上帝之手 解题报告

杭州第二中学 陈思禹

1 试题来源

2015年集训队互测

2 试题大意

2.1 题目描述

上帝之手操纵着四维空间。假设四维空间中上帝关心的部分共N天，定义第i天结束时一个三维世界的混乱度为X_i。由于一些自然的原因，第i天该世界的混乱度会增加D_i，即$X_i = X_{i-1} + D_i$。但为了世界的平衡，每天该世界都有一个混乱度的上限值L_i，即实际上$X_i = min(X_{i-1} + D_i, L_i)$。

上帝想对该四维空间作一系列测试，于是希望你帮忙建立一个模型。具体有以下三种测试：

（1）给定A、B和K，要求找到一个C（$A \leq C \leq B$），使得世界以L_{C-1}的初始混乱度从第C天开始发展将得到第K大的X_B。你只需告诉上帝第K大的X_B值即可。

（2）给定A、B和X_0，要求找到一个C（$A \leq C \leq B$），使得世界以X_0的初始混乱度从第C天开始发展将得到最大的X_B。你只需告诉上帝最大的X_B值即可（注意：X_0可能大于L_{C-1}）。

（3）给定A 和B，要求对于所有满足$A \leq C \leq B$的C，求出世界以L_{C-1}的初始混乱度从第C天开始发展将得到的X_B 的不同值的种数。

当然，上帝还会根据测试的结果修改某些位置的L_i。请你帮助上帝完成测试。

2.2 输入格式

第一行包含两个正整数N和M，按序表示总天数和总操作次数。

第二行为N个非负整数$D_1 - D_N$，第三行为$N+1$个非负整数$L_0 - L_N$。含义见题目描述。

第四行起的M行，每行第一个整数$type$表示操作种类。若$type = 0$，则后面跟有两个整数u和x，表示将L_u改为x。若$type > 0$，则$type$等于题目描述中对应的测试种类编号。$type = 1$时后面跟有三个整数A、B和K；$type = 2$时后面跟有三个整数A、B和X_0；$type = 3$时后面跟有两个整数A和B。具体含义见题目描述。

2.3 数据规模

对于10%的数据，$N, M \le 100$；

对于20%的数据，$N, M \le 5000$；

对于另10%的数据，$type \le 1$；

对于另20%的数据，$type \le 2$；

对于另15%的数据，$type = 0$或3；

对于100%的数据，$N \le 10^5$，$M \le 2 \times 10^5$，$0 \le D_i \le 10^4$，$0 \le L_i, X_0, x \le 10^9$，$0 \le u \le N$，$1 \le A \le B \le N$，$1 \le K \le B - A + 1$。

3 算法介绍

3.1 算法1

考虑10%的数据，$N, M \le 100$。我们可以考虑模拟。

如果给定了C，那么从第C天开始发展最终得到的X_B显然可以通过代入公式$X_i = min(X_{i-1} + D_i, L_i)$方便地以$O(N)$ 的时间复杂度模拟出。

这样，对于任意一个测试，我们直接枚举C，分别求出最后的X_B，对于单个询问时间复杂度为$O(N^2)$。对于第一种测试，可以通过排序得到第K 大；对于第二种测试在求X_B时记录下最大值即可；对于第三种测试，可以排序后枚举一遍统计出不同的个数。至于修改L_i，只需在L数组中修改即可。

容易看出，该算法时间复杂度为$O(MN^2)$。

期望得分：10

3.2 算法2

考虑20%的数据，$N, M \leq 5000$。

我们发现上一个算法的瓶颈在于求X_B。我们考虑能否不每次都套用题目描述中给出的公式。

我们先将公式写两次：

$$X_i = min\left(X_{i-1} + D_i, L_i\right), \tag{1}$$
$$X_{i-1} = min\left(X_{i-2} + D_{i-1}, L_{i-1}\right). \tag{2}$$

据此可得到$X_i = min\left(X_{i-2} + D_{i-1} + D_i, L_{i-1} + D_i, L_i\right)$。

令$S_{i,j} = \sum_{k=i}^{j} D_k$（为了方便，$S_{i+1,i} = 0$），$T_{i,j} = L_{i-1} + S_{i,j}$，则：

$$X_i = min(X_{i-2} + S_{i-1,i}, T_{i,i}, T_{i+1,i})$$

依此类推，可以得到以下规律：

Theorem 1.

$$X_B = min(X_{C-1} + S_{C,B}, min\{T_{i,B} \mid C < i \leq B+1\})$$

$S_{i,j}$和$T_{i,j}$都可以通过预处理D数组的后缀和$O(1)$求出。而且我们发现$min\{T_{i,B} \mid C < i \leq B+1\}$在枚举$C$时每次只变化一项。因此我们只要从后向前枚举$C$就可以对于每个$C$ $O(1)$求出X_B了。

