2017 年全国青少年信息学奥林匹克

浙江省队选拔赛第二试题解

竞赛时间：4 月 28 日 8:00 – 13:00

题目名称	汉诺塔	线段树	字符串
目录	hanoi	segment	string
可执行文件名	hanoi	segment	string
输入文件名	hanoi.in	segment.in	string.in
输出文件名	hanoi.out	segment.out	string.out
每个测试点时限	3s	2s	3s
内存限制	512MB	512MB	512MB
测试点数目	10	10	10
每个测试点分值	10	10	10
是否有部分分	否	否	否
题目类型	传统型	传统型	传统型
是否有附加文件	是	是	是

提交源程序必须加后缀

对于 C++ 语言	hanoi.cpp	segment.cpp	string.cpp
对于 C 语言	hanoi.c	segment.c	string.c
对于 Pascal 语言	hanoi.pas	segment.pas	string.pas

编译开关

对于 C++ 语言	-O2 -lm	-O2 -lm	-O2 -lm
对于 C 语言	-O2 -lm	-O2 -lm	-O2 -lm
对于 Pascal 语言	-O2	-O2	-O2

1 汉诺塔

这儿用 m 表示相同大小的盘子数。

如果不对最终状态进行限制，那么最优解显然和普通的汉诺塔一样，每一层都是分三步：把上面一些小的移到中间，把大小相同的 m 个移动到最后，把中间小的移动到最后。可以发现在这样的过程中，每一步所有盘子的相对顺序都会反过来。**我们把这样的移动称为初始的**。

现在对终止状态进行了限制，这就意味着每一层在某几次移动的时候与上述表现出来的行为不同。我们可以把每一次的移动看成对当前的盘子的相对顺序进行置换。即对于第 i 层，假设它被移动了 K 次，那么我们要做的是找到一些置换 $p_1, p_2, ..., p_K$，是的他们乘起来等于我们需要的排列。

在汉诺塔中这个 K 可能很大（2^n 级别），但是不难发现其实大部分移动都是初始的，至多只有 $m!$ 个会发生变化（因为排列总共只有 $m!$ 种，超过这个数一定会出现重复状态，不妨直接把重复状态去掉。）

我们的思路就是大部分情况下使用初始的移动，同时 DP 发生变化的部分。

现在考虑底层的移动对高层的影响，在假设高层都是初始的移动的情况下，可以发现每一层对高的层的影响只停留在高的层移动的奇偶性（奇数的时候再初始移动下是倒置的），更进一步的说，第 i 层只对第 $i+1$ 层有影响（更高层在初始移动的情况下影响的步数都是偶数。）同时因为我们的 DP 是对初始的移动进行修改，因此我们还需要记录 $i+1$ 层总共移动了几步，步数不需要确切记下来，只要对 $m!$ 取 min 就可以了。

因此我们的 DP 状态是 $f[i][t][b]$ 表示当前 DP 到第 i 层，第 i 层移动了 t 次，以及移动次数的奇偶性是 b（当然后面两个可以直接合在一起）。在转移的时候我们需要处理这样的值 $w[i][t][p][t']$ 表示在第 i 层用 t 步移动出排列 p 同时上一层的移动次数增加了 t' 的最小步数。接下来我们考虑怎么处理这个值。

首先我们需要计算 $g[i][p][t']$ 表示在第 i 层仅修改一次初始的移动得到置换 p，同时上一层的移动次数增加了 t' 的最小步数。这一步中我们只需要考虑当前层的盘子以及上面那一堆在哪里就可以了，状态数 S 只有不到 2000 种，把转移图建出来之后枚举 i 就跑分层的最短路就可以了，时间复杂度约为 $O(m!nS)$。

接着从 g 得到 w 就是一次简单的背包，时间复杂度约为 $O(n(m!)^4)$。

最后由 w DP 出 f 就根据定义就可以了，稍加优化时间复杂度可以降到 $O(m!Tn)$。

当然因为转移图非常特殊，可能不用像我的做法一样记录那么多东西也是对的，但是那样就有点玄学了，所以还是就这么出了。

2 线段树

我们把线段树节点分为两类，分别为左儿子节点（左节点）和右儿子节点（右节点）。观察 $S_{[l,r]}$ 的性质，它一定是连续的一段右节点接上连续的一段左节点。

不难发现对于一个右节点，它的后继右节点是唯一确定的同样对于一个左节点，它的前驱左节点也是唯一确定的。这样的话所有左节点构成了一棵树结构（左树），所有右节点也构成了一棵树结构（右树）。

