2020 Multi-University Training Contest 3, Editorial

By Tokitsukaze and Her Friends

出题人 (CF id)：Claris, bitetheDDDDt, cuizhuyefei, skywalkert, teitoku, tokitsukaze, winterzz1

验题人 (CF id)：Claris, Roundgod, dreamoon_love_AA, Subconscious, cuizhuyefei

- 特别感谢 Subconscious 在比赛当天凌晨验了一直没人验的 1011，还发现了 std 有锅
- 感谢 skywalkert 通宵修 statement，tls English best English
- 感谢 LCY 老师的支持和大家的参与！

以下是一些花絮：

1. 蔡队鸽了验题，原因：跟妹子约会。(这个仇我记住了.jpg)

2. 不到 ddl 不干活.jpg

Problem A. Tokitsukaze, CSL and Palindrome Game

Author: tokitsukaze, Claris

首先有个经典结论：

border： 对于一个长度为 L 的序列 A，若 $A[1,i] = A[L-i+1,L]$，则称 $A[1,i]$ 是 A 的一个 **border**。

令 a_i 表示 $A[1,i]$ 是否是 A 的一个 **border**，a_i 为 1 表示是，0 为不是。

则 $E(A) = \sum_{i=1}^{L} a_i \cdot m^i$，$m$ 为字符集大小，在本题中，$m = 26$。

此结论在 "国家集训队 **2018** 论文集" 中的第一篇，《浅谈生成函数在掷骰子问题上的应用》有介绍，例题是 **[CTSC2006]** 歌唱王国。

结合上述结论可知，本题只需要比较 $S_{a..b}$ 与 $S_{c..d}$ 的 **border** 长度组成的序列的字典序即可。

在本题中，由于 S 为回文串，根据 **border** 定义，$S[1,i] = S[L-i+1,L]$，而 $S[1,i] = reverse(S[L-i+1,L])$，所以 $S[L-i+1,L] = reverse(S[L-i+1,L])$，所以 S 的 **border** 也是回文串。

那么只要找到 S 在回文自动机上对应的节点，沿着 **fail** 指针跳到根，所经过的每个节点都是 S 的 **border**。只要把这些节点，每个节点所代表的回文串的长度拿出来，变成序列，从大到小排序后，再比较两个序列的字典序大小即可。

举个例子：

$S_{a..b} = aaaaa$，**border** 的长度序列为：**5,4,3,2,1**

$S_{c..d} = ababa$，**border** 的长度序列为：**5,3,1**

字典序 $5,4,3,2,1 > 5,3,1$，所以 $E(S_{a..b}) > E(S_{c..d})$，所以输出 **cslnb**

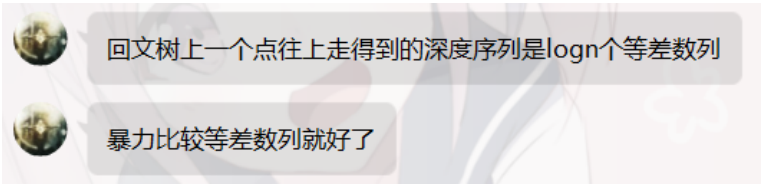
比较字典序有两种做法：

做法 **1(by tokitsukaze, 请见 sjf_nlogn.cpp)**：

建出回文自动机的 **fail** 树，用树上倍增找到 $S_{a..b}$ 和 $S_{c..d}$ 在树上对应的节点，假设分别为 x, y。

根据 **fail** 树性质，从根到某个节点的链上，节点代表的长度是递增的。那么直接从 x 和 y 开始，同时倍增往上跳相同深度，用 **hash** 判断往上跳的那条链上节点所代表的长度是否都相同，跳到第一个长度不同的节点，然后比较即可。

