THUPC2024 初赛

Elegia, E·Space, fzw, Itst, liuzhangfeiabc, SpiritualKhorosho[†]

2023 年 12 月 17 日

清华大学算法协会

除命题人外, 感谢 He-Ren, Mys.C.K.,PinkRabbait 等人参与题目准备工作.

† 按照字典顺序排列.

Part 0 (出题人 *liuzhangfeiabc*)

M 你说得对，但是 AIGC

Part I (出题人 *Elegia*)

C 前缀和

I 分治乘法

J 套娃

Part II (出题人 *E.Space*)

E 转化

F 机器人

G 采矿

Part III (出题人 *fzw*)

B 一棵树

H 二进制

Part IV (出题人 *ltst*)

A 排序大师

Part V (出题人 *SpiritualKhorosho*)

D 多折较差验证

K 三步棋

L 勇闯末日塔

Part 0 (出题人 *liuzhangfeiabc*)

M 你说得对，但是 AIGC by *liuzhangfeiabc*

判断输入串的前 19 个字符是不是 ‘You are right, but ‘，是则输出 ‘AI‘，否则输出 ‘Human‘

花絮

中文2字英文7字的游戏除了扫雷 Winmine，还有蔚蓝 Celeste，~~东方Project~~.

第3个样例是应验题人的要求加上去的.

~~出题人表示不知道“G****** I*****”是什么东西~~（投降喵）

Part I (出題人 *Elegia*)

C 前缀和 by *Elegia*

有一个 0 到 1 之间的实数 p，接下来生成 n 个随机数，随机方式如下:
每个 x_i 独立生成，有 p 的概率是 1，$(1-p)p$ 的概率是 2，以此类推.
给定 $1 \le l \le r \le n$，问序列 $\{x_i\}$ 的前缀和序列 $\{y_i\}$ 中，期望有多少个数落在 $[l, r]$ 之间?

问题转化

- 考虑有无穷多个灯，编号从 1 开始一字排开
- 每个灯独立有 p 的概率亮起，$1-p$ 的概率是灭的
- 我们从左往右数，第一次找到一个灯的编号，这就是 x_i 的分布
- 所以序列 $\{y_i\}$ 就是第 i 个亮起的灯的位置
- 所以问题就是在问：$[l,r]$ 之间，期望有多少亮起的灯？
- 答案就是 $(r-l+1)p$。

有三种构造集合的方式：

- 创建一个集合 $\{x\}$
- 将两个不交集合 X, Y 合并
- 将集合平移 X + y

之前构造的集合可以重复使用.

构造的总代价是构造出的所有集合大小之和.

一个数均不超过 5×10^5 的集合，用不超过 $5t \times 10^6$ 的代价构造出它.

两个初步的思路

不能直接按照标题分治吗？

两个初步的思路

不能直接按照标题分治吗？

- 按照题目的要求计算，分治的代价大概是 $N \lg N$，要比题目限制多出一倍.

两个初步的思路

不能直接按照标题分治吗？

- 按照题目的要求计算，分治的代价大概是 $N \lg N$，要比题目限制多出一倍.

什么样的集合能借助平移操作显著快速地构造?

两个初步的思路

不能直接按照标题分治吗？

- 按照题目的要求计算，分治的代价大概是 $N \lg N$，要比题目限制多出一倍.

什么样的集合能借助平移操作显著快速地构造?

- 考虑 $[1, n]$，可以先构造出 $\left[1, \frac{n}{2}\right]$，再平移，如果需要的话再补上最后一个元素.
- $T(N) = T\left(\frac{N}{2}\right) + O(N)$ 最后总代价还是 $O(N)$.

两个初步的思路

不能直接按照标题分治吗？

- 按照题目的要求计算，分治的代价大概是 $N \lg N$，要比题目限制多出一倍.

什么样的集合能借助平移操作显著快速地构造?

