《osu!》解题报告

南京外国语学校 乔明达

1 题目大意

对于一个01串，将其中的0删去，得到m个全是1的子串，设它们的长度分别为$L_1, L_2, \ldots, L_m$。则定义这个01串的分值为$\sum_{i=1}^{m}\left({L_i}^2 + L_i + 1\right)$。

已知一个长度为N的数列$a_1, a_2, \ldots, a_N$，对于$i = 1, 2, \ldots, N$，有$0 \le a_i \le 1$。

给出M个对这个数列的修改或询问。修改操作有两个参数x和$y(1 \le x \le N, 0 \le y \le 1)$，表示将$a_x$赋为$y$。询问操作有两个参数$x$和$y(1 \le x \le y \le N)$，表示：随机产生一个长度为$y - x + 1$的01串，其中第1位是1的概率为$a_x$，第2位是1的概率为$a_{x+1}$，……，第$y - x + 1$位是1的概率为$a_y$。输出这个01串的分值的期望。

2 数据规模和约定

20%的数据满足$N, M \le 5000$；

60%的数据满足$N, M \le 50000$；

100%的数据满足$N, M \le 500000$。

3 分析

我们不妨先不考虑修改操作和区间查询，只考虑计算数列$a_1, a_2, \ldots, a_N$的期望分值。

观察分值的计算式$\sum_{i=1}^{m}\left({L_i}^2 + L_i + 1\right)$。

不妨将其分为三部分分别计算期望：$\sum {L_i}^2$，$\sum L_i$和$\sum 1$。为了方便我们把这三部分分别记为S_2，S_1和S_0。记01串第i位为b_i，则$b_i = 1$的概率为a_i，$b_i = 0$的概率为$1 - a_i$。

4 $E[S_1]$的计算

不难发现，最终的01串中每一个1对S_1的贡献恰好为1，因此有：

$$E[S_1] = E\left[\sum_{i=1}^{N} b_i\right] = \sum_{i=1}^{N} E[b_i] = \sum_{i=1}^{N} a_i$$

计算上式的时间复杂度为$\Theta(N)$。

5 $E[S_0]$的计算

S_0的实际意义是01串中极大的由1构成的子串的数目。我们关注每个这样的子串的起始位置，这样的位置和子串是一一对应的，因此我们只要统计有多少位是子串的起始位置。

考虑第i位，它是“起始位置”当且仅当第i位是1且第$i-1$位是0。特别地，我们规定第0位一定是0，也就是说$a_0 = 0$。因此第i位是起始位置的概率为$a_i(1 - a_{i-1})$。

下面我们要求出“起始位置”的期望个数。定义N个随机变量$X_1, X_2, \ldots, X_N$，当第i位是“起始位置”时$X_i = 1$，否则$X_i = 0$。这样$E[X_i] = a_i(1 - a_{i-1})$那么不难得出下式：

$$E[S_0] = E\left[\sum_{i=1}^{N} X_i\right] = \sum_{i=1}^{N} E[X_i] = \sum_{i=1}^{N} a_i(1 - a_{i-1})$$

计算上式的时间复杂度为$\Theta(N)$。

6 $E[S_2]$的计算

由于出现了平方，计算S_2的期望并没有计算S_0和S_1那么简单，必须想办法将其转化为容易计算期望的形式。

定义以下两个数列：$Len_0, Len_1, \ldots, Len_N$以及$Sum_0, Sum_1, \ldots, Sum_N$。$Len_i$表示$b_1, b_2, \ldots, b_i$构成的01串最长的全是1的后缀。特别地，当$b_i = 0$时$Len_i = 0$，并且$Len_0 = 0$。$Sum_i$表示$b_1, b_2, \ldots, b_i$构成的01串$S_2$的期望值。特别地，$Sum_0 = 0$。我们希望求的值是$Sum_N$。

考虑如何利用Len_{i-1}和Sum_{i-1}推出Len_i和Sum_i。

第i位有a_i的概率是1，这样后缀长度就变为$Len_{i-1}+1$；第i位有$1-a_i$的概率是0，这样后缀长度将变为0。因此有以下等式：

$$Len_i = a_i(Len_{i-1}+1)$$

如果第i位是0，那么期望的得分不会改变。下面考虑在第i位是1的情况下期望得分的变化量。如果后缀长度由x增加到$x+1$，那么得分的变化量$\Delta = (x+1)^2 - x^2 = 2x+1$，是关于$x$的线性函数。由于期望具有线性性质，$E[\Delta] = E[2x+1] = 2E[x]+1 = 2Len_{i-1}+1$。因此，期望得分有$a_i$的概率增加$2Len_{i-1}+1$，有$1-a_i$的概率不变化。不难得出：

$$Sum_i = Sum_{i-1} + a_i(2Len_{i-1}+1)$$

结合以上两个递推式，求$E[S_2]$的时间复杂度为$\Theta(N)$。

7 处理修改和区间询问

综上，我们得到了一个时间复杂度为$\Theta(N)$的算法来计算一个长度为N的数列的期望分值。不过如果直接计算，整个算法的时间复杂度为$\Theta(MN)$，不能通过本题。我们需要更高效地处理修改操作和区间询问。

