小学数据结构教程

Falsyta

2018 年 6 月 12 日

T1

显然就是求线性无关组的个数。(线性相关的显然一次都构造出来)
高消

T2

大概是要弄个菊花图的形状
我的构造：
建 2K 个点，Ti 令 2i-1 和 2i 当根。
(2i-1, 2j) (2i, 2j-1) (j<i)
(2i-1, 2j-1) (2i, 2j) (j>i)

T3

f[i] 切 1..i 最后一个数最小的最后一刀的位置
g[i] 切 i..n 最后一个数最小，字典序最大的第一刀位置
注意到由于是数字比较，所以位数多的一定大（没有前导 0 的话）因此可以线性更新 DP 值

钓鱼题一

求一个 01-Trie 的后缀数组。
$O(n \log n)$ 暴力大家应该都会。

Sol

考虑 Trie 上后缀自动机的满足 $len_v = len_u + 1$ 的边 (u, v)，建出来是一棵生成树，DFS 一遍就可以很容易的建出后缀数组。

钓鱼题二 - BZOJ4231

给一棵边带字符的树，每次询问一个串 s 在 u 到 v 的路径上出现了多少次。

$n, q \leqslant 10^5$

Sol

注意到 lca 处可以暴力，转化成深度递增的一段路径。
时间复杂度 $O(n \log n)$。

钓鱼题三 - HNOI2016lcm

对于无向图 G，边有两个权值 a, b，q 次询问，每次询问 u, v 之间是否存在一条 $max(a_i) = A$，$max(b_i) = B$。
$n, m, q \leqslant 2 \times 10^5$
$O(n^{1.5})$ 的暴力应该大家都会。

钓鱼题四 - COT5

维护一个 Treap，支持插入一个节点（权值和权重给定），删除一个节点和询问两个节点之间的距离。
可以认为权值权重两两不同。
$m \leqslant 2 \times 10^5$

Sol

个人觉得如果只是嘴巴的话可能搞出来个 $O(npolylog(n))$ 的做法并不太难（雾）。
主要问题是怎么写的比较优雅。
两个点的 LCA 相信大家用脚也会求，就不说了。
那么就考虑怎么查询一个点的深度。
无非就是向左向右找单调上升的序列的长度。
维护
`len`[0/1] 区间内向左/右走能走出的最大长度
`max` 区间最大值

Deep Purple

给定一个长度为 n 的字符串 S，q 次询问，每次询问 $S[l:r]$ 的 $LBorder$（即最大的 i 使得 $S[l:l+i-1]=S[r-i+1:r]$）。
$n,q\leqslant 2\times 10^5$

Quasi Template

定义一个串 s 能匹配 S 当且仅当 s 能可超出头尾得覆盖 S。如下图。
给定 S，问不同的 s 的个数和字典序最小的 s。
$|S| \leqslant 2 \times 10^5$

Sol

考虑后缀树，唯一问题是匹配开头，匹配其他的可以用平衡树启发式合并。
考虑使用 KMP。
考虑每个后缀树节点到其父亲节点的边上的每一个位置到根的路径组成的字符串，令我们当前考虑的边中的可行字符串是 $S[p:l], S[p:l+1], \ldots S[p:r]$，其中 p 是当前后缀树节点的第一次出现位置。
那么该边的贡献为 $\sum_{i=l}^{r}[next_i \geq p-1]$。
该条件的必要性显然，充分性的话，
因为若 $i - next_i + 1 <= p$，则显然 p 不是该后缀节点的第一次出现位置，不可能。
时间复杂度 $O(n\log^2 n)$。

Sol - cont'd

注意到瓶颈在于平衡树的启发式合并，因为只是要支持合并和维护最小差值，所以用线段树合并就可以做到 $O(n \log n)$ 了。

Subsequence

考虑这样的一个程序，按照如此方式计算一个序列 a 的代价：

```
def calc_cost(a):
    cur = 0
    cost = 0
    for (cost0, cost1, col) in a:
        if cur == 0:
            cost += cost0
        else:
            cost += cost1
        cur = col
    return cost
```

对于一个序列 A 的所有子序列的代价，求出它们中前 k 大的。
$n, k \leqslant 5 \times 10^4$

Sol

好像就是个 k 短路嘛?
让我们试试这个代价函数?

```
def calc_cost(a):
    cur = 0
    cost = 0
    for (k0, k1, b0, b1, col) in a:
        if cur == 0:
            cost = k0 * cost + b0
        else:
            cost = k1 * cost + b1
        cur = col
    return cost
```