- 考虑 $[1, n]$，可以先构造出 $\left[1, \frac{n}{2}\right]$，再平移，如果需要的话再补上最后一个元素.
- $T(N) = T\left(\frac{N}{2}\right) + O(N)$ 最后总代价还是 $O(N)$.

如何将二者有效地结合?

四毛子

考虑设置一个块大小 L，将集合剖成 $\frac{N}{L}$ 个大小为 L 的块.

一个块有 $2L$ 种可能性，我们记每种可能性的块构成的集合为 S_i.

四毛子

考虑设置一个块大小 L，将集合剖成 $\frac{N}{L}$ 个大小为 L 的块.

一个块有 $2L$ 种可能性，我们记每种可能性的块构成的集合为 S_i.

所有 S_i 的大小总和是 $\frac{N}{L}$，我们先构造出每个 S_i，然后平移出它在原序列中的 $S_i + x$，这一步的总代价是 $\frac{N}{L} \lg N + N$.

四毛子

考虑设置一个块大小 L，将集合剖成 $\frac{N}{L}$ 个大小为 L 的块.

一个块有 $2L$ 种可能性，我们记每种可能性的块构成的集合为 S_i.

所有 S_i 的大小总和是 $\frac{N}{L}$，我们先构造出每个 S_i，然后平移出它在原序列中的 $S_i + x$，这一步的总代价是 $\frac{N}{L} \lg N + N$.

现在我们要合并 $L2^L$ 个集合，把它们用 Huffman 树合并，代价是 $N \lg(L2^L) = NL$.

四毛子

考虑设置一个块大小 L，将集合剖成 $\frac{N}{L}$ 个大小为 L 的块.

一个块有 $2L$ 种可能性，我们记每种可能性的块构成的集合为 S_i.

所有 S_i 的大小总和是 $\frac{N}{L}$，我们先构造出每个 S_i，然后平移出它在原序列中的 $S_i + x$，这一步的总代价是 $\frac{N}{L}\lg N + N$.

现在我们要合并 $L2^L$ 个集合，把它们用 Huffman 树合并，代价是 $N\lg(L2^L) = NL$.

取 $L = \sqrt{(\lg aN)}$，总代价是 $O(N\sqrt{\lg n})$.

常数

上面这个做法取 $L = 5$，挂在 Huffman 树上跑，我能构造出来的数据基本上代价最多也就是 3.8×10^6 的代价.

我用了一些手段严格证明了这个做法在数据范围内是严格正确的（代价不超过 4.9×10^6），所以开了这个数据范围，具体的不等式放缩赛后公开，~~跑了一个 L 个变量的凸优化~~.

常数

上面这个做法取 $L=5$，挂在 Huffman 树上跑，我能构造出来的数据基本上代价最多也就是 3.8×10^6 的代价.

我用了一些手段严格证明了这个做法在数据范围内是严格正确的（代价不超过 4.9×10^6），所以开了这个数据范围，具体的不等式放缩赛后公开, ~~跑了一个L 个变量的凸优化~~.

似乎这个问题还有很多其他奇怪的做法也很有希望卡到数据范围之内（而且这个数据范围看起来很像卡常），所以实际上可能八仙过海了.

欢迎交流可以证明更低的代价复杂度的做法!

给一个序列 a，对每个 k，询问

- 把所有长为 k 的子区间求 mex，对得到的答案集合再取 mex

的结果.

解法

首先考虑所有里层的 mex 的数会如何出现.

解法

首先考虑所有里层的 mex 的数会如何出现.

如果一段区间的 mex 值是 v，那么这个区间首先不存在一个值域为 v 的数，然后 $0, ..., v-1$ 这些数都出现过.

解法

首先考虑所有里层的 mex 的数会如何出现.

如果一段区间的 mex 值是 v，那么这个区间首先不存在一个值域为 v 的数，然后 $0,...,v-1$ 这些数都出现过.

