动态规划。最基本的想法是：枚举中间最高的一个人，接着对它的左边求最长上升序列（注意序列中最高的同学不应高过基准），对右边求最长下降序列（同样的，序列中最高的同学不应高过基准）。时间复杂度为O(n3)，算法实现起来也很简单。

接着对这个算法进行分析，我们不难发现，假如还是基于枚举一个同学的话，设Incsq[i]表示了1 - i的最长上升序列，Decsq[i]表示了i - n的最长下降序列，那么，

Current[i] = Incsq[i] + Decsq[i] - 1（两个数组中i被重复计算了）

那么，我们只需要先求好最长上升和下降序列，然后枚举中间最高的同学就可以了。

算法还可进一步优化：

求最长上升序列的经典状态转移方程为：

opt[i] = max{opt[j]+1, 其中i<j<=n, 且list[j]>list[i]}

我们对状态转移方程稍微做一些修改：

opt[i] = max{opt[i+1], min{j, rec[j]>=list[i]}}

rec[j] = list[i]

很明显可以看出，在opt[i]的寻找j的过程当中，查询序列是单调的，于是可以用二分法，就十分巧妙地在logn的时间内找到指定的j，而问题的总体复杂度为O(nlogn)。这样，这个问题的算法效率就得到了大幅度的提升，即便n是106，也可以轻松应对。

参考程序如下所示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | //合唱团  #include<stdio.h>  #include<stdlib.h>  int T[201];  int f[201],g[201];  int main()  {  freopen("chorus.in","r",stdin);  freopen("chorus.out","w",stdout);  int n,i,j,min,k,R,L;  scanf("%d",&n);  for(i=1;i<=n;i++)  scanf("%d",&T[i]);  for(i=1;i<=n;i++)  {  f[i]=1;  for(j=1;j<=i-1;j++)  if(T[j]<T[i]&&f[j]+1>f[i])  f[i]=f[j]+1;  }  for(i=n;i>=1;i--)  {  g[i]=1;  for(j=i+1;j<=n;j++)  if(T[j]<T[i]&&g[j]+1>g[i])  g[i]=g[j]+1;  }  min=n;  for(k=1;k<=n;k++)  {  L=f[k],R=g[k];  if(n-L-R+1<min)  min=n-L-R+1;  }  printf("%d\n",min);  } |