Solution

--for NOIp Day 1

Pb

我们的目的是要把 n 个区域分成 w 块，使得同时访问的代价最小。

我们可以通过一定的数学分析发现，估价越大的区域越应该先访问。

所以首先按照估价大小排序，问题转化成了把 n 个区域划出(w-1)个间隔，使得其成为 w 块。

尝试 DP.$f[i][j]$表示前 i 个区域分成 j 块的最小期望。

有 DP 方程：$f[i][j] = max\{f[k][j-1] + i \times (\sum_{l=k+1}^{i} p[l])\}$，其中 p[i]为第 i 个区域的存在概率，可以通过计算前缀和做到 O(1)出求和。总复杂$O(n^3)$。

Ball

首先来推导当球 i 与求 j 外切时，球半径的计算公式：

容易知道，此时球 i 的坐标为(x_i, r_i)

$$\therefore (x_i - x_j)^2 + (r_i - r_j)^2 = (r_i + r_j)^2$$

$$\therefore (x_i - x_j)^2 = 4r_i r_j$$

$$\therefore r_i = \frac{(x_i - x_j)^2}{4r_j}$$

考虑维护一个单调栈，栈内元素为各个已经确定圆大小的圆的圆心坐标。当新加一个元素时，当栈顶坐标的 x 坐标和 r 均小于小于当前点，栈顶元素退栈（显然对于后面的元素，一定会先碰到这个圆再碰到栈顶存的圆）。

如何确定当前的的圆心坐标(即圆的半径)？

考虑当前点 i，对其与栈顶元素求可外切的圆半径。若这个半径大于栈顶元素半径，那么栈顶退栈，直到与一个满足条件的圆相切或栈为空为止。因为如果当所求半径大于栈顶元素半径，有两种可能，第一种：与栈顶元素下方的元素没有相交。此时这是最大值，第二种是与栈顶元素下方元素相交。当前要求圆半径的这个圆一定是先与栈顶元素下方的元素外切再与栈顶元素外切，所以也可以去掉。

GFW

问题可抽象为：给一个长度为 n 的序列，删除一段连续的序列（或者不删），使得位置连续的递增序列长度最长。

由于给的递增序列的值很大，现对其离散化。

用 O(n)的方法依次找出原序列中所有的连续递增序列是很简单的是吧（我压了一个栈）。

对于一段连续递增序列，他可能对答案造成如下影响:

1. 这段序列的某一段前缀是答案的前半部分
2. 这段序列得某一段后缀是答案的后半部分

所以我们可以开一个以答案前半部分的最后一个数的值为关键字，前半部分的序列长度为值的线段树或树状数组。维护区间最大值。对于每一段序列的每一个后缀，在线段树中查询关键字比这一个后缀的第一个元素值小的最长序列长度值，和答案比较，更新答案。对于这段序列的所有前缀，全部加入线段树中更新线段树。由于每一个数只可能以前缀方式加入一次线段树，以后缀形式查询一次线 段树。故总复杂度为$O(nlog_n)$