Министерство высшего образования и науки Российской Федерации

Пензенский Государственный Университет

Кафедра «Вычислительная техника»

ОТЧЕТ

Лабораторная работа №2

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Оценка времени выполнения программ»

Выполнил студент группы 20ВВ2:

Кодиров И.Н.

Пантелеев И. А.

Приняли:

Митрохин М. А.

Юрова О. В.

Пенза 2021

**Листинг первой программы:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

void timer()

{

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

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

clock\_t start, end; // объявляем переменные для определения времени выполнения

int i = 0, j = 0, r;

int\*\* a, \*\* b, \*\* c, elem\_c, size = 1000;

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

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

{

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

srand(time(NULL)); // инициализируем параметры генератора случайных чисел

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

{

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

}

}

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

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

{

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

srand(time(NULL)); // инициализируем параметры генератора случайных чисел

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

{

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

}

}

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

start = clock();

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

{

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

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

{

elem\_c = 0;

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

{

elem\_c = elem\_c + (a[i][r] \* b[r][j]);

c[i][j] = elem\_c;

}

}

}

end = clock();

printf("%f", difftime(end, start));

free(\*a);

free(\*b);

free(\*c);

}

Задания:

**1-ое:**

1)Вычислить порядок сложности программы (*О*-символику).   
Порядок сложности O(n3).

2) Время выполнения программы при:

100 эл. = 4.000000  
200 эл. = 27.000000  
400 эл. = 215.000000  
1000 эл. = 4016.000000  
2000 эл. = 41499.000000  
4000 эл. = 365302.000000  
10000 эл. = много

3) График:

Вывод: Полученные данные подтвердили О-символику.

**2-ое:**

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тип сорт. | Кол-во элементов и время выполнения каждого типа | | | | |
|  | 100 | 1000 | 10000 | 100000 | 1000000 |
| Шелла | 0,000000 | 0,000000 | 1,000000 | 15,000000 | 165,000000 |
| Быстрая | 0,000000 | 0,000000 | 1,000000 | 7,000000 | 78,000000 |
| Qsort | 0,000000 | 0,000000 | 1,000000 | 9,000000 | 84,000000 |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тип сорт. | Кол-во элементов и время выполнения каждого типа | | | | |
|  | 100 | 1000 | 10000 | 100000 | 1000000 |
| Шелла | 0,000000 | 0,000000 | 0,000000 | 5,000000 | 54,000000 |
| Быстрая | 0,000000 | 0,000000 | 0,000000 | 3,000000 | 35,000000 |
| Qsort | 0,000000 | 0,000000 | 1,000000 | 10,000000 | 121,000000 |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тип сорт. | Кол-во элементов и время выполнения каждого типа | | | | |
|  | 100 | 1000 | 10000 | 100000 | 1000000 |
| Шелла | 0,000000 | 0,000000 | 1,000000 | 7,000000 | 78,000000 |
| Быстрая | 0,000000 | 0,000000 | 0,000000 | 3,000000 | 30,000000 |
| Qsort | 0,000000 | 0,000000 | 1,000000 | 11,000000 | 126,000000 |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тип сорт. | Кол-во элементов и время выполнения каждого типа | | | | |
|  | 100 | 1000 | 10000 | 100000 | 1000000 |
| Шелла | 0,000000 | 0,000000 | 1,000000 | 6,000000 | 65,000000 |
| Быстрая | 0,000000 | 0,000000 | 1,000000 | 3,000000 | 35,000000 |
| Qsort | 0,000000 | 0,000000 | 1,000000 | 10,000000 | 126,000000 |

**Листинг второй программы:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

void ShellSort(int n, int\* mass)

{

int i, j, step;

int tmp;

for (step = n / 2; step > 0; step /= 2)

for (i = step; i < n; i++)

{

tmp = mass[i];

for (j = i; j >= step; j -= step)

{

if (tmp < mass[j - step])

mass[j] = mass[j - step];

else

break;

}

mass[j] = tmp;

}

}

void qs(int\* items, int left, int right) //вызов функции: qs(items, 0, count-1);

{

int i, j;

int x, y;

i = left; j = right;

/\* выбор компаранда \*/

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);

}

void rand(int\* items, int size)

{

int i;

srand(time(NULL));

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

{

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

}

}

void half(int\* items, int size)

{

int i, num;

num = size;

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

{

if (i < size / 2)

{

items[i] = i;

}

else {

items[i] = num;

num--;

}

}

}

void incr(int\* items, int size)

{

int i;

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

{

items[i] = i;

}

}

void decr(int\* items, int size)

{

int i;

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

{

items[i] = size - i;

}

}

int compare(const void\* x1, const void\* x2) // функция сравнения элементов массива

{

return (\*(int\*)x1 - \*(int\*)x2); // если результат вычитания равен 0, то числа равны, < 0: x1 < x2; > 0: x1 > x2

}

int main()

{

setlocale(0,"");

int x, \*fm, \*sm, i, size;

clock\_t start, end;

printf("Размер массива = ");

scanf("%d", &size);

fm = (int\*)malloc(size \* sizeof(int));

sm = (int\*)malloc(size \* sizeof(int));

printf("Тип массива:\n1 - убывание\n2 - возрастание\n3 - рандом\n4 - половины\n");

scanf("%d", &x);

switch (x)

{

case 1:

decr(fm, size);

decr(sm, size);

start = clock();

qs(fm, 0, size);

end = clock();

printf("QuickSort: %f\n", difftime(end, start));

decr(fm, size);

break;

case 2:

incr(fm, size);

incr(sm, size);

start = clock();

qs(fm, 0, size);

end = clock();

printf("QuickSort: %f\n", difftime(end, start));

incr(fm, size);

break;

case 3:

rand(fm, size);

rand(sm, size);

start = clock();

qs(fm, 0, size);

end = clock();

printf("QuickSort: %f\n", difftime(end, start));

rand(fm, size);

break;

case 4:

half(fm, size);

half(sm, size);

start = clock();

qs(fm, 0, size);

end = clock();

printf("QuickSort: %f\n", difftime(end, start));

half(fm, size);

break;

}

start = clock();

ShellSort(size, fm);

end = clock();

printf("Shellsort: %f\n", difftime(end, start));

start = clock();

qsort(sm, size, sizeof(int), compare);

end = clock();

printf("QSLib: %f", difftime(end, start));

free(fm);

free(sm);

return 0;

}

Вывод: На каждом отдельном количестве элементов каждый алгоритм показывает себя по – разному чем больше элементов, тем выше разница во времени работы каждого из них. Среди этих трех алгоритмов сортировки сильнее всех выделилась быстрая сортировка, выполняя одно и тоже задание, значительно быстрее других видов сортировки (Шелла и QSlib).