

Министерство образования и науки Российской Федерации  
Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

## Отчет по лабораторной работе

### “Методы поиска и сортировки”

**Выполнил:** студент группы 381606-2

\_\_\_\_\_ Тимакин Н. Е.

Подпись

**Проверил:** к.ф.-м.н., доц.

\_\_\_\_\_ Баркалов К.А.

Подпись

## Введение

Сортировка данных - это процесс изменения порядка расположения элементов в некоторых упорядоченных структурах данных таким образом, чтобы обеспечить возрастание или убывание числового значения элемента данных или определенного числового параметра, связанного с каждым элементом данных (ключа), при переходе от предыдущего элемента к последующему. Поиском наилучшего алгоритма сортировки данных человечество заинтересовалось ещё в конце XIX века, когда были созданы первые статистические табуляторы - электромеханические машины, предназначенные для автоматической обработки информации. И с тех пор данная проблема не утратила своей актуальности в силу того, что информации становится всё больше и больше. Человек не способен справиться с таким объёмом данных, поэтому этим занимаются ЭВМ, для которых и создаются новые алгоритмы сортировки, чтобы уменьшить время решения данной задачи.

## Постановка задачи

В программе должны быть реализованы как минимум 4 вида сортировки массивов:

- Пузырьковая
- Выбором
- Вставками
- Слиянием

- и 2 вида поиска элемента в массиве:

- Линейный
- Бинарный

Виды сортировок должны сравниваться по времени, количеству обменов и сравнений.

## Описание алгоритмов

- Пузырьковая сортировка (принимает на вход массив и количество элементов)
  - Копирование элементов из оригинального массива в копию
  - Цикл по  $i$  ( $0 < i < \text{кол-во элементов}$ )
  - Цикл по  $j$  ( $0 < j < \text{кол-во} - 1 - i$ )
  - Если текущий элемент больше следующего, то обменять их и увеличить кол-во обменов
  - Увеличить кол-во сравнений
  - Закрыть оба цикла
  - Печать кол-ва сравнений и обменов
  - Если кол-во элементов в массиве  $< 50$ , то напечатать массив

- Сортировка выбором (принимает на вход массив и количество элементов)
  - Копирование элементов из оригинального массива в копию
  - Цикл по  $i$  ( $0 < i < \text{кол-во элементов}$ )
  - Минимум =  $i$ -ый элемент
  - Позиция =  $i$
  - Цикл по  $j$  ( $i < j < \text{кол-во элементов}$ )
  - Если  $j$ -ый элемент < минимума, то минимум =  $j$ -ый элемент, позиция =  $j$
  - Увеличить кол-во сравнений
  - Закрывать цикл по  $j$
  - Обменять минимальный элемент с  $i$ -ым
  - Увеличить кол-во обменов
  - Закрывать цикл по  $i$
  - Печать кол-ва сравнений и обменов
  - Если кол-во элементов в массиве < 50, то напечатать массив
  
- Сортировка вставками (принимает на вход массив и количество элементов)
  - Копирование элементов из оригинального массива в копию
  - Цикл по  $i$  ( $0 < i < \text{кол-во элементов}$ )
  - Позиция = -1
  - Цикл по  $j$  ( $i-1 > j \geq 0$ )
  - Если  $j$ -ый элемент меньше  $i$ -ого, то позиция =  $j$ ; break;
  - Увеличить кол-во сравнений
  - Закрывать цикл по  $j$
  - Временной переменной присвоить  $i$ -ый элемент
  - Цикл по  $j$  ( $i-1 > j > \text{позиция}$ )
  - Присвоить следующему элементу текущий
  - Закрывать цикл по  $j$
  - Элементу[позиция+1] присвоить значение временной переменной
  - Увеличить кол-во обменов
  - Закрывать цикл по  $i$
  - Печать кол-ва сравнений и обменов
  - Если кол-во элементов в массиве < 50, то напечатать массив
  
- Слияние (принимает на вход 2 массива (mas и arr), начало1, начало2, конец1, конец2)
  - Цикл (пока  $i \leq \text{конец1}$  и  $j \leq \text{конец2}$ )
  - Если  $i$ -ый элемент меньше  $j$ -ого, то  $\text{arr}[k]$  присвоить значение  $\text{mas}[i]$ , увеличить счётчики  $k$  и  $i$
  - Иначе  $\text{arr}[k]$  присвоить значение  $\text{mas}[j]$ , увеличить счётчики  $k$  и  $j$
  - Закрывать цикл

