5. ИССЛЕДОВАНИЕ АЛГОРИТМОВ СОРТИРОВКИ

Цель работы – изучить основные алгоритмы сортировки, овладеть практическими навыками разработки, программирования и применения алгоритмов сортировки.

5.1 Подготовка к работе

При подготовке к работе необходимо изучить алгоритмы сортировки, методы оценки быстродействия различных алгоритмов сортировки.

5.2 Теоретические сведения

Сортировка массива – это переразмещение элементов массива в заданном порядке (по возрастанию или убыванию их значений). *Основная цель сортировки – облегчить последующий поиск*. Методы условно разделяют по главной идее алгоритма, а реализуются они целым семейством алгоритмов.

Для оценки быстродействия алгоритмов различных методов сортировки, как правило используют два показателя:

- прямые методы сортировки;
- улучшенные методы сортировки.

Прямые методы сортировки по принципу, лежащему в основе метода, в свою очередь разделяются на три группы:

- сортировка вставкой (включением);
- сортировка выбором (выделением);
- сортировка обменом («пузырьковая» сортировка).

Улучшенные методы сортировки основываются на тех же принципах, что и прямые, но используют некоторые оригинальные идеи для ускорения процесса сортировки. Прямые методы на практике используют довольно редко, так как имеют довольно низкое быстродействие. Однако они хорошо показывают суть основанных на них улучшенных методов. Кроме того, в некоторых случаях (как правило, при небольшой длине массива и/или особом исходном расположении элементов массива) некоторые из прямых методов могут даже превзойти улучшенные методы.

Сортировка вставкой

Основная идея данного алгоритма заключается в том, что массив сортируемых объектов условно разбивается на две части: отсортированную и не отсортированную. На каждом шаге, начиная с i=2 (т.е. со второго элемента массива) из исходной последовательности извлекается один элемент и перемещается в готовую (отсортированную) половину последовательности в соответствующую позицию так, чтобы упорядоченность массива не нарушилась.

Условием завершения вложенного цикла по “j” будет либо ситуация, когда с выделенным элементом дошли до начала отсортированной последовательности, не найдя ни одного элемента, который бы был меньше, чем выделенный, а следовательно, новый элемент должен быть вставлен в самое начало ряда. Либо цикл будет завершен, когда будет найден такой элемент с номером “j”, который оказался меньше чем выделенный, и, значит, после него в позицию “j+1” необходимо поместить выделенный элемент.

Программа, реализующая рассмотренный алгоритм, будет иметь следующий вид:

```
int arr [] =  {8,7,6,0,4,3,2,1};
```

```
const int N = 8;
cout<< "Массив:\n";            //Вывод массива
for (int i=0; i < N; i++)
        cout<< arr [i]<< " ";
cout<<"\n";
//сортировка массива
for (int i=1; i< N; i++)
{
        int tmp = arr [i];  //взятие неотсортированного элемента
        int j;
        for (j=i - 1; j >=0 && arr [j] > tmp; j--)
                arr [j + 1] = arr [j];
        arr [j + 1] = tmp;
}
cout<< "\nОтсортированный массив:\n";   //Вывод массива
for (int i=0; i < N; i++)
        cout<< arr [i]<< " ";
```

Сортировка выбором

Данный алгоритм сортировки основан на многократном выполнении следующего набора шагов:

1) в исходном массиве находится элемент с наименьшим значением;

2) найденный элемент меняется местами с самым первым элементом;

3) эти два шага повторяются многократно для оставшейся неупорядоченной части массива, начиная со второго, потом третьего и так далее до тех пор, пока не дойдем до конца массива и не отсортируем его целиком.

Программа, реализующая рассмотренный алгоритм, будет иметь следующий вид:

```
//Ввод массива
  for (int i=0; i< N; i++)
  {
        int posMin = i;
        for (int j=i + 1; j < N; j++)
                if (arr [j] < arr[posMin])
                        posMin = j;
        int tmp = arr [posMin];
        arr [posMin] = arr [i];
        arr [i] = tmp;
  }
//Вывод массива
```

Сортировка обменом («пузырьковая» сортировка)

Принцип метода. Слева направо поочередно сравниваются два соседних элемента, и если их взаиморасположение не соответствует заданному условию упорядоченности, то они меняются местами. Далее берутся два следующих соседних элемента и так далее до конца массива.

После одного такого прохода на последней n-ой позиции массива будет стоять максимальный элемент («всплыл» первый «пузырек»). Поскольку максимальный элемент уже стоит на своей последней позиции, то второй проход обменов выполняется до (n-1)-го. И так далее. Всего требуется (n-1) проход.

Программа, реализующая рассмотренный алгоритм, будет иметь следующий вид:

```
//Ввод массива
  for (int i=N-1; i > 0; --i)
    // «Всплывание» очередного максимального элемента на i -ю позицию
          for (int j=0; j < i; ++j)
                if (arr [j] > arr [j+1])
                {
                      int tmp = arr [j];
                      arr [j] = arr [j+1];
                      arr [j+1] = tmp;
                }
  //Вывод массива
```

Шейкер-сортировка

Шейкер-сортировка является усовершенствованным методом пузырьковой сортировки.

Принцип метода. От последней перестановки до конца (начала) массива находятся отсортированные элементы. Учитывая данный факт, просмотр осуществляется не до конца (начала) массива, а до конкретной позиции. Границы сортируемой части массива сдвигаются на 1 позицию на каждой итерации.

