第 45 届国际大学生程序设计竞赛（ICPC）亚洲区域赛（济南）题解

A. Matrix Equation

$A \times B = B \odot C$

可以发现这两个运算都是列独立的 , 我们令 $B_{*,i}$ 表示 B 的第 i 列 , 则以上式子可以写成 n 个式子:

$\forall i \in [1, n], \ \ A \times B_{*,i} = C \odot B_{*,i}$

也就是 $\forall j \in [1, n] \sum_{k=1}^{n} A_{j,k} B_{k,i} = B_{j,i} \odot C_{j,i}$

我们把 B, C 第二维的 i 省略, 相当于 $\forall j \in [1, n] \sum_{k=1}^{n} A_{j,k} B_k = B_j \odot C_j$

接下来想办法处理 $B_j \odot C_j$, 如果 $C_j = 0$ 则直接不管, 如果 $C_j = 1$ 则相当于把左侧的 $A_{j,j} B_j$ 改成 $(A_{j,j} - 1) B_j$

所以变成 $\forall j \in [1, n] \sum_{k=1}^{n} (A_{j,k} - [k = j \land C_j]) B_k = 0$

我们把 B_k 看成第 k 个向量选不选,相当于变成了从 n 个向量里选若干个, 使得他们的异或值为 0 向量

可以在 $O(n^3/w)$ 的时间内求出基的个数 d_i, 答案就是 $\prod_{i=1}^{n} 2^{n-d_i}$

时间复杂度: $O(n^4/w)$

B. Number Game

可以发现, 每个数的 sg 在 $0...20$ 这个范围内,只要我们能求出对于每个 sg 值有几个数的 sg 值等于它,就可以用一个简单的 $O(k^2)$ dp 计算出答案

我们可以用一个非常大力的状压 DP 来计算: $f[n][S]$ 表示 $[n-19...n]$ 的 sg 值为 S, 每个 sg 值有几个数

这个状压 DP 的转移非常简单, 但是复杂度过大无法通过本题

我们可以考虑倍增的思想, 令 $f[i][S][w]$ 表示, 状态 S 从 $k2^i$ 走到 $(k+1)2^i - 1$ 后所到的状态, 并且 k 有 w 个 1 (显然 w 只要记录奇偶性)

$f[i][S][..]$ 可以通过 $f[i-1][S][..]$ 和 $f[i-1][f[i-1][S][..]][..]$ 转移, 这个跟倍增求 LCA 是类似的方法

然后询问时也类似倍增 LCA 来询问就行了, 本质的原理就是从高往低每次放 1

时间复杂度: $O(|State| \log n)$

但是直接这样做的话会因为 S 的数量过多而 TLE

我们可以暴力做 $i \leq 15$ 的情况, 这个直接 for 一下就可以了,不用再往下拆分,这样可以减少很多新的状态

神奇的是, 如果这样做, 绝大多数情况下 $|State| = 3$, 最坏的情况下 $|State|$ 大概是 10 这个级别的.

出题人以及验题人目前都没能证明这个神奇的性质, 如果有选手知道怎么证明请联系我

C. Stone Merge

可以发现,如果石子的个数是 3 的倍数, 则他跟别人合并的作用都是自己消失然后不产生任何 cost, 所以可以忽略 %3=0 的石子堆

假设 %3=1 的和 %3=2 的石子堆分别是 a_1, a_2 个, 则:

- 如果有两堆石子分别是 $1, 2$,肯定优先合并他们两个
- 如果只剩 1 或者只剩 2, 则将 2/3 的该石堆合并, 然后重复上一步

时间复杂度: $O(\log a_i)$

D. Fight against involution

考虑按照 R 从小往大贪心, 显然在这个顺序下给他们规定的 w_i 是不递减的, 而且越小越好, 所以贪心即可

需要注意的是 R 相同的学生他们的 w_i 必须相同, 处理一下即可

时间复杂度: $O(n \log n)$

E. Tree Transform

出题人的做法:

如果可以在 $O(n)$ 的时间内把一个点与另一个点连上的话, 就做完了, 只要 bfs 一遍结果的树,然后每次给每个点挂到他的父亲上即可

假设现在是要把点 x 挂到 y 上, 如果 $x \to y$ 的链大于等于 4 个点, 显然可以每次操作把链的大小缩小 1, 最后链上只有 4 个点的时候直接挂上去就行了

如果小于等于 3 个点, 先想个办法把 x 挂到更远的地方, 然后再做上一步, 这是一个比较简单的分类讨论

验题人的做法:

考虑该操作的逆变换是把任意 4 个连通的点变成一条链

先把 S 通过变换变成一条链, 再把 T 通过逆变换变成一条链, 之后链到链的变换就是冒泡排序

时间复杂度: $O(n^2)$

F. Gcd Product

$C_k = \sum_{i=1}^{k} A_{gcd(i,k)} B_{gcd(k+1-i,k)}$

$= \sum_{d_1|k} \sum_{d_2|k} A_{d_1} B_{d_2} \sum_{d_1|i} [gcd(i/d_1, k/d_1) = 1][gcd(k+1-i, k) = d2]$

考虑容斥掉后面的 $gcd = 1$, 这个只需要再枚举一个约数

$= \sum_{d_1|k} \sum_{d_2|k} A_{d_1} B_{d_2} \sum_{c_1|k/d_1} \sum_{c_2|k/d_2} \mu(c_1)\mu(c_2) \sum_i [c_1 d_1|i][c_2 d_2|k+1-i]$

设 $D_1 = c_1 d_1, D_2 = c_2 d_2$

$= \sum_{D_1|k} \sum_{D_2|k} f(D_1) g(D_2) \sum_i [D_1|i][D_2|k+1-i]$

其中 $f(D_1) = \sum_{d|D_1} A_d \mu(D_1/d), g(D_2) = \sum_{d|D_2} B_d \mu(D_2/d)$

其实前面那些都是比较好处理的, 主要是最后的 $\sum_i [D_1|i][D_2|k+1-i]$ 比较难处理

首先, 因为 $D_1|k, D_2|k$, 所以 D_1 和 D_2 需要互质, 否则 $gcd(D_1, D_2)|k+1$,但是 k 和 $k+1$ 是互质的

设 $i = k_1D_1, k+1-i = k_2D_2$

则 $k_1D_1 + k_2D_2 = k+1$, 相当于要求这个方程的正整数解 (k_1, k_2) 个数

分别将方程对 D_1, D_2 取模

$k_2D_2 = 1(mod\ D_1)$

$k_1D_1 = 1(mod\ D_2)$

因为 D_1, D_2 互质,所以 k_1, k_2 在模 D_2, D_1 下分别有且仅有一个解, 假设是 x_1, x_2

因为 $x_1D_1 < D_1D_2$ 且 $x_2D_2 < D_1D_2$, 所以 $x_1D_1 + x_2D_2 = D_1D_2 + 1$

之后可以认为就是将剩余的 $(k - D_1D_2)$ 分配到 x_1D_1, x_2D_2 上,

所以解的个数是 $k/(D_1D_2)$

$c_k = \sum_{D_1|k} \sum_{D_2|k} f(D_1)g(D_2)k/(D_1D_2)[gcd(D_1, D_2) = 1]$

设 $w(D) = \sum_{D_1|D} [gcd(D_1, D/D_1) = 1]f(D_1)g(D/D_1)$

则答案就是 $c_k = \sum_{D|k} w(D)k/D$

求 f, g 都是 $O(n \log n)$ 的, 求 w 用一些枚举约数的技巧也是 $O(n \log n)$ 的, 最后求 c 也是 $O(n \log n)$ 卷一下即可

总而言之, 答案就是 $id * ((a * \mu) \times (b * \mu))$, 其中 $\times$ 是互质狄利克雷卷积, $*$ 是普通狄利克雷卷积

