МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ»

Федеральное государственное бюджетное образовательное учреждение
высшего образования
АДЫГЕЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Инженерно-физический факультет
Кафедра автоматизированных систем обработки информации и управления

Отчет по практике

«Сортировка Быстрая и Слиянием.»

Вариант 4

2 курс, группа 2ИВТ1

Выполнил:
О.О. Богомолов 2024 г

Руководитель:

С.В. Теплоухов 2024 г

Майкоп, 2024 г.

0.1. Введение

1. Текстовая формулировка задачи
2. Код данной задачи
3. Скриншот программы

0.2. Вариант 4

0.2.1. Задание

Сортировки Быстрая и Слиянием..

0.2.2. Теория

Быстрая сортировка, хоть и была разработана более 40 лет назад, является наиболее широко применяемым и одним их самых эффективных алгоритмов. Метод основан на подходе "разделяй-и-властвуй". Общая схема такова:

- из массива выбирается некоторый опорный элемент a[i],
- запускается процедура разделения массива, которая перемещает все ключи, меньшие, либо равные a[i], влево от него, а все ключи, большие, либо равные a[i] - вправо,
- теперь массив состоит из двух подмножеств, причем левое меньше, либо равно правого,
- для обоих подмассивов: если в подмассиве более двух элементов, рекурсивно запускаем для него ту же процедуру.

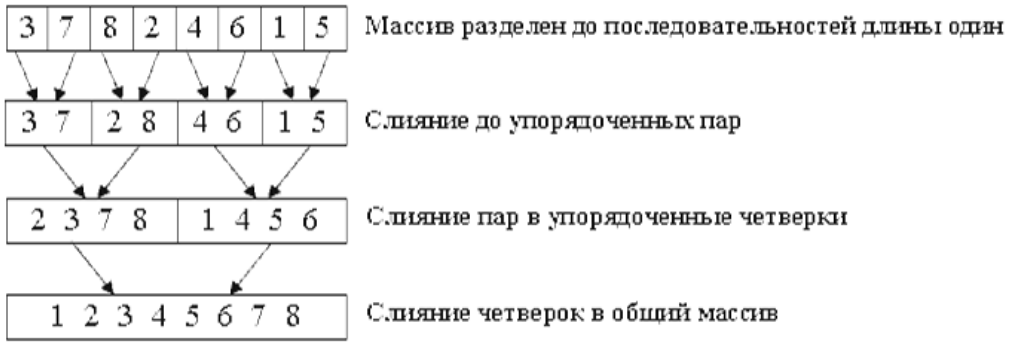
В конце получится полностью отсортированная последовательность. Разделение массива: На входе массив a[0]...a[N] и опорный элемент p, по которому будет производиться разделение.

- Введем два указателя: i и j. В начале алгоритма они указывают, соответственно, на левый и правый конец последовательности.
- Будем двигать указатель i с шагом в 1 элемент по направлению к концу массива, пока не будет найден элемент a[i] >= p. Затем аналогичным образом начнем двигать указатель j от конца массива к началу, пока не будет найден a[j] <= p.
- Далее, если i <= j, меняем a[i] и a[j] местами и продолжаем двигать i,j по тем же правилам...
- Повторяем шаг 3, пока i <= j.

Рассмотрим работу процедуры для массива a[0]...a[6] и опорного элемента p = a[3].
В простейшем случае алгоритм выглядит так:

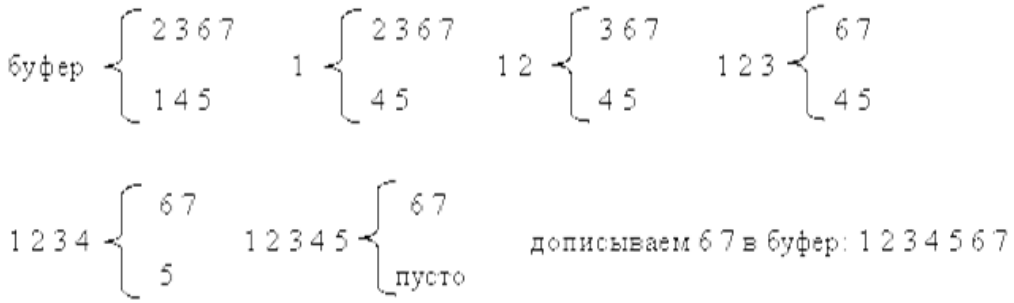
Теперь массив разделен на две части: все элементы левой меньше либо равны p, все элементы правой - больше, либо равны p. Разделение завершено.

Сортировка слиянием также построена на принципе "разделяй-и-властвуй однако реализует его несколько по-другому, нежели quickSort. А именно, вместо разделения по опорному элементу массив просто делится пополам. Функция merge на месте двух упорядоченных массивов a[lb]...a[split] и a[split+1]...a[ub] создает единый упорядоченный массив a[lb]...a[ub]. Пример работы алгоритма на массиве 3 7 8 2 4 6 1 5..



