Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский химико-технологический университет имени Д.И. Менделеева»

Кафедра информационных компьютерных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

Выполнил студент группы КС-38 (Косарев Виталий Владимирович)

Ссылка на репозиторий: (https://github.com/harbatium/ALGOS)

Приняли: Пысин Максим Дмитриевич

Краснов Дмитрий Олегович

Лобанов Алексей Владимирович

Крашенинников Роман Сергеевич

Дата сдачи: …………………….27.02.2023…………………………………………………………………..

Оглавление

[Описание задачи. 2](#_Toc63548272)

[Описание метода/модели. 2](#_Toc63548273)

[Выполнение задачи. 3](#_Toc63548274)

[Заключение. 10](#_Toc63548275)

# Описание задачи.

Вариант 1. Сортировка пузырьком.

В рамках лабораторной работы необходимо изучить и реализовать метод сортировки соответствующий вашему варианту (приведены ниже).  
Для реализованного метода сортировки необходимо провести серию тестов для всех значений N из списка (1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000), при этом:

* в каждом тесте необходимо по 20 раз генерировать вектор, состоящий из N элементов
* каждый элемент массива заполняется случайным числом с плавающей запятой от -1 до 1, для этого можно использовать как C функцию rand(), так и С++ генераторы
* каждый массив после генерации необходимо отсортировать и замерить время, требуемое на сортировку, для замера времени использовать следующий код
* Результат замера для каждой попытки каждого теста записать в файл общий файл.
* При замере времени выбираем только один из варивантов (nano\_diff, micro\_diff, milli\_diff, sec\_diff).

По окончанию всех тестов необходимо нанести все точки, полученные в результате замеров времени на график где на ось абсцисс(Х) нанести N, а на ось ординат(Y) нанести значения времени на сортировку. По полученным точкам построить график лучшего (минимальное время для каждого N), худшего (максимальное время для каждого N) и среднего (среднее время для каждого N) случая.

# Описание метода/модели.

# Сортировка пузырьком - это метод сортировки массивов и списков путем последовательного сравнения и обмена соседних элементов, если предшествующий оказывается больше последующего.

# В процессе выполнения данного алгоритма элементы с большими значениями оказываются в конце списка, а элементы с меньшими значениями постепенно перемещаются по направлению к началу списка.

В сортировке методом пузырька количество итераций внешнего цикла определяется длинной списка минус единица, так как когда второй элемент становится на свое место, то первый уже однозначно минимальный и находится на своем месте.

Количество итераций внутреннего цикла зависит от номера итерации внешнего цикла, так как конец списка уже отсортирован, и выполнять проход по этим элементам смысла нет.

Если мы говорим, что пузырьковая сортировка выполняется за время O(n^2), это значит, что медленнее она точно работать не будет. Здесь n это количество операций, которое предстоит выполнить алгоритму.

В среднем, сортировка пузырьком выполняется за квадратичное время, поэтому ожидаемая производительность алгоритма будет Θ(n^2).

Сортировка пузырьком — крайне неэффективный алгоритм.

Сортировка пузырьком расходует постоянное количество памяти. Дело в том, что мы сортируем уже имеющийся массив и не создаём дополнительных структур данных, в которых теоретически могли что-нибудь хранить.

К плюсам сортировки пузырьком относится простота реализации алгоритма. Алгоритм сортировки пузырьком сводится к повторению проходов по элементам сортируемого массива. Проход по элементам массива выполняет внутренний цикл. За каждый проход сравниваются два соседних элемента и, если порядок неверный, элементы меняются местами. Внешний цикл будет работать до тех пор, пока массив не будет отсортирован.

Основной минус алгоритма сортировки «пузырьком» - его медленная скорость

# Выполнение задачи.

Использовался язык Python. Время измеряли в миллисекундах.

1. Код:

import random

import time

import matplotlib.pyplot as plt

plt.style.use('Solarize\_Light2')

'''добавляем переменные для циклов и пустые массивы, которые позже заполнятся временем обработки итераций'''

N = 20

n = 1000

v1k = []

v2k = []

v4k = []

v8k = []

v16k = []

v32k = []

v64k = []

v128k = []

'''открываем файл с атрибутом "w" и сразу закрываем для очищения файла при каждом запуске программы'''

f = open('results.txt', 'w')

f.close()

'''составление массива из случайных чисел от -1.0 до 1.0 20раз для каждого n'''

for l in range(7):

'''открываем файл с атрибутом "a" (дописывает значения после имеющихся в файле)'''

f = open('results.txt', 'a')

f.write("\nCount:")

n = str(n)

f.write(n)

f.close()

n = int(n)

for k in range(N):

v1000\_1 = []

for i in range(n):

v1000\_1.append(random.uniform(-1, 1))

'''засекаем время начала метода пузырьков'''

start = time.time()