通过拆分每组询问，可以转化为询问左（右）树上某个点的所有祖先（包括自己）在原树上和 u 的距离和。为了简化问题，首先假设 u 也是左节点。

现在考虑如何用左树上的信息和尽可能少的原树信息求左节点 u 和 v 在原树上的距离 w。设 D_i 为 i 在原树中的深度，d_i 为 i 在左树中的深度。接下来的叙述中，如果不特殊指明，都是指左树上的关系。不妨设 $d_u \geq d_v$。

1) 若 $u = v$，则 $w = 0$。

2) 若 v 是 u 的祖先，则 $w = D_u - D_v + 2$。

3) 令 f 为 u 和 v 的 LCA，$p1$ 为 f 的儿子中 u 的祖先，$p2$ 为 f 的儿子中 v 的祖先，p 为 $p1$ 和 $p2$ 中 D 较小的那个。通常情况下，$w = D_u + D_v - 2D_p + 2$，要特判 p 恰好等于 v ，这时 $w = D_u - D_v$。

通过简单的分类讨论，我们已经可以计算左树上两个节点的距离了。现在考虑如何计算左树上一条到根节点的链和某一个节点的距离和：只需要预处理左树上到根路径的 D 的前缀和，再通过类似的讨论就可以了。

现在考虑如何求一个由节点 u 到一条左链 l 的距离和。令 u' 为 u 的兄弟节点，显然 u' 是左节点。我们可以先求出 u' 到 l 的距离和，然后可以发现对于所有在原树中 u 子树中的节点，贡献多了 2，在 u' 子树中的节点，贡献少了 2，其余节点贡献不变。而左链 l 有两个很好的性质：

1) 左链中一旦存在一个节点 v 在原树中 u 的子树内（不包括 u），那么 l 中所有比 v 深的节点，都在 u 的子树内。

2) 一旦 u 出现在了左链 l 中，那么 l 中所有比 u 深的节点，都在 u' 在原树中的子树内。

利用这两个性质，我们可以简单地把贡献的变化计算出来并累加。右链的情况类似。

最后考虑如何把 $S_{[l,r]}$ 转化成一些左链和右链，这只需要一个倍增就可以了。

时间复杂度 $O(n \log n)$。

3 字符串

对于字符串 $s = uv$，v 是 s 好的后缀当且仅当存在字符串 t 使得 vt 是 st 的最小后缀。可以发现对于长度为 n 的字符串 s，s 的好的后缀不超过 $\log n$ 个。

证明：对于 s 的两个好的后缀 a 和 b，$|a| > |b|$。可以发现 $|b|$ 是 $|a|$ 的前缀，若 $2|b| > |a|$，那么 a 的有一个长度为 $|a| - |b|$ 的周期。设 $a = TTc$，$b = Tc$，则 c 是 b 的前缀。现在考虑串 $TTct$，若 $Tct < TTct$，则必有 $ct < Tct$，所以 bt 不可能作为 st 的最小后缀，矛盾。因此 s 中任意两个好的后缀长度至少相差两倍，数量不超过 $\log n$。

现在对 $[1, n]$ 建线段树，每一个节点维护它所代表的子串的所有好的后缀，那么只要能合并两个串就可以了。现在考虑合并 u 和 v，因为在线段树上有 $|v| \leq |u| \leq |v| + 1$，因此 u 中至多只有一个 uv 的好的后缀。

对于 u 的两个好的后缀 x 和 y，我们比较 xv 和 yv，若 yv 不是 xv 的前缀，那么只要保留字典序较小的那个就可以了，否则保留较长的那个（原因见证明）。最后我们得到了唯一的可

能的好的后缀，这儿不需要进一步的检验这个后缀是否真的是好的后缀以及 v 中的好的后缀是否仍然还是好的后缀，直接合并复杂度也是对的。

在询问的时候我们先定位出 $O(\log n)$ 个区间，这 $O(\log n)$ 个区间带来了 $O(\log^2 n)$ 个候选答案，我们只要比较出最优的那个就可以了。

于是问题转化成了区间加减，比较两个子串的字典序大小。一个简单的做法是线段树维护哈希值，然后比较时二分出两个子串的 LCP（最长公共前缀）。这样一次比较的时间复杂度是 $O(\log^2 n)$，总的时间复杂度为 $O(n\log^3 n + m\log^4 n)$。

可以发现线段树维护并不优，我们可以对序列分块然后维护哈希值，这样修改是 $O(\sqrt{n})$，询问时 $O(1)$ 的，复杂度降到了 $O(n\log^2 n + m\log^3 n + m\sqrt{n})$。