
牛客竞赛
AC.NOWCODER.COM
牛客暑期多校
训练营
南京外国语学校命题组
HR.NOWCODER.COM

AC.

A-Image Scaling

牛客竞赛
AC.NOWCODER.COM

首先，我们需要找到 x 组成的矩形的长宽。我们可以求出输入矩阵的所有 x 中横纵坐标的最小值和最大值，两者相减就是长宽。记长宽分别为 a, b，则等比例缩到最小后的长宽为 $\frac{a}{\gcd(a,b)}$，$\frac{b}{\gcd(a,b)}$，按长宽输出一个 x 组成的矩阵即可。

AC.

K-Kill The Monsters

牛客竞赛 AC.NOWCODER.COM

考虑对于单个位置，如果进行 1 和 2 操作各一次，顺序对答案的影响。容易发现按 12 的顺序进行会变成 $\lfloor \frac{a_i-1}{k} \rfloor$，而按 21 的顺序进行会变成 $\lfloor \frac{a_i-k}{k} \rfloor$。所以容易得出所有 2 操作会在 1 操作的前面。

考虑枚举 1 操作的次数 num，也就是说所有 a_i 都需要先靠 2 操作减到小于等于 num。容易发现每个 a_i 除以 $\log V$ 次就变成了 0，也就是说总共不同的 num 应该只有 $n \log V$ 种，使用堆维护或者离线排序即可。

总复杂度 $O(n \log n \log V)$。

AC.

I-Interesting Numbers

牛客竞赛
AC.NOWCODER.COM

显然可以将问题转化为求R以内满足条件的数x的个数

简单分类讨论即可：

- x的高$\frac{n}{2}$位比R的小：此时x的高$\frac{n}{2}$位的方案数为$\left\lceil \sqrt{\frac{R}{10^{\frac{n}{2}}}} \right\rceil$，低$\frac{n}{2}$位的方案数为$\left\lceil \sqrt{10^{\frac{n}{2}}} \right\rceil$
- x的高$\frac{n}{2}$位和R的相等且为完全平方数，且x的低$\frac{n}{2}$位不比R的大：此时x的高$\frac{n}{2}$位的方案数为1，低$\frac{n}{2}$位的方案数为$\left\lfloor \sqrt{R \mod 10^{\frac{n}{2}}} \right\rfloor + 1$

AC.

