МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»

(БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Дисциплина: Алгоритмы и структуры данных

Лабораторная работа №3

Встроенные структуры данных (Pascal/C)

|  |  |
| --- | --- |
|  | Выполнила: ст. группы ПВ-21  Зановская А.И.  Проверил: Синюк В.Г. |

Белгород

2017

Л а б о р а т о р н а я р а б о т а № 3

**Сравнительный анализ методов сортировки (Pascal/C)**

**Цель работы:** изучение методов сортировки массивов иприобретение навыков в проведении сравнительного анализа различных методов сортировки.

# З а д а н и е

# 1. Изучить временные характеристики алгоритмов.

2. Изучить методы сортировки:

1) включением;

2) выбором;

3) обменом:

3.1) улучшенная обменом 1;

3.2) улучшенная обменом 2;

4) Шелла;

5) Хоара;

6) пирамидальная.

3. Программно реализовать методы сортировки массивов.

/\*============СОРТИРОВКА ОБМЕНОМ=============\*/

/\* функция сортировки обменом \*/

void BblSort(int A[],int nn)

{

int i,j,k;

for ( i=0; i<nn-1; i++ )

{

for (j=nn-1; j>i; j--)

if (A[j] <A[j-1])

{

k = A[j];

A[j] = A[j-1];

A[j-1] = k;

}

}

}

void BblSort1(int A[],int nn)

{

int i,j,k,p;

for ( i=0; i<nn-1; i++ )

{

p = 0;

for (j=nn-1; j>i; j--)

if (A[j] <A[j-1])

{

k = A[j];

A[j] = A[j-1];

A[j-1] = k;

p = 1;

}

/\* Если перестановок не было, то сортировка выполнена \*/

if (p == 0)

break;

}

}

void BblSort2(int A[],int nn)

{

int i,j,k,p,q;

for ( i=0; i<nn-1; i=q )

{

p = 0;

for (j=nn-1; j>i; j--)

if (A[j] <A[j-1])

{

k = A[j];

A[j] = A[j-1];

A[j-1] = k;

p = 1;

q=j-1;

}

/\* Если перестановок не было, то сортировка выполнена \*/

if (p == 0)

break;

}

}

/\*============СОРТИРОВКА МЕТОДОМ ШЕЛЛА=============\*/

/\* функция сортировки методом Шелла \*/

void ShellSort(int a [], int n)

{

int i,j,k,hh,t,s;

int h [1000];

t = round(log(n)/log(3))-1;

if (t < 1)

t = 1;

h[t] = 1;

for (k=t; k >= 1; k--)

h[k-1] = 3\*h[k]+1;

for (s=t-1;s>=0;s--)

{

hh = h[s];

for (j = hh;j<=n;j++)

{

i = j-hh;

k = a[j];

while ((k <= a[i])&&(i >= 0))

{

a[i+hh] = a[i];

i = i-hh;

}

a[i+hh] = k;

}

}

}

/\*============СОРТИРОВКА МЕТОДОМ ХОАРА=============\*/

void QSort(int a [], int L, int R)

{

int x = a[L], i = L, j = R, t; // в качестве разделителя выбираем первый элемент

while (i<=j)

{

while (a[i]<x)

i++;

while (a[j]>x)

j--;

if (i<=j)

{

t = a[i];

a[i] = a[j];

a[j] = t;

i++;

j--;

}

}

if (L<j)

QSort(a,L,j);

if (i<R)

QSort(a,i,R);

}

/\*функция сортировки методом Хоара\*/

void HoarSort(int a[], int n)

{

QSort(a,1,n);

}

/\*============ПИРАМИДАЛЬНАЯ СОРТИРОВКА=============\*/

/\* пирамидальная функция сортировки \*/

void HeapSort(int A[],int nn)

{

int L,R,x,i;

L = nn/2 ; R = nn-1;

/\* Построение пирамиды из исходного массива \*/

while ( L>0 )

{

L = L - 1;

Sift(A,L,R);

}

/\* Сортировка: пирамида => отсортированный массив \*/

while ( R>0 )

{ x = A[0]; A[0] = A[R]; A[R] = x;

R--;

Sift(A,L,R);

}

}

/\* ============================================ \*/

void Sift(int A[],int L,int R)

{

int i,j,x,k;

i = L;

j = 2\*L+1;

x = A[L];

if ((j<R) && (A[j]<A[j+1]))

j++;

while ((j<=R) && (x<A[j]))

{

k=A[i]; A[i] = A[j]; A[j]=k;

i = j;

j = 2\*j+1;

if ((j<R) && (A[j]<A[j+1]))

j++;

}

}

4. Разработать и программно реализовать средство для проведения экспериментов по определению временных характеристик алгоритмов сортировки.

