Балтийский государственный технический университет  
«ВОЕНМЕХ» им. Д. Ф. Устинова

Кафедра О7 «Информационные системы и программная инженерия»

**Практическая работа №3**по дисциплине «Структуры и организация данных»  
на тему «Оценка эффективности алгоритмов»  
часть 1 «Алгоритмы сортировки»  
  
вариант 4

Выполнил:  
Студент Антонов Н. М.  
Группа И924Б   
  
Преподаватель:  
Гладевич А.А.

Санкт-Петербург  
2023г.

Уровень сложности – базовый. Провести сравнение указанных алгоритмов сортировки массивов, содержащих *N1*, *N2*, *N3* и *N4* элементов, по указанному в вариативной части критерию и объему требуемой дополнительно памяти.

Порядок: по возрастанию элементов.

Методы: пузырька, шейкера, быстрая сортировка, сортировка естественным слиянием.

N1 = 10000, N2 = 50000, N3 = 100000, N4 = 150000. Критерий – количество сравнений.

1. Алгоритм сортировки пузырьком:

Алгоритмпроходит по списку несколько раз, сравнивая два соседних элемента и меняя их местами, если они находятся в неправильном порядке.

Трудоемкость сортировки пузырьком по количеству сравнений:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Случай** | **Ситуация, соответст-вующая случаю** | **Обоснование** | **Ожидаемое число**  **сравнений элементов массива** | **Асимптотическая оценка сложности по количеству сравнений** | **Ожидаемое число вспомогат.**  **сравнений** |
| Наилучший  Средний  Наихудший | Массив может быть любым | Сортировка не зависит от упорядоченности элементов, она работает всегда одинаково | N1: 49995000  N2: 1249975000  N3: 499950000  N4: 11249925000 | O(n2) | N1: 50015001  N2: 1250075001  N3: 5000150001  N4: 11250225001 |

Пространственная сложность – O(1) (две переменных цикла и вспомогательная переменная).

1. Алгоритм шейкерной сортировки:

Алгоритм последовательно просматривает элементы массива слева направо, сравнивая соседние элементы и, если они находятся в неправильном порядке, меняет их местами. Затем алгоритм движется в обратном направлении — справа налево, опять выполняя сравнения и перемещения. Этот процесс продолжается до тех пор, пока не будет достигнут правильный порядок элементов.

Трудоемкость шейкерной сортировки по количеству сравнений:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Случай** | **Ситуация, соответст-вующая случаю** | **Обоснование** | **Ожидаемое число сравнений элементов массива** | **Асимптотическая оценка сложности по количеству сравнений** | **Ожидаемое число вспомогат. сравнений** |
| Наилучший | Массив упорядочен по возрастанию | Массив уже полностью отсортирован | *n-1*  N1: 9999  N2: 49999  N3: 99999  N4: 149999 | Ω(n) | *n+2*  N1: 10002  N2: 50002  N3: 100002  N4: 150002 |
| Наихудший | Массив упорядочен по убыванию | Потребуется отсортировать массив в обратном порядке | N1: 49995000  N2: 1249975000  N3: 4999950000  N4:11249925000 | O(n2) | N1: 50015001  N2: 1250075001  N3: 5000150001  N4: 11250225001 |
| Средний | Частично упорядоченный массив | Массив может быть частично упорядочен | N1: 33330000  N2: 833317000  N3: 3333300000  N4: 7499950000 | O(n2) | N1: 33330000  N2: 833317000  N3: 3333300000  N4: 7499950000 |

Пространственная сложность – O(1) (две переменных цикла и 3 вспомогательных переменных).

1. Алгоритм быстрой сортировки:

Алгоритм сначала выбирает опорный элемент, далее массив делится на две подгруппы: меньшие опорного элемента и большие опорного элемента. Далее рекурсивно повторяется алгоритм, пока подгруппы не станут размером меньше или равному 1.