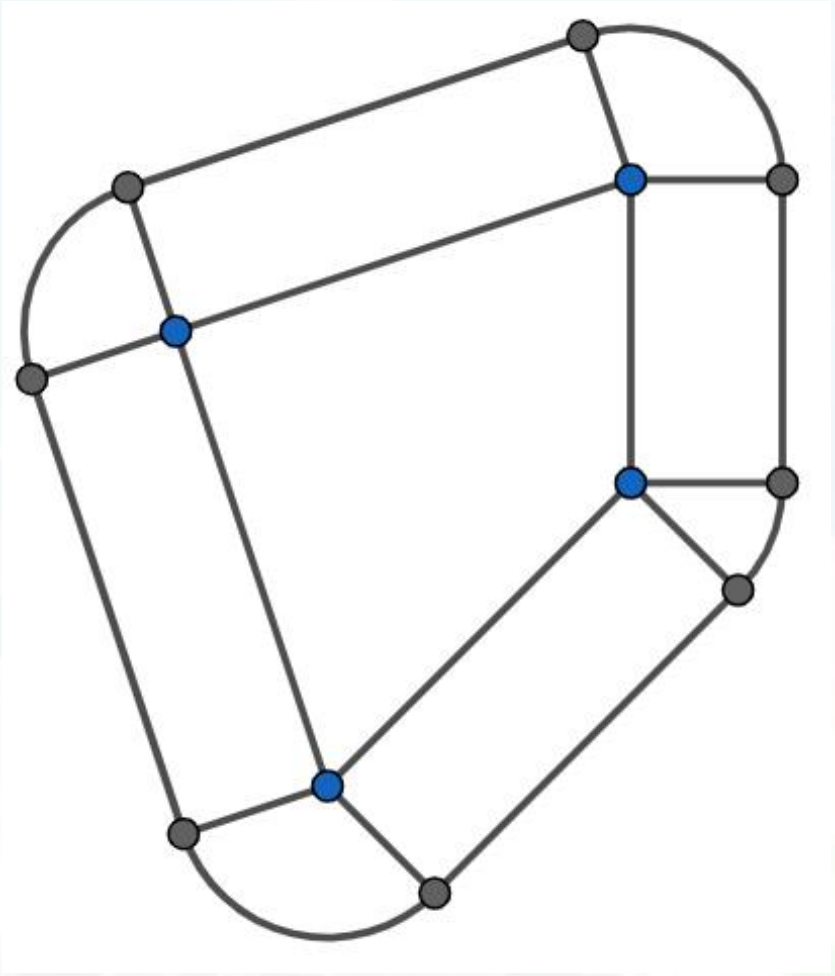
H-Two Convex Polygons

首先，答案为 A 的周长 $+ 2 \times \pi \times B$ 的直径。

一方面，考虑找到 B 的直径，让这条直径在 A 的周围覆盖一圈，可以发现此时的图形为若干条线段连接了若干段圆弧，如图所示。这些线段的长度和即为 A 的周长，而这些圆弧可以拼成一个整圆，半径是 B 的直径，所以该图形内的点都能被覆盖到。

另一方面，对于该图形外部的点，到 A 内部点的距离 $> B$ 的直径，如果能够覆盖到，说明 B 中应该存在两点距离 $> B$ 的直径，而这显然不满足，所以该图形外部的点都无法被覆盖。

至于凸多边形直径的求法，参见维基百科旋转卡壳。



AC.

B-Break Sequence

考虑 dp_i 表示 i 前缀，分成若干个合法的段的方案数。

考虑转移，有显然的 $O(n^2)$ 解法，即枚举 j，判断 $i+1 \sim j$ 一段是否合法。

考虑优化，注意到在每个右端点，对于一种值，其出现次数在 S 中的情况对应的左端点一定形成 $O(|S|)$ 个区间。

于是，在每次扫到 a_i 的时候，更新一下这一些区间。我们不允许从这些区间内转移，于是可以在这些区间上用线段树打 tag，区间查询 tag 最少的位置 j 对应的 dp_j 值的和，如果 tag 的 $\min$ 是 0 那么就转移到目前位置的 dp_i 即可。

总复杂度 $O(nm \log n)$。

AC.

C-Change Matrix

牛客竞赛 AC.NOWCODER.COM

注意到$\gcd(i,j)=\sum_{x|i,x|j}\phi(x)$，考虑维护$n$个矩阵$M_i$，大小分别为$\left\lfloor\frac{n}{i}\right\rfloor$，初始全为$\phi(i)$，每次将原矩阵的第$x$行/列乘$y$时改为对于$x$的每个因数$t$，将$M_t$的第$\frac{x}{t}$行/列乘$y$，可以证明所有$M_i$的和等于原矩阵的和

动态的维护每个M_i的每行和每列总计被乘的数$r_{i,j},c_{i,j}$，可以证明M_i的和等于$\phi(i)\times\sum r_{i,j}\times\sum c_{i,j}$

由于$[1,n]$随机的x的期望因数个数为$\Theta(\log n)$，所以总时间复杂度为$\Theta(n+q\log n)$

AC.

D-Luner XOR

考虑对函数距离做一个转化：

$$\sum_{x\in F_2^n}(f(x)\oplus g(x))=2^{n-1}+\frac{1}{2}\sum_{x\in F_2^n}(-1)^{f(x)}\cdot(-1)^{g(x)}$$

发现后面的求和部分可以 FWT，于是就做完了。时间复杂度 $\mathcal{O}(2^n n)$。

AC.

G-Sticky Pistons

为了便于描述，将序列 a_i 进行「翻转」。初始对于 $i \in [0, n]$，有 $a_i = i$，目标变为将 a_0 从位置 0 推到位置 $-n$ 并拉回来。当 a_0 被推到 $-n$ 时，显然有 $a_{i+1} = a_i + 2$。

先不考虑并行，并记 $S_1(n)$ 表示第一部分的最优操作序列，$S_2(n)$ 表示第二部分的最优操作序列，使用符号 $+$ 表示操作序列的拼接。

对于第一部分，首先考虑没有 k 的限制的情况。显然可以从大到小枚举 i，将 $< i$ 的数往左推，可以在 n 个时刻内完成。即 $S_1(n) = n + S_1(n-1)$。

倘若有 k 的限制，则可能会出现「推不动」的情况。此时可以先从 k 到 1 依次向左推，再将 $2k$ 到 1 依次向左推，以此类推，最后将 n 到 1 向左推。形式化地，对操作序列进行描述：

$$S_1(n) = \begin{cases} S_1(\lfloor (n-1)/k \rfloor \times k) + n + \ldots + 1 & n > k \\ n + \ldots + 1 & n \le k \end{cases}$$

不难发现，在上述操作序列中，不会出现某个位置推不动的情况。

AC.

