Курс «Основы программирования»

Федорук Елена Владимировна ст. преподаватель каф РК-6 МГТУ им.Н.Э.Баумана

Лекция №10

Сортировка одномерных массивов

Сортировка — это процесс упорядочивания информации по определенному признаку.

Цель сортировки — облегчение последующего поиска элементов. Существует большое количество методов сортировки и, соответственно, алгоритмов их реализации, которые имеют свои достоинства и недостатки. При выборе алгоритма, реализующего определенный метод, следует выполнить анализ его производительности в определенных условиях.

В качестве оценки производительности метода обычно используют функциональную зависимость времени работы программы от размерности исходного массива t(n). При анализе алгоритма в первую очередь интерес представляет характер зависимости при достаточно большой размерности массива ($n \to \infty$).

В программировании порядок зависимости времени выполнения программы, реализующей некоторый метод, от размерности исходных данных n называют *вычислительной сложностью* данного метода.

Вычислительная сложность O(const) означает, что время решения задачи данным методом не зависит от размерности, O(n) — время работы пропорционально размерности задачи, $O(n^2)$ — время работы пропорционально квадрату размерности задачи и т.д.

Временную сложность можно оценивать, используя в качестве единицы измерения временные единицы (мкс, с и т.д.), а можно - используя время выполнения основных, характеризующих процесс операций, количество которых соответствует количеству итераций цикла, например, операций сравнения, операций пересылки. Время выполнения этих операций можно считать постоянным. Следовательно, функциональная зависимость выполняемых операций от размерности задачи по характеру будет совпадать с временной зависимостью.

На практике интересны методы сортировки, которые позволяют экономно использовать оперативную память, поэтому целесообразно рассматривать методы, не требующие использования дополнительных массивов. Такие методы называют прямыми.

Самыми простыми из прямых методов являются:

- метод выбора;
- метод вставки;
- метод обменов (метод пузырька).

Метод выбора

Метод выбора - один из самых простых прямых методов сортировки.

Находим минимальный элемент массива размерностью **n**. Найденный элемент меняется местами с первым элементом. Затем процесс повторяется, начиная со второго элемента массива и т.д.

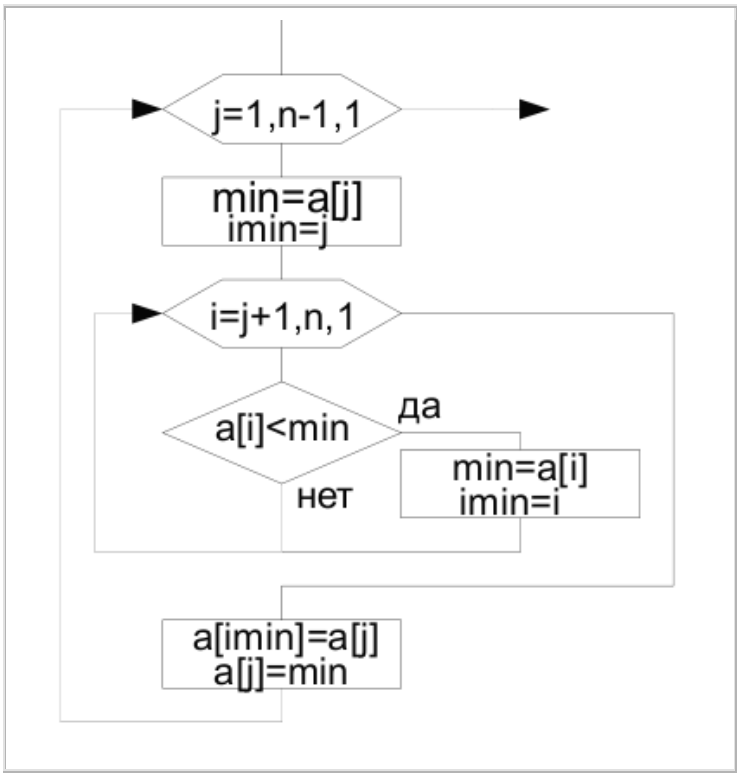

Рис. 1. Алгоритм метода выбора

В табл. 1-табл. 6 приведены результаты сортировки после выполнения очередного прохода.