Рекурсивный алгоритм обходит получившееся дерево слияния в прямом порядке. Каждый уровень представляет собой проход сортировки слияния - операцию, полностью переписывающую массив. Обратим внимание, что деление происходит до массива из единственного элемента. Такой массив можно считать упорядоченным, а значит, задача сводится к написанию функции слияния merge. Один из способов состоит в слиянии двух упорядоченных последовательностей при помощи вспомогательного буфера, равного по размеру общему количеству имеющихся в них элементов. Элементы последовательностей будут перемещаться в этот буфер по одному за шаг. merge (упорядоченные последовательности A, B , буфер C) пока A и B непусты сравнить первые элементы A и B переместить наименьший в буфер если в одной из последовательностей еще есть элементы дописать их в конец буфера, сохраняя

имеющийся порядок Пример работы на последовательностях 2 3 6 7 и 1 4 5



Результатом является упорядоченная последовательность, находящаяся в буфере. Каждая операция слияния требует n пересылок и n сравнений, где n - общее число элементов, так что время слияния: Theta(n).

0.3. Ход работы

0.3.1. Код программы

Метод быстрой сортировки

```
#include <iostream>
#include <ctime>
using namespace std;
const int n = 20;
void quicksortHoara(int* M, int l, int r)
{
int i = l, j = r;
double count, x;
x = M[(l + r) / 2];
do
{
while (M[i] < x) i++;
while (M[j] > x) j--;
if (i <= j)
{
count = M[i];
M[i] = M[j];
M[j] = count;
i++; j--;
}
} while (i <= j);

if (i < r)
quicksortHoara(M, i, r);
```

```
if (j > l)
quicksortHoara(M, l, j);
}

//главная функция
int main()
{
setlocale(LC_ALL, "Russian");
srand(time(NULL));

int A[n], i; //объявление динамического массива
cout << "Вывод массива:\n";
for (i = 0; i < n; i++) //ввод массива
{
A[i] = rand() % 100;
cout << A[i] << " ";
}
cout << endl;
quicksortHoara(A, 0, n - 1);
cout << "Отсортированный массив:\n";
for (i = 0; i < n; i++) //ввод массива
{
cout << A[i] << " ";
}
cout << endl;
system("pause");
return 0;
}
```

Сортировка Методом слияния

```
#include <iostream>
using namespace std;
//функция, сливающая массивы
void Merge(int* A, int first, int last)
{
int middle, start, final, j;
int* mas = new int[100];
middle = (first + last) / 2; //вычисление среднего элемента
start = first; //начало левой части
final = middle + 1; //начало правой части
```

```
for (j = first; j <= last; j++) //выполнять от начала до конца
if ((start <= middle) && ((final > last) || (A[start] < A[final])))
{
mas[j] = A[start];
start++;
}
else
{
mas[j] = A[final];
final++;
}
//возвращение результата в список
for (j = first; j <= last; j++) A[j] = mas[j];
delete[]mas;
};
//рекурсивная процедура сортировки
void MergeSort(int* A, int first, int last)
{
{
if (first < last)
{
MergeSort(A, first, (first + last) / 2); //сортировка левой части
MergeSort(A, (first + last) / 2 + 1, last); //сортировка правой части
Merge(A, first, last); //слияние двух частей
}
}
};
//главная функция
void main()
{
setlocale(LC_ALL, "Rus");
int i, n;
int* A = new int[100];
cout << "Размер массива > "; cin >> n;
for (i = 1; i <= n; i++)
{
cout << i << " элемент > "; cin >> A[i];
}
MergeSort(A, 1, n); //вызов сортирующей процедуры
cout << "Упорядоченный массив: "; //вывод упорядоченного массива
for (i = 1; i <= n; i++) cout << A[i] << " ";
```

```
delete[]A;
system("pause>>void");
}
```

0.4. Скриншот программы

Сортировки Быстрая и Слиянием.

