Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №2

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему « Оценка времени выполнения программ»

Выполнил:

студент группы 20ВВ2

Кочергин В.П.

Принял:

к.т.н. доцент Юрова О.В.

д.т.н. профессор Митрохин М. А.

Пенза 2021

**Цель работы:** измерить время перемножения матриц и выполнения два вида сортировок с разными массивами.

**Задание 1**

Листинг:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <windows.h>

int main(void)

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setvbuf(stdin, NULL, \_IONBF, 0);

setvbuf(stdout, NULL, \_IONBF, 0);

clock\_t start, end, start1, end1;

int i = 0, j = 0, r;

int \*\*a, \*\*b, \*\*c, proizv;

int max, max2;

printf("Укажите размер матрицы: ");

scanf\_s("%d", &max );

a = (int\*\*)malloc(max \* sizeof(int));

b = (int\*\*)malloc(max \* sizeof(int));

c = (int\*\*)malloc(max \* sizeof(int));

for (i = 0; i < max; i++)

{

a[i] = (int\*)malloc(max \* sizeof(int));

}

for (i = 0; i < max; i++)

{

b[i] = (int\*)malloc(max \* sizeof(int));

}

for (i = 0; i < max; i++)

{

c[i] = (int\*)malloc(max \* sizeof(int));

}

i = 0;

start = clock();

srand(time(NULL));

while (i < max)

{

while (j < max)

{

a[i][j] = rand() % 10 + 1;

j++;

}

j = 0;

i++;

}

srand(time(NULL));

i = 0; j = 0;

while (i < max)

{

while (j < max)

{

b[i][j] = rand() % 10 + 1;

j++;

}

j = 0;

i++;

}

start1 = clock();

for (i = 0; i < max; i++)

{

for (j = 0; j < max; j++)

{

proizv = 0;

for (r = 0; r < max; r++)

{

proizv = proizv + a[i][r] \* b[r][j];

c[i][j] = proizv;

}

}

}

end1 = clock();

end = clock();

printf("Время выполнения работы: %.4f\n", ((double)end - start) / ((double)CLOCKS\_PER\_SEC));

printf("Время перемножения: %.4f\n", ((double)end1 - start1) / ((double)CLOCKS\_PER\_SEC));

return(0);

}

Результаты работы программы:

1.1. Вычислить порядок сложности программы.

O(n3)

1.2. Оценить время выполнения программы и кода, выполняющего перемножение матриц



Рисунок 1

1.3. Построить график зависимости времени выполнения программы от размера матриц и сравнить полученный результат с теоретической оценкой.

Проанализировав график, мы подтвердили предположения, что сложность программы равна O(n3)

**Задание 2:**

2.1. Оценить время работы каждого из реализованных алгоритмов на случайном наборе значений массива.

Листинг:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <windows.h>

void shell(int\* items, int count)

{

int i, j, gap, k;

int x, a[5];

a[0] = 9; a[1] = 5; a[2] = 3; a[3] = 2; a[4] = 1;

for (k = 0; k < 5; k++) {

gap = a[k];

for (i = gap; i < count; ++i) {

x = items[i];

for (j = i - gap; (x < items[j]) && (j >= 0); j = j - gap)

{

items[j + gap] = items[j];

}

items[j + gap] = x;

}

}

}

void qs(int\* items, int left, int right)

{

int i, j;

int x, y;

i = left; j = right;

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

do {

while ((items[i] < x) && (i < right)) i++;

while ((x < items[j]) && (j > left)) j--;

if (i <= j) {

y = items[i];

items[i] = items[j];

items[j] = y;

i++; j--;

}

} while (i <= j);

if (left < j) qs(items, left, j);

if (i < right) qs(items, i, right);

}

int main(void)

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setvbuf(stdin, NULL, \_IONBF, 0);

setvbuf(stdout, NULL, \_IONBF, 0);

clock\_t start, end, start1, end1;

int i = 0, j = 0, max, sort;

int\* a;

printf("Укажите размер массива: ");

scanf\_s("%d", &max);

a = (int\*)malloc(max \* sizeof(int));

while (i < max)

{

a[i] = rand() % 100 + 1;

i++;

}

printf("\nВыберите тип сортировки\nСортировка Шелла - 1\nБыстрая сортировка - 2\n");

scanf\_s("%d", &sort);

start = clock();

switch (sort)

{

case 1: shell(a, max);

case 2: qs(a, 0, max);

}

end = clock();

