Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

По лабораторной работе №2

по курсу «Логика и основы алгоритмизации в ИЗ»

на тему «Оценка времени выполнения программ»

**Выполнил студент группы 21вв1:**

Вартанов Алексей

**Приняли**

Юрова О.В.

Акифьев И.В.

Пенза 2022

Задание 1:

1. Вычислить порядок сложности программы (*О*-символику).
2. Оценить время выполнения программы и кода, выполняющего перемножение матриц, используя функции библиотеки time.h для матриц размерами от 100, 200, 400, 1000, 2000, 4000, 10000.
3. Построить график зависимости времени выполнения программы от размера матриц и сравнить полученный результат с теоретической оценкой.

Сложность алгоритма -

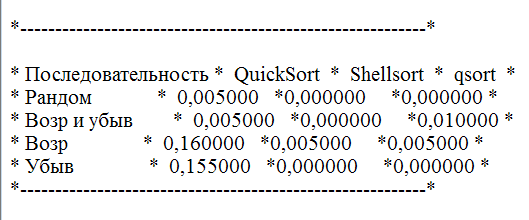
|  |  |  |
| --- | --- | --- |
| № теста | Размер массива | Время выполнения алгоритма в секундах |
| 1 | 100 | 0,00300 |
| 2 | 200 | 0,001600 |
| 3 | 400 | 0,22300 |
| 4 | 1000 | 6,537 |
| 5 | 2000 | 67,611 |
| 6 | 4000 | 1427,623 |
| 7 | 10000 | ---- |

**Задание 2:**

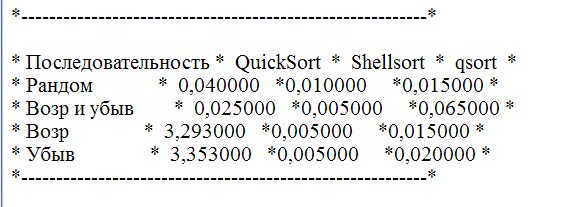
1. Оценить время работы каждого из реализованных алгоритмов на случайном наборе значений массива.
2. Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой возрастающую последовательность чисел.
3. Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой убывающую последовательность чисел.
4. Оценить время работы каждого из реализованных алгоритмов на массиве, одна половина которого представляет собой возрастающую последовательность чисел, а вторая, – убывающую.
5. Оценить время работы стандартной функции qsort, реализующей алгоритм быстрой сортировки на выше указанных наборах данных.

Программа выводит всю информацию с консоли в текстовый файл.

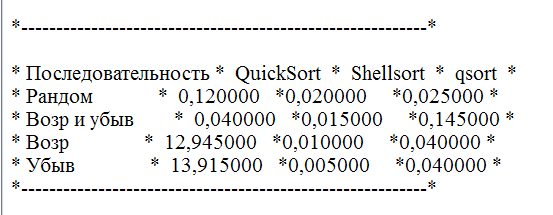
При наборе из 10000 чисел:



При наборе из 50000 чисел:

****

При наборе из 100000 чисел:

****

Листинг программы:

Задание 1:

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<locale.h>

#define N 10000

int main(void)

{

setlocale(LC\_ALL, "Rus");

inti, j, k, elem\_c;

int \*\*A = (int\*\*)malloc(N \* sizeof(int\*));

int \*\*B = (int\*\*)malloc(N \* sizeof(int\*));

int \*\*C = (int\*\*)malloc(N \* sizeof(int\*));

for (i = 0; i< N; i++){

A[i] = (int\*)malloc(N \* sizeof(int));

B[i] = (int\*)malloc(N \* sizeof(int));

C[i] = (int\*)malloc(N \* sizeof(int));

}

srand(time(NULL));

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

for (j = 0; j < N; j++){

A[i][j] = rand() % 100;

B[i][j] = rand() % 100;

}

doubletime\_spent = 0.0; // хранение времени выполнения кода

clock\_t begin = clock();

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

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

{

C[i][j] = 0;

for(k = 0; k < N; k++)

C[i][j] += A[i][k] \* B[k][j];

