#### 动态规划

我们仔细分析，发现两个条件可以总结为：求一个最长序列，使得该序列的任意三个相邻元素，中间的元素是三个中最大的，或者最小的；

设dp[i][0]表示当前元素i是以i为中心的三个相邻的元素中最大的。

设dp[i][1]表示当前元素i是以i为中心的三个相邻的元素中最小的。

则当h[i]<h[i-1]时，若选择i，则dp[i][0]是前一个元素即i-1作为最低点时的长度即dp[i-1][1]的值+1，或不选择，则继承前一个元素即i-1作为最高点的值即dp[i-1][0]。

h[i]<h[i-1]时同理，反向操作即可。

参考代码如下所示。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | //花匠  #include<bits/stdc++.h>  using namespace std;  int h[100010],dp[100010][2];  int n;  int main()  {  scanf("%d",&n);  for(int i=1; i<=n; i++)  scanf("%d",&h[i]);  dp[1][0]=dp[1][1]=1;  for(int i=2; i<=n; i++)  {  h[i]>h[i-1]?dp[i][0]=dp[i-1][1]+1:dp[i][0]=dp[i-1][0];  h[i]<h[i-1]?dp[i][1]=dp[i-1][0]+1:dp[i][1]=dp[i-1][1];  }  cout<<max(dp[n][0],dp[n][1])<<endl;  return 0;  } |

贪心法

这道题可以看成是找到一个最长的波浪序列（每一盆花都是波峰或波谷）。

首先，对于第一盆花，不论如何都要选，因为如果不选，第二盆花就相当于第一盆，而花的总数却减少了，所以一定不会更优。

然后我们可以发现，一个连续的上升或下降序列中，**有且只有一个能被选入最终的子序列中**，不然就不满足波浪的需求，因此我们只需要统计连续的上升与下降序列的个数，最终的答案等于统计数+1。

另外，因为是要求严格波浪，所以对于相等的连续序列我们可以直接忽略，相当于只有一个点。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | //花匠  #include<bits/stdc++.h>  using namespace std;  int n,now,last,type,ans; //type记录上一个单调序列的种类,1为上升,-1为下降  int main()  {  cin>>n>>last; //last为上一个高度，即先读入第一个点  for(int i=2; i<=n; ++i)  {  cin>>now;  if(now==last) continue;  if(type!=1 && now>last)//若不满足上一个序列单调性  ++ans,type=1;  if(type!=-1 && now<last)  ++ans,type=-1;  last=now;  }  cout<<ans+1<<endl;  return 0;  } |