'''применяем метод пузырьков для сформированного массива случайных чисел'''

for i in range(n - 1):

for j in range(n - i - 1):

if v1000\_1[j] > v1000\_1[j + 1]:

v1000\_1[j], v1000\_1[j + 1] = v1000\_1[j + 1], v1000\_1[j]

'''завершаем отсчёт времени для метода пузырьков'''

end = time.time()

finaltime = (end - start) \* 10 \*\* 3

'''в зависимости от n на текущей итерации заносим значения времени в соотв.массив'''

if n == 1000:

v1k.append(finaltime)

elif n == 2000:

v2k.append(finaltime)

elif n == 4000:

v4k.append(finaltime)

elif n == 8000:

v8k.append(finaltime)

elif n == 16000:

v16k.append(finaltime)

elif n == 32000:

v32k.append(finaltime)

elif n == 64000:

v64k.append(finaltime)

elif n == 128000:

v128k.append(finaltime)

finaltime = "\n" + str(finaltime)

'''дописываем в файл время текущей итерации метода пузырьков'''

f = open('results.txt', 'a')

f.write(finaltime)

f.close()

n \*= 2

'''создаём общее окно для графиков и отдельное поле для каждого из 9 графиков'''

figure = plt.figure()

ax1 = figure.add\_subplot(3, 3, 1)

ax2 = figure.add\_subplot(3, 3, 2)

ax3 = figure.add\_subplot(3, 3, 3)

ax4 = figure.add\_subplot(3, 3, 4)

ax5 = figure.add\_subplot(3, 3, 5)

ax6 = figure.add\_subplot(3, 3, 6)

ax7 = figure.add\_subplot(3, 3, 7)

ax8 = figure.add\_subplot(3, 3, 8)

ax9 = figure.add\_subplot(3, 3, 9)

'''выводим точки времени для n=1000 на первый график'''

y1 = v1k

avg\_time1000 = 0

min\_time1000 = min(y1)

max\_time1000 = max(y1)

for i in range(len(y1)):

avg\_time1000 += y1[i]

avg\_time1000 /= len(y1)

m = len(y1)

x = [1000] \* m

ax1.set\_xlim(999, 1001)

ax1.set\_ylim(min\_time1000 - 0.5, max\_time1000 + 0.5)

for i in range(len(y1)):

ax1.plot(x[i], y1[i], marker=".", color="red")

'''выводим точки времени для n=2000 на второй график'''

y2 = v2k

avg\_time2000 = 0

min\_time2000 = min(y2)

max\_time2000 = max(y2)

for i in range(len(y2)):

avg\_time2000 += y2[i]

avg\_time2000 /= len(y2)

m = len(y2)

x = [2000] \* m

ax2.set\_xlim(1999, 2001)

ax2.set\_ylim(min\_time2000 - 1, max\_time2000 + 1)

for i in range(len(y2)):

ax2.plot(x[i], y2[i], marker=".", color="red")

'''выводим точки времени для n=4000 на третий график'''

y3 = v4k

avg\_time4000 = 0

min\_time4000 = min(y3)

max\_time4000 = max(y3)

for i in range(len(y3)):

avg\_time4000 += y3[i]

avg\_time4000 /= len(y3)

m = len(y3)

x = [4000] \* m

ax3.set\_xlim(3999, 4001)

ax3.set\_ylim(min\_time4000 - 1.5, max\_time4000 + 1.5)

for i in range(len(y3)):

ax3.plot(x[i], y3[i], marker=".", color="red")

'''выводим точки времени для n=8000 на четвертый график'''

y4 = v8k

avg\_time8000 = 0

min\_time8000 = min(y4)

max\_time8000 = max(y4)

for i in range(len(y4)):

avg\_time8000 += y4[i]

avg\_time8000 /= len(y4)

m = len(y4)

x = [8000] \* m

ax4.set\_xlim(7999, 8001)

ax4.set\_ylim(min\_time8000 - 2, max\_time8000 + 2)

for i in range(len(y4)):

ax4.plot(x[i], y4[i], marker=".", color="red")

'''выводим точки времени для n=16000 на пятый график'''

y5 = v16k

avg\_time16000 = 0

min\_time16000 = min(y5)

max\_time16000 = max(y5)

for i in range(len(y5)):

avg\_time16000 += y5[i]

avg\_time16000 /= len(y5)

m = len(y5)

x = [16000] \* m

ax5.set\_xlim(15999, 16001)

ax5.set\_ylim(min\_time16000 - 2, max\_time16000 + 2)

for i in range(len(y5)):

ax5.plot(x[i], y5[i], marker=".", color="red")

