**Министерство образования Республики Беларусь**

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

|  |
| --- |
|  |
|  |

ОТЧЕТ

по лабораторной работе

на тему:

СОРТИРОВКИ

|  |  |  |
| --- | --- | --- |
| Выполнил  Студент гр. 051005 |  | К.Н. Волков |
| Проверил |  | Асс. C.В. Болтак |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Минск, 2021

1. **Задание на лабораторную работу**

Произвести анализ сортировок массивов в соответствии с вариантом, выданным преподавателем. Для чего отсортировать по возрастанию массивы целочисленных элементов различной размерности: 10 элементов, 100 элементов, 2000 элементов. Анализ произвести по числу сравнений и перестановок двух элементов. Исследования производить над массивами трех типов:

1. Массив, содержащий не отсортированные элементы;
2. Массив, содержащий отсортированные элементы;
3. Массив, содержащий элементы, отсортированные в обратном порядке.

Виды сортировок: сортировка прямым выбором, Сортировка пузырьком, Quicksort

Требования к программе:

- программа должна работать в режиме формы, а не консоли;

- интуитивно понятный пользовательский интерфейс;

- построение гистограмм сравнений и перестановок, используя свойство Canvas;

Результат работы программы: таблица, гистограммы сравнений и перестановок.

**2.Описание сортировок с примерами**

2.1 Сортировка прямым выбором

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Исходный массив: | | | | | |
| 176 | 190 | 165 | 162 | 185 | 176 |

1. Ищем минимальный элемент массива (min=162) начиная с i=1 элемента
2. Меняем местами min и первый элемент массива

Теперь массив имеет следующий вид:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 162 | 190 | 165 | 176 | 185 | 176 |

3)Начинаем поиск min элемента с i=2 позиции (min=165)

4)Меняем местами второй элемент и min

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 162 | 165 | 190 | 176 | 185 | 176 |

Вид массива после второго прохода:

5) Начинаем поиск min элемента с i=2 позиции (min=176)

6)Меняем местами i=3 и min элементы

Массив:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 162 | 165 | 176 | 190 | 185 | 176 |

7) меняем местами i=4 min =176

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 162 | 165 | 176 | 176 | 185 | 190 |

Массив:

Массив отсортирован.

2.2 QuickSort

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Исходный массив: | | | | | | | |
| 44 | 55 | 12 | 42 | 94 | 6 | 18 | 67 |

N=8, L=1, R=8, Mid= (1+8) div 2 = 4, Х=42, где

N – кол-во эл-тов массива

L ­– индекс крайнего левого элемента

R ­– индекс крайнего правого элемента

Mid ­– индекс среднего элемента

X ­– значение среднего элемента

После первого прохода по массиву массив разделился на две части относительно числа 42:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 18 | 6 | 12 | **42** | 94 | 55 | 44 | 67 |

Последние значения индексов: i=5, j=3

Сортируем левую часть (от элемента 42) с границами 1 и 3. (правая часть имеет границы 5 и 8)

n=3, Mid= ((L=1)+(R=3)) div 2 = 2, Х=6

Меняем местами 18 и 6:

|  |  |  |
| --- | --- | --- |
| **6** | 18 | 12 |

Последние значения индексов: i=2, j=1. Так как L=j=1, то левый рекурсивный вызов не срабатывает. Так как i=2 и R=3 (i<R), то срабатывает правый рекурсивный вызов:

Mid= ((L=2)+(R=3)) div 2 = 2, Х=18.

Меняем местами 18 и 12:

|  |  |
| --- | --- |
| 12 | 18 |

Последние значения индексов: i=3, j=2. Так как L=j=2, то левый рекурсивный вызов не срабатывает.

Так как i=3=R, то правый рекурсивный вызов не срабатывает.

Сортируем правую часть (от элемента 42) с границами 5 и 8.

n=4, Mid= (5+8) div 2 = 6, Х=55

|  |  |  |  |
| --- | --- | --- | --- |
| 44 | **55** | 94 | 67 |

Переставляем элемент 55 сам с собой.

Последние значения индексов: i=7, j=5. Так как L=j=5, то левый рекурсивный вызов не срабатывает. Так как i=7 и R=8, то срабатывает правый рекурсивный вызов. Исходный массив:

|  |  |
| --- | --- |
| **94** | 67 |

Mid= ((L=7)+(R=8)) div 2 = 7, Х=94. После прохода по массиву меняем местами 94 и 67:

|  |  |
| --- | --- |
| 67 | 94 |

Последние значения индексов: i=8, j=7. Так как L=7=j, i=8=R, то рекурсивных вызовов нет.

Массив:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 6 | 12 | 18 | **42** | 44 | 55 | 67 | 94 |

Массив отсортирован.