G-Sticky Pistons

牛客竞赛 AC.NOWCODER.COM

考虑如何将上述操作序列并行。显然 $n + \ldots + 1$ 的操作是要按顺序进行的，这部分已经无法并行优化。

先考虑简单的情况，当 $n \leq 2k$ 时，看看能不能将 $n + \ldots + 1$ 与 $k + \ldots + 1$ 进行并行优化。

- 在时刻 1，显然只有 k 可以推动。
- 在时刻 2，除了 $k - 1$ 可以推动之外，由于 k 与 $k - 1$ 不再相邻，因此 n 也可以推动。
- 在时刻 3，除了 $k - 2$ 可以推动之外，由于 $k - 1$ 与 $k - 2$ 不再相邻，因此 $n - 1$ 也可以推动。

这启发我们，拓展到一般情况下，记 $f(n) = \lfloor (n-1)/k \rfloor \times k$，可以按照下表的方式进行并行：

$$
\begin{array}{cccccccc}
f(f(n)) & f(f(n))-1 & f(f(n))-2 & f(f(n))-3 & \ldots & 1 & & \\
 & f(n) & f(n)-1 & f(n)-2 & \ldots & \ldots & 1 & \\
 & & n & n-1 & \ldots & \ldots & \ldots & 1
\end{array}
$$

所需的时刻 $Q_1 = n + \lfloor (n-1)/k \rfloor$。

AC.

G-Sticky Pistons

牛客竞赛 AC.NOWCODER.COM

对于第二部分，还是先不考虑并行，先构造出一个将 a_0 拉回来的方法。

初始时，有 $a_0 = -n$，$a_{i+1} = a_i + 2$。每次操作最多将 1 个位置的 a_i 增加 1，最终状态为 $a_i = i$，因此总操作数的下界为 $1 + 2 + \ldots + n = n(n+1)/2$。

考虑直接根据操作数的下界，递归地构造一个方案：

$$S_2(n) = \begin{cases} n + \ldots + 1 + S_2(n-1) & n > 1 \\ 1 & n = 1 \end{cases}$$

即，先递归地将前 $n-1$ 个拉回来，再将前 $n-1$ 个值整体向右拉一格。

AC.

G-Sticky Pistons

继续考虑能否并行。考虑什么情况下，并行会导致操作出问题。

若某一时刻，$a_{i+1} > a_i + 2$，这一定是不优的，因为这将导致 i 无法将 $i+1$ 拉回。

因此，考虑以下情况：

- 在时刻 1，只能操作 1。
- 在时刻 2，只能操作 2。
- 在时刻 3，除了操作 1 之外，注意到在此之后，1 和 2 已经相邻，此时操作 3 并将 2 向右移动不会使得 $a_2 > a_1 + 2$。因此可以并行操作 3。
- 在时刻 4，操作 $2, 4$。

拓展到一般情况下，按照下表的方式进行：

$$
\begin{array}{cccccccc}
 & & & & 1 & & & \\
 & & & 2 & & 1 & & \\
 & & & & 3 & 2 & & 1 \\
 & & & & & 4 & 3 & 2 & 1 \\
 & & & & & & \cdots &
\end{array}
$$

所需的时刻 $Q_2 = 2n - 1$。

AC.

E-Easy Tree Problem

首先对于每次询问，有一个简单的 dp。考虑 dp_i 表示 i 子树答案的最大值。每次转移 i 选择最大的 b_i 个儿子求和设为 dp_i 的答案。

考虑有修改怎么办。容易想到动态 dp。

在重链上，儿子对父亲的影响一定是形如 $f(x) = x + t$ 的一个函数。我们可以在重链上用线段树维护，做函数复合。

在轻链上，我们考虑开一个对顶可删堆来维护所有轻链第 b_i 和 $b_i - 1$ 大的答案。每次跳轻链的时候更新父亲的对顶可删堆，并求出目前重链那边距离第 b_i 大的距离，更新该位置的 $f(x) = x + t$ 中的 t 即可。

总复杂度 $O(n \log^2 n)$。

AC.