Трудоемкость быстрой сортировки по количеству сравнений:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Случай** | **Ситуация, соответст-вующая случаю** | **Обоснование** | **Ожидаемое число сравнений элементов массива** | **Асимптотическая оценка сложности по количеству сравнений** | **Ожидаемое число вспомогат. сравнений** |
| наилучший | Каждый опорный элемент является медианой сортируемой части | Достаточно совершать вызовы только медианы | n\*log(n)  N1: 132877  N2: 780482  N3: 1660960  N4: 2579190 | Ω(n\*log(n)) | *n/2 \* log(n/2)*  N1: 61438  N2: 365241  N3: 780482  N4: 1214600 |
| наихудший | Каждый опорный элемент является минимальным или максимальным в сортируемой части | На каждом этапе массив из n элементов будет делиться на левую часть, состоящую из n-1 элементов, а правая часть будет одним единственным числом, будут совершаться n2 проходов | N1: 49995000  N2: 1249975000  N3: 4999950000  N4: 11249925000 | O(n2) | N1: 24997500  N2: 624987500  N3: 2499975000  N4: 5624962500 |
| средний | При делении массива на подгруппы, получившиеся массивы будут не очень сильно отличаться по длине | Разделяй и властвуй принцип | *1.33\*n\*log(n)*  N1: 176726  N2: 1038042  N3: 2209080  N4: 3430320 | O(n\*log(n)) | N1: 88363  N2: 519021  N3: 1104540  N4: 1715160 |

Пространственная сложность – O(logn).

1. Алгоритм сортировки естественным слиянием:

Алгоритм начинается с деления исходного списка на отдельные подсписки. Это делается путем прохода по списку и нахождения участков, где элементы находятся в возрастающем порядке. Когда обнаруживается участок, где элементы идут в порядке убывания, он считается концом текущего подсписка. Затем этот подсписок добавляется в список отсортированных подсписков. Если список имеет только один подсписок (т.е. все элементы уже упорядочены), алгоритм сортировки считается завершенным.

Трудоемкость сортировки естественным слиянием по количеству сравнений:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Случай** | **Ситуация, соответст-вующая случаю** | **Обоснование** | **Ожидаемое число сравнений элементов массива** | **Асимптотическая оценка сложности по количеству сравнений** | **Ожидаемое число вспомогат. сравнений** |
| Наилучший | Массив уже полностью отсортирован | В этом случае сортировка естественным слиянием может быстро обнаружить отсортированные подсписки и объединит их | *n-1*  N1: 9999  N2: 49999  N3: 99999  N4: 149999 | O(n) | 0  N1: 0  N2: 0  N3: 0  N4: 0 |
| Наихудший | Массив обратно упорядочен | В этом случае каждый элемент будет рассматриваться как отдельный подсписок, и при каждом объединении будет создаваться новый подсписок | *n-1*  N1: 9999  N2: 49999  N3: 99999  N4: 149999 | O(n2) | N1: 50015001  N2: 1250075001  N3: 5000150001  N4: 11250225001 |
| Средний | Массив частично упорядочен | Некоторые подмассивы будут уже находиться в правильном порядке | *n\*log(n)*  N1: 132877  N2: 780482  N3: 1660960  N4: 2579190 | O(n\*log(n)) | n\*3\*log(n)  N1: 398631  N2: 2341450  N3: 4982890  N4: 7737570 |

Пространственная сложность – O(1) (две переменных цикла и 8 вспомогательных переменных).

Текст программы:

#include <iostream>

#include <fstream>

#include <algorithm>

#include <ctime>

#include <Windows.h>

#define MAXSTACK 2048

using namespace std;

unsigned long int memory, cntosn, cntvtr;

bool compare\_lower(long int a, long int b) {

return (a < b) ? true : false;

}

void BubbleSort(long\* arr, int n) {

int x = 0;

cntosn = 0; // Счетчик основных сравнений

cntvtr = 0; // Счетчик второстепенных сравнений

for (int i = 1; cntvtr++, i < n; i++)

for (int j = n - 1; cntvtr++, j >= i; j--)

if (cntosn++, arr[j - 1] < arr[j])

{

x = arr[j - 1];

arr[j - 1] = arr[j];

arr[j] = x;

}

memory = 3 \* sizeof(int);

}

void cocktailSort(long a[], int n) {

int k = n - 1, left = 1, right = n - 1, x;

cntosn = 0;

cntvtr = 0;

memory = 0;

memory += sizeof(int) \* 5;

bool swapped = true;

int start = 0;

int end = n - 1;

do

{

for (int j = right; j >= left; j--)

{

cntosn++; // Увеличиваем счетчик основных сравнений

if (a[j - 1] > a[j])

{

x = a[j - 1];

a[j - 1] = a[j];

a[j] = x;

k = j;

}

}

left = k + 1;

for (int j = left; j <= right; j++)

{

cntosn++; // Увеличиваем счетчик основных сравнений

if (a[j - 1] > a[j])

