Сортировки массива

Сортировка пузырьком

Сортировка пузырьком – простейший алгоритм сортировки, применяемый чисто для учебных целей.

Недостатки:
Практического применения этому алгоритму нет, так как он не эффективен, особенно если необходимо отсортировать массив большого размера.

Достоинство:
Простота реализации алгоритма.

Сложность алгоритма: $O(n^2)$.

Связь с другими сортировками: лежит в основе некоторых более совершенных алгоритмов, таких как шейкерная сортировка, пирамидальная сортировка и быстрая сортировка.

Видео: [http://www.youtube.com/watch?v=lyZQPjUT5B4]

Алгоритм

1. При первом проходе по массиву элементы попарно сравниваются между собой: первый со вторым, затем второй с третьим, следом третий с четвертым и т.д. Если предшествующий элемент оказывается больше последующего, то их меняют местами.

2. Не трудно догадаться, что постепенно самое большое число оказывается последним. Остальная часть массива остается не отсортированной, хотя некоторое перемещение элементов с меньшим значением в начало массива наблюдается.

3. При втором проходе незачем сравнивать последний элемент с предпоследним. Последний элемент уже стоит на своем месте. Значит, число сравнений будет на одно меньше.

4. На третьем проходе уже не надо сравнивать предпоследний и третий элемент с конца. Поэтому число сравнений будет на два меньше, чем при первом проходе.

5. В конце концов, при проходе по массиву, когда остаются только два элемента, которые надо сравнить, выполняется только одно сравнение.

6. После этого первый элемент не с чем сравнивать, и, следовательно, последний проход по массиву не нужен. Другими словами, количество проходов по массиву равно m-1, где m – это количество элементов массива.

7. Количество сравнений в каждом проходе равно m-i, где i – это номер прохода по массиву (первый, второй, третий и т.д.).

8. При обмене элементов массива обычно используется "буферная" (третья) переменная, куда временно помещается значение одного из элементов.

Пример

Возьмём массив с числами «5 1 4 2 8» и отсортируем значения по возрастанию, используя сортировку пузырьком. Выделены те элементы, которые сравниваются на данном этапе.

Первый проход:

(**5 1** 4 2 8) (**1 5** 4 2 8), Здесь алгоритм сравнивает два первых элемента и меняет их местами.

(1 **5 4** 2 8) (1 **4 5** 2 8), Меняет местами, так как 5 > 4

(1 4 **5 2** 8) (1 4 **2 5** 8), Меняет местами, так как 5 > 2

(1 4 2 **5 8**) (1 4 2 **5 8**), Теперь, ввиду того, что элементы стоят на своих местах (8 > 5), алгоритм не меняет их местами.

Второй проход:

(**1 4** 2 5 8) (**1 4** 2 5 8)

(1 **4 2** 5 8) (1 **2 4** 5 8), Меняет местами, так как 4 > 2

(1 2 **4 5** 8) (1 2 **4 5** 8)

(1 2 4 **5 8**) (1 2 4 **5 8**)

Теперь массив полностью отсортирован, но алгоритм не знает так ли это. Поэтому ему необходимо сделать полный проход и определить, что перестановок элементов не было.

Третий проход:

(**1 2** 4 5 8) (**1 2** 4 5 8)

(1 **2 4** 5 8) (1 **2 4** 5 8)

(1 2 **4 5** 8) (1 2 **4 5** 8)

(1 2 4 **5 8**) (1 2 4 **5 8**)

Теперь массив отсортирован и алгоритм может быть завершён.

Реализация на Паскале

```
const
   m = 10;
 var
   arr: array[1..m] of integer;
   i, j, k: integer;
begin
   randomize;

   write ('Исходный массив: ');
   for i := 1 to m do begin
      arr[i] := random(256);
      write (arr[i]:4);
   end;
   writeln; writeln;
    for i := 1 to m-1 do
      for j := 1 to m-i do
         if arr[j] > arr[j+1] then begin
            k := arr[j];
            arr[j] := arr[j+1];
            arr[j+1] := k
         end;
    write ('Отсортированный массив: ');
   for i := 1 to m do
      write (arr[i]:4);
    writeln;
 readln
end.
```

Сортировка выбором

Сортировка выбором – второй простейший алгоритм сортировки.

Недостатки:
Такие же недостатки, что и у сортировки пузырьком.