2.3 Сортировка пузырьком

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 0 | 1 | 8 | 7 |

1. Берем первый элемент массива i=1;
2. Сравниваем i=1 и следующий i=2 элементы и меняем местами, если i=1 больше i=2
3. i=1 больше i=2,меняем местами
4. получаем:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 3 | 1 | 8 | 7 |

5)Далее берем элемент i=2

6) Сравниваем i=2 и i=3 элементы

7) i=2 больше i=3, меняем местами

8) Получаем:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 3 | 8 | 7 |

9)Далее берем элемент i=3

10) Сравниваем i=3 и i=4 элементы

11) i=3 меньше i=4, следовательно, ничего не меняем

12) Получаем:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 3 | 8 | 7 |

13)Далее берем элемент i=4

14)Сравниваем i=4 и i=5 элементы

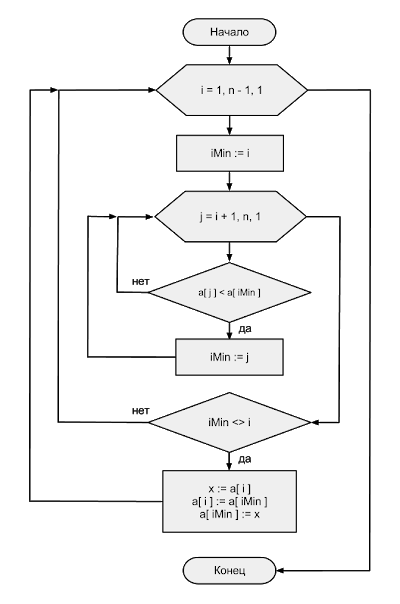
15) i=4 больше i=5, следовательно, меняем местами

16) Получаем:

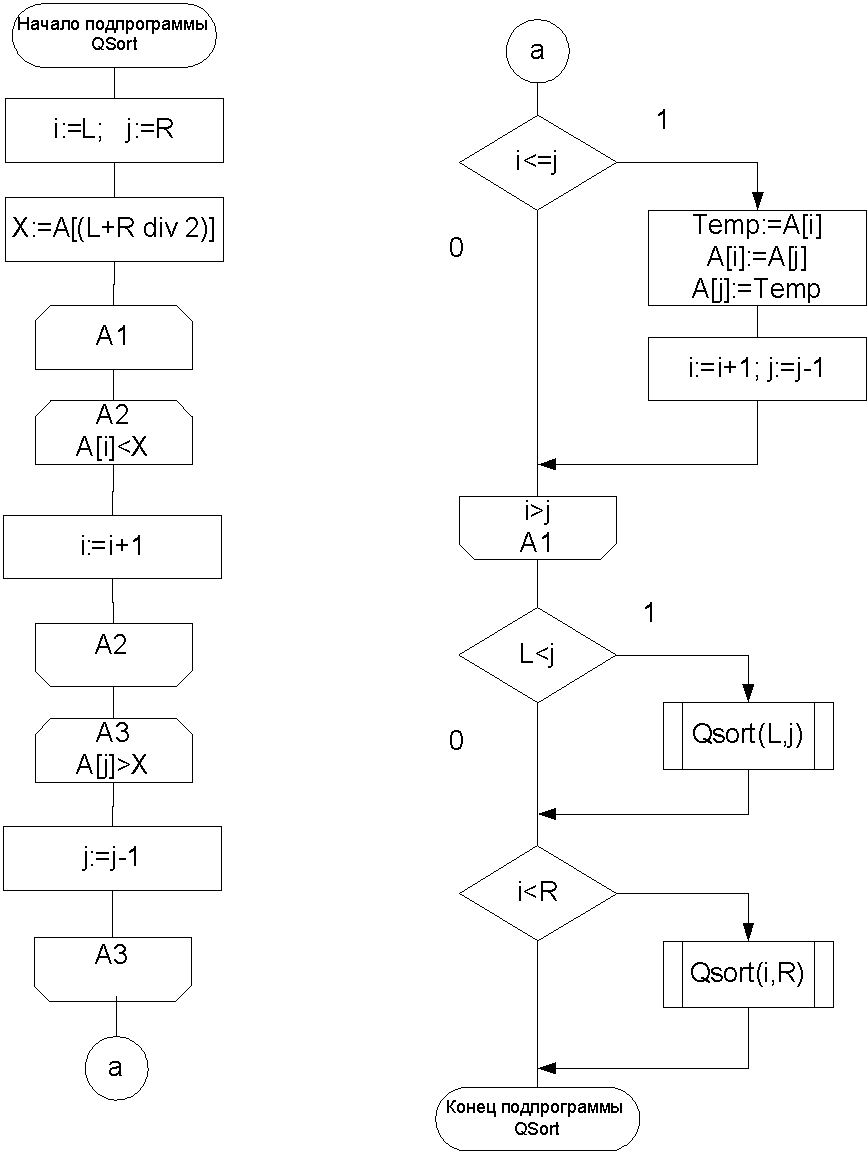
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 3 | 7 | 8 |