Таблица 1

Массив	7.8	-6.4	5.7	1.3	8.5	4.6

Таблица 2

1 проход	-6.4	7.8	5.7	1.3	8.5	4.6

Таблица 3

2 проход	-6.4	1.3	5.7	7.8	8.5	4.6

Таблица 4

3 проход	-6.4	1.3	4.6	7.8	8.5	5.7

Таблица 5

4 проход	-6.4	1.3	4.6	5.7	8.5	7.8

Таблица 6

5 проход	-6.4	1.3	4.6	5.7	7.8	8.5

Пример 1

```
for(j=0; j<n-1; j++)
  {
   min=a[j];
   imin=j;
   for(i=j+1;i<n;i++)
      if(a[i]<min)
        {
         min=a[i];
         imin=i;
        }
   }
```

Примечание 1

В языке Си индексация элементов массива начинается с 0.

Оценим вычислительную сложность метода, используя в качестве основной операции операцию сравнения.
Для поиска минимального элемента в каждом проходе потребуется выполнить n-1, n-2,..., 1 операцию сравнения, то есть всего n(n-1)/2 операций сравнения. Следовательно, вычислительная сложность данного метода $O(n^2)$. Причем время сортировки не зависит от исходного порядка элементов.

Метод вставки

Метод вставки — один из прямых методов сортировки.

В исходном состоянии считаем, что сортируемая последовательность состоит из двух последовательностей: уже сортированной (на первом шаге состоит из единственного — первого элемента) и последовательности, которую еще необходимо сортировать. На каждом шаге из сортируемой последовательности извлекается элемент и вставляется в первую последовательность так, чтобы она оставалась сортированной. Поиск места вставки осуществляется с конца первой последовательности, сравнивая вставляемый элемент a_i с очередным элементом сортированной последовательности a_j. Если элемент a_i больше a_j, его вставляют вместо a_{j+1}, иначе сдвигают a_j вправо и уменьшают j на единицу. Поиск места вставки завершают, если элемент вставлен или достигнут левый конец массива. В последнем случае элемент a_i вставляют на первое место.

Существует прием, который позволяет отменить проверку достижения левой границы массива. Этот прием называется "проверка с барьером". С использованием этого приема проверка осуществляется так, чтобы из цикла поиска вставки в любом случае выход происходил по первому условию. Для этого достаточно поместить вставляемый элемент перед первым элементом массива, как элемент с индексом 0. Этот элемент и станет естественным барьером для ограничения выхода за левую границу массива.