- Если  $i$  больше конец1, то открыть цикл по  $j$  ( $j \leq \text{конец2}$ )
- $arr[k]$  присвоить значение  $mas[j]$ , увеличить счётчик  $k$
- Закрыть цикл по  $j$
- Иначе открыть цикл по  $i$  ( $i \leq \text{конец1}$ )
- $arr[k]$  присвоить значение  $mas[i]$ , увеличить счётчик  $k$
- Закрыть цикл по  $i$
- Закрыть иначе
- Открыть цикл по  $i$  ( $i \leq \text{конец2}$ )
- $mas[i]$  присвоить значение  $arr[i]$
- Закрыть цикл по  $i$
  
- Сортировка слиянием (принимает на вход два массива и границы)
  - Если правая граница равна левой, то return
  - Сортировка слиянием (принимает на вход два массива, левую границу и середину)
  - Сортировка слиянием (принимает на вход два массива, правую границу и середину+1)
  - Слияние (принимает на вход два массива, правую и левую границы, середину и середину+1)
  
- Линейный поиск (принимает на вход искомый элемент, массив и его длину)
  - Цикл по  $i$  ( $0 < i < \text{длина массива}$ )
  - Если  $i$ -ый элемент равен искомому, то запомнить позицию
  - Увеличить кол-во сравнений
  - Закрыть цикл
  - Напечатать кол-во сравнений
  - Вернуть позицию
  
- Бинарный поиск (принимает на вход искомый элемент, массив и его длину)
  - Цикл (пока левая граница меньше или равна правой)
  - Если средний элемент массива равен искомому, то увеличить и напечатать кол-во сравнений, вернуть середину
  - Иначе если средний элемент массива больше искомого, то правая граница = середина -1
  - Иначе левая граница = середина +1
  - Закрыть цикл
  - Напечатать кол-во сравнений
  - Вернуть -1

## Описание структуры программы

Программа состоит из одного модуля, в котором находятся следующие функции:

`void genarray(int mas[], int n)` - генерирует исходный массив

`void printarray(int mas[],int n)` - печатает исходный массив

`void bubblesort(int mas[],int n)` - пузырьковая сортировка

`void selectionsort(int mas[],int n)` - сортировка выбором

`void insertsort(int mas[], int n)` - сортировка вставками

`void combine(int mas[],int n1,int k1,int n2,int k2,int arr[])` - слияние двух массивов

`void combinesort(int mas[],int left,int right,int arr[])` - сортировка слиянием

`int linearsearch(int elem,int mas[],int n)` - линейный поиск

`int binarysearch(int elem,int mas[],int n)` - бинарный поиск

`void main()` - главная функция, реализованная следующим образом:

пользователь вводит число от 0 до 9, в зависимости от этого программа выполняет то или иное действие:

1. генерация массива с заданным пользователем числом элементов
2. применение пузырьковая сортировка
3. применение сортировка выбором
4. применение сортировка вставками
5. применение сортировка слиянием
6. печать исходного массива
7. сравнение видов сортировок (на экран выводятся время работы и количество сравнений и обменов для всех сортировок, и программа выбирает лучшую по времени)
8. линейный поиск
9. бинарный поиск
0. завершение программы

