【题目大意】

维护两个数组，A[1~N]，B[1~N]，支持三种类型的操作：

1、$A[i] = \max\{A[i], (i - l) * a + b\}$

2、$B[i] += (i - l) * a + b$

3、输出 $A[i] + B[i]$

总共 Q 个操作。

【解题报告】

因为我们只需要输出 $A[i] + B[i]$，且操作间不会互相影响，所以我们可以分开来处理 A[]和 B[]两数组。

一、先考虑 B[]这边，要简单许多。

首先，我们可以把操作 2 改写成：

$B[i] += i * a + l * a + b$

亦可以写成：

$B[i] += i * a' + b'$

此外，显然有：

$(i * a + b) + (i * c + d) = i * (a + c) + (b + d)$

因此，我们可以维护一棵线段树。对于每个代表区间[l,r]的结点，我们记录 a 的和以及 b 的和（作用于整个区间）。标记不需要下传，当询问 B[i]时，我们从根[1,N]走到[i,i]并合并答案即可。

所有与 B[]相关的操作均可在 O(logn)时间内完成。

二、考虑 A[]这边，略显棘手，不过也很容易理解。

对于每个代表区间[l,r]的结点，我们记录一对数 a 和 b，代表对于任意 i（1<= i <= r），A[i]应当比 i * a + b 小。当新的一对 a'、b'操作于该结点时，由于这两个以 i 为变量的函数均为线性，因而最多具有一个交点。这是一个非常重要且美妙的性质——它表明只有以下两种情形：

1、a'、b'完全比 a、b 优或劣。意即，对于任意 i（1 <= i <=r），a * i + b <= a' * i + b'，或对于任意 i（1 <= i <=r），a * i + b >= a' * i + b'。

2、令 m = (l + r) / 2，在[l,m]或[m+1,r]区间中满足情形 1。

因此，当遇到情形 2 时，我们可以递归的更新。又因为一个区间最多对应 O(logN)个结点，所以一次更新操作复杂度不超过 $O(\log^2 N)$。对于一次询问，我们依然可以从根[1,N]走到[i,i]，对路径上每组 a、b 形成的函数，代入自变量 i，取函数值的最大值。故查询可以在 O(logn)时间中完成。

综上，此题可在 O(QlogN)到 $O(Q\log^2 N)$的时间范围内完成。由于 N 的规模较大，我们不能建出整棵线段树。因此我们需要使用动态开结点的方法，对每次操作最多新开 logN 个结点，所以空间复杂度为 O(QlogN)。