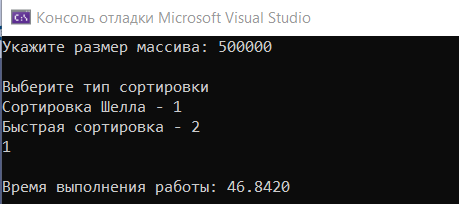
printf("\nВремя выполнения работы: %.4f\n", ((double)end - start) / ((double)CLOCKS\_PER\_SEC));

return(0);

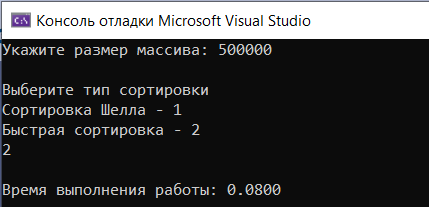
}

Результаты работы программы:

Сортировка Шелла



Быстрая сортировка



2.2. Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой возрастающую последовательность чисел.

Листинг:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <windows.h>

void shell(int\* items, int count)

{

int i, j, gap, k;

int x, a[5];

a[0] = 9; a[1] = 5; a[2] = 3; a[3] = 2; a[4] = 1;

for (k = 0; k < 5; k++) {

gap = a[k];

for (i = gap; i < count; ++i) {

x = items[i];

for (j = i - gap; (x < items[j]) && (j >= 0); j = j - gap)

{

items[j + gap] = items[j];

}

items[j + gap] = x;

}

}

}

void qs(int\* items, int left, int right)

{

int i, j;

int x, y;

i = left; j = right;

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

do {

while ((items[i] < x) && (i < right)) i++;

while ((x < items[j]) && (j > left)) j--;

if (i <= j) {

y = items[i];

items[i] = items[j];

items[j] = y;

i++; j--;

}

} while (i <= j);

if (left < j) qs(items, left, j);

if (i < right) qs(items, i, right);

}

int main(void)

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setvbuf(stdin, NULL, \_IONBF, 0);

setvbuf(stdout, NULL, \_IONBF, 0);

clock\_t start, end, start1, end1;

int i = 0, j = 0, max, sort;

int\* a;

printf("Укажите размер массива: ");

scanf\_s("%d", &max);

a = (int\*)malloc(max \* sizeof(int));

for (i = 0; i < max; i++)

{

a[i] = i;

printf("\nВыберите тип сортировки\nСортировка Шелла - 1\nБыстрая сортировка - 2\n");

scanf\_s("%d", &sort);

start = clock();

switch (sort)

{

case 1: shell(a, max);

case 2: qs(a, 0, max);

}

end = clock();

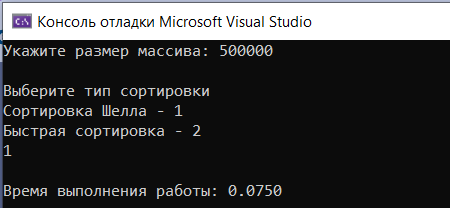
printf("\nВремя выполнения работы: %.4f\n", ((double)end - start) / ((double)CLOCKS\_PER\_SEC));

return(0);

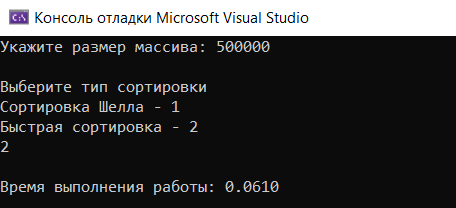
}

Результаты работы программы:

Сортировка Шелла



Быстрая сортировка



2.3. Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой убывающую последовательность чисел.

Листинг:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <windows.h>

void shell(int\* items, int count)

{

int i, j, gap, k;

int x, a[5];

a[0] = 9; a[1] = 5; a[2] = 3; a[3] = 2; a[4] = 1;

for (k = 0; k < 5; k++) {

gap = a[k];

for (i = gap; i < count; ++i) {

x = items[i];

for (j = i - gap; (x < items[j]) && (j >= 0); j = j - gap)

{

items[j + gap] = items[j];

}

items[j + gap] = x;

}

}

}

void qs(int\* items, int left, int right)

{

int i, j;

int x, y;

i = left; j = right;

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

do {

while ((items[i] < x) && (i < right)) i++;

while ((x < items[j]) && (j > left)) j--;

if (i <= j) {

y = items[i];

items[i] = items[j];

items[j] = y;

i++; j--;

}

} while (i <= j);

if (left < j) qs(items, left, j);

if (i < right) qs(items, i, right);

}

int main(void)

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setvbuf(stdin, NULL, \_IONBF, 0);

setvbuf(stdout, NULL, \_IONBF, 0);

clock\_t start, end, start1, end1;

int i = 0, j = 0, max, sort;

int\* a;

printf("Укажите размер массива: ");

scanf\_s("%d", &max);

a = (int\*)malloc(max \* sizeof(int));

for (i = max; i > 0; i--)