## Результаты экспериментов

```
C:\Windows\system32\cmd.exe

Enter 6 to print the array
Enter 7 to compare kinds of sort (Recommended to use with large arrays)
Enter 8 to search the element using linear search
Enter 9 to search the element using binary search
Enter 0 to exit
1
length=10000
?
Bubble:
Number of swaps=25341218
Number of compares=49995000
0.3430000 seconds
Selection:
Number of swaps=10000
Number of compares=50005000
0.1560000 seconds
Insert:
Number of swaps=9999
Number of compares=25346477
0.1400000 seconds
Combine:
0.0000000 seconds

Combine sort is the fastest sort
```

```
C:\Windows\system32\cmd.exe

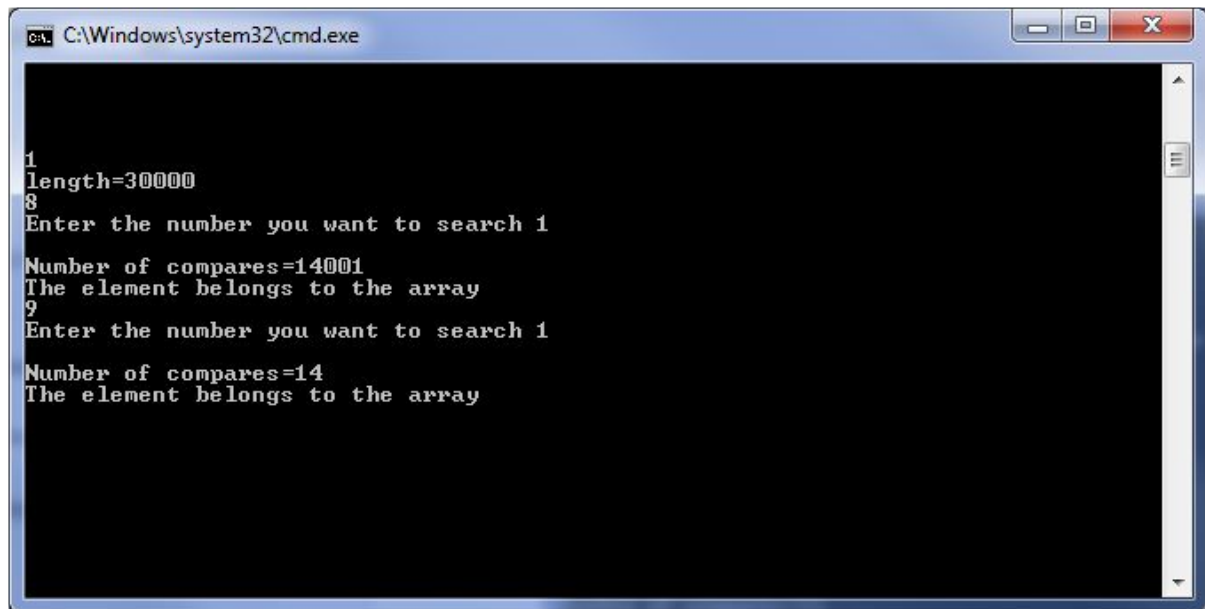
1
length=30000
?
Bubble:
Number of swaps=225348497
Number of compares=449985000
3.0920000 seconds
Selection:
Number of swaps=30000
Number of compares=450015000
1.4360000 seconds
Insert:
Number of swaps=29999
Number of compares=225394485
1.2490000 seconds
Combine:
0.0000000 seconds

Combine sort is the fastest sort
```

```
C:\Windows\system32\cmd.exe
1
length=50000
7
Bubble:
Number of swaps=624831557
Number of compares=1249975000
8.7240000 seconds
Selection:
Number of swaps=50000
Number of compares=1250025000
3.9240000 seconds
Insert:
Number of swaps=49999
Number of compares=624959496
3.4790000 seconds
Combine:
0.0150000 seconds
Combine sort is the fastest sort
-
```

Как видно из приведённых скриншотов, сортировка слиянием показывает наилучший результат

```
C:\Windows\system32\cmd.exe
Enter 1 to input the length of array (max 50000)
Enter 2 to use a bubble sort
Enter 3 to use a selection sort
Enter 4 to use a insert sort
Enter 5 to use a combine sort
Enter 6 to print the array
Enter 7 to compare kinds of sort (Recommended to use with large arrays)
Enter 8 to search the element using linear search
Enter 9 to search the element using binary search
Enter 0 to exit
1
length=10000
8
Enter the number you want to search 67
Number of compares=595
The element belongs to the array
9
Enter the number you want to search 67
Number of compares=11
The element belongs to the array
-
```



```
C:\Windows\system32\cmd.exe

1
length=30000
8
Enter the number you want to search 1
Number of compares=14001
The element belongs to the array
9
Enter the number you want to search 1
Number of compares=14
The element belongs to the array
```

А бинарный поиск на порядок эффективнее линейного

## Заключение

На примере данной программы легко убедиться в том, что приведённые методы сортировок массивов можно расположить в порядке возрастания эффективности в среднем случае: пузырьковая → выбором → вставками → слиянием. А среди видов поиска элемента лучшим является бинарный.

