**МИНОБРНАУКИ РОССИИ**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

Лабораторная работа №3

Дисциплина: Алгоритмы и структуры данных

по теме «Сравнительный анализ методов сортировки (Pascal/C)»

Выполнил: ст. группы ВТ-22  
Макаров Даниил Сергеевич

Проверил: Синюк В.Г.

Белгород 2018

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.

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

5. Провести эксперименты по определению временных характеристик алгоритмов сортировки. Результаты экспериментов представить в виде таблицы 9, клетки которой содержат количество операций сравнения при выполнении 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).

6. Построить график зависимости количества операций сравнения от количества элементов в сортируемом массиве.

7. Определить аналитическое выражение функции зависимости количества операций сравнения от количества элементов в массиве.

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

**Результаты для неупорядоченного массива**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Сортировка | Количество элементов в массиве, тыс. | | | | | | | |
| 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 |
| Включением | 0,287239 | 0,629897 | 1,118036 | 1,736714 | 2,5208 | 3,39888 | 4,47264 | 5,644624 |
| Шелла | 0,004054 | 0,006018 | 0,008753 | 0,011186 | 0,013964 | 0,016976 | 0,02 | 0,021568 |
| Обменом | 0,72519 | 1,65706 | 2,964902 | 4,653678 | 6,696716 | 9,10268 | 11,91813 | 15,12998 |
| Выбором | 0,252739 | 0,572064 | 1,015679 | 1,596948 | 2,290152 | 3,115368 | 4,078248 | 5,156384 |
| Хоара | 0,001784 | 0,002726 | 0,003607 | 0,004562 | 0,00558 | 0,006744 | 0,007368 | 0,008656 |
| Пирамидальная | 0,003648 | 0,5782 | 0,00811 | 0,01064 | 0,012968 | 0,01564 | 0,01808 | 0,020448 |
| Обменом 1 | 0,95552 | 2,152372 | 3,838156 | 6,004222 | 8,642632 | 11,77018 | 15,38587 | 19,47797 |
| Обменом 2 | 0,893539 | 2,011777 | 3,573276 | 5,588436 | 8,057012 | 10,96694 | 14,32085 | 18,14799 |

**Результаты для упорядоченного массива**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Сортировка | Количество элементов в массиве, тыс. | | | | | | | | |
| 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 |
| Включением | 0,00016 | 0,000192 | 0,000272 | 0,000336 | 0,000416 | 0,000496 | 0,000544 | 0,00064 |
| Шелла | 0,001952 | 0,002832 | 0,004048 | 0,005264 | 0,005984 | 0,007824 | 0,009072 | 0,010368 |
| Обменом | 0,000048 | 0,00008 | 0,000112 | 0,000128 | 0,000208 | 0,000208 | 0,00024 | 0,00024 |
| Выбором | 0,257232 | 0,578368 | 1,029248 | 1,60824 | 2,31168 | 3,152336 | 4,114 | 5,2124 |
| Хоара | 0,000784 | 0,001104 | 0,0016 | 0,001952 | 0,002384 | 0,002704 | 0,003152 | 0,003552 |
| Пирамидальная | 0,003312 | 0,005152 | 0,007 | 0,009008 | 0,0112 | 0,013344 | 0,015296 | 0,017264 |
| Обменом 1 | 0,000064 | 0,00008 | 0,000096 | 0,000144 | 0,00016 | 0,000176 | 0,000208 | 0,000256 |
| Обменом 2 | 0,000048 | 0,000064 | 0,000112 | 0,000144 | 0,00016 | 0,000176 | 0,000192 | 0,000224 |