{

x = a[j - 1];

a[j - 1] = a[j];

a[j] = x;

k = j;

cntvtr++; // Увеличиваем счетчик второстепенных сравнений

}

}

right = k - 1;

} while (left < right);

}

void swap(long\* a, long\* b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void swap(int\* a, int\* b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void quicksort(long\* a, int n) {

const int M = log(n) / log(2) + 1;

int i, j, left, right, s, x, w;

struct stack { int left, right; } \*stack;

stack = new struct stack[M];

s = 0;

stack[0].left = 0;

stack[0].right = n - 1;

do /\*выбор из стека последнего запроса\*/

{

left = stack[s].left;

right = stack[s].right;

s--;

do /\*разделение а[left]... a[right]\*/

{

i = left; j = right;

x = a[(left + right) / 2];

do

{

while (cntosn++, a[i] < x) i++;

while (cntosn++, x < a[j]) j--;

if (cntvtr++, i <= j)

{

w = a[i]; a[i] = a[j]; a[j] = w;

i++; j--;

}

} while (cntvtr++, i < j);

if (cntvtr++, i < right && (cntvtr++, right - i >= j - left)) /\*если правая часть не меньше левой\*/

{ /\*запись в стек границ правой части\*/

s++;

stack[s].left = i;

stack[s].right = right;

right = j; /\*теперь left и right ограничивают левую часть\*/

}

else if (cntvtr++, j > left && (cntvtr++, j - left > right - i)) /\*если левая часть больше правой\*/

{ /\*запись в стек границ левой части\*/

s++;

stack[s].left = left;

stack[s].right = j;

left = i; /\*теперь left и right ограничивают правую часть\*/

}

else left = right; /\*делить больше нечего, интервал "схлопывается"\*/

} while (cntvtr++, left < right);

} while (cntvtr++, s > -1);

delete stack;

memory += sizeof(stack);

memory += sizeof(int);

}

void mergeSort(long\* a, int n)

{

memory = 0;

cntosn = 0;

cntvtr = 0;

int split;

int last, end;

long\* p = a, \* tmp;

char flag = 0, sorted = 0;

int pos1, pos2, pos3;

tmp = new long[n];

do

{

end = n;

pos2 = pos3 = 0;

do

{

p += pos2;

end = n - pos3;

for (split = 1; split < end && p[split - 1] <= p[split]; split++)

cntosn++;

if (split == n)

{

sorted = 1;

break;

}

pos1 = 0;

pos2 = split;

while (pos1 < split && pos2 < end)

{

if (p[pos1] > p[pos2])

{

tmp[pos3++] = p[pos2++];

cntvtr++;

}

else

{

tmp[pos3++] = p[pos1++];

cntvtr++;

}

}

while (pos2 < end && tmp[pos3 - 1] <= p[pos2])

{

tmp[pos3++] = p[pos2++];

cntvtr++;

}

while (pos1 < split)

{

tmp[pos3++] = p[pos1++];

cntvtr++;

}

} while (pos3 < n);

if (sorted)

break;

p = tmp;

tmp = a;

a = p;

flag = !flag;

} while (split < n);

if (flag)

{

for (pos1 = 0; pos1 < n; pos1++)

tmp[pos1] = a[pos1];

delete[] a;

}

else

{

delete[] tmp;

}

memory += 6 \* sizeof(int);

memory += 2 \* sizeof(long);

memory += 2 \* sizeof(char);

}

int main(int argc, char\* argv[]) {

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

const int n1 = 10000, n2 = 50000, n3 = 100000, n4 = 150000;

clock\_t start = 0;

clock\_t end = 0;

long\* arr1 = new long[10000], \* arr2 = new long[50000], \* arr3 = new long[100000], \* arr4 = new long[150000];

int w, s;

ifstream infile("C:\\Users\\NiKiToS\\Downloads\\test\_numbers (2).txt");

if (!infile)

{

cout << "Error openning file";

return 0;

}

infile.seekg(0);

for (int i = 0; i < n1; i++) {

infile >> w;

arr1[i] = w;

}

infile.seekg(0);

for (int i = 0; i < n2; i++) {

infile >> w;

arr2[i] = w;

}

infile.seekg(0);

for (int i = 0; i < n3; i++) {

infile >> w;

arr3[i] = w;

}

infile.seekg(0);

for (int i = 0; i < n4; i++) {

infile >> w;

arr4[i] = w;

}

infile.seekg(0);

cout << "СОРТИРОВКА ПУЗЫРЬКОМ" << endl << endl;

cout << "----------------------------------------" << endl;

cout << "10 000 Elements" << endl;

cout << "Неупорядоченный массив" << endl;

start = 0;

end = 0;

start = clock();