## Литература

Б. Керниган, Д. Ритчи “Язык программирования Си”

А.О. Грудзинский, И.Б. Мееров, А.В. Сысоев “Методы программирования”

<https://ru.wikipedia.org>

## Приложение

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX_SIZE 50000

void genarray(int mas[], int n);

void printarray(int mas[],int n);

void bubblesort(int mas[],int n);
```



```

void selectionsort(int mas[],int n);

void insertsort(int mas[], int n);

void combine(int mas[],int n1,int k1,int n2,int k2,int arr[]);

void combinesort(int mas[],int left,int right,int arr[]);

int linearsearch(int elem,int mas[],int n);

int binarysearch(int elem,int mas[],int n); // Требуется сортировка данных перед
использованием

void main()
{
    int k=0,mas[MAX_SIZE],arr[MAX_SIZE],size,elemL,elemB,posL,posB;
    double timeB,timeS,timeI,timeC;
    printf("Enter 1 to input the length of array (max 50000)\n");
    printf("Enter 2 to use a bubble sort\n");
    printf("Enter 3 to use a selection sort\n");
    printf("Enter 4 to use a insert sort\n");
    printf("Enter 5 to use a combine sort\n");
    printf("Enter 6 to print the array\n");
    printf("Enter 7 to compare kinds of sort (Recomended to use with large
arrays)\n");
    printf("Enter 8 to search the element using linear search\n");
    printf("Enter 9 to search the element using binary search\n");
    printf("Enter 0 to exit\n");
    while(1)
    {
        scanf("%i",&k);
        if(k==0) break;
        switch(k)
        {
            case 1:
                printf("length=");
                scanf("%i",&size);
                genarray(mas,size);
                break;
            case 2: bubblesort(mas,size); break;
            case 3: selectionsort(mas,size); break;

```

```

        case 4: insertsort(mas,size); break;
        case 5: combinesort(mas,0,size-1,arr); break;
        case 6: printarray(mas,size); break;
        case 7:
            if(size<51)
            {
                printf("\nError 01\n");
                break;
            }
            printf("Bubble:\n");
            clock_t start, finish;
            start = clock();
            bubblesort(mas,size);
            finish = clock();
            timeB = (double)(finish - start) /
CLOCKS_PER_SEC;

            printf( "%2.7f seconds\n", timeB );
            printf("Selection:\n");
            start = clock();
            selectionsort(mas,size);
            finish = clock();
            timeS = (double)(finish - start) /
CLOCKS_PER_SEC;

            printf( "%2.7f seconds\n", timeS );
            printf("Insert:\n");
            start = clock();
            insertsort(mas,size);
            finish = clock();
            timeI = (double)(finish - start) /
CLOCKS_PER_SEC;

            printf( "%2.7f seconds\n", timeI );
            printf("Combine:\n");
            start = clock();
            combinesort(mas,0,size-1,arr);
            finish = clock();
            timeC = (double)(finish - start) /
CLOCKS_PER_SEC;

            printf( "%2.7f seconds\n", timeC );
            if(timeB<timeS && timeB<timeI && timeB<timeC)
printf("\nBuble sort is the fastest sort\n");
            else if(timeS<timeB && timeS<timeI && timeS<timeC)
printf("\nSelection sort is the fastest sort\n");

```

```

        else if(timeI<timeB && timeI<timeS && timeI<timeC)
printf("\nInsert sort is the fastest sort\n");
        else if(timeC<timeB && timeC<timeS && timeC<timeI)
printf("\nCombine sort is the fastest sort\n");
        else printf("\nError 01\n");
        break;
    case 8:
        printf("Enter the number you want to search ");
        scanf("%i",&elemL);
        printf("\n");
        posL=linearsearch(elemL,mas,size);
        if(posL>=0) printf("The element belongs to the array\n");
        else printf("The element don't belongs to the array\n");
        break;
    case 9:
        combinesort(mas,0,size-1,arr);
        printf("Enter the number you want to search ");
        scanf("%i",&elemB);
        printf("\n");
        posB=binarysearch(elemB,mas,size);
        if(posB>=0) printf("The element belongs to the array\n");
        else printf("The element don't belongs to the array\n");
        break;
    default: printf("Input error\n"); break;
}
}
}

void genarray(int mas[], int n)
{
    int i;
    for(i=0;i<n;i++)
    {
        mas[i]=rand()%10000;
    }
}

void printarray(int mas[],int n)
{
    int i;
    printf("mas=");
    for(i=0;i<n;i++)

```