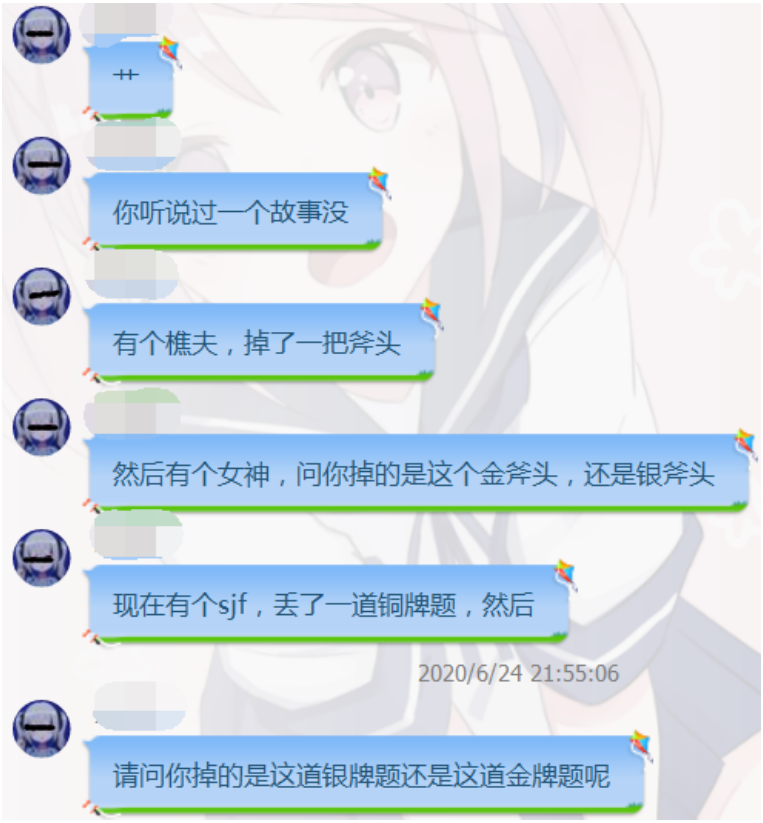
做法 **2(by Claris, 请见 palgame.cpp)**：



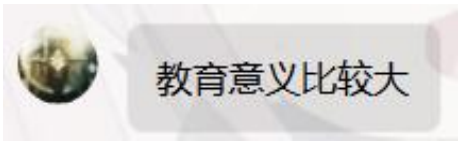
复杂度都是 $O(n \log n)$

P.S.

这题本来是个比较简单的 **fail** 树上计数题，可能是个铜牌题。**Claris** 觉得太裸，想了不到 **10** 分钟，就蹦出了现在这个题，然后我不知道那个 "经典结论"，感觉是个金牌题，于是就有了下面这个故事。



别问，问就是



Problem B. Lady Layton and Stone Game

Author: bitetheDDDDt

首先我们考虑合并约束为 [2,R] 时候，直接按 k-ary Huffman coding 的方式进行构造即可，每次合并 $x, R, \ldots, R$ 堆，记为长度为 p 的序列 m（由于可能 $(\sum_{i=1}^{n} a_i) - 1$ 不为 $R-1$ 的倍数，所以第一次合并的堆数 x 可能小于 R）。

此时我们将情况扩展到 [L,R]，这个时候可能需要不只一次让合并小于 R 才能使得 $sum - 1 = \sum m_i - 1$，所以我们应该让合并序列变为 $L, \ldots, L, x, R, \ldots, R$ 因为这样使得序列 m 在长度不增加的情况下可以得到尽可能小的字典序。

注意，显然 $\sum m_i - 1$ 的值域在 [$p*(L-1)$,$p*(R-1)$], 用右边界求得最小的 p 之后，如果 $(\sum_{i=1}^{n} a_i) - 1$ 不在值域内，显然继续扩大 p, 也仍然不在值域内，此时无解。

结论的证明可以参考 "1002 口胡证明 (by winterzz1).pdf"

Problem C. Tokitsukaze and Colorful Tree

Author: tokitsukaze

容易发现，节点 x 的贡献，不包括根到 x 这条链上的节点，和 x 子树中所有的节点。

但直接维护异或的贡献不太方便。异或无非就是 0 与 1，1 与 0 的组合。所以拆位，对于每一位，维护每个节点，根到节点这条链上 0 与 1 的个数，和节点子树的 0 与 1 的个数，再用总的减去，就是每一位每个节点的贡献。

利用 **dfs** 序维护子树，是个常规操作。而维护根到节点这条链，若是使用树链剖分维护的话，复杂度 $O(20n\log^2 n)$ 级别，无法接受。

注意到修改都是单点修改。单点修改，树链查询，可以用 **dfs** 序 + 差分转换为单点修改，区间查询。修改节点 x 的权值，当且仅当 y 在 x 的子树中时，x 对 y 的值有贡献。用差分前缀和，y 的权值等于 **dfs** 序中 $[1, l[y]]$ 的区间和。更新就是单点修改：$add(l[x], v), add(r[x]+1, -v)$。

复杂度 $O(20n\log n)$

现在有两种姿势来做这道题。

1. 对每种颜色都用一个线段树来维护。**std** 最开始就是这么写的，但发现常数巨大，大概是现在的 4 到 6 倍。

2. 离线颜色，把每个拆分成删除和插入，然后用树状数组进行维护，跑的飞快。

P.S.

