 Homework 5  
令A[1..n]是一个由n个数所组成的数组。序列A[1], A[2], … , A[n]被称为是单模的(unimodal)，当且仅当存在顶点序号1≤p≤n，使得数组的元素从A[1]、A[2]开始到A[p]单调增加，而从A[p]、A[p+1]开始到A[n]则单调下降。  
问题：对于一个给定的单模序列A[1], A[2], … , A[n]，请找出其顶点序号p。设计一个求解此问题的算法并分析其最坏时间复杂性。

解：

遍历数组A[1..n]，若后一个数比前一个数大，则继续遍历，否则，停止遍历，输出结果

for (int i = 0 ; i < n - 1 ; i++){

2 if (A[i + 1] >= A[i]){

3 continue;

4 }

5 else{

6 return i;

7 break;

8 }

9 }

由单模序列的定义可知，最坏情况为从A[1]到A[n - 1]均为单调增加，从A[n - 1]到A[n]为单调下降。此时，需要比较 n - 2 次。故，最坏情况下算法复杂度为 O(n - 2)即O(n)。

Home Work 6

设有一个算法Median能在O(n)的时间内计算一个数组的中位值(即将数组的元素按大小顺序排列正好位于中间的值)。给定一个有n个元素的数组，能否以Median算法为基础设计一个算法，对任意的整数1≤i≤n，该算法在O(n)的时间内求出数组中第i大小的元素。如果能，请给出一个这样的算法并分析其最坏时间复杂性。

1.将输入的数组A[n]中的n个元素分为n/5组，每组5个元素，且至多只有一组有n%5个元素；

2.调用Median算法得到每组的中位数，所有的中位数构成数组M[1,...,n/5]；

3.再次调用Median算法求得数组M[1,...,n/5]的中位数Med；

4.根据Med将原数组A[n]进行划分，比Med小的元素全部放到Med左边，比Med大的元素全部置于Med的右侧，得到Med的位置k

5.如果i = k，则返回中位数Med，如果 i < k，对Med左边的所有元素重复算法，如果i < k，对Med所有右侧的元素重复算法

public int Select\_i (ArrayList<Integer> A, int index\_s, int index\_e, int i){

ArrayList<Integer> M = new ArrayList<Integer>();

//将数组划分为n/5组，每组5个元素，求得每组中位数，得到数组M

for (int m = 0; m < A.size()/5 ; m++){

if (m\*5 + 5 < A.size() - 1){

M.add(Median(A,m\*5, m\*5+5));

}

else{

M.add(Median(A, m\*5, A.size()));

}

}

//求得数组M的中位数Med

int Med = Median(M, 0, M.size());

//根据中位数Med对数组A进行划分，调用函数Divede，得到Med在数组A中的位置k

int k = Divede(A,Med);

//结果处理

if (i == k){

return Med;

}

else if (i < k){

return Select\_i(A, 0, k-1, i);

}

else{

return Select\_i(A, k+1, A.size() - 1, i);

}

}

复杂度为线性O(n)满足要求。