```

    {
        printf("%i ",mas[i]);
    }
    printf("\n");
}

void bubblesort(int mas[],int n)
{
    int i,j,buf,comp=0,swap=0,mas1[MAX_SIZE],a;
    for(i=0;i<n;i++)
    {
        mas1[i]=mas[i];
    }
    for(i=0;i<n;i++)
    {
        for(j=0;j<n-1-i;j++)
        {
            if(mas1[j]>mas1[j+1])
            {
                buf=mas1[j];
                mas1[j]=mas1[j+1];
                mas1[j+1]=buf;
                swap++;
            }
            comp++;
        }
    }
    printf("Number of swaps=%i\n",swap);
    printf("Number of compares=%i\n",comp);
    if(n<=50) printarray(mas1,n);
}

void selectionsort(int mas[],int n)
{
    int i,j,pos=0,min=0,comp=0,swap=0,mas1[MAX_SIZE],a;
    for(i=0;i<n;i++)
    {
        mas1[i]=mas[i];
    }
    for(i=0;i<n;i++)
    {
        min=mas1[i];

```

```

        pos=i;
        for(j=i;j<n;j++)
        {
            if(mas1[j]<min)
            {
                min=mas1[j];
                pos=j;
            }
            comp++;
        }
        mas1[pos]=mas1[i];
        mas1[i]=min;
        swap++;
    }
    printf("Number of swaps=%i\n",swap);
    printf("Number of compares=%i\n",comp);
    if(n<=50) printarray(mas1,n);
}

```

```

void insertsort(int mas[], int n)
{
    int i,j,pos=0,buf=0,comp=0,swap=0,mas1[MAX_SIZE],a;
    for(i=0;i<n;i++)
    {
        mas1[i]=mas[i];
    }
    for(i=1;i<n;i++)
    {
        pos=-1;
        for(j=i-1;j>=0;j--)
        {
            if(mas1[j]<mas1[i])
            {
                pos=j;
                break;
            }
        }
        comp++;
    }
    buf=mas1[i];
    for(j=i-1;j>pos;j--)
    {
        mas1[j+1]=mas1[j];
    }
    mas1[pos+1]=buf;
}

```

```

        }
        mas1[pos+1]=buf;
        swap++;
    }
    printf("Number of swaps=%i\n",swap);
    printf("Number of compars=%i\n",comp);
    if(n<=50) printarray(mas1,n);
}

```

```

void combine(int mas[],int n1,int k1,int n2,int k2,int arr[])
{
    int i=n1,j=n2,k=n1;
    while(i<=k1 && j<=k2)
    {
        if(mas[i]<mas[j]) arr[k++]=mas[i++];
        else arr[k++]=mas[j++];
    }
    if(i>k1)
    {
        for(j;j<=k2;j++)
        {
            arr[k++]=mas[j];
        }
    }
    else
    {
        for(i;i<=k1;i++)
        {
            arr[k++]=mas[i];
        }
    }
    for(i=n1;i<=k2;i++)
    {
        mas[i]=arr[i];
    }
}

```

```

void combinesort(int mas[],int left,int right,int arr[])
{
    int mid=(left+right)/2;
    if (left==right) return;
    combinesort(mas,left,mid,arr);

```

```

        combinesort(mas,mid+1,right,arr);
        combine(mas,left,mid,mid+1,right,arr);
    }

int linearsearch(int elem,int mas[],int n)
{
    int i,pos=-1,comp=1;
    for(i=0;i<n;i++)
    {
        if(mas[i]==elem)
        {
            pos=i;
            break;
        }
        comp++;
    }
    printf("Number of compares=%i\n",comp);
    return pos;
}

int binarysearch(int elem,int mas[],int n)
{
    int left=0,right=n-1,mid,pos=-1,comp=0;
    while (left<=right)
    {
        mid=(left+right)/2;
        if(mas[mid]==elem)
        {
            comp++;
            printf("Number of compares=%i\n",comp);
            return mid;
        }
        else if(mas[mid]>elem)
            right=mid-1;
        else
            left=mid+1;
        comp++;
    }
    printf("Number of compares=%i\n",comp);
    return -1;
}

```