BubbleSort(arr1, n1);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный по возрастанию" << endl;

start = 0;

end = 0;

start = clock();

BubbleSort(arr1, n1);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный в обратном порядке" << endl;

start = 0;

end = 0;

for (int i = 0; i < n1 / 2; i++)

arr1[i] = arr1[n1 - i - 1];

start = clock();

BubbleSort(arr1, n1);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

cout << "----------------------------------------" << endl;

cout << "50 000 Elements" << endl;

cout << "Неупорядоченный массив" << endl;

start = 0;

end = 0;

start = clock();

BubbleSort(arr2, n2);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный по возрастанию" << endl;

start = 0;

end = 0;

start = clock();

BubbleSort(arr2, n2);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный в обратном порядке" << endl;

start = 0;

end = 0;

for (int i = 0; i < n2 / 2; i++)

arr2[i] = arr2[n2 - i - 1];

start = clock();

BubbleSort(arr2, n2);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

cout << "----------------------------------------" << endl;

cout << "100 000 Elements" << endl;

cout << "Неупорядоченный массив" << endl;

start = 0;

end = 0;

start = clock();

BubbleSort(arr3, 100000);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный по возрастанию" << endl;

start = 0;

end = 0;

start = clock();

BubbleSort(arr3, 100000);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный в обратном порядке" << endl;

start = 0;

end = 0;

for (int i = 0; i < n3 / 2; i++)

arr3[i] = arr3[n3 - i - 1];

start = clock();

BubbleSort(arr3, 100000);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

cout << "----------------------------------------" << endl;

cout << "150 000 Elements" << endl;

cout << "Неупорядоченный массив" << endl;

start = 0;

end = 0;

start = clock();

BubbleSort(arr4, n4);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный по возрастанию" << endl;

start = 0;

end = 0;

start = clock();

BubbleSort(arr4, n4);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный в обратном порядке" << endl;

start = 0;

end = 0;

s = n4;

for (int i = 0; i < n4 / 2; i++)

arr4[i] = arr4[n4 - i - 1];

start = clock();

BubbleSort(arr4, n4);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

infile.seekg(0);

for (int i = 0; i < n1; i++) {

infile >> w;

arr1[i] = w;

}

infile.seekg(0);

for (int i = 0; i < n2; i++) {

infile >> w;

arr2[i] = w;

}

infile.seekg(0);

for (int i = 0; i < n3; i++) {

infile >> w;

arr3[i] = w;

}

infile.seekg(0);

for (int i = 0; i < n4; i++) {

infile >> w;

arr4[i] = w;

}

infile.seekg(0);

cout << "ШЕЙКЕРНАЯ СОРТИРОВКА" << endl << endl;

cout << "----------------------------------------" << endl;

cout << "10 000 Elements" << endl;

cout << "Неупорядоченный массив" << endl;

start = 0;

end = 0;

start = clock();

cocktailSort(arr1, n1);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный по возрастанию" << endl;

start = 0;

end = 0;

start = clock();

cocktailSort(arr1, n1);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный в обратном порядке" << endl;

start = 0;

end = 0;

for (int i = 0; i < n1 / 2; i++)

arr1[i] = arr1[n1 - i - 1];

start = clock();

cocktailSort(arr1, n1);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

cout << "----------------------------------------" << endl;

cout << "50 000 Elements" << endl;

cout << "Неупорядоченный массив" << endl;

start = 0;

end = 0;

start = clock();

cocktailSort(arr2, n2);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный по возрастанию" << endl;

start = 0;

end = 0;

start = clock();

cocktailSort(arr2, n2);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный в обратном порядке" << endl;

start = 0;

end = 0;

for (int i = 0; i < n2 / 2; i++)

arr2[i] = arr2[n2 - i - 1];

start = clock();

cocktailSort(arr2, n2);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

cout << "----------------------------------------" << endl;

cout << "100 000 Elements" << endl;

cout << "Неупорядоченный массив" << endl;

start = 0;

end = 0;

start = clock();

cocktailSort(arr3, n3);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный по возрастанию" << endl;

start = 0;

end = 0;

start = clock();

cocktailSort(arr3, n3);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный в обратном порядке" << endl;

start = 0;

end = 0;

for (int i = 0; i < n3 / 2; i++)

arr3[i] = arr3[n3 - i - 1];