{

a[i] = i;

}

printf("\nВыберите тип сортировки\nСортировка Шелла - 1\nБыстрая сортировка - 2\n");

scanf\_s("%d", &sort);

start = clock();

switch (sort)

{

case 1: shell(a, max);

case 2: qs(a, 0, max);

}

end = clock();

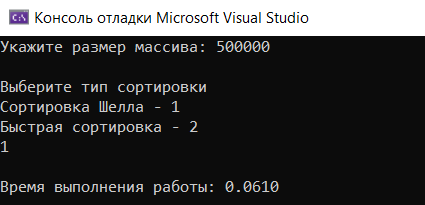
printf("\nВремя выполнения работы: %.4f\n", ((double)end - start) / ((double)CLOCKS\_PER\_SEC));

return(0);

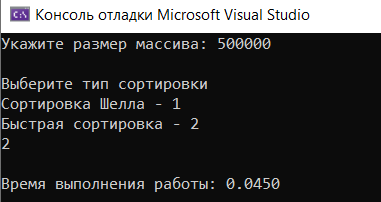
}

Результаты работы программы:

Сортировка Шелла



Быстрая сортировка



2.4. Оценить время работы каждого из реализованных алгоритмов на массиве, одна половина которого представляет собой возрастающую последовательность чисел, а вторая, – убывающую.

Листинг:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <windows.h>

void shell(int\* items, int count)

{

int i, j, gap, k;

int x, a[5];

a[0] = 9; a[1] = 5; a[2] = 3; a[3] = 2; a[4] = 1;

for (k = 0; k < 5; k++) {

gap = a[k];

for (i = gap; i < count; ++i) {

x = items[i];

for (j = i - gap; (x < items[j]) && (j >= 0); j = j - gap)

{

items[j + gap] = items[j];

}

items[j + gap] = x;

}

}

}

void qs(int\* items, int left, int right)

{

int i, j;

int x, y;

i = left; j = right;

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

do {

while ((items[i] < x) && (i < right)) i++;

while ((x < items[j]) && (j > left)) j--;

if (i <= j) {

y = items[i];

items[i] = items[j];

items[j] = y;

i++; j--;

}

} while (i <= j);

if (left < j) qs(items, left, j);

if (i < right) qs(items, i, right);

}

int main(void)

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setvbuf(stdin, NULL, \_IONBF, 0);

setvbuf(stdout, NULL, \_IONBF, 0);

clock\_t start, end, start1, end1;

int i = 0, j = 0, max, sort;

int\* a;

printf("Укажите размер массива: ");

scanf\_s("%d", &max);

a = (int\*)malloc(max \* sizeof(int));

for (i = 0; i < max/2; i++)

{

a[i] = i;

}

for (i = max/2; i < max; i++)

{

a[i] = i;

}

printf("\nВыберите тип сортировки\nСортировка Шелла - 1\nБыстрая сортировка - 2\n");

scanf\_s("%d", &sort);

start = clock();

switch (sort)

{

case 1: shell(a, max);

case 2: qs(a, 0, max);

}

end = clock();

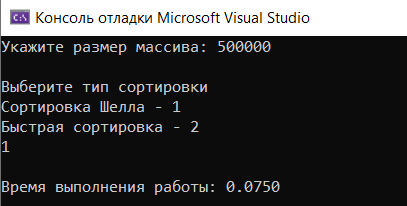
printf("\nВремя выполнения работы: %.4f\n", ((double)end - start) / ((double)CLOCKS\_PER\_SEC));

return(0);

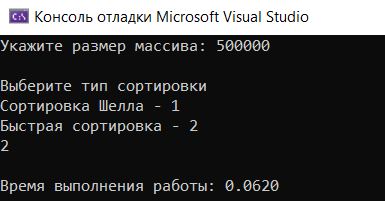
}

Результаты работы программы:

Сортировка Шелла



Быстрая сортировка



2.5. Оценить время работы стандартной функции qsort, реализующей алгоритм быстрой сортировки на выше указанных наборах данных.

