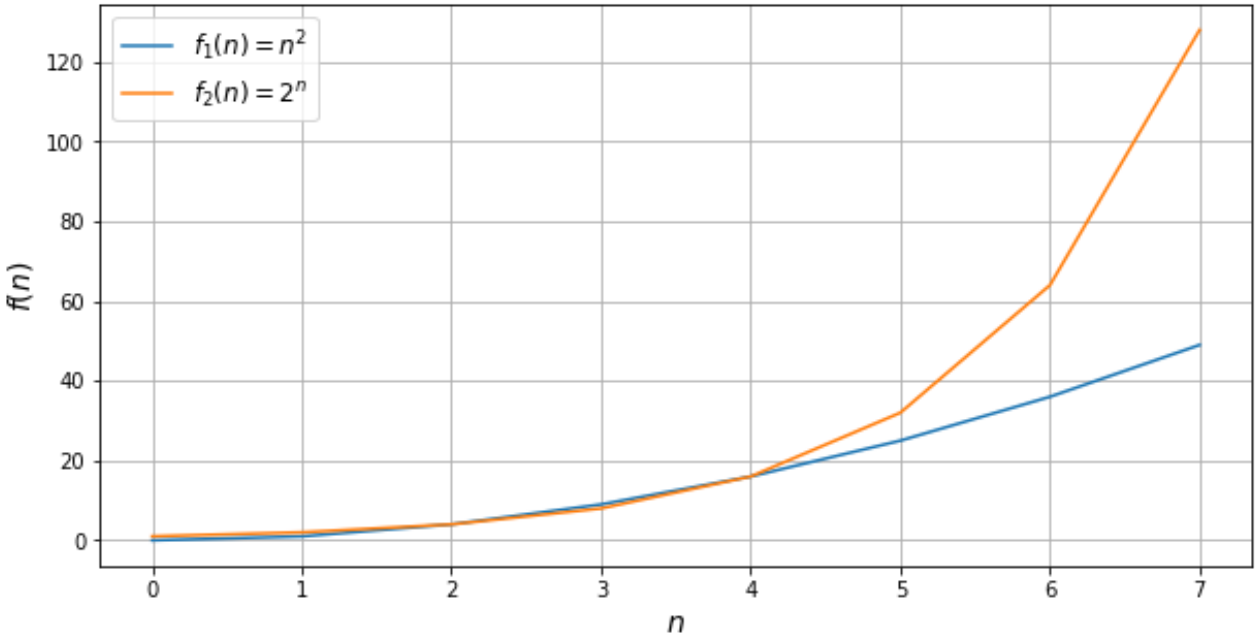

Рисунок 3.2 – Зависимость между количеством элементов и количеством шагов для алгоритмов сложности от количества шагов $O(n^2)$ и $O(2^n)$

Заключение

Таким образом, для каждой задачи была написана программа на языке программирования Python. Был изучен алгоритм сортировки массива пузырьком и рассмотрен встроенный метод sort(). Также написаны алгоритмы с различными сложностями. Для каждого решения приведены программы.

Все программы можно найти на GitHub [1].

Список литературы

1. GitHub [Электронный ресурс] – https://github.com/estle/ADs (Дата обращения – 03.11.2022).