如果 k 短路还能做我就……向 k 短路低头

Sol - cont'd

如果只是求最优解的话 DP 是显然的。
把 int[][] 改成 Heap[][] 就可以求第 k 大了。
不过普通的可并堆复杂度是 $O(\log size)$ 的，这里不出意外的话 *size* 是指数级别的。
说一个我的合并策略：
我们用一些 *F* 节点来组成堆的结构。

```
struct FNode { Heap<FNode> son; Info info; };
```

大概就是用一个可并堆来维护一个 F 节点所有的儿子（这些儿子也是 F 节点）。

Sol - cont'd

合并的时候就是直接让一个变成另外一个的儿子，意会一下大概长这样子：

```
FNode fnode_merge(FNode *u, FNode *v) {
    if(u->val<v->val) swap(u, v);
    return new FNode(heap_merge(u->son, new
        HeapNode(v)), u->info);
}
```

求第 k 大的时候：

```
int kth(FNode *u, int k) {
    HeapNode *root = new HeapNode(u);
    for (int i = 1; i < k; ++i) {
        root = merge(heap_pop(root), heap_merge(
            root, u->son));
    }
    return (root->ptr->info).val;
}
```

Sol - cont'd

以上所有数据结构都是可持久化的。
打标记留给大家思考吧。
时间复杂度 $O(n \log n)$。

Excalibur

维护一个小写字母字符串的字符串序列 S，支持操作：

- `I k s` 在第 k 个位置后插入串 s
- `Q l r s` 查询串 s 在位置在 $[l, r]$ 中的多少个串中出现过

强制在线
要求复杂度 $O(Mpolylog(M))$，M 是总串长加询问数。

Sol

先做离线的版本，考虑维护后缀树的 DFS 序。
每加入一个字符串，将其加入后缀树，并求出其所有后缀节点构成的虚树，我们要对所有虚树边经过的点答案加一。
对每个点计算贡献，并在对应的数据结构里修改。
把最终的后缀树建出来，用线段树套平衡树维护。
时间复杂度 $O(M\log^2 M)$

Sol - cont'd

为了在线，后缀树和虚树都要改成在线的版本，外层数据结构只要改成重量平衡树即可。
后缀树的在线，只要暴力复杂度就是对的。
求虚树需要求一个点的深度，LCA 和按 DFS 序排序。
深度可以直接维护，用重量平衡树维护 DFS 序就可以排序，LCA 可以用类似 ST 的想法维护（虽然标解写的不是这个）。
时间复杂度 $O(M \log^2 M)$。
注：跑的并没有根号快。

Vortex

有一门野鸡语言，其中有四种语句和一个寄存器 x。

- `xor/or/and a` 即执行 $x = x$ *xor*/*or*/*and* a
- `jump p` 跳转到第 p 行的语句继续执行，保证不会死循环

这门野鸡语言还有一个版本控制器，支持三种操作：

- `M v p s` 将版本号为 v 的程序的第 p 条语句改为 s，另存为一个版本
- `R v s t` x_0 寄存器 x 的初始值是 x_0，从版本号为 v 的程序的第 s 条语句执行到第 t 条语句结束，输出最终 x 的值
- 询问版本号为 v 的程序中，能到达第 p 条语句的编号最小的语句。

程序有 n 条语句，进行 m 次操作。
$n, m \leqslant 5 \times 10^4$

Sol

题面写了很长，但是要做的事情很简单：

- 询问链信息
- 询问子树信息
- 更改一个点的父亲
- 更改一个点的信息
- 可持久化地维护

这种事情基本就只能考虑树链剖分了（在 $O(npolylog(n))$ 的算法里）。
主要问题是如何换父亲和如何可持久化。

Sol - Persistent Treap

先考虑一个很重要的问题，怎么可持久化平衡树呢?
我相信大家都会可持久化 Treap，那么怎么记父亲呢?
考虑如下的策略:
每个节点开两个 $(verion, father)$，要记父亲的时候先看这两个桶是否全被占用了，若全被占用就新开一个节点。
复杂度的话，考虑令势能为当前两个桶都被占用了的节点数。
时间复杂度 $O(n \log n)$。

Sol - cont'd

回到正题。
如果这棵树只会加叶节点，该怎么维护它的树链剖分呢?
考虑一个点到根的路径上只有 $O(\log n)$ 条轻边，对每条边判断它是否应该变成重边，用 Treap 维护 DFS 序。
删除叶节点怎么办?
对每个点维护 $s_u = 2size_u - size_{fa}$，当 $s_u \leqslant 0$ 时应把 u 到父亲的边改为轻边，维护 $\min\{s_u\}$ 即可。
要维护的东西很多……
$top_u, begin_u, end_u, s_u, size_u$……
如果没搞错的话时间复杂度是 $O(n \log^2 n)$ 的。