**3.Схемы алгоритмов сортировок**

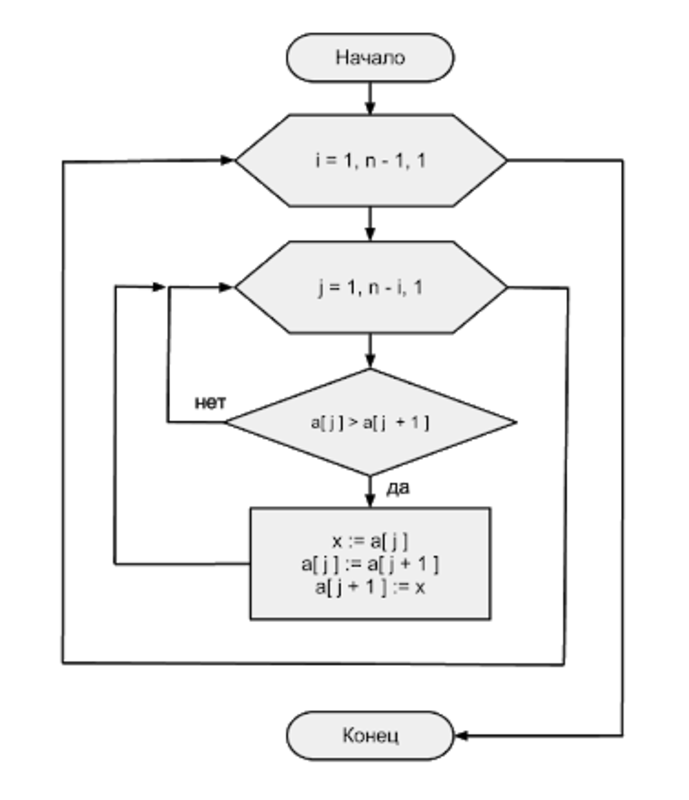
3.1 Сортировка прямым выбором



3.2 Quicksort



3.1 Сортировка пузырьком



**4.Анализ сортировок**

4.1 Сортировка прямым выбором

Число сравнений ключей С не зависит от порядка ключей:

C=½(n^2-n).

Число перестановок минимально

Mmin=3\*(n-1)

в случае изначально упорядоченных ключей и максимально

Mmax= n\*2/4 +3\*(n-1),

если первоначально ключи располагаются в обратном порядке.

Среднее число перестановок

Mср≈n(ln n + g), где g = 0,577216... — константа Эйлера

4.2 Сортировка пузырьком

Плюс данной сортировки — это простота реализации. Однако это один самых медленных методов сортировки (время выполнения квадратично зависит от длины массива n^2).

C=½(n^2-n).

Mmin=0

Mmax=3(n^2-n)/4

Mavg = 3(n^2-n)/2

* 1. Quicksort

Эффективность быстрой сортировки в значительной степени определяется правильностью выбора опорных (ведущих) элементов при формировании блоков. В худшем случае трудоемкость метода имеет ту же сложность, что и пузырьковая сортировка, то есть порядка O(n2). При оптимальном выборе ведущих элементов, когда разделение каждого блока происходит на равные по размеру части, трудоемкость алгоритма совпадает с быстродействием наиболее эффективных способов сортировки, то есть порядка O(n log n). В среднем случае количество операций, выполняемых алгоритмом быстрой сортировки, определяется выражением T(n) = O(1.4n log n)

Число сравнений:

Число перестановок:

1. **Таблица с результатами работы сортировок (сравнения, перестановки)**

Quicksort:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Cavg | Mmin | Mavg |
| 10 | 33 | 7 | 23 |
| 100 | 664 | 153 | 460 |
| 2000 | 21 920 | 5 067 | 15 201 |

Сортировка прямым выбором:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Cavg | Mmin | Mavg |
| 10 | 45 | 27 | 32 |
| 100 | 4950 | 297 | 347 |
| 2000 | 1999000 | 5997 | 6997 |

Сортировка пузырьком:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Cavg | Mmin | Mavg |
| 10 | 45 | 0 | 135 |
| 100 | 4950 | 0 | 14850 |
| 2000 | 1999000 | 0 | 5997000 |

**7.Вывод об эффективности сортировок**

Сортировка пузырьком медленнее чем сортировка прямым выбором и быстрая сортировка. Если сравнивать количество сравнений, то сортировка пузырьком и сортировка прямым выбором уступают быстрой сортировке, поскольку зависят от количества элементов в массиве. Сортировка прямым выбором выигрывает в случае уже отсортированного массива поскольку никаких перестановок не происходит, в отличие от остальных. Быстрая сортировка является самой эффективной на случайном массиве, но может быть неэффективна в случае если массив малых размеров или если он почти отсортирован.