}

clock\_t end = clock();

time\_spent += (double)(end - begin) / CLOCKS\_PER\_SEC;// рассчитатьпрошедшеевремя, найдяразницу (end - begin) иделениеразницына CLOCKS\_PER\_SEC дляпереводавсекунды

printf("Затраченно время в секундах %f ", time\_spent);

for (i = 0; i< N; i++){

free(A[i]);

free(B[i]);

free(C[i]);

}

free(A);

free(B);

free(C);

getchar();

}

Задание 2:

#define \_CRT\_SECURE\_NO\_WARNINGS // для корректной работы scanf()

#include <stdio.h>

#include <conio.h>

#include <locale.h>

#include <stdlib.h>

#include <windows.h>

#include <time.h>

#pragma comment(linker, "/STACK:1468006400")

void qs(int С[], int left, int right)

{

int pivot; // разрешающий элемент

int l\_hold = left; //левая граница

int r\_hold = right; // правая граница

pivot = С[left];

while (left < right) // пока границы не сомкнутся

{

while ((С[right] >= pivot) && (left < right))

right--; // сдвигаем правую границу пока элемент [right] больше [pivot]

if (left != right) // если границы не сомкнулись

{

С[left] = С[right]; // перемещаем элемент [right] на место разрешающего

left++; // сдвигаем левую границу вправо

}

while ((С[left] <= pivot) && (left < right))

left++; // сдвигаем левую границу пока элемент [left] меньше [pivot]

if (left != right) // если границы не сомкнулись

{

С[right] = С[left]; // перемещаем элемент [left] на место [right]

right--; // сдвигаем правую границу влево

}

}

С[left] = pivot; // ставим разрешающий элемент на место

pivot = left;

left = l\_hold;

right = r\_hold;

if (left < pivot) // Рекурсивно вызываем сортировку для левой и правой части массива

qs(С, left, pivot - 1);

if (right > pivot)

qs(С, pivot + 1, right);

}

// Функция сортировки Шелла

void ShellSort(int n, int С[])

{

int i, j, step;

int tmp;

for (step = n / 2; step > 0; step /= 2) //пока шаг больше нуля будет выполняться код ниже

for (i = step; i < n; i++) //текущее значение шага записывается в i, пока i< значений массива будет выполняться код ниже

{

tmp = С[i]; //в tmp записывается первое значение массива

//перебираем элементы, отстоящие от j

for (j = i; j >= step; j=j-step) // текущее значение i записывается в j, пока j больше или равно step выполняется код ниже

{

if (tmp < С[j - step]) //Если первый элемент массива меньше, чем отстоящий, то выполняется код ниже

С[j] = С[j - step];//переходим к следующему отстающему элементу

else

break;

}

С[j] = tmp;//сюда помещаем сохраненный элемент

}

}

int compare (const void \*a, const void \*b)

{

return (\*(int \*)a - \*(int \*)b);

}

int main()

{

srand(time(NULL)); //функция для генерации рандомных чисел

setlocale(LC\_ALL, "Rus");//русская локализация

FILE \*F = fopen ( "prikol.txt", "wt" );

int numb, i, j, n, b, k, d, numb1, numb2;

//ввод количества значений массива

printf("Введите numb: ");

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

printf("\n");

numb1 = numb/2;

numb2 = numb/2;

//создание массива

int\* A;

A = (int \*)malloc(numb1 \* sizeof(int));//выделение памяти под массив

int\* B;

B = (int \*)malloc(numb2 \* sizeof(int));//выделение памяти под массив

int\* C;

C = (int \*)malloc(numb \* sizeof(int));//выделение памяти под массив

int\* D;

D = (int \*)malloc(numb \* sizeof(int));//выделение памяти под массив

int\* G;

G = (int \*)malloc(numb \* sizeof(int));//выделение памяти под массив

int\* H;

H = (int \*)malloc(numb \* sizeof(int));//выделение памяти под массив

//заполнение массива возрастающими и убывающими значениями

for (int i = 0; i <= numb1; i++) {

for(i = numb1; i != 0; i--) {

A[i] = i - 1;

//fprintf(F, "%d\t", A[i]);