结合算法1的想法，时间复杂度降为$O(mnlogn)$。

期望得分：20

3.3 算法3

注意到有10%的数据仅有第一种测试。

不妨令$X_{C,B} = min(X_{C-1} + S_{C,B}, min\{T_{i,B} \mid C < i \leq B+1\})$。对于第一种测试，$X_{C-1} = L_{C-1}$，即$X_{C,B} = min\{T_{i,B} \mid C \leq i \leq B+1\}$。

容易观察出，$X_{C,B}$随C的减小不增，所以使得$X_{C,B}$第K大的$C = B - K + 1$。

这时问题转化为单点修改和快速询问$X_{B-K+1,B} = min\{T_{i,B} \mid B-K+1 \leq i \leq B+1\}$，即区间最小值。显然使用线段树即可。时间复杂度为$O(mlogn)$。

期望得分：20（结合算法2）+10=30

3.4 算法4

对于接下来的20%数据，需要分析第二种测试。

我们可以从函数的角度来思考这个问题。令$f(C) = X_{C-1} + S_{C,B} = X_0 + S_{C,B}$，$g(C) = min\{T_{i,B} \mid C < i \le B + 1\}$。

由于$D_i \ge 0$，所以$S_{C,B}$随C的减小不减，即$f(C)$随C的减小不减。而由算法3可知$g(C)$随C的减小不增。而我们要求的是这两个函数在区间$[A, B]$上较小值的最大值。

结合图形我们很容易发现：

Theorem 2. *若$f(C)$和$g(C)$两函数图象无交点，则最大值在边界上取到；否则该最大值在交点附近取到。*

对于有交点的情况，由于都是单调的函数，我们可以用二分快速找到交点位置。鉴于单次求$g(C)$与第一种测试类似，需要$O(logn)$的时间复杂度，找到交点的时间复杂度为$O(log^2n)$。求出交点后取其附近的C求出$f(C)$和$g(C)$较小值的最大值即可。

至此，对于第二种测试我们得到了一个单次时间复杂度$O(log^2n)$的算法。总复杂度$O(mlog^2n)$。

期望得分：30（结合算法3）+20=50

3.5 算法5

最后50%的分数均涉及到第三种测试。

第三种测试与第一种测试的计算方式类似，$X_{C,B} = min\{T_{i,B} \mid C \le i \le B + 1\}$。显然$X_{C,B} < X_{C+1,B} \iff T_{C,B} < min\{T_{i,B} \mid C < i \le B + 1\}$。只要能统计出满足$X_{C,B} < X_{C+1,B}$的位置数，也就是$X_{C,B}$减小的次数，就能得到所要求的$X_{C,B}$的种数。

从上面的分析可以得出，问题其实已转化为：

定义p表示当前位置，v表示当前最小值。初始$p = B$，$v = min(T_{B,B}, T_{B+1,B})$。每次找到$i = max\{x \mid x < p且T_{x,B} < v\}$，然后令$p = i$，$v = T_{p,B}$，递归找下去，直至$p < A$。求递归的次数，或者形象地说每次从$p$跳到$i$，求跳的步数。

首先这里有一个简单的规律：

Theorem 3. 如果从一个区间右端开始跳，要么最后跳到该区间的最小值处，要么不会跳到该区间的任何点。

利用这个规律就可以用线段树解决这个问题。对于线段树的节点o（假设对应的区间$[l, r]$，左子节点为$left_o$，右子节点为$right_o$），定义$query(o, qv)$表示初始令$p = r + 1$，$v = qv$时能跳的步数，令$step_o = query(o, +\infty)$，$minT_o$表示线段树维护的区间最小值。则合并左右节点信息的时候$step_o = step_{right_o} + query(left_o, minT_{right_o})$。询问的时候将询问区间$[A, B]$按线段树分成至多$logn$个区间依次调用$query$即可。

现在剩下的问题就是如何实现$query$函数。如果$minT_{right_o} \geq qv$，说明不会跳到$right_o$对应的区间，$query(o, qv) = query(left_o, qv)$；否则，经过$right_o$后会停在其最小值处。这时我们注意到这里的情形和求$step_o$时情形完全一致，剩下要跳的步数正是$step_o - step_{left_o}$。所以在这种情况下$query(o, qv) = query(right_o, qv) + step_o - step_{left_o}$。容易看出单次$query$的时间复杂度是$O(logn)$的。这样单次修改或询问的时间复杂度为$O(log^2 n)$。

用类似方法解决的题目还有中国国家集训队清华集训2012-2013的楼房重建等。

至此此题已完整解决。总时间复杂度$O(mlog^2 n)$。

期望得分：15或100（结合算法4）

4 总结

本题难点主要在于分析问题的一些性质。预计平均得分将会在50以上。