线段树是处理序列询问的工具之一，如果要使用线段树处理本题，必须满足以下几个条件：

1. 我们要设计出若干个参数描述一个区间的属性，并能通过这些参数计算出这个区间相应的答案（在本题中即为S_0，S_1和S_2）

2. 对于长度为1的区间，我们要能够快速地得到相应参数

3. 如果一个区间被分为两部分，并且这两部分的所有参数已知，我们要能够快速地得到原区间的所有参数

下面我们分别对S_0，S_1和S_2解决这三个问题。

8 维护S_0

我们尝试直接把S_0作为区间的一个参数。如果一个区间只包含a_i，那么根据定义$S_0 = a_i$。

考虑区间的合并，我们不妨考虑一个实例：L是一个包含a,b,c的区间，R是一个包含d,e,f的区间，a,b,c,d,e,f代表概率。我们将区间L和R的S_0值分别记为$L.S_0$和$R.S_0$。

那么$L.S_0 = a + b(1-a) + c(1-b)$，$R.S_0 = d + e(1-d) + f(1-e)$。

合并后S_0应该等于$a + b(1-a) + c(1-b) + d(1-c) + e(1-d) + f(1-e)$，和$L.S_0 + R.S_0$相比，减去了$d$，同时加上了$d(1-c)$，相当于减去了$cd$。

经过对更加一般的例子的验证，假如L的最后一个元素为x，R的第一个元素为y，那么$S_0 = L.S_0 + R.S_0 - xy$。

9 维护S_1

对S_1的维护是最简单的，因为S_1等于区间内所有a_i的和，所以我们直接把S_1当作区间的一个参数。如果一个区间只包含a_i，那么$S_1 = a_i$。考虑一个由两个区间L和R拼接而成的区间，有$S_1 = L.S_1 + R.S_1$。

10 维护S_2

由于计算S_2时使用了递推式，所以维护S_2是比较困难的。我们观察之前写出的递推式：

$$Len_i = a_i(Len_{i-1} + 1) = a_i Len_{i-1} + a_i$$

$$Sum_i = Sum_{i-1} + a_i(2Len_{i-1} + 1) = 2a_i Len_{i-1} + Sum_{i-1} + a_i$$

如果把a_i视为常数，那么以上是对行向量$(Len_{i-1}, Sum_{i-1}, 1)$的线性变换，我们不难构造出矩阵：

$$\begin{pmatrix} a_i & 2a_i & 0 \\ 0 & 1 & 0 \\ a_i & a_i & 1 \end{pmatrix}$$

使得：

$$(Len_{i-1}, Sum_{i-1}, 1)\begin{pmatrix} a_i & 2a_i & 0 \\ 0 & 1 & 0 \\ a_i & a_i & 1 \end{pmatrix}$$

$$= (a_i Len_{i-1} + a_i, 2a_i Len_{i-1} + Sum_{i-1} + a_i, 1)$$

$$= (Len_i, Sum_i, 1)$$

这样一来，我们只要用初始的行向量$(0,0,1)$依次右乘若干个3×3的矩阵，就能得到最终的S_2。

这样转化的意义在于，矩阵乘法是具有结合律的，我们在每个区间上记录相应的矩阵。一个只包含a_i的区间对应的矩阵就是上面写出的矩阵。合并区间时直接将这两个矩阵相乘即可。

11 优化

综合以上三点，我们需要对每个区间记录以下信息：S_0，S_1，一个3×3的矩阵以及左右端点等区间基础信息。合并区间时，计算量最大的是矩阵乘法这一步，需要$3^3=27$次乘法运算和若干次加法。如果将合并的时间其视为一个常数，那么利用线段树，时间复杂度为$\Theta(N+M\log N)$。

上述算法最大的缺陷在于巨大的常数因子，我们应该试图在矩阵乘法处寻求减小常数的方法。观察我们给出的矩阵：

$$\begin{pmatrix} a_i & 2a_i & 0 \\ 0 & 1 & 0 \\ a_i & a_i & 1 \end{pmatrix}$$

可以发现其中只有4个位置不是常数，进一步可以发现：

$$\begin{pmatrix} a & b & 0 \\ 0 & 1 & 0 \\ c & d & 1 \end{pmatrix}\begin{pmatrix} e & f & 0 \\ 0 & 1 & 0 \\ g & h & 1 \end{pmatrix}=\begin{pmatrix} ae & af+b & 0 \\ 0 & 1 & 0 \\ ce+g & cf+d+h & 1 \end{pmatrix}$$

令人惊讶的是，两个矩阵相乘之后，矩阵中不是常数的元素仍然只有4个！如果我们在区间上只保存这四个值，那么合并区间的时候只需要进行4次乘法和4次加法，明显地减小了计算量。

加上上述优化之后，虽然算法的时间复杂度仍为$\Theta(N+M\log N)$，但是常数明显减小，能够通过所有数据。