Массив просматривается поочередно справа налево и слева направо.Просмотр массива осуществляется до тех пор, пока все элементы не встанут в порядке возрастания (убывания).

Количество просмотров элементов массива определяется моментом упорядочивания его элементов.

Программа, реализующая рассмотренный алгоритм, будет иметь следующий вид:

```
//Ввод массива
      int left= 0, right = N-1;
      double t;
      int flag = 1;  // флаг наличия перемещений
      while((left < right) && flag > 0)
     {
            flag = 0;
            for(int i=left; i<right; i++) //двигаемся слева направо
             if(arr[i]>arr[i+1])
            {
              t=arr[i];
              arr[i]=arr[i+1];
              arr[i+1]=t;
              flag = 1;
            }
             right--;          //сдвигаем правую границу
             for(int i=right; i>left; i--) //двигаемся справа налево
             if(arr[i-1]>arr[i])
             {
               t=arr[i];
               arr[i]=arr[i-1];
               arr[i-1]=t;
               flag = 1;
             }
            left++;  // сдвигаем левую границу
     }
    //Вывод массива
```

Сортировка включениями с убывающими приращениями

В 1959 г. Д. **Шеллом** было предложено усовершенствование сортировки с помощью прямого включения.

Сначала отдельно группируются и сортируются элементы, отстоящие друг от друга на 4 позиции. Такой процесс называется ***четвертной сортировкой***.

После первого прохода элементы перегруппировываются — теперь каждый элемент группы отстоит от другого на 2 позиции — и вновь сортируются (***двойная сортировка***).

На третьем проходе идет обычная сортировка.

На каждом этапе либо сортируется относительно мало элементов, либо элементы уже довольно хорошо упорядочены и требуется сравнительно немного перестановок.

Расстояния в группах можно уменьшать по-разному, главное чтобы последнее Расстояние было единичным. В самом плохом случае последний проход сделает всю работу.

Программа, реализующая рассмотренный алгоритм, будет иметь следующий вид:

```
//Ввод массива
int i, j, increment= 4, temp;
while (increment > 0)
{
        for (i=0; i < N; i++)
        {
                j = i;
                temp = arr[i];
                while((j>=increment) && (arr[j-increment]>temp))
                {
                        arr[j] = arr[j - increment];
                        j = j - increment;
                }
                 arr[j] = temp;
        }
         if (increment/2 != 0)
         increment = increment/2;
        else if (increment == 1)
                  increment = 0;
                  else
                        increment = 1;
 }
//Вывод массива
```

Сравнение прямых методов сортировки

Теоретические и практические исследования алгоритмов прямых методов сортировки показали, что как по числу сравнений, так и по числу присваиваний они имеют квадратичную зависимость от длины массива n. Исключение составляет метод выбора, который имеет число присваиваний порядка $n*\ln(n)$. Это свойство алгоритма выбора полезно использовать в задачах сортировки в сложных структурах данных, когда на одно сравнение выполняется большое число присваиваний. В таких задачах метод выбора успешно конкурирует с самыми быстрыми улучшенными методами сортировки [7].

Если сравнить рассмотренные прямые методы между собой, то в среднем методы вставки и выбора оказываются приблизительно эквивалентными и в несколько раз (в зависимости от длины массива) лучше, чем метод обмена («пузырька»).

5.3 Варианты заданий

Выполнить сортировку массива, используя алгоритмы в соответствии с вариантом (табл.5.1).

Таблица 5.1 – Варианты заданий

№ вар.	Тип и размер массива	Алгоритм сортировки				
		вставкой	выбором	обменом	Шейкер сортировка	сортировка Шелла
1.	целые числа C(25)	1	-	3	-	-
2.	целые числа D(n)	-	2	-	5	-
3.	действительные числа B(20)	-	-	6	-	4
4.	действительные числа *L(20)*		3	-	4	-
5.	целые числа *D(k)*	2	-	-	7	-
6.	целые числа *M(15)*	-	1	-	-	6
7.	действительные числа *L(20)*	-	-	5	1	-
8.	целые числа *C(25)*	3	-	6	-	-
9.	целые числа *K(n)*	-	4	-	1	-
10.	действительные числа *L(20)*	-	7	-	-	5
11.	целые числа *D(k)*	-	3	4	-	-
12.	целые числа *M(15)*	2	-	-	-	1
13.	целые числа A(k)	-	7	-	-	4
14.	целые числа C(25)	2	-	-	6	-
15.	целые числа D(n)	-	3	5	-	-
16.	действительные числа B(20)	-	1	-	7	-

17.	целые числа A(15)	1	-	5	-	-
18.	целые числа C(25)	-	-	6	4	-

где 1- сортировка первой половины массива;

2- сортировка элементов массива, расположенных за максимальным элементом;

3- сортировка второй половины массива;

4- сортировка элементов массива, расположенных за минимальным элементом;

5- сортировка всего массива;

6- сортировка элементов массива, расположенных перед максимальным элементом;

7- сортировка элементов массива, расположенных перед минимальным элементом;

5.4 Контрольные вопросы

1 Классификация алгоритмов сортировки?
2 Алгоритм сортировки вставкой?
3 Алгоритм сортировки выбором?
4 Алгоритм сортировки обменом?
5 Алгоритм Шейкер-сортировки?
6 Алгоритм сортировки Шелла?