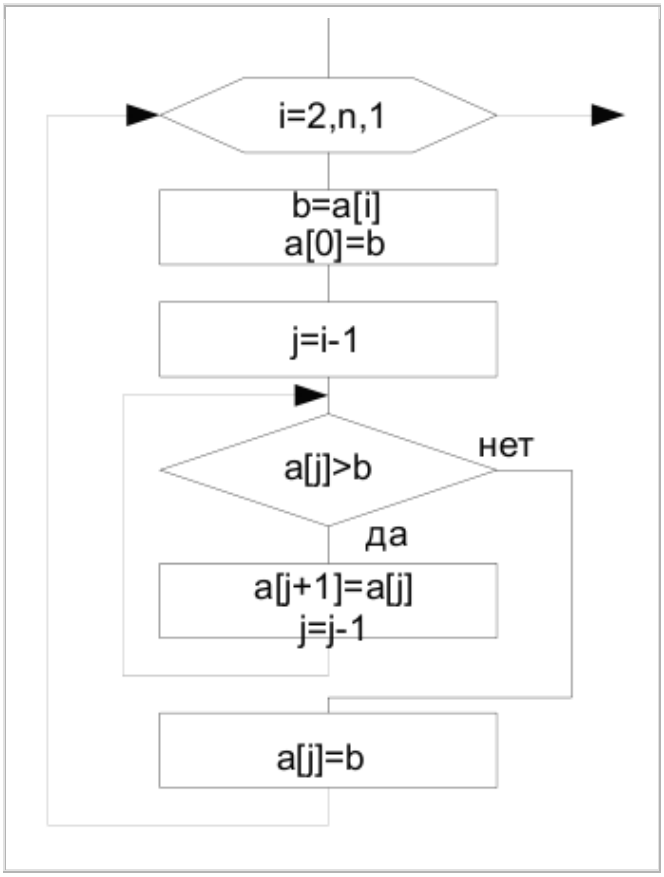

Рис. 1. Алгоритм метода вставки

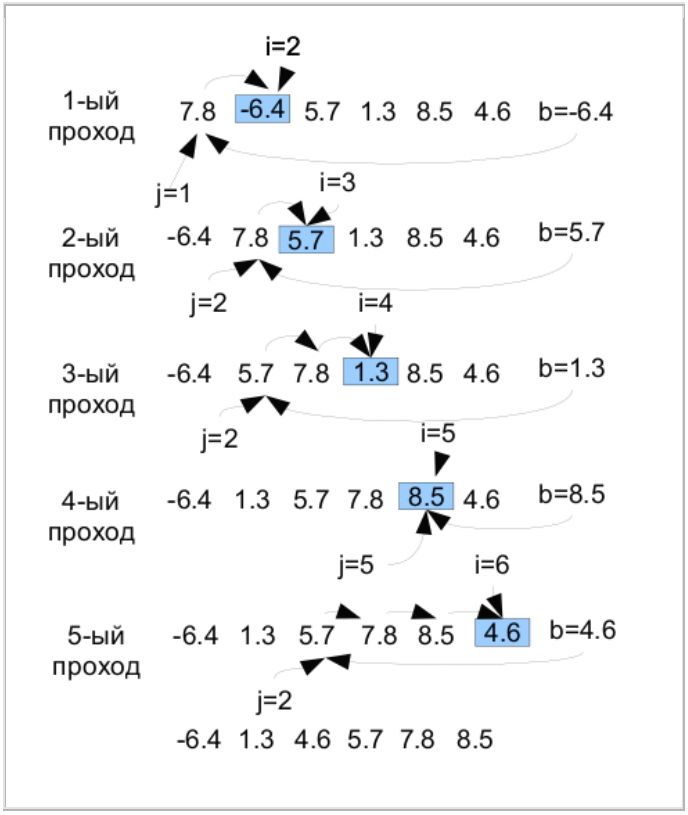

Рис. 2.

Примечание 1

В программе элементы сортируемого массива расположены, начиная с индекса 1. В нулевом элементе массива будет располагаться барьер.

```
/*n-число элементов сортируемого массива*/
for (i=2; i<n+1; i++)      /*начиная со второго элемента до конца массива*/
 {
  b=a[i];
  a[0]=b;                  /* барьер */
  j=i-1;
  while(b<a[j])            /* ока очередной элемент сортированного массива */
    {                      /*  больше i-го элемента */
     a[j+1]=a[j ]          /* сдвигаем элемент */
     j=j-1;
     }
   a[j+1]=b;               /*как только найдено место, туда записываем b*/
  }
```

Оценим вычислительную сложность этого метода, определив количество операций сравнения.

Для поиска места i-го элемента каждый раз потребуется выполнить от 1 до i-1 операции сравнения, т.е. в среднем i/2 операций сравнения. Значение i изменяется от 2 до n, т.е. выполняется n-1 проход, в каждом из которых происходит в среднем от 1 до n/2 сравнения. Суммарно в среднем

для решения задачи требуется выполнить $(n-1)(n/2+1)/2=(n^2+n-2)/4$ операции сравнения. Откуда вычислительная сложность метода в среднем равна $O_{ср}(n^2)$, хотя время выполнения примерно в два раза меньше, чем в методе выбора. В данном случае вычислительная сложность зависит от исходного расположения элементов массива.

В лучшем случае, когда массив уже упорядочен, поиск места вставки требует одного сравнения для каждого элемента, и количество сравнений равно n-1. Вычислительная сложность равна O(n).

В худшем случае, если элементы в исходном состоянии расположены в обратном порядке, поиск места вставки для каждого элемента потребует 1, 2, 3,..., n-1 сравнения, следовательно, всего потребуется n(n-1)/2 операций сравнения. Вычислительная сложность равна $O(n^2)$.

Таким образом, за счет ускорения сортировки в лучших случаях данный метод имеет лучшие временные характеристики, чем метод выбора.

Метод обменов

Метод обменов или метод пузырька является прямым методом сортировки массива.

Алгоритм метода обмена основан на сравнении пары соседних элементов. Если расположение элементов не удовлетворяет условиям сортировки, то их меняют местами. Сравнения и перестановки продолжаются до тех пор, пока не будут упорядочены все элементы. Определить, что элементы упорядочены, можно, считая количество выполненных перестановок: если количество перестановок равно нулю, то массив отсортирован.