时间复杂度: $O(n \log n)$

G. Xor Transform

如果 $x\ xor\ y < x$,则直接一步到位

否则先 xor 上 y 变成 $x\ xor\ y$, 然后 xor 上 x 变成 y

H. Path Killer

假设最后每个 path 被删是在第 w_i 次操作

我们要求的是 $E[\max(w_{1\dots n})]$

而 $\max S = \sum_{T \subseteq S} (-1)^{|T|-1} \min(T)$

所以答案是 $\sum_{T \subseteq S} (-1)^{|T|-1} E[\min(T)]$

而 $E[\min(T)]$ 的值是很好算的, 相当于求 T 中第一个被删除的 path 的删除时间

假设 T 的path 覆盖了 k 个点

那其实就是一直抛硬币, 正面概率是 k/n, 反面概率是 $(n-k)/n$,抛到出正面为止期望要几次

根据一些数学常识答案是 n/k

所以我们要求的就是对每个 k: 有几个带权($(-1)^{|T|-1}$)集合 T 满足覆盖了 k 个点

这个可以树形 DP, $f[x][i][h]$ 表示底部在子树 x 里的路径选完了, 子树 x 内覆盖了 i 个点, 里面的路径往上延伸最长的延伸了 h

这是一个非常简单的树形 DP , 可以在 $O(n^3)$ 的时间内计算

I. Random Walk On Tree

我们可以用生成函数 $\sum_i x^i p_i$ 来表示一系列随机游走事件, p_i 指的是走 i 步的概率

如果我们能求出这次随机游走的生成函数 $f(x)$ 的话, 答案其实就是 $f'(1) + f''(1)$

我们考虑 S 到 T 的路径 $< x_1 \dots x_d >$, 设从 x_i 出发随机游走, 第一次到达 x_{i+1} 就停下的生成函数是 $f_i(x)$

那么答案的生成函数就是 $\prod_{i=1}^{d-1} f_i(x)$

我们可以对于树上每一条边 (v, fa_v) , 预处理出两个生成函数:

$up_v(x)$: 从 v 出发随机游走到 fa_v 停下来的生成函数

$down_v(x)$: 从 fa_v 出发随机游走到 v 停下来的生成函数

可以发现:

$up_v(x) = x/d(1 + \sum_{y \in son(v)} up_y(x) up_v(x))$

$(1 - x/d(\sum_{y \in son(v)} up_y(x))) up_v(x) = x/d$

$up_v(x) = x/(d(1 - x/d(\sum_{y \in son(v)} up_y(x))))$

其中 d 是 v 的度数

也就是说 , 可以通过自下而上的树形 DP 求出所有 $up(x)$

$down_v(x)$ 也同理, 本质是个自上而下的树形 DP

那么对于一次询问 S, T, 对于 $a \in sub(S), b \in sub(T)$, 答案就是:

$\prod_{v \in path[a, lca(a,b))} up_v(x) \prod_{v \in path[b, lca(a,b))} down_v(x)$

a 到 b 的路径可以拆成三个部分: $a \to S \to T \to b$

设 $G(x)$ 是 S 到 T 的路径的生成函数

则 a 到 b 的生成函数就是 $G(x) \prod_{v \in path[a,S)} up_v(x) \prod_{v \in path[b,T)} down_v(x)$

而我们要求的是对 $sub(S)$ 中的每个 a 和 $sub(T)$ 中的每个 b 的答案, 也就是

$G(x) \sum_{a \in sub(S)} \sum_{b \in sub(T)} \prod_{v \in path[a,S)} up_v(x) \prod_{v \in path[b,T)} down_v(x)$

等于:

$G(x)(\sum_{a \in sub(S)} \prod_{v \in path[a,S)} up_v(x))(\sum_{b \in sub(T)} \prod_{v \in path[b,T)} down_v(x))$

其实要求的就是: S 中每个点到 S 的路径的 $up(x)$ 或者 $down(x)$ 的乘积之和, 这个用树形 DP 可以求 , $G(x)$ 也可以用倍增或者树链剖分的方法求

那么说了这么多, 其实有一个很关键的问题: 生成函数怎么维护 ?