Достоинство:
Может быть реализован и как устойчивый и как неустойчивый.

Сложность алгоритма: $O(n^2)$.

Видео: [http://www.youtube.com/watch?v=Ns4TPTC8whw]

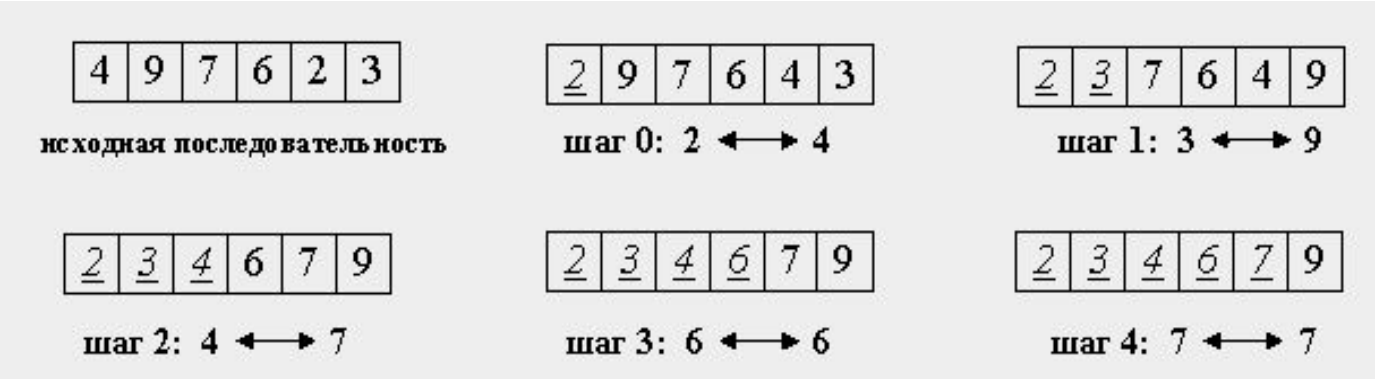
Алгоритм

Шаги алгоритма:

- находим номер минимального значения в текущем списке;
- производим обмен этого значения со значением первой неотсортированной позиции (обмен не нужен, если минимальный элемент уже находится на данной позиции);
- теперь сортируем хвост списка, исключив из рассмотрения уже отсортированные элементы.

Для реализации устойчивости алгоритма необходимо в пункте 2 минимальный элемент непосредственно вставлять в первую неотсортированную позицию, не меняя порядок остальных элементов.

Пример



Реализация на Паскале

```
for i := 1 to n - 1 do begin
    min := i;
    for j := i + 1 to n do
        if a[min] > a[j] then
            min := j;
    if min<>i then begin
        t := a[i];
        a[i] := a[min];
        a[min] := t;
    end;
end;
```

Сортировка вставками

Сортировка вставками — третий и последний из простых алгоритмов сортировки. Сначала он сортирует два первых элемента массива.

Недостатки:
Такие же недостатки, как и у сортировки пузырьком и выбором.

Достоинство:
- эффективен на небольших наборах данных, на наборах данных до десятков элементов может оказаться лучшим;
- эффективен на наборах данных, которые уже частично отсортированы;
- это устойчивый алгоритм сортировки (не меняет порядок элементов, которые уже отсортированы);
- может сортировать список по мере его получения;

Сложность алгоритма: $O(n^2)$.

Видео: [http://www.youtube.com/watch?v=ROalU379l3U]

Алгоритм

Сначала он сортирует два первых элемента массива. Затем алгоритм вставляет третий элемент в соответствующую порядку позицию по отношению к первым двум элементам. После этого он вставляет четвертый элемент в список из трех элементов.

Пример

Например, при сортировке массива **dcab** каждый проход алгоритма будет выглядеть следующим образом:

Начало	**d c a b**
Проход 1	**c d a b**
Проход 2	**a c d b**
Проход 3	**a b c d**

Реализация на Паскале

```
const N = 255;
var x : array [1..N] of integer;
  i, j, buf: integer;
begin
  for i := 2 to N do
  begin
    buf := x[i];
    j := i - 1;
    while (j >= 1) and (x[j] >
buf) do
    begin
      x[j + 1] := x[j];
      j := j - 1;
    end;
    x[j + 1] := buf;
  end;
end;
```

Домашнее задание

Создать программу, для сортировки массива.

Спасибо за внимание!