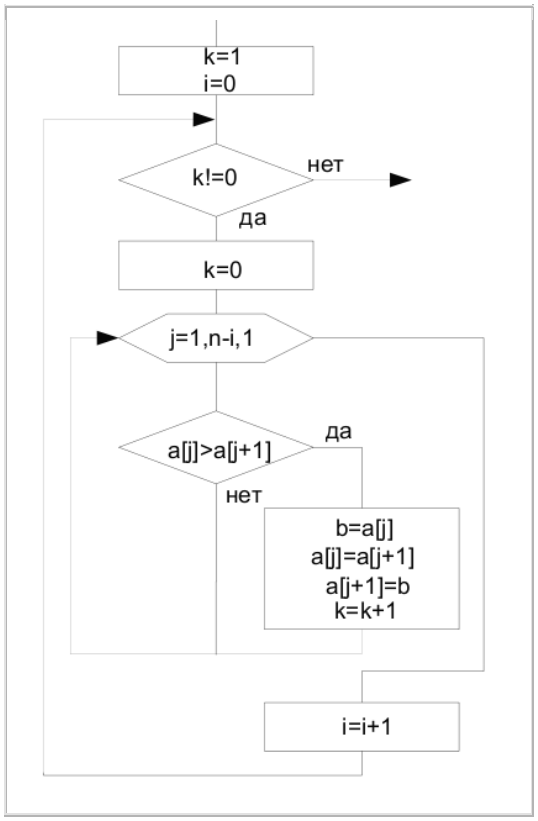

Рис. 1. Алгоритм метода обменов

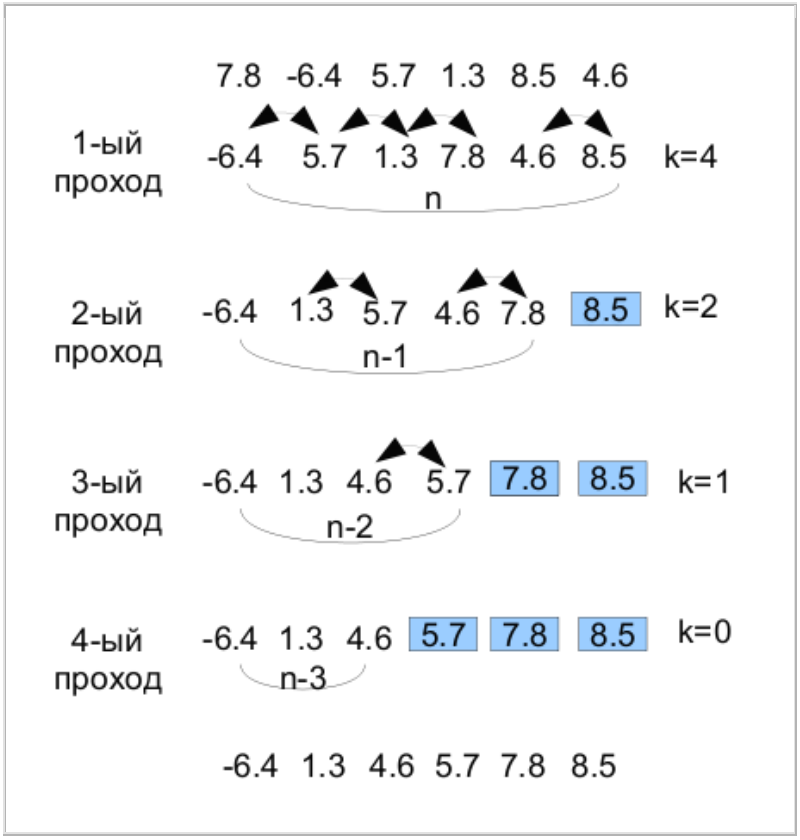

Рис. 2.

```
k=1;                    /*количество перестановок, в начале не 0*/
i=1;                    /*номер очередного просмотра, в начале 1*/
while(k!=0)             /*пока есть перестановки*/
  {
   k=0;                 /*обнуляем счетчик перестановок*/
   for(j=0; j<n-i; j++) /*цикл сравнения соседних элементов*/
     if(a[j]>a[j+1])    /*если предыдущий элемент больше последующего, то*/
       {
        b=a[j];         /*осуществляем перестановку*/
        a[j]=a[j+1];
        a[j+1]=b;
        k=k+1;          /*счетчик перестановок увеличиваем на 1*/
        }
    i=i+1;              /*увеличиваем номер просмотра на 1*/
    }
```

Оценим вычислительную сложность данного метода. Очевидно, что она сильно зависит от исходного расположения элементов массива. Так в лучшем случае, когда массив уже отсортирован, потребуется выполнить n-1

сравнение для проверки правильности расположения элементов массива, т.е. вычислительная сложность равна O(n).

В худшем случае, если массив отсортирован в обратном порядке, будет выполнено n-1, n-2, ... 1 операций сравнения, т.е. всего $(n^2-n)/2$ операций сравнения, следовательно, вычислительная сложность равна $O(n^2)$.

Выполнить усредненную оценку данного метода достаточно сложно, так как зависимость времени выполнения от количества неправильно стоящих элементов не является линейной.