start = clock();

cocktailSort(arr3, n3);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

cout << "----------------------------------------" << endl;

cout << "150 000 Elements" << endl;

cout << "Неупорядоченный массив" << endl;

start = 0;

end = 0;

start = clock();

cocktailSort(arr4, n4);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный по возрастанию" << endl;

start = 0;

end = 0;

start = clock();

cocktailSort(arr4, n4);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный в обратном порядке" << endl;

start = 0;

end = 0;

for (int i = 0; i < n4 / 2; i++)

arr4[i] = arr4[n4 - i - 1];

start = clock();

cocktailSort(arr4, n4);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

infile.seekg(0);

for (int i = 0; i < n1; i++) {

infile >> w;

arr1[i] = w;

}

infile.seekg(0);

for (int i = 0; i < n2; i++) {

infile >> w;

arr2[i] = w;

}

infile.seekg(0);

for (int i = 0; i < n3; i++) {

infile >> w;

arr3[i] = w;

}

infile.seekg(0);

for (int i = 0; i < n4; i++) {

infile >> w;

arr4[i] = w;

}

infile.seekg(0);

cout << "БЫСТРАЯ СОРТИРОВКА" << endl << endl;

cout << "----------------------------------------" << endl;

cout << "10 000 Elements" << endl;

cout << "Неупорядоченный массив" << endl;

start = 0;

end = 0;

cntosn = 0;

cntvtr = 0;

start = clock();

quicksort(arr1, n1);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный по возрастанию" << endl;

start = 0;

end = 0;

cntosn = 0;

cntvtr = 0;

start = clock();

quicksort(arr1, n1);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный в обратном порядке" << endl;

cntosn = 0;

cntvtr = 0;

start = 0;

end = 0;

for (int i = 0; i < n1 / 2; i++)

arr1[i] = arr1[n1 - i - 1];

start = clock();

quicksort(arr1, n1);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

cout << "----------------------------------------" << endl;

cout << "50 000 Elements" << endl;

cout << "Неупорядоченный массив" << endl;

start = 0;

end = 0;

cntosn = 0;

cntvtr = 0;

start = clock();

quicksort(arr2, n2);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный по возрастанию" << endl;

start = 0;

end = 0;

cntosn = 0;

cntvtr = 0;

start = clock();

quicksort(arr2, n2);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный в обратном порядке" << endl;

start = 0;

end = 0;

cntosn = 0;

cntvtr = 0;

for (int i = 0; i < n2 / 2; i++)

arr2[i] = arr2[n2 - i - 1];

start = clock();

quicksort(arr2, n2);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

cout << "----------------------------------------" << endl;

cout << "100 000 Elements" << endl;

cout << "Неупорядоченный массив" << endl;

start = 0;

end = 0;

cntosn = 0;

cntvtr = 0;

start = clock();

quicksort(arr3, n3);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный по возрастанию" << endl;

start = 0;

end = 0;

cntosn = 0;

cntvtr = 0;

start = clock();

quicksort(arr3, n3);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный в обратном порядке" << endl;

start = 0;

end = 0;

cntosn = 0;

cntvtr = 0;

for (int i = 0; i < n3 / 2; i++)

arr3[i] = arr3[n3 - i - 1];

start = clock();

quicksort(arr3, n3);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

cout << "----------------------------------------" << endl;

cout << "150 000 Elements" << endl;

cout << "Неупорядоченный массив" << endl;

start = 0;

end = 0;

cntosn = 0;

cntvtr = 0;

start = clock();

quicksort(arr4, n4);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный по возрастанию" << endl;

start = 0;

end = 0;

cntosn = 0;

cntvtr = 0;

start = clock();

quicksort(arr4, n4);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный в обратном порядке" << endl;

start = 0;

end = 0;

cntosn = 0;

cntvtr = 0;

for (int i = 0; i < n4 / 2; i++)

arr4[i] = arr4[n4 - i - 1];

start = clock();

quicksort(arr4, n4);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

infile.seekg(0);

for (int i = 0; i < n1; i++) {

infile >> w;

arr1[i] = w;

}

infile.seekg(0);

for (int i = 0; i < n2; i++) {

infile >> w;

arr2[i] = w;

}

infile.seekg(0);

for (int i = 0; i < n3; i++) {

infile >> w;

arr3[i] = w;

}

infile.seekg(0);

for (int i = 0; i < n4; i++) {

infile >> w;

arr4[i] = w;