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <math.h>

#include <time.h>

#define N 4500

void Sift(int A[],int L,int R);

int main()

{

double start, end, s;

setlocale(LC\_ALL,"Rus");

int a[N];

form(a,1);

start=clock();

Sis(a,N);

end=clock();

s=((double) end - start) / ((double) CLOCKS\_PER\_SEC);

printf ("timeSis = %f\n",s);

form(a,3);

start=clock();

StrSel(a,N);

end=clock();

s=((double) end - start) / ((double) CLOCKS\_PER\_SEC);

printf ("timeStrSel = %f\n",s);

form(a,3);

start=clock();

BblSort(a,N);

end=clock();

s=((double) end - start) / ((double) CLOCKS\_PER\_SEC);

printf ("timeBBl = %f\n",s);

form(a,3);

start=clock();

BblSort1(a,N);

end=clock();

s=((double) end - start) / ((double) CLOCKS\_PER\_SEC);

printf ("timeBBl1 = %f\n",s);

form(a,3);

start=clock();

BblSort2(a,N);

end=clock();

s=((double) end - start) / ((double) CLOCKS\_PER\_SEC);

printf ("timeBBl2 = %f\n",s);

form(a,3);

start=clock();

ShellSort(a,N);

end=clock();

s=((double) end - start) / ((double) CLOCKS\_PER\_SEC);

printf ("timeShell = %f\n",s);

form(a,3);

start=clock();

HoarSort(a,N);

end=clock();

s=((double) end - start) / ((double) CLOCKS\_PER\_SEC);

printf ("timeHoar = %f\n",s);

form(a,3);

start=clock();

HeapSort(a,N);

end=clock();

s=((double) end - start) / ((double) CLOCKS\_PER\_SEC);

printf ("timeHeap = %f\n",s);

}

void form (int a[], int k)

{

int i;

switch (k)

{

case 1:

srand(N);

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

a[i]=rand();

break;

case 2:

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

a[i]=i;

case 3:

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

a[i]=N-i;

}

}

5. Провести эксперименты по определению временных характеристик алгоритмов сортировки. Результаты экспериментов представить в виде таблицы 9, клетки которой содержат время выполнения алгоритма сортировки массива с заданным количеством элементов. Провести эксперимент для упорядоченных, неупорядоченных и упорядоченных в обратном порядке массивов (для каждого типа массива заполнить отдельную таблицу).

**Результаты экспериментов**

1. Неупорядоченный массив

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Сортировка | Количество элементов в массиве | | | | | | | | |
| 500 | 1000 | 1500 | 2000 | 2500 | 3000 | 3500 | 4000 | 4500 |
| Включением | 0,002 | 0,002 | 0,004 | 0,008 | 0,014 | 0,015 | 0,02 | 0,031 | 0,033 |
| Выбором | 0,001 | 0,001 | 0,006 | 0,013 | 0,028 | 0,017 | 0,025 | 0,042 | 0,044 |
| Обменом | 0 | 0,007 | 0,013 | 0,048 | 0,071 | 0,056 | 0,076 | 0,105 | 0,119 |
| Обменом 1 | 0,003 | 0,005 | 0,014 | 0,036 | 0,047 | 0,049 | 0,081 | 0,1 | 0,112 |
| Обменом 2 | 0,001 | 0,006 | 0,015 | 0,028 | 0,038 | 0,055 | 0,076 | 0,1 | 0,112 |
| Шелла | 0 | 0,002 | 0,001 | 0,003 | 0,003 | 0,004 | 0,007 | 0,01 | 0,01 |
| Хоара | 0 | 0 | 0 | 0 | 0,001 | 0 | 0,001 | 0 | 0,01 |
| Пирамидаль-ная | 0 | 0 | 0,001 | 0,001 | 0,001 | 0,002 | 0,001 | 0,001 | 0,002 |

