1、旅行

很容易想到 2^n 的算法。直接枚举每个人是否在船上,计算取最大值，可以拿到 30%的分数。

很明显，直接枚举是没有必要的。我们可以枚举人数 K，然后算出在这种情况下每个人的快乐程度。然后贪心地取前 K 大的数，这样就得到了 $n^2\log n$ 的算法。

2、数据

方法一、

设 f(i)表示要将前 i 个数改得合法需要的最少步数。则

f(i)=min{f(j)+|A(j+1)-(i-j-1)|} 其中 1<=j<i

边界 f(1)=A(1)

时间复杂度为 O(n^2)

分析方程继续优化：

方程变形为：f(i)=min{f(j)+|(A(j+1)+j+1)-i|} 其中 1<=j<i

我们将决策分成两类，即：

◆第一类决策：当 i<=(A(j+1)+j+1)时，f(i)=min{ f(j) + (A(j+1)+j+1) } - i

◆第二类决策：当 i> (A(j+1)+j+1)时，f(i)=min{ f(j) - (A(j+1)+j+1) } + i

我们将两类决策分开计算。

随着 i 的增加，会有一些决策从第一类转到第二类。

我们用两个二叉堆维护两类决策，用对于第一类决策用 f(j)+A(j+1)+j+1 作为关键字，第二类决策用 f(j)-(A(j+1)+j+1)作为关键字。

二叉堆可以进行的操作有：插入元素、删除元素、取最值。

时间复杂度 O(nlogn)

方法二、

设 f(i)表示要将 a[i]..a[n]改合法需要的最少步数。则：

f(i) = min { f(j) + |(A[i]+i+1)-j)| } 其中 i<j<=n

同理可以讨论两类决策：

◆第一类决策：当 j<=(A(i)+i+1)时， f(i)=min{ f(j) - j } + A[i]+i+1

◆第二类决策：当 j>(A(i)+i+1)时，f(i)=min{ f(j) + j } - A[i]+i+1

此时可以用线段树维护两类决策：

对于第一类：查询区间[i+1, min(A(i)+i+1,n)]最小的 f(j)-j 的值

对于第二类：查询区间[A[i]+i+1, n]最小的 f(j)+j 的值

时间复杂度 O(nlogn)

3、业务

令：

ans[x][y]=从 x 到 y 的最小过路费！计算 ans[x][y]可以用 DIJKstra 算法或 SPFA 算法！

令：

dist[i][0]=从 k 出发到 i 点经过的点费用最大值为 a[k]的情况的最小边费用和

dist[i][1]=从 k 出发到 i 点经过的点费用次大值为 a[k]的情况的最小边费用和

于是:

ans[x][y]=min{dist[x][0]+dist[y][1],dist[x][1]+dist[y][0]}+a[k]}其中 1<=k<=n

计算出 ans[x][y]后，回答询问就很简单了。

时间复杂度 $O(n^2\log m+k)$ (DIJKstra 堆优化后的时间复杂度)