J-String Problem

首先考虑如何判断一个 01 串是否合法，发现可以通过建一个 2^{k-1} 个点的带权有向图，合法当且仅当每个环长度恰好为 0，如果加入反向边（权值为正向边的相反数），那么合法当且仅当不存在负环。

有些位置不确定该怎么办，考虑差分约束，注意到每个点最多只会加入队列 k 次（否则说明存在负环），复杂度 $O(k \times 2^k)$，再结合从前往后确定的部分，复杂度 $O(k \times 2^k)$。

跑完上一次差分约束之后，只会增加或减小一条边的权值，使用 Johnson 刻画（从 0 开始跑最短路）之后相当于是有一张边权只有 01 的有向图，现在把某条边的权值加或减 1，问是否存在负环（之后的图都默认为是 Johnson 操作之后的图）。

AC.

J-String Problem

显然如果要存在负环首先只有可能是把某条权值为 0 的边变成 -1。

比如说现在把 (u,v) 边权从 0 变成 -1，当且仅当存在 $v \to u$ 的边权为 0 的路径，也就是说，如果 (u,v) 在边权为 0 的边组成的强连通分量内，那么就不能减，不然就能减。

如果能减，那么直接对 v 走 0 边到达的点进行 DFS，注意到势能总和是 $O(k \times 2^k)$ 的，所以如果我们可以 $O(1)$ 判断是否合法，那么总复杂度就是 $O(k \times 2^k)$。

然后做一点简单的观察，我们可以发现两个事情：一是如果确定了字符串某个位置的字符，那么对应图上的这两条有向边权值都变成了 0，二是我们确定边的顺序是 $(0,0),(0,1),(1,2),(1,3),\cdots$ 也就是说确定的边至少有一端是和 0 在同一强连通分量内。

根据 Johnson 的定义，0 到所有点的最短路都是 0，于是我们只需要维护所有能通过走 0 边到达 0 的点的点集即可快速判断，总复杂度 $O(k \times 2^k)$。

AC.

F-Minesweeper

令 q_1 为修改个数，q_2 为询问个数。

做法 1

注意到在 DFS 序上，一个点的父亲是它左边离他最近的、深度比它小的点。

所以考虑查询的路径如果是 1 到 u，可以认为是区间 $[1,u]$ 上所有后缀深度最小值所在的位置。

所以是查询一个前缀的单调栈上的权值和，可以直接用单侧递归线段树维护，复杂度 $O((q_1+q_2)\log^2 n)$，常数较大应该难以通过。

但是可以发现，修改操作根本不会改变树形态，所以区间后缀最小深度的点都不变。所以我们可以对于每个结点不进行上传操作，而是改为计算本次区间加有当前结点的单调栈中有多少个元素被影响。这可以通过每个结点返回自己的单调栈内被修改的元素在单调栈上的区间做到。

复杂度 $O(q_1\log n+q_2\log^2 n)$。

F-Minesweeper

做法 2

验题人的做法。

树剖，维护每个点到根的前缀和。

先对于修改拆成两次前缀修改，那么 1 到 r 的编号就直接加上深度乘以权值。

大于 r 的部分，对于根到 r 路径中每条重链下的子树考虑。

对于 i 来说，就是加上 i 到这条重链的对应点的深度。

那么就是 i 往上深度第 j 小的轻边对应的深度。

这个 j 对于这条重链下所有点都是一样的，并且 j 是 $O(\log n)$ 的，可以对每个 j 开一棵树状数组，就好修改了。

查询时对于每一棵树状数组进行单点查询。

复杂度 $O((q_1+q_2)\log^2 n)$，常数较小可以通过。

AC.

F-Minesweeper

牛客竞赛 AC.NOWCODER.COM

做法 3

先把全局平衡二叉树建出来，简化处理，我们只考虑 DFS 序前缀修改。

记这个前缀里最后一个点是 x。修改相当于，将 x 的所有祖先，以及所有祖先的靠前面的子树修改。

考虑从 x 往上跳重链，每次更换一条重链的时候，在全局平衡树上，那些要进行子树加的子树，形成一个类似线段树上区间的东西，打标记。以及，对重链上要修改的部分做区间加。查询时，对这个标记下传。

复杂度 $O((q_1+q_2)\log n)$，未实现，应该能过。

F-Minesweeper

做法 4

来自场上老哥（队伍：栀晓）。

考虑树剖，然后用线段树维护原始的 DFS 序。不过线段树上不维护区间和，而是维护区间所有属于这个区间内编号（也就是 DFS 序）最小的重链在这个区间的和。

修改直接对线段树进行区间修改。查询就树剖正常查询即可。容易发现这是正确的。

复杂度 $O(q_1 \log n + q_2 \log^2 n)$。

THANKS

AC.NOWCODER.COM