**Результаты для упорядоченного в обратном порядке массива**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Сортировка | Количество элементов в массиве, тыс. | | | | | | | |
| 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 |
| Включением | 0,5572 | 1,26 | 2,2388 | 3,49632 | 5,02976 | 6,845184 | 8,939648 | 11,3161 |
| Шелла | 0,002464 | 0,003792 | 0,00536 | 0,006752 | 0,00784 | 0,009696 | 0,01168 | 0,012992 |
| Обменом | 0,905 | 2,0284 | 3,615 | 5,636352 | 8,119232 | 11,07373 | 14,42406 | 18,2737 |
| Выбором | 0,2648 | 0,602 | 1,059376 | 1,656704 | 2,388128 | 3,272128 | 4,245472 | 5,373472 |
| Хоара | 0,000656 | 0,00096 | 0,001344 | 0,001792 | 0,002016 | 0,002336 | 0,002848 | 0,003232 |
| Пирамидальная | 0,00312 | 0,004832 | 0,006672 | 0,00848 | 0,01072 | 0,012352 | 0,014496 | 0,016224 |
| Обменом 1 | 0,967 | 2,1698 | 3,867344 | 6,03152 | 8,694688 | 11,91408 | 15,44371 | 19,53715 |
| Обменом 2 | 0,897 | 2,0239 | 3,593984 | 5,611552 | 8,08832 | 11,01728 | 14,36774 | 18,16877 |

**Заголовочный файл модуля sort**

#ifndef SORT

#define SORT

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

void SelSort(int\* A,int nn);

void BSort(int\* A,int nn);

void InsSort(int\* A,int nn);

void ShellSort(int \*mass,int n);

void QSort(int \*a, int L, int R);

void HoarSort(int \*a, int n);

void ba1Sort(int \*array, unsigned n);

void ba2Sort(int \*array, unsigned n);

void Sift(int \*array, int left, int right);

void HeapSort(int \*array, unsigned n);