注意到, 我们最后只需要 $f'(1)+f''(1)$, 所以对于一个生成函数 $f(x)$, 我们用 $<f(1),f'(1),f''(1)>$ 表示它

根据四则运算求导规则, 我们可以发现:

$f(x)+g(x)\to<f(1)+g(1),f'(1)+g'(1),f''(1)+g''(1)>$

$f(x)g(x)\to<f(1)g(1),f(1)g'(1)+f'(1)g(1),f(1)g''(1)+2f'(1)g'(1)+f''(1)g(1)>$

对于 $1/f(x)$, 假设 $1/f(x)=g(x)$, 则 $f(x)g(x)$ 的三元组表示等于 $<1,0,0>$, 我们可以根据上面的式子列出方程解出 $<g(1),g'(1),g''(1)>$

所以这种表示方法是支持四则运算的, 我们上面所有的生成函数都可以用这种方法表示, 并 $O(1)$ 运算

时间复杂度: $O((n+Q)\log n)$

J. Tree Constructer

首先树是一个二分图,考虑怎么对二分图构造 $a_{1...n}$

首先拿出最后两位, 让左右分别是 01 和 10 ,这样可以保证左右内部不会产生边

之后我们可以采用一种思想 : 让左边当锁, 右边当钥匙

对于左边第 i 个点, 令他的值为全集去掉第 i 位, 也就是说, 只要右边的点第 i 位为 1 就可以和他有边

对于右边的, 对于所有和他相连的 j, 加上第 j 位即可

这样使用的位数是左边的点的个数+2, 可以发现二分图一定有一边不超过 $n/2$ 个点, 所以只需要用 $n/2+2$ 位

K. Kth Query

我们先将 $a_{1...n}$ 从高位到低位插入, 建立一棵 Trie 树

对于 Trie 上的每个点 x , 我们预处理出一个数组 $kth[x][1...sz(x)]$ 表示对于 x 的子树导出的数组 (因为是从 x 的子树导出,所以更高位的信息相当于都清零了), 任意选择异或的值 S, 第 k 小的最小值

设 $ch[x][0..1]$ 为 x 的两个儿子, 则 $kth[x][k]=\min(kth[ch[x][0]][k],kth[ch[x][1]][k])$

如果 $k>sz(ch[x][0])$ 的话 $kth[x][k]$ 还可以等于 $(2^{bit(x)}+kth[ch[x][1]][k-sz(ch[x][0])])$, 比右边大同理

这样我们就可以在 $O(n\log n)$ 的时间内处理出 $kth[x]$ (因为 trie 树每个点树高都是 $O(\log n)$ 的,所以暴力就可以了)

然后对于一次询问 L,R,k, 假设我们最后选择的是 S, 那么我们从高位往低位看的话, S 肯定经历了这么三个过程:

- 与 L,R 相同
- 与 L,R 中某个相同
- 无限制

所以我们可以枚举前面两部分有几位且是哪些情况, 剩下的位其实就没有上下界的限制了

现在我们相当于枚举了 S 的高 w 位, 我们在 trie 树上从根往下走, 当这一位已经确定了之后, 我们是可以确定第 k 小是在哪个子树的(S 这一位是 1 相当于交换了两个子树), 走到对应的子树即可

当 S 没有限制后直接用预处理好的 $kth[x]$ 计算

以上过程看上去是枚举 $O(\log n)$ 个情况然后每个 $O(\log n)$ 计算,但显然可以边走边算, 所以时间复杂度是 $O((n+Q)\log n)$ 的

L. Bit Sequence

我们枚举 x 最后 8 位的值, 以及 8 位后连续有几个 1, 以及 8 位后总共有几个 1

这样的话, $x+i$ 只要讨论进不进位, 就可以知道 $f(x+i)\ mod\ 2$ 的值 ,我们可以暴力枚举 $i=0...255$ 判断这组条件合不合法

如果满足条件, 就可以计算满足枚举的条件的数的个数, 这个是 $O(1)$ 的

时间复杂度: $O(Tm^2)$, 可以优化到 $O(Tm)$

M. Cook Pancakes

经典小学数学题

相当于有 $2N$ 个数 $-1...-N$ 和 $1...N$,每次可以选择删掉 K 个数, x 和 $-x$ 不能一起删(一个煎饼的两面不能同时煎), 求最少删几次删完

按照上面的方式排好然后每次删 K 个即可, 答案就是 $2N/K$ 上取整, 注意 $K>N$ 的情况