1. 由于卡树剖的数据难造，时限放的可能比较紧 (虽然基本是 **std** 的两倍，但是 **Claris** 的 **std** 实在是太快了**.jpg**)，于是变成了一个代码又臭又长还可能被卡常的题。

2. 赛前两天让 **Claris** 写了个暴力爬树链，结果比 **std** 还快。紧急加了一组" 人" 字形数据，因为单链的话，不管什么修改答案都是 0，担心被特判判过去。

3. 数据构成：**6** 组卡树剖数据 **+1** 组完全随机 **+1** 组卡暴力爬树链

4. 本题的 **idea** 是某天做梦梦见的 (对，你没看错)，梦见的是没有异或也没有修改的版本。

赛后

5. 赛前怕卡常，赛中一个个跑的都比我们快，怎么回事**.jpg**

6. 但还是有人被卡常！对不起！

Problem D. Tokitsukaze and Multiple

Author: tokitsukaze

题意其实就是：把序列切分成若干段，然后看看有几段的和是 p 的倍数。枚举所有切法，把所有结果取 max 就是答案。（题面说成 “合并”，是为了增加一些迷惑性）

假设把所有和是 p 的段拿出来，问题就转化为，选最多的不相交线段能选几条。于是这就变成一个经典问题。

引用百度上某 blog 的贪心做法：“按右端点排序，能选就选，不能就不选。因为如果不能选的话，它的加入一定会导致至少一条线段的退出，并且还使得当前线段集的最右端点右移了，一定不优。”

那么这个题就是用 map 维护前缀和，当遇到 $[L, i]$ 的和是 p 的倍数时，根据 “能选就选”，答案 $+1$，然后把 map 清空即可。

上述做法请见 sjf-std.cpp

还有 dp 做法，请见 Roundgod-multiple.cpp

Problem E. Little W and Contest

Author: winterzz1

首先一个显然的算法是用并查集维护关系，同时记录每个连通块中战斗力为 1、2 的人数。

接下来对于每一个操作计算组合数贡献即可。

std 使用了母函数计算贡献，实际上用母函数计算组合数贡献，从时间复杂度和常数的角度讲都并不算优秀（相比直接计算贡献的话）。

它最大的好处就是可以将稍复杂的排列组合问题转换成模板化，套路化的多项式问题。

能够在一定程度上解决选手的思考时间和思维难度。

并且有时，转化成多项式卷积后可以进行一些额外的优化。

例如本题，我们可以构造一个二元多项式。

借助 x 表示人数，借助 y 表示战斗力。

则多项式中的项 kx^ay^b 表示：组成由 a 人组成战斗力为 b 的团队，方案总数为 k。

对于一个有 n 人战斗力为 1，m 人战斗力为 2 的连通块，则可以用多项式 $(nxy + mxy^2 + 1)$ 来表示。

然后就简单了嘛... 要添加连通块只要多项式乘进去，要删除连通块消除贡献只要多项式除回来即可。

由于幂较小，借助二维背包来做多项式乘除法即可。

正贡献 01 背包为多项式乘法，负贡献完全背包为多项式除法。

上述做法请见 std.cpp

直接算贡献做法请见 dreamoon-ac.cpp

Problem F. X Number

Author: teitoku, winterzz1

看上去第一眼好像是数位 DP，但是难点在于状态好像没有共性，无法表示。

要想做出本题的话需要对数位 DP 以及对其中的 limit 限制（表示当前 bool）有一定理解。

首先数位 DP 的复杂度为 O(进制数 * 位数 + 进制数 * 新增状态数)

它并不像其他的记忆化搜索一样，搜索到最后复杂度会降低到 O(1) 直接返回。

即使最终再也没有新状态，其复杂度仍然为 O(进制数 * 位数)