我们首先考虑数列中所有出现过的数字 v，它们出现过的位置将这个序列分割成 c_v 段（我们有 $\sum c_v = O(n)$），mex 值是 v 的段肯定只能是其中某一段的子区间. 而且如果某一段里存在的话，那么这一整段肯定存在，所以有一个长度区间 $[d,u]$，表示 $d,...,u$ 这些长度都有作为 v 的 mex 存在过.

解法

如果处理出了这些 $[d,u]$，我们只需要扫描线一下就就可以求出要输出的所有答案了，复杂度 $O(n\log n)$.

解法

如果处理出了这些 $[d,u]$，我们只需要扫描线一下就就可以求出要输出的所有答案了，复杂度 $O(n\log n)$.

接下来考虑如何确定这些 $[d,u]$。从小到大考虑 v，对于每个 i 我们维护一个最小的数 p_i 表示 $[i,p_i]$ 里出现过所有 $0,...,v-1$ 的数, 那么 p_i 这个阶梯型可以用线段树维护.

解法

如果处理出了这些 $[d,u]$，我们只需要扫描线一下就就可以求出要输出的所有答案了，复杂度 $O(n\log n)$.

接下来考虑如何确定这些 $[d,u]$。从小到大考虑 v，对于每个 i 我们维护一个最小的数 p_i 表示 $[i,p_i]$ 里出现过所有 $0,...,v-1$ 的数, 那么 p_i 这个阶梯型可以用线段树维护.

查询被 v 分割出的一段 $[l,r]$ 里最小的区间时, 我们先想办法二分出最大的 i 满足 $p_i<r$，然后用线段树顺带维护出 p_i-i+1 的最小值，对此区间查询.

解法

如果处理出了这些 $[d,u]$，我们只需要扫描线一下就就可以求出要输出的所有答案了，复杂度 $O(n\log n)$.

接下来考虑如何确定这些 $[d,u]$。从小到大考虑 v，对于每个 i 我们维护一个最小的数 p_i 表示 $[i,p_i]$ 里出现过所有 $0,...,v-1$ 的数, 那么 p_i 这个阶梯型可以用线段树维护.

查询被 v 分割出的一段 $[l,r]$ 里最小的区间时, 我们先想办法二分出最大的 i 满足 $p_i<r$，然后用线段树顺带维护出 p_i-i+1 的最小值，对此区间查询.

复杂度 $O(n\log n)$.

Part II (出題人 *E.Space*)

E 转化 by *E.Space*

本题改编自桌游 Gizmos 的部分玩法，定位简单题.

有 n 种球，第 i 种有 a_i 个.

- 可以把 i 变成任意颜色不超过 b_i 次.
- 可以把 i 变成两个 i 不超过 c_i 次.

对每个 i 分别求最多能有多少个 i，以及求最多能有多少个球.

正解

首先对于同一种颜色，$x+1$ 个第一种工具和 x 个第二种工具可以组合起来用，把一个 i 变成 $x+1$ 个任意颜色.

正解

首先对于同一种颜色，$x+1$ 个第一种工具和 x 个第二种工具可以组合起来用，把一个 i 变成 $x+1$ 个任意颜色.

对于存在这样的组合的颜色，如果 $a_i \neq 0$，那么可以用原来的球直接变出 $x+1$ 个任意颜色，否则如果有任意颜色的球，可以用一个任意颜色的球变出来.

正解

首先对于同一种颜色，$x+1$ 个第一种工具和 x 个第二种工具可以组合起来用，把一个 i 变成 $x+1$ 个任意颜色.

对于存在这样的组合的颜色，如果 $a_i \neq 0$，那么可以用原来的球直接变出 $x+1$ 个任意颜色，否则如果有任意颜色的球，可以用一个任意颜色的球变出来.

如果组合后有剩下的第一种工具和对应颜色的球，那么就可以用这些再变几个任意颜色的球.

正解

首先对于同一种颜色，$x+1$ 个第一种工具和 x 个第二种工具可以组合起来用，把一个 i 变成 $x+1$ 个任意颜色.