}

for(i = 0; i != numb2; i++) {

B[i] = i + 1;

//fprintf(F, "%d\t", B[i]);

}

}

k = -1; i = 0; j = 0;

for(int i = 0; i <= numb1; i++) {

C[k++] = A[i];

}

for(int i = 0; i <= numb2 ; i++) {

C[k++] = B[i];

}

fprintf(F, "\n\n");

for(i = numb; i != -1; i--) {

G[i] = i - 1;

}

fprintf(F, "\n\n");

for(i = 0; i != numb; i++) {

H[i] = i + 1;

}

fprintf(F, "\n\n");

srand(time(NULL));

for (i = 0; i < numb; i++){

D[i] = rand() % 100;

}

double time\_potr = 0.0; // для хранения времени выполнения кода

clock\_t nach = clock();

qs(C, 0, numb-1);

clock\_t konec = clock();

time\_potr += (double)(konec - nach) / CLOCKS\_PER\_SEC;

//printf("Затраченно время в секундах на quick sort %f ", time\_potr);

printf("\n");

//сортировка методом Шелла для возрастающего и убывающего

double time\_p = 0.0; // для хранения времени выполнения кода

clock\_t na = clock();

qs(D, 0, numb-1);

clock\_t ko = clock();

//for (int i = 0; i < numb; i++){

// fprintf(F, "%d\t ", D[i]);

//}

time\_p += (double)(ko- na) / CLOCKS\_PER\_SEC;

double time\_pot = 0.0; // для хранения времени выполнения кода

clock\_t nac = clock();

qs(H, 0, numb-1);

clock\_t kon = clock();

//for (int i = 0; i < numb; i++){

// fprintf(F, "%d\t ", H[i]);

//}

time\_pot += (double)(kon- nac) / CLOCKS\_PER\_SEC;

double time\_pota = 0.0; // для хранения времени выполнения кода

clock\_t naca = clock();

qs(G, 0, numb-1);

clock\_t konc = clock();

time\_pota += (double)(konc- naca) / CLOCKS\_PER\_SEC;

for (int i = 0; i <= numb1; i++) {

for(i = numb1; i != 0; i--) {

A[i] = i - 1;

}

for(i = 0; i != numb2; i++) {

B[i] = i + 1;

}

}

k = -1; i = 0; j = 0;

for(int i = 0; i <= numb1; i++) {

C[k++] = A[i];

}

for(int i = 0; i <= numb2 ; i++) {

C[k++] = B[i];

}

fprintf(F, "\n\n");

for(i = numb; i != -1; i--) {

G[i] = i - 1;

}

fprintf(F, "\n\n");

for(i = 0; i != numb; i++) {

H[i] = i + 1;

//fprintf(F, "%d\t", H[i]);

}

fprintf(F, "\n\n");

//fprintf(F, "Первородный с рандомными значениями массив:\n");//запись фразы в файл

srand(time(NULL));

for (i = 0; i < numb; i++){

D[i] = rand() % 100;

}

double time\_spent = 0.0; // для хранения времени выполнения кода

clock\_t begin = clock();

ShellSort(numb-1, C);//вызов функции метода шелла

clock\_t end = clock();

time\_spent += (double)(end - begin) / CLOCKS\_PER\_SEC;// рассчитать прошедшее время, найдя разницу (end - begin) и деление разницы на CLOCKS\_PER\_SEC для перевода в секунды

double time\_s = 0.0; // для хранения времени выполнения кода

clock\_t be = clock();

ShellSort(numb-1, D);//вызов функции метода шелла

clock\_t en = clock();

time\_s += (double)(en - be) / CLOCKS\_PER\_SEC;// рассчитать прошедшее время, найдя разницу (end - begin) и деление разницы на CLOCKS\_PER\_SEC для перевода в секунды

double time\_st = 0.0; // для хранения времени выполнения кода

clock\_t beg = clock();