2. Упорядоченный по возрастанию массив

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Сортировка | Количество элементов в массиве | | | | | | | | |
| 500 | 1000 | 1500 | 2000 | 2500 | 3000 | 3500 | 4000 | 4500 |
| Включением | 0 | 0,001 | 0,003 | 0,008 | 0,01 | 0,014 | 0,018 | 0,027 | 0,068 |
| Выбором | 0 | 0,002 | 0,006 | 0,013 | 0,02 | 0,024 | 0,037 | 0,059 | 0,129 |
| Обменом | 0 | 0,006 | 0,012 | 0,023 | 0,038 | 0,055 | 0,08 | 0,114 | 0,246 |
| Обменом 1 | 0,002 | 0,006 | 0,013 | 0,022 | 0,042 | 0,053 | 0,092 | 0,102 | 0,266 |
| Обменом 2 | 0,001 | 0,007 | 0,014 | 0,025 | 0,040 | 0,055 | 0,1 | 0,096 | 0,167 |
| Шелла | 0 | 0,001 | 0,003 | 0,004 | 0,007 | 0,009 | 0,012 | 0,014 | 0,017 |
| Хоара | 0,002 | 0,002 | 0,003 | 0,007 | 0,01 | 0,013 | 0,02 | 0,026 | 0,026 |
| Пирамидаль-ная | 0 | 0 | 0 | 0 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 |

3. Упорядоченный по убыванию массив

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Сортировка | Количество элементов в массиве | | | | | | | | |
| 500 | 1000 | 1500 | 2000 | 2500 | 3000 | 3500 | 4000 | 4500 |
| Включением | 0,001 | 0 | 0,003 | 0,005 | 0,01 | 0,014 | 0,019 | 0,026 | 0,032 |
| Выбором | 0,001 | 0,003 | 0,006 | 0,01 | 0,02 | 0,026 | 0,032 | 0,058 | 0,057 |
| Обменом | 0,001 | 0,007 | 0,013 | 0,023 | 0,037 | 0,052 | 0,072 | 0,108 | 0,118 |
| Обменом 1 | 0,002 | 0,008 | 0,013 | 0,024 | 0,04 | 0,051 | 0,068 | 0,102 | 0,116 |
| Обменом 2 | 0,001 | 0,009 | 0,017 | 0,027 | 0,038 | 0,054 | 0,073 | 0,103 | 0,129 |
| Шелла | 0 | 0,001 | 0,002 | 0,005 | 0,006 | 0,008 | 0,011 | 0,013 | 0,021 |
| Хоара | 0,001 | 0,002 | 0,004 | 0,006 | 0,01 | 0,013 | 0,018 | 0,024 | 0,042 |
| Пирамидаль-ная | 0,001 | 0 | 0 | 0,001 | 0,001 | 0 | 0 | 0,002 | 0,001 |

6. Построить графики функций временной сложности алгоритмов сортировки.

Сортировка включением

Сортировка выбором

Сортировка обменом

Сортировка обменом 1

Сортировка обменом 2

Сортировка Шелла

Сортировка Хоара

Пирамидальная сортировка

7. Определить аналитическое выражение функции временной сложности алгоритмов сортировки.

Сортировка включением

Сортировка выбором

Сортировка обменом

Сортировка обменом 1

Сортировка обменом 2

Сортировка Шелла

Сортировка Хоара

Пирамидальная сортировка

8. Определить порядок функций временной сложности алгоритмов сортировки при сортировке упорядоченных, неупорядоченных и упорядоченных в обратном порядке массивов.

Сортировка включением

Неупорядоченный массив

Упорядоченный по возрастанию массив

Упорядоченный по убыванию массив

Сортировка выбором

Неупорядоченный массив

Упорядоченный по возрастанию массив

Упорядоченный по убыванию массив

Сортировка обменом

Неупорядоченный массив O(n2)

Упорядоченный по возрастанию массив O(n2)

Упорядоченный по убыванию массив O(n2)

Сортировка обменом 1

Неупорядоченный массив O(n2)

Упорядоченный по возрастанию массив O(n)

Упорядоченный по убыванию массив O(n2)

Сортировка обменом 2

Неупорядоченный массив O(n1,5)

Упорядоченный по возрастанию массив O(n)

Упорядоченный по убыванию массив O(n2)

Сортировка Шелла

Неупорядоченный массив O(n1,5)

Упорядоченный по возрастанию массив O(n\*log2n)

Упорядоченный по убыванию массив O(n2)

Сортировка Хоара

Неупорядоченный массив O(n\*log2n)

Упорядоченный по возрастанию массив O(n\*log2n)

Упорядоченный по убыванию массив O(n2)

Пирамидальная сортировка

Неупорядоченный массив O(n2)

Упорядоченный по возрастанию массив O(n\*log2n)

Упорядоченный по убыванию массив O(n2)