'''выводим точки времени для n=32000 на шестой график'''

y6 = v32k

avg\_time32000 = 0

min\_time32000 = min(y6)

max\_time32000 = max(y6)

for i in range(len(y6)):

avg\_time32000 += y6[i]

avg\_time32000 /= len(y6)

m = len(y6)

x = [32000] \* m

ax6.set\_xlim(31999, 32001)

ax6.set\_ylim(min\_time32000 - 2, max\_time32000 + 2)

for i in range(len(y6)):

ax6.plot(x[i], y6[i], marker=".", color="red")

'''выводим точки времени для n=64000 на седьмой график'''

y7 = v64k

avg\_time64000 = 0

min\_time64000 = min(y7)

max\_time64000 = max(y7)

for i in range(len(y7)):

avg\_time64000+=y7[i]

avg\_time64000/=len(y7)

m = len(y7)

x = [64000] \* m

ax7.set\_xlim(63999, 64001)

ax7.set\_ylim(min\_time64000-2, max\_time64000+2)

for i in range(len(y7)):

ax7.plot(x[i], y7[i], marker=".", color="red")

'''

#выводим точки времени для n=128000 на восьмой график

y8 = v128k

avg\_time128000 = 0

min\_time128000 = min(y8)

max\_time128000 = max(y8)

for i in range(len(y8)):

avg\_time128000+=y8[i]

avg\_time128000/=len(y8)

m = len(y8)

ax8.set\_xlim(127999, 128001)

ax8.set\_ylim(min\_time128000-2, max\_time128000+2)

for i in range(len(y8)):

ax8.plot(x[i], y8[i], marker=".", color="red")

'''

'''выводим график лучшего времени-n на девятый график'''

graphbest = [min\_time1000, min\_time2000, min\_time4000, min\_time8000, min\_time16000, min\_time32000, min\_time64000]

''', min\_time128000]'''

ybestgraph = [1000, 2000, 4000, 8000, 16000, 32000,64000]

''',128000]'''

ax9.plot(graphbest, ybestgraph, color="red", marker='o')

'''выводим график худшего времени-n на девятый график'''

graphworst = [max\_time1000, max\_time2000, max\_time4000, max\_time8000, max\_time16000, max\_time32000, max\_time64000]

''', max\_time128000]'''

yworstgraph = [1000, 2000, 4000, 8000, 16000, 32000, 64000]

''',128000]'''

ax9.plot(graphworst, yworstgraph, color="blue", marker='o')

'''выводим график среднего времени-n на девятый график'''

graphavg = [avg\_time1000, avg\_time2000, avg\_time4000, avg\_time8000, avg\_time16000, avg\_time32000, avg\_time64000]

''', avg\_time128000]'''

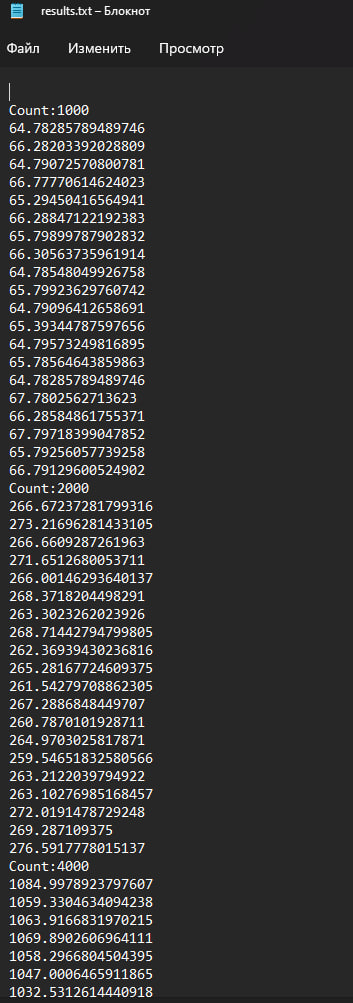
yavggraph = [1000, 2000, 4000, 8000, 16000, 32000,64000]

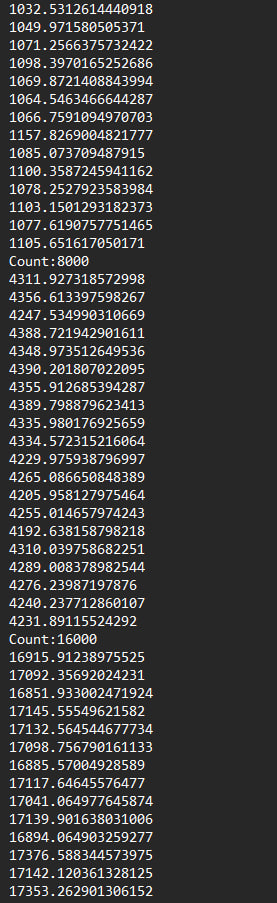
''',128000]'''