#endif

**Содержимое файла sort.c**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <math.h>**

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

**int t;**

**t=\*b;**

**\*b=\*a;**

**\*a=t;**

**}**

**void SelSort(int \*A,int nn){**

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

**int min = i;**

**for (int j = i + 1; j < nn; j++) {**

**if (A[j] < A[min]) {**

**min = j;}}**

**if (A[min] != A[i]) {**

**swap(&A[i],&A[min]);**

**min = i;}**

**}}**

**void BSort(int\* A,int nn)**

**{ int i,j,k,r;**

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

**r = 0;**

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

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

**swap(&A[j-1],&A[j]);**

**r=1;**

**}**

**if ( r == 0)**

**break;}**

**}**

**void InsSort(int \*A,int nn)**

**{ int i,j,k;**

**for ( j=1; j<nn; j++ )**

**{ k = A[j];**

**i = j -1;**

**while ( k < A[i] && i >= 0){**

**A[i+1] = A[i];**

**i -= 1;**

**}**

**A[i+1] = k;**

**}**

**}**

**void ShellSort(int \*mass,int n)**

**{**

**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 QSort(int \*a, int L, int R){**

**int x = a[(L+R)/2], i = L, j = R, t;**

**while (i<=j){**

**while (a[i]<x)**

**i++;**

**while (a[j]>x)**

**j--;**

**if (i<=j){**

**swap(&a[i],&a[j]);**

**i++;**

**j--;**

**}}**

**if (L<j)**

**QSort(a,L,j);**

**if (i<R)**

**QSort(a,i,R);**

**}**

**void HoarSort(int \*a,int n){**

**QSort(a,0,n);**

**}**

**void Sift(int \*array, int left, int right){**

**int i = left;**

**int j = 2\*left+1;**

**int x = array[left];**

**if (j<right && array[j]<array[j+1]) j++;**

**while (j<=right && x<array[j]){**

**swap(&array[i], &array[j]);**

**i = j;**

**j = 2\*j+1;**

**if (j<right && array[j]<array[j+1]) j++;**

**}**

**}**

**void HeapSort(int \*array, unsigned n){**

**int left = n/2 ;**

**int right = n-1;**

**while(left>0){**

**left = left - 1;**

**Sift(array, left, right);**

**}**

**while(right>0){\**

**swap(&array[0], &array[right]);**

**right--;**

**Sift(array, left, right);**

**}**

**}**

**void ba1Sort(int \*array, unsigned n){**

**int i = 1;**

**int flag = 0;**

**do {**

**for (int j = n-1; j>=i; j--){**

**if (array[j-1] >= array[j]){**

**swap(&array[j-1], &array[j]);**

**flag = 1;**

**}**

**}**

**i++;**

**} while(flag && i<n);**

**}**

**void ba2Sort(int \*array, unsigned n){**

**unsigned i = 0;**

**do {**

**unsigned last\_swapped = n;**

**for (int j = n-1; j>i; j--){**

**if (array[j-1] >= array[j]){**

**swap(&array[j-1], &array[j]);**

**last\_swapped = j;**

**}**

**}**

**i = last\_swapped;**

**}while(i<n);**

**}**

**Заголовочный файл модуля inputdataGen**

#ifndef DGEN

#define DGEN

int\* sorted\_arrGen(int size);

int\* random\_arrGen(int size);

int\* sorted\_RarrGen(int size);

int\* init\_arr(int size);

void write\_arr(int\* arr,int size);

#endif

**Содержимое файла sort.c**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "inputdataGen.h"

int\* init\_arr(int size){

int\* p;

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

return p;

}

void write\_arr(int\* arr,int size){

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

printf("%i ",arr[i]);}

printf("\n");

}

int\* sorted\_arrGen(int size){

int\* result\_arr;

result\_arr=init\_arr(size);

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

result\_arr[i]=i;

}

return result\_arr;

}

int\* sorted\_RarrGen(int size){

int\* result\_arr;

result\_arr=init\_arr(size);

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

result\_arr[i]=size-i;

}

return result\_arr;

}

int\* random\_arrGen(int size){

int\* result\_arr;

result\_arr=init\_arr(size);

srand(time(NULL));

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

result\_arr[i]=rand()%1000;

}

return result\_arr;

}

**График реузльтатов для неупорядоченного массива**

**Результаты для упорядоченного массива**

**График для упорядоченного в обратном порядке массива**

**Пирамидальная сортировка**

В среднем O(n logn)

y = 0,0244x-0,086  
R² = 0,0009

Лучший случай O(n logn)

y = 0,0003x1,1068  
R² = 0,9998

Худший случай O(n logn)

y = 0,0002x1,1058  
R² = 0,9996

**Быстрая сортировка**

В среднем O(n logn)

y = 0,0002x1,0448   
R² = 0,999

Лучший случай O(n logn)

y = 7E-05x1,0168  
R² = 0,9978

Худший случай O(n2)

y = 6E-05x1,0601  
R² = 0,9968

**Сортировка Шелла**

В среднем O(n3/2)

y = 0,0003x1,1496  
R² = 0,998

Лучший случай O(n logn)

y = 0,0001x1,1243  
R² = 0,9957

Худший случай O(n2)

y = 0,0002x1,1062  
R² = 0,9982

**Сортировка включением**

В среднем O(n2)

y = 0,0029x1,9843  
R² = 1

Лучший случай O(n)

y = 2E-05x0,9611  
R² = 0,9847

Худший случай O(n2)

y = 0,0056x2,0006  
R² = 1

**Сортировка выбором**

В среднем O(n2)

y = 0,0025x2,0043  
R² = 1

Лучший случай O(n2)

y = 0,0026x2,0002  
R² = 1

Худший случай O(n2)

y = 0,0027x2,0002  
R² = 1

**Сортировка обменом**

В среднем O(n2)

y = 0,007x2,0171  
R² = 1

Лучший случай O(n)

y = 4E-06x1,1229  
R² = 0,9741

Худший случай O(n2)

y = 0,0091x1,9987  
R² = 1

**Сортировка обменом 2**

В среднем O(n2)

y = 0,0089x2,0018  
R² = 1

Лучший случай O(n)

y = 4E-06x1,046  
R² = 0,971

Худший случай O(n2)

y = 0,009x2,0001  
R² = 1

**Сортировка обменом 1**

В среднем O(n2)

y = 0,0095x2,0045  
R² = 1

Лучший случай O(n)

y = 7E-06x0,9269  
R² = 0,9719

Худший случай O(n2)

y = 0,0096x2,0005  
R² = 1