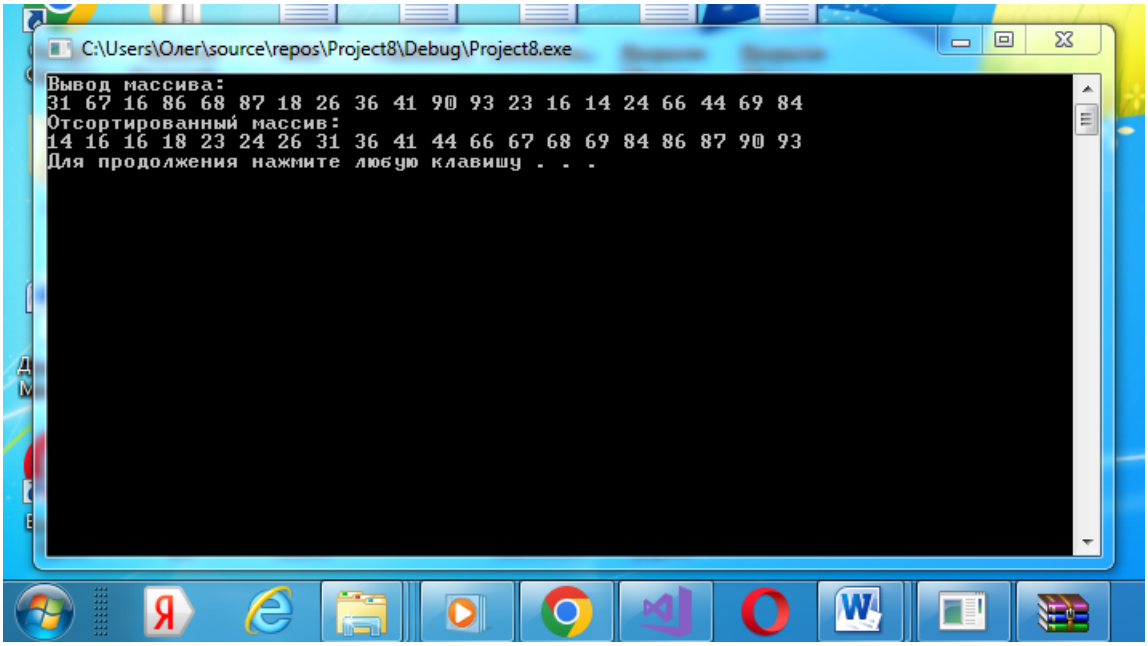


Рис. 0.1: Скриншот программы метод быстрой сортировки

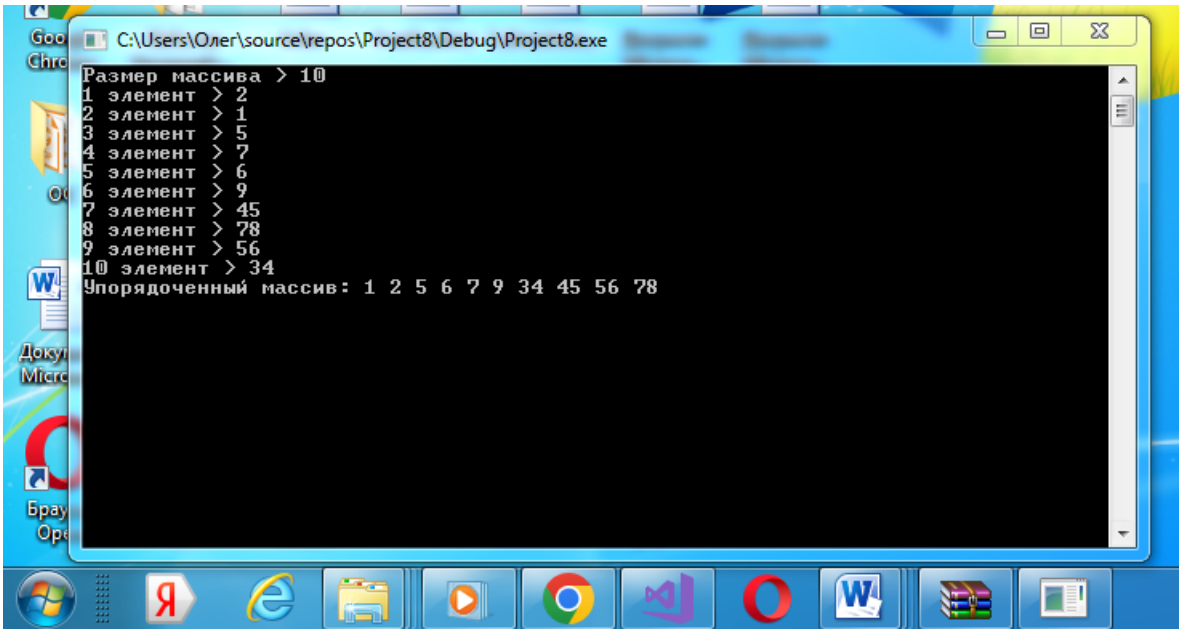


Рис. 0.2: Скриншот программы сортивка методом слиянием

0.5. Библиографические ссылки

Для изучения «внутренностей» TEX необходимо изучить [1], а для использования LATEX лучше почитать [2, 3].

Литература

[1] Кнут Д.Э. Всё про TEX. —- Москва: Изд. Вильямс, 2003 г. 550 с.

[2] Львовский С.М. Набор и верстка в системе LATEX. —- 3-е издание, исправленное и дополненное, 2003 г.

[3] Воронцов К.В. LATEX в примерах. 2005 г.