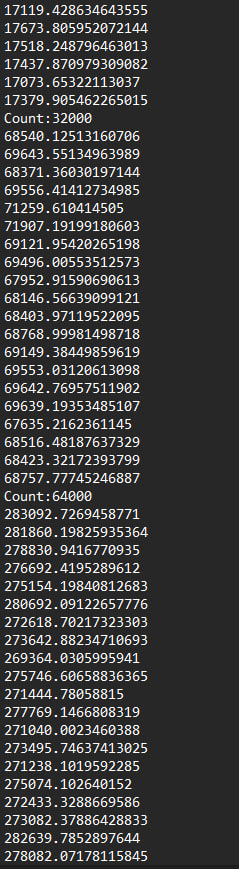
ax9.plot(graphavg, yavggraph, color="green", marker='o')

plt.show()

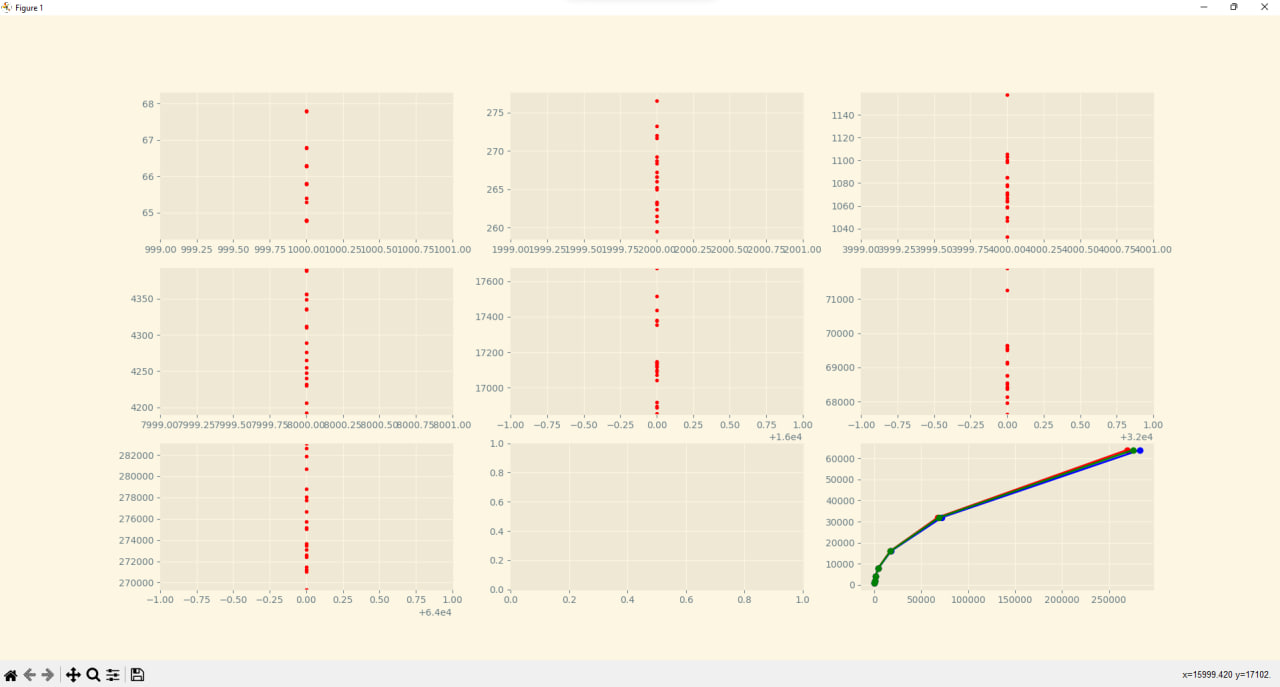
По мере выполнения программы данные записываются в файл.







1. Графики выводятся в отдельном окне по окончанию выполнения программы. Графики идут соответственно для каждого значения N из списка (1000, 2000, 4000, 8000, 16000, 32000, 64000). Предпоследний график зарезервирован для N = 128000, но не использован ввиду слишком долгого времени выполнения программы (ввиду использования работы со списками, а не массивами). На последнем окне показаны графики лучшего (красный), среднего (зеленый) и худшего (синий) замера времени в соответствии с его N.



# Заключение.

В ходе проделанной работы мы приходим к выводу, что метод сортировки пузырьками один из наиболее простых в реализации, поскольку он сводится к повторению проходов по элементам сортируемого массива . Однако кроме как в учебных целях он почти не применяется ввиду своей низкой эффективности. Несмотря на это, на нём основаны многие другие методы. Например, шейкерная сортировка и сортировка расчёской.

Каждый раз, когда увеличиваем количество элементов в массиве для его сортировки, разница во времени значительно увеличивается. К примеру, при N = 2000 время варьируется от 259 до 276 миллисекунд, а при N = 4000 уже от 1032 до 1157 миллисекунд.