ShellSort(numb-1, H);//вызов функции метода шелла

clock\_t eni = clock();

time\_st += (double)(eni - beg) / CLOCKS\_PER\_SEC;// рассчитать прошедшее время, найдя разницу (end - begin) и деление разницы на CLOCKS\_PER\_SEC для перевода в секунды

double time\_sta = 0.0; // для хранения времени выполнения кода

clock\_t begi = clock();

ShellSort(numb-1, G);//вызов функции метода шелла

clock\_t enid = clock();

time\_sta += (double)(enid - begi) / CLOCKS\_PER\_SEC;// рассчитать прошедшее время, найдя разницу (end - begin) и деление разницы на CLOCKS\_PER\_SEC для перевода в секунды

//вывод отсортированного массива с возрастающей и убывающей последовательностями на экран

for (int i = 0; i <= numb1; i++) {

for(i = numb1; i != 0; i--) {

A[i] = i - 1;

//

}

for(i = 0; i != numb2; i++) {

B[i] = i + 1;

}

}

k = -1; i = 0; j = 0;

for(int i = 0; i <= numb1; i++) {

C[k++] = A[i];

}

for(int i = 0; i <= numb2 ; i++) {

C[k++] = B[i];

}

fprintf(F, "\n\n");

for(i = numb; i != -1; i--) {

G[i] = i - 1;

}

fprintf(F, "\n\n");

for(i = 0; i != numb; i++) {

H[i] = i + 1;

}

fprintf(F, "\n\n");

srand(time(NULL));

for (i = 0; i < numb; i++){

D[i] = rand() % 100;

}

double time\_potan = 0.0; // для хранения времени выполнения кода

clock\_t nacan = clock();

qsort(C, numb, sizeof(int), compare);

clock\_t koncn = clock();

time\_potan += (double)(koncn- nacan) / CLOCKS\_PER\_SEC;

double time\_potani = 0.0; // для хранения времени выполнения кода

clock\_t nacani = clock();

qsort(D, numb, sizeof(int),compare);

clock\_t koncni = clock();

time\_potani += (double)(koncni- nacani) / CLOCKS\_PER\_SEC;

double time\_potanic = 0.0; // для хранения времени выполнения кода

clock\_t nacanic = clock();

qsort(G, numb, sizeof(int),compare);

clock\_t koncnic = clock();

time\_potanic += (double)(koncnic- nacanic) / CLOCKS\_PER\_SEC;

double time\_potanica = 0.0; // для хранения времени выполнения кода

clock\_t nacanica = clock();

qsort(H, numb, sizeof(int),compare);

clock\_t koncnica = clock();

time\_potanica += (double)(koncnica- nacanica) / CLOCKS\_PER\_SEC;

printf("Работу программы можно посмотреть в текстовом файле 'prikol.txt'");

fprintf(F,"\n\n\*----------------------------------------------------------\*");

fprintf(F,"\n\n");

fprintf(F,"\* Последовательность \* QuickSort \* Shellsort \* qsort \*");

fprintf(F,"\n");

fprintf(F,"\* Рандом \* %f",time\_p);

fprintf(F," \*%f", time\_s);

fprintf(F," \*%f", time\_potani);

fprintf(F," \*");

fprintf(F,"\n");

fprintf(F,"\* Возр и убыв \* %f",time\_potr);

fprintf(F," \*%f", time\_spent);

fprintf(F," \*%f", time\_potan);

fprintf(F," \*");

fprintf(F,"\n");

fprintf(F,"\* Возр \* %f",time\_pot);

fprintf(F," \*%f", time\_st);

fprintf(F," \*%f", time\_potanic);

fprintf(F," \*");

fprintf(F,"\n");

fprintf(F,"\* Убыв \* %f",time\_pota);

fprintf(F," \*%f", time\_sta);

fprintf(F," \*%f", time\_potanica);

fprintf(F," \*");

fprintf(F,"\n");

fprintf(F,"\*----------------------------------------------------------\*");

getchar();

getchar();

}

Вывод:

Сортировка ShellSort показало себя намного лучше, чем все остальные. Quicksort работает намного медленнее и занимает много времени на полную сортировку большого количества чисел.

Выполнив данную лабораторную, работу я закрепил свои знания о том, как правильно измерять время выполнения определенных алгоритмов.