}

infile.seekg(0);

cout << "СОРТИРОВКА ЕСТЕСТВЕННЫМ СЛИЯНИЕМ" << endl << endl;

cout << "----------------------------------------" << endl;

cout << "10 000 Elements" << endl;

cout << "Неупорядоченный массив" << endl;

start = 0;

end = 0;

cntosn = 0;

cntvtr = 0;

start = clock();

mergeSort(arr1, n1);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный по возрастанию" << endl;

start = 0;

end = 0;

cntosn = 0;

cntvtr = 0;

start = clock();

mergeSort(arr1, n1);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный в обратном порядке" << endl;

cntosn = 0;

cntvtr = 0;

start = 0;

end = 0;

for (int i = 0; i < n1 / 2; i++)

arr1[i] = arr1[n1 - i - 1];

start = clock();

mergeSort(arr1, n1);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

cout << "----------------------------------------" << endl;

cout << "50 000 Elements" << endl;

cout << "Неупорядоченный массив" << endl;

start = 0;

end = 0;

cntosn = 0;

cntvtr = 0;

start = clock();

mergeSort(arr2, n2);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный по возрастанию" << endl;

start = 0;

end = 0;

cntosn = 0;

cntvtr = 0;

start = clock();

mergeSort(arr2, n2);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный в обратном порядке" << endl;

start = 0;

end = 0;

cntosn = 0;

cntvtr = 0;

for (int i = 0; i < n2 / 2; i++)

arr2[i] = arr2[n2 - i - 1];

start = clock();

mergeSort(arr2, n2);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

cout << "----------------------------------------" << endl;

cout << "100 000 Elements" << endl;

cout << "Неупорядоченный массив" << endl;

start = 0;

end = 0;

cntosn = 0;

cntvtr = 0;

start = clock();

mergeSort(arr3, n3);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный по возрастанию" << endl;

start = 0;

end = 0;

cntosn = 0;

cntvtr = 0;

start = clock();

mergeSort(arr3, n3);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный в обратном порядке" << endl;

start = 0;

end = 0;

cntosn = 0;

cntvtr = 0;

for (int i = 0; i < n3 / 2; i++)

arr3[i] = arr3[n3 - i - 1];

start = clock();

mergeSort(arr3, n3);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

cout << "----------------------------------------" << endl;

cout << "150 000 Elements" << endl;

cout << "Неупорядоченный массив" << endl;

start = 0;

end = 0;

cntosn = 0;

cntvtr = 0;

start = clock();

mergeSort(arr4, n4);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный по возрастанию" << endl;

start = 0;

end = 0;

cntosn = 0;

cntvtr = 0;

start = clock();

mergeSort(arr4, n4);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

start = clock();

cout << "Упорядоченный в обратном порядке" << endl;

start = 0;

end = 0;

cntosn = 0;

cntvtr = 0;

for (int i = 0; i < n4 / 2; i++)

arr4[i] = arr4[n4 - i - 1];

start = clock();

mergeSort(arr4, n4);

end = clock();

cout << "Основные сравнения: " << cntosn << endl;

cout << "Второстепенные сравнения: " << cntvtr << endl;

cout << "Количество памяти: " << memory << endl;

cout << "Затраченное время: " << double(end - start) / CLOCKS\_PER\_SEC << endl << endl;

return 0;

}

Результаты работы программы:

Анализ полученных данных:

*Сравниваем полученные результаты с расчетными, делаем выводы.*

*Для каждого алгоритма сравниваем количество основных операций с количеством вспомогательных, анализируем, как изменяется соотношение при увеличении количества элементов, делаем выводы.*

*Сравниваем между собой трудоемкости анализируемых алгоритмов сортировки по асимптотике и по полученным программно результатам (в лучшем, худшем и среднем случаях), делаем выводы.*

*Сравниваем между собой время работы рассматриваемых алгоритмов (в лучшем, среднем и худшем случаях), соотносим их с количеством подсчитанных операций, делаем выводы.*

*Сравниваем пространственную сложность анализируемых алгоритмов, соотносим ее с полученным временем сортировки, делаем выводы.*

*При выполнении задания на повышенном уровне сложности анализируем результаты, полученные для последовательностей с повторяющимися ключами, и сравниваем их со сделанным ранее прогнозом, делаем выводы.*

Вывод: по результатам проведенного анализа самым эффективным из рассмотренных алгоритмов по соотношению время-память является алгоритм сортировки *методом случайных перестановок*.