Листинг:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <windows.h>

void shell(int\* items, int count)

{

int i, j, gap, k;

int x, a[5];

a[0] = 9; a[1] = 5; a[2] = 3; a[3] = 2; a[4] = 1;

for (k = 0; k < 5; k++) {

gap = a[k];

for (i = gap; i < count; ++i) {

x = items[i];

for (j = i - gap; (x < items[j]) && (j >= 0); j = j - gap)

{

items[j + gap] = items[j];

}

items[j + gap] = x;

}

}

}

void qs(int\* items, int left, int right)

{

int i, j;

int x, y;

i = left; j = right;

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

do {

while ((items[i] < x) && (i < right)) i++;

while ((x < items[j]) && (j > left)) j--;

if (i <= j) {

y = items[i];

items[i] = items[j];

items[j] = y;

i++; j--;

}

} while (i <= j);

if (left < j) qs(items, left, j);

if (i < right) qs(items, i, right);

}

int comp(const int\* i, const int\* j)

{

return \*i - \*j;

}

int main(void)

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setvbuf(stdin, NULL, \_IONBF, 0);

setvbuf(stdout, NULL, \_IONBF, 0);

clock\_t start, end, start1, end1;

int i = 0, j = 0, max, sort, type;

int\* a;

printf("Укажите размер массива: ");

scanf\_s("%d", &max);

a = (int\*)malloc(max \* sizeof(int));

printf("\nКак заполнить массив?\nСлучайная генерация - 1\nПо возрастанию - 2\nПо убыванию - 3\nПоловина на половину - 4\n");

scanf\_s("%d", &type);

switch (type)

{

case 1: while (i < max)

{

a[i] = rand() % 100 + 1;

i++;

}

case 2: for (i = 0; i < max; i++)

{

a[i] = i;

}

case 3: for (i = max; i > 0; i--)

{

a[i] = i;

}

case 4: for (i = 0; i < max / 2; i++)

{

a[i] = i;

}

for (i = max / 2; i < max; i++)

{

a[i] = i;

}

}

printf("\nВыберите тип сортировки\nСортировка Шелла - 1\nБыстрая сортировка - 2\nqsort - 3\n");

scanf\_s("%d", &sort);

start = clock();

switch (sort)

{

case 1: shell(a, max);

case 2: qs(a, 0, max);

case 3: qsort(a, max, sizeof(int), (int(\*) (const void \*, const void \*)) comp);

}

end = clock();

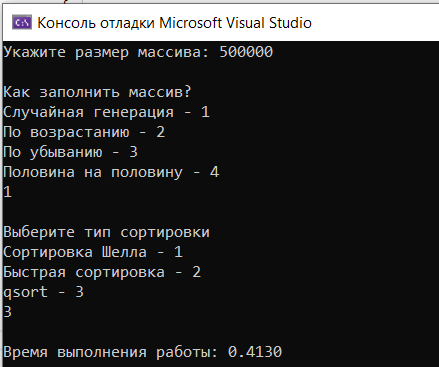
printf("\nВремя выполнения работы: %.4f\n", ((double)end - start) / ((double)CLOCKS\_PER\_SEC));

return(0);

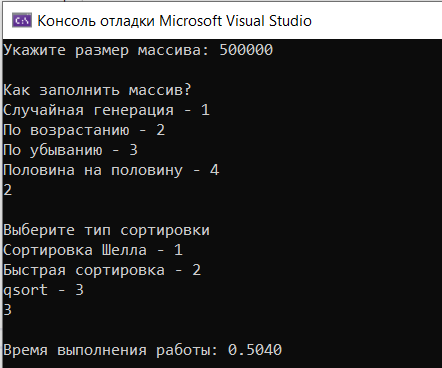
}

Результаты работы программы:

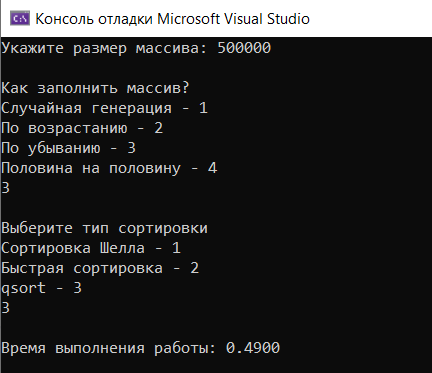
Случайная генерация



По возрастанию



По убыванию



Половина на половину

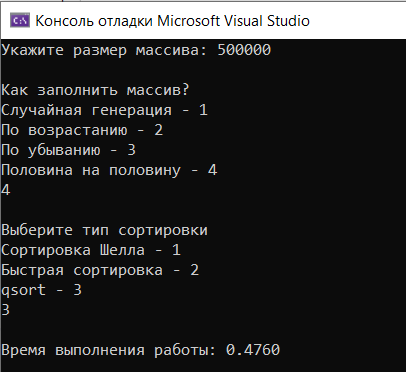




Рисунок 2

Вывод: Проанализировав три вида сортировки массивов (сортировка Шелла, быстрая сортировка и функция qsort) можно подвести итоги. Каждый из них более или менее продуктивен для разных видов заполнения массивов. По результатам тестирования, самой продуктивной сортировкой оказалась быстрая.