|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** ***М-КФ «Машиностроительный»***

**КАФЕДРА** \_***М10-КФ «Высшая математика и физика»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

**ЛАБОРАТОРНАЯ РАБОТА №3**

**«Алгоритмы сортировки»**

**ДИСЦИПЛИНА: «Дискретная математика»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-32Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Карельский М.К. )  (Подпись) |
| Проверил: доцент кафедры М10-КФ | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Булычев В.А. )  (Подпись) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2021

**Цель:** реализация и анализ различных алгоритмов сортировки.

**Задание:**

1. Организуйте заполнение массива (списка) A из N чисел (N вводится пользователем) четырьмя способами (на выбор пользователя):
   1. с консоли ввода;
   2. случайными целыми числами;
   3. возрастающими числами 1, 2, 3, …, N;
   4. убывающими числами N, N-1, …, 3, 2, 1.
2. Реализуйте сортировку полученного в п.1 массива (списка) тремя алгоритмами квадратичной сортировки (выбором, обменом, вставками) и двумя алгоритмами быстрой сортировки (фон Неймана и QuickSort).
3. В качестве контрольного значения выведите после каждой сортировки минимальную из соседних разностей.
4. С помощью системного таймера сравните время работы этих пяти алгоритмов на случайном массиве A из N=500, 1000 и 5000 элементов. Полученные результаты сведите в таблицу
5. Составьте еще две аналогичные таблицы: одну для массива A, в котором записаны по возрастанию все числа от 1 до N, а другую – когда те же числа записаны в обратном порядке от N до 1.

**Листинг:**

import random

import math

from datetime import datetime

N = int(input("Введите размер массива: "))

print("\nВыберете способ ввода массива")

print("1. В ручную")

print("2. Случайно")

print("3. Возрастающими числами")

print("4. Убывающими числами")

means = input(">>> ")

print()

A = []

if means == '1':

A = [int(input("{}: ".format(i + 1))) for i in range(N)]

elif means == '2':

minimum = int(input("Введите минимальное число: "))

maximum = int(input("Введите максимальное число: "))

A = [random.randint(minimum, maximum) for i in range(N)]

elif means == '3':

A = [i + 1 for i in range(N)]

elif means == '4':

A = [i for i in range(N, 0, -1)]

else:

print("Ошибка ввода")

print("\nСозданный массив: " + ' '.join(str(a) for a in A))

print("\nВыберите способ сортировки")

print("1. Выбором")

print("2. Обменом")

print("3. Вставками")

print("4. Слиянием")

print("5. QuickSort")

means = input(">>> ")

print()

start = datetime

end = datetime

if means == '1':

start = datetime.now()

for i in range(N - 1):

minimum = A[i]

index = i

for j in range(i + 1, N):

if A[j] < minimum:

minimum = A[j]

index = j

if i != index:

A[i], A[index] = A[index], A[i]

end = datetime.now()

elif means == '2':

start = datetime.now()

for i in range(N - 1):

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

if A[j] > A[j + 1]:

A[j], A[j + 1] = A[j + 1], A[j]

end = datetime.now()

elif means == '3':

start = datetime.now()

for i in range(1, N):

for j in range(i):

if A[i] < A[j]:

for k in range(i, j, -1):

A[k], A[k - 1] = A[k -1], A[k]

end = datetime.now()

elif means == '4':

def MergeSort(A):

if len(A) > 2:

L = A[:len(A)//2]

R = A[len(A)//2:]

MergeSort(L)

MergeSort(R)

l = r = a = 0

while l < len(L) and r < len(R):

if L[l] < R[r]:

A[a] = L[l]

l += 1

else:

A[a] = R[r]

r += 1

a += 1

if l < len(L):

for i in range(l, len(L)):

A[a] = L[i]

a += 1

else:

for i in range(r, len(R)):

A[a] = R[i]

a += 1

elif len(A) == 2:

if A[0] > A[1]:

A[0], A[1] = A[1], A[0]

start = datetime.now()

MergeSort(A)

end = datetime.now()

elif means == '5':

def QuickSort(A):

if len(A) > 1:

pivot = A[0]

L = []

R = []

for a in A[1:]:

if a < pivot:

L.append(a)

else:

R.append(a)

L = QuickSort(L)

R = QuickSort(R)

L.append(pivot)

A = L + R

return A

start = datetime.now()

A = QuickSort(A)

end = datetime.now()

else:

print("Ошибка ввода")

print("Отсортированный массив: " + ' '.join(str(a) for a in A))

minimum = math.inf

for i in range(len(A) - 1):

if A[i + 1] - A[i] < minimum:

minimum = A[i + 1] - A[i]

print("Контрольное значение: {}".format(minimum))

print("Время работы: {}".format(end - start))

**Время работы:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Выбором | Обменом | Вставками | Слиянием | QuickSort |
| N = 500 | 00.040006 | 00.058595 | 00.064011 | 00.008023 | 00.007999 |
| N = 1000 | 00.112024 | 00.208027 | 00.168053 | 00.008026 | 00.016001 |
| N = 5000 | 01.979666 | 03.743149 | 03.368978 | 00.040008 | 00.256068 |

Табл. 1. Случайные числа

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Выбором | Обменом | Вставками | Слиянием | QuickSort |
| N = 500 | 00.031989 | 00.032011 | 00.032007 | 00.008003 | 00.032009 |
| N = 1000 | 00.105867 | 00.112046 | 00.112025 | 00.007997 | 00.104039 |
| N = 5000 | 01.774623 | 03.072832 | 02.671515 | 00.031987 | 03.641797 |

Табл. 2. Возрастающие числа

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Выбором | Обменом | Вставками | Слиянием | QuickSort |
| N = 500 | 00.048030 | 00.104048 | 00.096021 | 00.007980 | 00.024030 |
| N = 1000 | 00.104037 | 00.308305 | 00.302449 | 00.008002 | 00.116076 |
| N = 5000 | 02.345279 | 06.132961 | 05.410560 | 00.032003 | 02.113784 |

Табл. 3. Убывающие числа

**Вывод:** в ходе выполнения лабораторной работы составлены программы, осуществляющие различные по эффективности алгоритмы сортировки, и проведён их сравнительный анализ.

**Контрольные вопросы:**

1. В этой работе рассмотрены 5 алгоритмов сортировки. Какова сложность каждого из них «в среднем»? Какова их сложность в лучшем случае? В худшем случае?

|  |  |  |  |
| --- | --- | --- | --- |
|  | Худший случай | В среднем | Лучший случай |
| Выбором | O(n2) | O(n2) | O(n2) |
| Обменом | O(n2) | O(n2) | O(n) |
| Вставками | O(n2) | O(n2) | O(n) |
| Слиянием | O(n log n) | O(n log n) | O(n log n) |
| QuickSort | O(n2) | O(n log n) | O(n log n) |

Табл. 4. Сложности

1. Какая сортировка называется устойчивой? Какие из рассмотренных 5-ти алгоритмов являются устойчивыми?

Сортировку называют устойчивой, если она не меняет расположение элементов с одинаковыми ключами.

Устойчивые сортировки: обменом, вставками, слиянием.

1. Объясните почему, несмотря на одинаковую сложность, реальное время выполнения квадратичных алгоритмов сильно отличается?

На скорость выполнения алгоритма помимо входных данных также влияют использованные команды в коде, сторонние процессы во время выполнения программы.