一、选择排序(排成从小到大的顺序)（不稳定，时间复杂度：最差、平均都是O(n2)）

   1.算法思路:

      ①找出数组中最小的数与第一个数进行交换;

      ②找出数组中第二小的数与第二个数进行交换

      ③依此类推，直到数组排序完毕.

  2.例子:

void SelectSort(int Arr[],int length)  
{  
   for(int m=0;m<length;m++)  
   {  
    int min=Arr[m];  
    int index=m;  
    for(int n=m+1;n<length;n++)  
    {  
      if(Arr[n]<min)  
   {   
    min=Arr[n];  
    index=n;  
   }   
 }  
 if(index!=m)  
 {  
   int temp=Arr[index];  
   Arr[index]=Arr[m];  
   Arr[m]=temp;  
 }  
   }  
}

二、冒泡排序(排成从小到大的顺序，大冒情况)(稳定，时间复杂度:最差、平均都是O(n2)，最好是O(n)（正序情况）)

1.算法思路:

  ①顺序遍历数组，相邻两个元素两两比较，大的往后交换

  ②第一遍将最大的数冒到最后一位

  ③第二遍将第二大的数冒到倒数第二位

  ④依次类推，遍历n-1遍即可排好序

2.例子:

void BubbleSort(int Arr[],int length)  
{  
   for(int m=0;m<length;m++)  
   {   
    for(int n=0;n<length-m-1;n++)  
    {  
      if(Arr[n]>Arr[n+1])  
   {   
    int temp=Arr[n];  
    Arr[n]=Arr[n+1];  
    Arr[n+1]=temp;  
   }   
 }  
   }  
}

三、直接插入排序(排成从小到大的顺序)(稳定，时间复杂度：最差、平均都是O(n2)、最好:O(n))

1.算法思路:

 ①从第二个元素开始执行第一遍“插入”算法，将当前元素与前面(已经排好序)的元素依次进行比较

 ②当前元素是从后往前比较前面已经排好序的元素的，遇到比自己大的元素时与之交换，遇到比自己小的元素或到数组起始位置时停留在原位置不动，算是“插入”结束

 ③接着第三个元素开始执行第二遍"插入"算法，依次类推，数组中所有元素都"插入"完毕也就排好序了

2.例子:

  void InsertSort(int Arr[],int length)  
{  
   for(int m=1;m<length;m++)  
   {   
    for(int n=m;n>0;n--)  
    {  
      if(Arr[n-1]>Arr[n])  
   {   
    int temp=Arr[n];  
    Arr[n]=Arr[n-1];  
    Arr[n-1]=temp;  
   }   
 }  
   }  
}

四、二分插入排序(稳定，时间复杂度：最差、平均都是O(n2)、最好:O(nlogn))

1.算法思路:

 ①.从数组第二个元素开始执行算法，将当前元素作为待插元素，折半查找待插元素在前面已经排好的序列中应该插入的位置index

 ②.将应该插入的位置index到待插元素当前位置前一位这一段元素全部后移一位

 ③.将待插元素插入到待插位置

 ④.当所有元素插入完毕时，即可完成排序。

2.例子:

void MiddleSort(int Arr[],int length)  
{  
  for(int m=1;m<length;m++)  
  {  
    int low=0;  
    int high=m-1;  
    int key=Arr[m];  
    while(low<=high)  
    {  
      int middle=(low+high)/2;  
      if(key<Arr[middle])  
           high=middle-1;  
      else   
           low=middle+1;          
 }  
   
 for(int n=m-1;n>=low;n--)  
 {  
      Arr[n+1]=Arr[n];  
 }  
 Arr[low]=key;  
  }  
}

五、希尔排序(不稳定,时间复杂度：O(nlogn))

1.算法思路：

①希尔排序本质上是直接插入排序的变种，直接插入排序是相邻元素的交换移动，这样做效率低，希尔排序进行交换的元素的间隔为dl(dl>=1)，这个dl叫做增量

②给增量一个初始值,将数组划分为dl个字数组，所有距离为dl的元素互为同一子数组，用直接插入法排序所有子数组

③变小增量值，重复上一操作

④直到增量值为1，希尔排序跟直接插入排序操作一样（只不过这时，数组有很多段是连续的），排序完毕

2.例子:

  void ShellPass(int Arr[],int length,int dl)  
{  
  for(int m=0+dl;m<length;m++)  
  {   
 for(int n=m;n>=0+dl;n=n-dl)  
 {  
      if(Arr[n-dl]>Arr[n])  
      {  
     int temp=Arr[n-dl];  
     Arr[n-dl]=Arr[n];  
     Arr[n]=temp;  
   }  
 }  
  }  
}

void int ShellSort(int Arr,int lenth,int n)  
{  
   int dl=n;  
   do{  
     dl=dl/3+1;  
     ShellPass(Arr,lenth,dl);          
   }while(dl>1);  
}

六、快速排序(不稳定,时间复杂度：最差O(n2)、平均O(nlogn))

1.算法思路：

 ①快排采用的是分治的策略，把数组中一个元素作为基数，将数组分成左右两字段，左边所有元素均小于等于该基数，右边所有元素均大于等于该基数，递归操作字段，到字段长度为1时，即可排好序。

 ②将待排序列第一项作为该次排序的基数，有两个指针left、right，初始值分别为待排序列最左端和最右端；

 ③首先right向左边移动（不越过left,这是因为此时left指向的位置要存入待转移的元素，要保证right不能超过该位置），找到一个小于基数的元素的位置，将该元素转移到left指向的位置，同时left后移一位；

 ④然后left向右边移动（不越过right,这是因为此时right指向的位置存入待转移的元素，要保证left不能超过该位置），找到一个大于基数的元素的位置，将该元素转移到right指向的位置，同时right前移一位;

 ⑤待left和right重逢时(left==right)，left所指向的位置就是该次待排序列的分割点，将基数存入left位置，left左边的元素均小于等于基数，右边均大于等于基数

 ⑥递归操作被分割出来的两个子序列序列，直到序列长度等于1时，排序结束

2.例子:  
void QuickSort(int Arr[] ,int m,int n)  
{  
   if(m==n) return;  
   int left=m,right=n;  
   int base=Arr[left];  
   while(left<right)  
   {  
     while((left<right)&&(Arr[right]>base))  
  {  
    right--;  
  }    
  if(left<right)  
  {  
    Arr[left]=Arr[right];  
    left++;  
  }  
      
  while((left<right)&&(Arr[left]<base))  
  {  
    left++;  
  }  
    
  if(left<right)  
  {  
    Arr[right]=Arr[left];  
    right--;    
  }  
   }  
   Arr[left]=base;  
   QuickSort(Arr,m,left-1);  
   QuickSort(Arr,left+1,n);  
}