## subtask1

暴力枚举每个人属于哪个楼梯。

复杂度 $O(2^n)$。

## subtask2

令 $pre_i$ 表示前 $i$ 个的最大值。显然两个序列必定有一个当前最大值为 $pre_i$，不妨设 $f(i,j)$ 表示前 $i$ 个元素划分完，第二个序列的最大值为 $j$ 的最小代价。

考虑如何转移，假如 $A_i = pre_i$，那么显然要放入第一个序列中，答案直接加上 $A_i$ 即可。

否则我们考虑 $A_i$ 是否成为第二个序列的最大值。如果不成为的话，那么对于 $j \in [A_i, pre_i]$ ，显然可以放入第二个序列，即 $f(i,j) = f(i-1,j) + j$，对于 $j \in [1, A_i)$，显然必须放入第一个序列，即 $f(i,j) = f(i-1,j) + pre_i$。

最后一种情况就是成为第二个序列的最大值的情况，即 $f(i,A_i) = \min\limits_{j \le A_i} f(i-1,j) + A_i$。

## subtask23

直接暴力 dp，复杂度 $O(n * pre_n)$。

## subtask4

注意到我们其实要维护的操作有，区间加常数，区间加下标，单点修改，区间查最小值。

考虑到区间加下标这个操作，我们可以使用分块凸包来维护。将序列分为 $\sqrt{n}$ 块，每块根据 $(j, f[j])$ 维护一个凸包。注意到下标较大的数字，如果已经不是最优了，那么永远不是最优了，所以每次询问的时候只要暴力把前面的弹出就好了，并不需要在凸包上面二分。每次修改过后，把零散块涉及到的部分全部重构就好了。

时间复杂度 $O(n\sqrt{n})$。

## subtask5

之前做法的瓶颈在于，修改的边界处会打乱凸包的维护，因此需要暴力重构，只能选择分块维护。

考虑一个更加简单粗暴的做法。我们直接维护维护一个单调栈，一旦当我们发现单调栈中存在元素比下一个元素更优的时候，就把下一个元素弹出。

注意到区间加常数不影响这段区间内部的单调栈，区间加下标只会减少这段区间内部的单调栈的大小，而问题就出在边界处可能会多出 $O(n)$ 级别的新的元素出来。

重新考虑这题的修改操作具体是怎么样的，查询的是前缀的最小值，每次修改操作相当于前缀加一个常数，然后紧跟着一个单点修改，然后后缀加下标。

## subtask5

这个过程的分界点就发生在 $A_i$，而左侧的加常数可能导致右侧多出来新的单调栈元素。

但是我们考虑一个已经被弹出的元素 $k$ 满足 $A_i < k$。假设存在一个 $j$ 满足 $A_i < j < k$，且 $f(i-1,j) < f(i-1,k)$，那么在这一次操作过后，依然满足 $f(i,j) < f(i-1,k)$。

如果不存在，那么必定存在一个 $j$ 满足 $j < A_i < k$ 满足 $f(i-1,j) < f(i-1,k)$，注意到 $f(i,A_i) \le f(i-1,j) + A_i \le f(i-1,k) + k$，所以这次操作过后，$k$ 依然不可能存在于单调栈中。

所以每次操作后，有可能加入单调栈的只有 $A_i$ 一个元素，我们只要用一个线段树维护每个位置还需要多少次操作就能小于下一个元素，用一个 set 维护单调栈即可。

时间复杂度 $O(n \log n)$。