而这个 O(进制数 * 位数) 是 dfs 进行数位枚举爆搜带来的复杂度。

而这部分正是 limit 为真表示有边界限制时才需要进行的。

当 limit 为假的时候表示没有边界限制，相当于进制内的数字均可以使用。

正常情况下的数位 DP 本应该分情况讨论：

当 limit 为真：继续 dfs 展开

当 limit 为假：进行记忆化搜索，以便于之后 O(1) 返回。

明白原理之后就可以开始魔改了。

如果 limit 为假，即没有限制时，有能够快速计算出当前情况的数学方法（比如 O(1) 或者 O(log)）。

那就不需要记忆化搜索了（数位 DP 没有 DP，还行）。

回到本题，例如用这种方式进行 dfs 搜索 $1-123456789$ 中符合条件的数字时

比如搜索到 123455XXX

由于在第 6 位取消了 limit 限制，所以 X 可以是 $0-9$ 任意的数字。

问题就转化成了，现在有 1 个 1,1 个 2,1 个 3,1 个 4,2 个 5，和 3 个任意数字。

问有多少种方案可以令某个数字成为众数。

这明显是一个组合数问题，运用我们的组合数知识计算贡献或者用指数型母函数是可以解决这个问题的。

不过本题需要继续优化，不然仍然会 TLE。

我们注意到对于非众数的数字，其实我们并不关心他们具体是谁

例如假设我们需要 3 成为众数，那么

已有 1 个 1,2 个 2, 和已有 2 个 1,1 个 2

这两个问题完全等价。

我们可以用最小字典序表示，问题，来进行记忆化处理。

这样总状态数为 18 的拆分数，大大降低了复杂度。

Problem G. Tokitsukaze and Rescue

Author: Claris

边权随机的情况下，最短路的边数很少。

所以只要每次跑一下最短路，抓一条最短路出来，枚举删除最短路上的哪条边，然后递归，变成删除 $(k-1)$ 条边的子问题。

重复这过程直到 $k=0$，然后再跑一次 1 到 n 最短路，把结果取 max 即可。

复杂度 $O(n^2 * c^k)$，c 为最短路边数。

P.S.

题面背景来源于炼金工作室系列第 18 作 "菲丽丝工作室：不可思议之旅的炼金术士"，简称 A18，是不可思议三部曲中的第二部。statement 第一段基本上是复制 steam 上的游戏简介。

炼金系列强力推荐！(虽然不建议 A18 入坑)

Problem H. Triangle Collision

Author: winterzz1

灵感来自用激光笔往镜子中射入激光被多次反射。

人眼从激光笔的视角来看，光束并没有被 “反射”，而是穿入了 “镜子中的世界”。

这道题也是利用了这个思路。不要去计算入射角反射角之类。

首先二分答案，考虑已知时间求在时间内的碰撞次数。

我们把三角形边缘看成是镜子，这样的话问题将转化成在无限密铺等边三角形的空间中小球的运动轨迹穿过了多少条边缘线。

首先只考虑这些线中与 x 轴平行的线，这个是最好算的，只要求 $abs\left(\left\lfloor\frac{y}{L*\frac{\sqrt{3}}{2}}\right\rfloor\right)$

那对于剩下两种歇着的直线，我们可以将坐标点绕等边三角形的中心点分别旋转 120,240 度。

即可算出小球在时间内的碰撞次数。

Problem I. Parentheses Matching

Author: skywalkert

本场第二简单题，可以采取贪心策略。

注意到，如果有最短长度的合法解，则不会存在一个被替换为 "(" 的 "*" 在一个被替换为 ")" 的 "*" 右侧。假设存在这样的情况，则交换它们不会使解不合法，但交换后可以发现它们都可以替换为 ""，这意味着存在更短长度的解，从而导出矛盾。

假设已经得到一个最短长度的合法解，则对于每个被替换为 ")" 的 "*"，如果其右侧存在被替换为 "" 的 "*"，则我们可以交换这两个字符的位置，新解的字典序不会更大，对于 "(" 同理。

因此最短长度的字典序最小合法解一定是替换最左侧的一部分 "*" 为 "("，最右侧的一部分 "*" 为 "("，而且它们的数量 c_l, c_r 满足 $c_l - c_r$ 是定值，于是我们可以二分查找最小的 c_l 使得构造出的串为合法解。

实际上，我们可以注意到 c_l 和 c_r 的值是可以直接根据已有括号算出的，因为我们加括号就是为了抵消对应的括号带来的限制。

时空复杂度：$\mathcal{O}(n)$

Problem J. Play osu! on Your Tablet

Author: skywalkert

考虑到目标同时出现、支持滑块这些部分不是解决这道题的关键所在，因此删去了这些内容，感兴趣的同学可以去玩玩 **osu!** 的标准模式。

令 $dist(i,j)$ 表示第 i 个目标到第 j 个目标的距离，定义 $dp(i,j)$ 表示在考虑前 i 个出现的目标时，一根手指最后放在了第 i 个目标，另一根则在第 j 个目标，这种情况下产生的最小代价。初始情况可以为 $dp(1,j)=0$，表示两只手的起始位置分别是第 1 个目标和第 j 个目标。$dp(i)$ 到 $dp(i+1)$ 的转移有两种：