对于存在这样的组合的颜色，如果 $a_i \neq 0$，那么可以用原来的球直接变出 $x+1$ 个任意颜色，否则如果有任意颜色的球，可以用一个任意颜色的球变出来.

如果组合后有剩下的第一种工具和对应颜色的球，那么就可以用这些再变几个任意颜色的球.

所以第一问答案就是这种颜色剩下的球的数量加上任意颜色球的数量加上剩下的第二种工具的数量，需特判前二者均为 0 的情况.

正解

对于第二问，显然如果 $a_i \neq 0$ 或者用过第 i 种颜色的组合，那么所有关于第 i 种颜色的第二类工具都会用上.

正解

对于第二问，显然如果 $a_i \neq 0$ 或者用过第 i 种颜色的组合，那么所有关于第 i 种颜色的第二类工具都会用上.

此外，如果还有剩下的第一类工具（$a_i = 0$ 且 $b_i = 1$ 的情况），且有任意颜色的球，那么可以免费利用第一类工具转化而用上所有的第二类工具.

正解

对于第二问，显然如果 $a_i \neq 0$ 或者用过第 i 种颜色的组合，那么所有关于第 i 种颜色的第二类工具都会用上.

此外，如果还有剩下的第一类工具（$a_i = 0$ 且 $b_i = 1$ 的情况），且有任意颜色的球，那么可以免费利用第一类工具转化而用上所有的第二类工具.

剩下的情况（$a_i = b_i = 0$）需要按照 c_i 从大到小排序，每用一个颜色的工具就要消耗一个任意颜色的球.

花絮

还是写题解的时候最清醒.

我在写题解的时候发现了 std 的 bug 并加强了数据.

还是验题人 ltst 比较厉害，一次就写对了.

F 机器人 by *E.Space*

一堆机器人.

每个机器人有两只手和一些指令.

求执行指令的过程与结果.

题目定位是模拟题.

正解

有几点需要注意.

正解

有几点需要注意.

指令不能直接深拷贝（可能很长），因此修改的时候不能直接在原来的指令上改（可能有别的指令包含这部分）.

正解

有几点需要注意.

指令不能直接深拷贝（可能很长），因此修改的时候不能直接在原来的指令上改（可能有别的指令包含这部分）.

然后就没了.

正解

有几点需要注意.

指令不能直接深拷贝（可能很长），因此修改的时候不能直接在原来的指令上改（可能有别的指令包含这部分）.

然后就没了.

指令可以用一个东西打包一下，验题人 zyb 用了下标指针+操作时判断类型，我写的是类继承+虚拟函数.

正解

有几点需要注意.

指令不能直接深拷贝（可能很长），因此修改的时候不能直接在原来的指令上改（可能有别的指令包含这部分）.

然后就没了.

指令可以用一个东西打包一下，验题人 zyb 用了下标指针+操作时判断类型，我写的是类继承+虚拟函数.

我的写法会更长一些，不过 VS Code 的根据声明自动生成定义还是好用的.

G 采矿 by *E.Space*

一棵有根二叉树，初始有个机器人在 s.

有 4 种指令：

1. 将机器人往祖先移至少一步
2. 将机器人往子树里移至少一步
3. 将一个人类加到根
4. 将一个人类从根移走

移动时只能点上站人且一个点只能包含不超过一个人.

每个点有产出，根据上面的工人，在每个指令结算.

指令之间人类可以任意移动，求最大总产出.

正解

当前状态只和机器人位置、当前是哪个计划，以及被机器人分开的每个连通块里有几个人类有关.

正解

当前状态只和机器人位置、当前是哪个计划，以及被机器人分开的每个连通块里有几个人类有关.

全部记录到状态里暴力 DP.

正解

当前状态只和机器人位置、当前是哪个计划，以及被机器人分开的每个连通块里有几个人类有关.

全部记录到状态里暴力 DP.