- 手指从第 i 个目标移到第 $(i+1)$ 个目标：$dp(i+1,j) \leftarrow dp(i,j)+dist(i,i+1)$；
- 手指从第 j 个目标移到第 $(i+1)$ 个目标：$dp(i+1,i) \leftarrow dp(i,j)+dist(j,i+1)$。

注意到第一种转移相当于给每个 j 对应的信息增加定值，第二种转移相当于对所有 j 求出信息最小值后对单个 dp 值进行更新，这启发我们使用数据结构维护信息。

令 $f(i,j)=dp(i,j)-\sum_{k=1}^{i-1} dist(k,k+1)$，则转移变化为

- $f(i+1,j) \leftarrow f(i,j)$；
- $f(i+1,i) \leftarrow f(i,j)+dist(j,i+1)-dist(i,i+1)$。

这本质上可以复用信息从而消除第一维 i，所以我们只用考虑对 j 维护信息。

注意到距离的定义是曼哈顿距离，为了化简其中的绝对值符号，我们可以以第 $(i+1)$ 个目标为中心把空间划分成四个部分，每个部分里 j 的贡献是同一种形式，因此我们可以用数据结构维护二维平面上的区域询问和单点修改。

这里数据结构有很多选择，我们可以使用线段树套线段树（也即划分树，又名时代的眼泪）维护信息，也可以用树状数组代替一部分线段树，甚至可以使用 KD 树来进行操作，前提是写得足够正确、效率够高，注意这里不能使用空间复杂度为 $\mathcal{O}(n\log^2 n)$ 的树状数组套函数式线段树。

此外，这道题没有强制在线（例如根据前 i 个目标的 dp 信息来解码第 $(i+1)$ 个目标的位置），我们也可以发现操作过程和动态二维数点问题的相似性，从而采取分治配合扫描线与树状数组进行操作。

时间复杂度：$\mathcal{O}(n\log^2 n)$ 或 $\mathcal{O}(n\sqrt{n})$

空间复杂度：$\mathcal{O}(n)$ 或 $\mathcal{O}(n\log n)$

Problem K. Game on a Circle

Author: cuizhuyefei

为表述方便，这里令 $q=1-p$。

令 a_i 为 c 号点是第 $i+1$ 个消失的概率，我们希望求出 $A(x)=\sum a_i x^i$。则有

$$\begin{aligned}A(x)&=\sum_{t=0}^{\infty}q^tp\left(q^{t+1}+(1-q^{t+1})x\right)^{c-1}\left(q^t+(1-q^t)x\right)^{n-c}\\&=\sum_{t=0}^{\infty}q^tp\left(q^{t+1}(1-x)+x\right)^{c-1}\left(q^t(1-x)+x\right)^{n-c}\\&=p\sum_i\sum_j\binom{c-1}{i}\binom{n-c}{j}\sum_{t=0}^{\infty}q^i(1-x)^{i+j}x^{n-1-i-j}q^{t(1+i+j)}\\&=p\sum_i\sum_j\binom{c-1}{i}\binom{n-c}{j}q^i(1-x)^{i+j}x^{n-1-i-j}\frac{1}{1-q^{1+i+j}}\end{aligned}$$

记 $f_k=\sum\limits_{i+j=k}\binom{c-1}{i}\binom{n-c}{j}q^i$，则有

$$A(x)=p\sum_i\frac{f_i}{1-q^{i+1}}(1-x)^ix^{n-1-i}$$

由于 c 为常数，f_n 是一个卷积的形式，可以 FFT。现在考虑求解 $A(x)$，继续推导可知

$$[x^{n-i+j-1}]A(x)=p\sum_i\sum_j\binom{i}{j}(-1)^j\frac{f_i}{1-q^{i+1}}$$

这也是卷积的形式，问题即可使用 2 次 FFT 得到解决。

事实上我们可以做得更快。记 $F(x)$ 为 f_i 对应的普通型生成函数，我们发现 $F(x)=(1+qx)^{c-1}(1+x)^{n-c}$

通过对生成函数求导，可以得到

$$F'(x)=(c-1)q\frac{F(x)}{1+qx}+(n-c)\frac{F(x)}{1+x}$$

对比系数，得到 f_i 的递推式：

$$f_{i+1}=\frac{((c-1)q+n-c-(q+1)i)\,f_i+q(n-i)f_{i-1}}{i+1}$$

综上所述，我们只需要 1 次 FFT 就解决了这个问题，时间复杂度 $\mathcal{O}(n\log n)$。

时限是 std 的 2 倍，所以你即使没有使用最后这个优化也大概率可以 AC。