注意每个时刻人类总数是已知的，所以只要记录两个子树里分别有多少人类.

正解

当前状态只和机器人位置、当前是哪个计划，以及被机器人分开的每个连通块里有几个人类有关.

全部记录到状态里暴力 DP.

注意每个时刻人类总数是已知的，所以只要记录两个子树里分别有多少人类.

状态总数为对每个点计算两个子树 size 的积加起来，是平方的.

正解

当前状态只和机器人位置、当前是哪个计划，以及被机器人分开的每个连通块里有几个人类有关.

全部记录到状态里暴力 DP.

注意每个时刻人类总数是已知的，所以只要记录两个子树里分别有多少人类.

状态总数为对每个点计算两个子树 size 的积加起来，是平方的.

前两种计划需要额外记录机器人是否移动过.

正解

当前状态只和机器人位置、当前是哪个计划，以及被机器人分开的每个连通块里有几个人类有关.

全部记录到状态里暴力 DP.

注意每个时刻人类总数是已知的，所以只要记录两个子树里分别有多少人类.

状态总数为对每个点计算两个子树 size 的积加起来，是平方的.

前两种计划需要额外记录机器人是否移动过.

转移的时候需要枚举下一步往里走的那个子树里的人类怎么分配.

正解

当前状态只和机器人位置、当前是哪个计划，以及被机器人分开的每个连通块里有几个人类有关.

全部记录到状态里暴力 DP.

注意每个时刻人类总数是已知的，所以只要记录两个子树里分别有多少人类.

状态总数为对每个点计算两个子树 size 的积加起来，是平方的.

前两种计划需要额外记录机器人是否移动过.

转移的时候需要枚举下一步往里走的那个子树里的人类怎么分配.

可以用前缀和优化做到 $O(1)$ 单次转移，总复杂度 $O(qn^2)$

花絮

这题是我两年前看着堵死的停车位出的.

改着改着就成这样了.

虽然看着题目描述很像桌游说明书，但这题和桌游没有一点关系.

Part III (出题人 *fzw*)

B 一棵树 by *fzw*

简单写一下题意。

正解

注意到当前询问形如权值为边权的权值求和，考虑树形 dp，直接的 dp 为 $f_{i,j}$ 表示 i 子树内有 j 个黑色节点的最小代价.

正解

注意到当前询问形如权值为边权的权值求和，考虑树形 dp，直接的 dp 为 $f_{i,j}$ 表示 i 子树内有 j 个黑色节点的最小代价.

有转移是易于计算的，子树的合并是树形背包的合并 $f_{x,j+k} = \min(f_{x,j} + f_{to,k})$，考虑当前点 x 到父亲这条边的代价是多少.

正解

注意到当前询问形如权值为边权的权值求和，考虑树形 dp，直接的 dp 为 $f_{i,j}$ 表示 i 子树内有 j 个黑色节点的最小代价.

有转移是易于计算的，子树的合并是树形背包的合并 $f_{x,j+k} = \min(f_{x,j} + f_{to,k})$，考虑当前点 x 到父亲这条边的代价是多少.

注意到，形如如果当前子树有 A 个黑点，子树外有 $k-A$ 个白点，代价为 $|2A-k|$.

正解

观察函数的形式，你注意到如果将 x 坐标作为黑点个数，y 坐标作为代价，则每次代价增量为一个下凸函数，有树形背包合并不影响函数凸性.

正解

观察函数的形式，你注意到如果将 x 坐标作为黑点个数，y 坐标作为代价，则每次代价增量为一个下凸函数，有树形背包合并不影响函数凸性.

注意到凸函数的维护可以尝试维护其差分数组，两个下凸函数做 min 卷积可以之际的视作差分数组的归并.

正解

观察函数的形式，你注意到如果将 x 坐标作为黑点个数，y 坐标作为代价，则每次代价增量为一个下凸函数，有树形背包合并不影响函数凸性.

注意到凸函数的维护可以尝试维护其差分数组，两个下凸函数做 min 卷积可以之际的视作差分数组的归并.

现在需要考虑的问题是往父亲方向的增量怎么处理.

正解

考虑代价增量的差分，形如一段前缀差分为负数，一段后缀差分为正数，中间一个点根据 k 的奇偶性讨论是否存在差分 $= 0$.

正解

考虑代价增量的差分，形如一段前缀差分为负数，一段后缀差分为正数，中间一个点根据 k 的奇偶性讨论是否存在差分 $= 0$.

有这一段前缀的长度总为定长度 $\lfloor \frac{k}{2} \rfloor$ 于是我们维护可并堆顶堆，使得左侧堆大小始终为前缀定长度，然后可以打加法标记实现.

正解

考虑代价增量的差分，形如一段前缀差分为负数，一段后缀差分为正数，中间一个点根据 k 的奇偶性讨论是否存在差分 $= 0$.

有这一段前缀的长度总为定长度 $\lfloor \frac{k}{2} \rfloor$ 于是我们维护可并堆顶堆，使得左侧堆大小始终为前缀定长度，然后可以打加法标记实现.

总复杂度 $O(n \log n)$.

H 二进制 by *fzw*

简要写一下题意。

正解

观察到当前过程形如你需要每次维护使得可以查找某些子串出现次数与位置，尝试观察询问，注意到查找子串的长度不会超过 $\log_2(n)$.

正解

观察到当前过程形如你需要每次维护使得可以查找某些子串出现次数与位置，尝试观察询问，注意到查找子串的长度不会超过 $\log_2(n)$.

我们对于每个二进制数都维护这个数在原串中的出现位置，注意到当前形如只需要检查原串的长度为 $\log_2(n)$ 的子串.

正解

观察到当前过程形如你需要每次维护使得可以查找某些子串出现次数与位置，尝试观察询问，注意到查找子串的长度不会超过 $\log_2(n)$.

我们对于每个二进制数都维护这个数在原串中的出现位置，注意到当前形如只需要检查原串的长度为 $\log_2(n)$ 的子串.

每次删除后我们需要重新检查删除这一段和前后 $\log_2(n)$ 个元素，每次检查加插入总复杂度为 $\log_2^3(n)$.

正解

观察到当前过程形如你需要每次维护使得可以查找某些子串出现次数与位置，尝试观察询问，注意到查找子串的长度不会超过 $\log_2(n)$.

我们对于每个二进制数都维护这个数在原串中的出现位置，注意到当前形如只需要检查原串的长度为 $\log_2(n)$ 的子串.

每次删除后我们需要重新检查删除这一段和前后 $\log_2(n)$ 个元素，每次检查加插入总复杂度为 $\log_2^3(n)$.

注意到，执行删除操作的次数不会超过 $\frac{n}{\log_2(n)}$ 有总复杂度在 $n\log_2^2(n)$ 级别.

正解

观察到当前过程形如你需要每次维护使得可以查找某些子串出现次数与位置，尝试观察询问，注意到查找子串的长度不会超过 $\log_2(n)$.

我们对于每个二进制数都维护这个数在原串中的出现位置，注意到当前形如只需要检查原串的长度为 $\log_2(n)$ 的子串.

每次删除后我们需要重新检查删除这一段和前后 $\log_2(n)$ 个元素，每次检查加插入总复杂度为 $\log_2^3(n)$.

注意到，执行删除操作的次数不会超过 $\frac{n}{\log_2(n)}$ 有总复杂度在 $n\log_2^2(n)$ 级别.

由于常数极小，所以可以直接通过。

Part IV (出題人 *Itst*)

A 排序大师 by *ltst*

Help!!!

Part V (出題人 *SpiritualKhorosho*)

D 多折较差验证 by *SpiritualKhorosho*

快写!

K 三步棋 by *SpiritualKhorosho*

快写!

L 勇闯末日塔 by *SpiritualKhorosho*

快写!

感谢倾听!