AtCoder试题选讲

by SYC from Shaoxing No.1 High School

AtCooder简介

- 一个日本的编程比赛网站
- 基本每周都会有比赛
- 一场比赛的时间大约在2小时左右,每场比赛有4到6个难度递增的题目。
- 主要举办3个不同难度的比赛,ABC,ARC,AGC
- B = Beginner, R = Regular, G = Grand
- 今天带来的主要是AGC中的试题
- AtCoder题风类似于TC,以小码量的脑洞题和智商题为主。

寿司

- 有个长为 N 数列 A, 初始全为 0.
- 有 Q 次操作,每次操作有两个参数 X, Y:
  - 1.在A[1],A[2],..A[X]中找出最小的数,如果有多个找下标最小的一个,设找到了u.
  - 2.A[u] = A[u] + 1
  - 3.重复这个过程 Y 次.
- 输出最后 A 的序列.
- N, Q <= 10^5, X <= N, Y <= 10^12.
- From Square869120Contest #3

Analysis

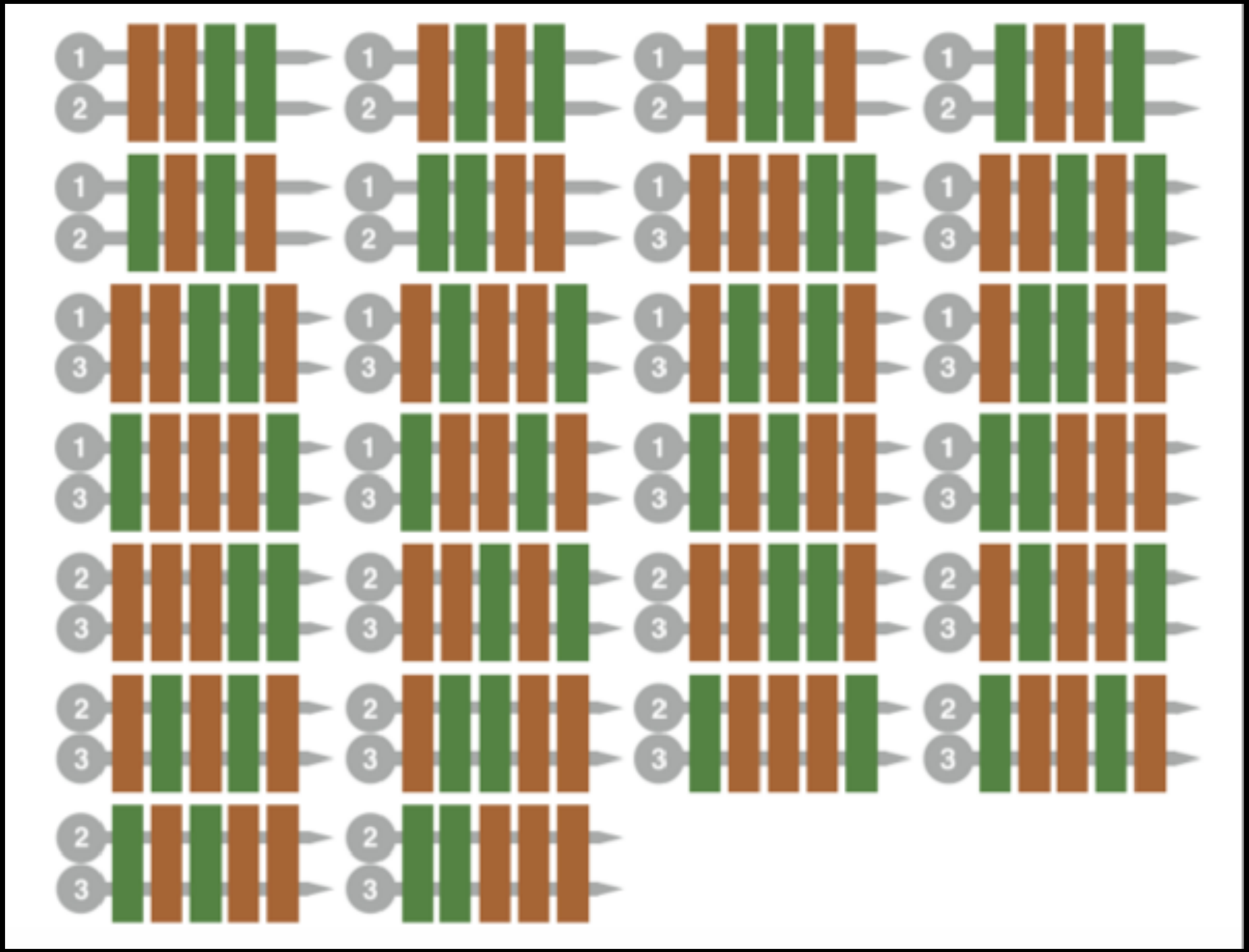
- A[i] >= A[i + 1]恒成立
- 可以把数列按照相同的数字分成一段段的.
- 然后用线段树模拟这些段即可.
- 时间复杂度O((N+Q)logN).

BBQ Hard

- 有N个食物包,第i个内有1个竹签,A[i]块肉,B[i]个菜
- 竹签是两两可区分的,但是肉和菜不是
- 你要做一个烤串,你会挑选两个数字 i, j(i < j), 然后把 i, j 两个包中的食材和竹签都拿出来串成一串,食材可以任意排列.
- 问有多少种不同的方案(mod 10^9+7)
- N<=2*10^5, A[i], B[i] <= 2000
- From AGC001

Sample:

- Input:
  - 3
  - 1 1
  - 1 1
  - 2 1
- Output:
  - 26



FFT

- 答案就是 $$\sum_{i=1}^{n}\sum_{j=i+1}^{n}\frac{(A[i]+A[j]+B[i]+B[j])!}{(A[i]+A[j])!(B[i]+B[j])!}$$
- 二维FFT?
- 4000*4000的规模,非NTT模数
- 看来需要O(松)优化.

A Simple A[i]^2 DP

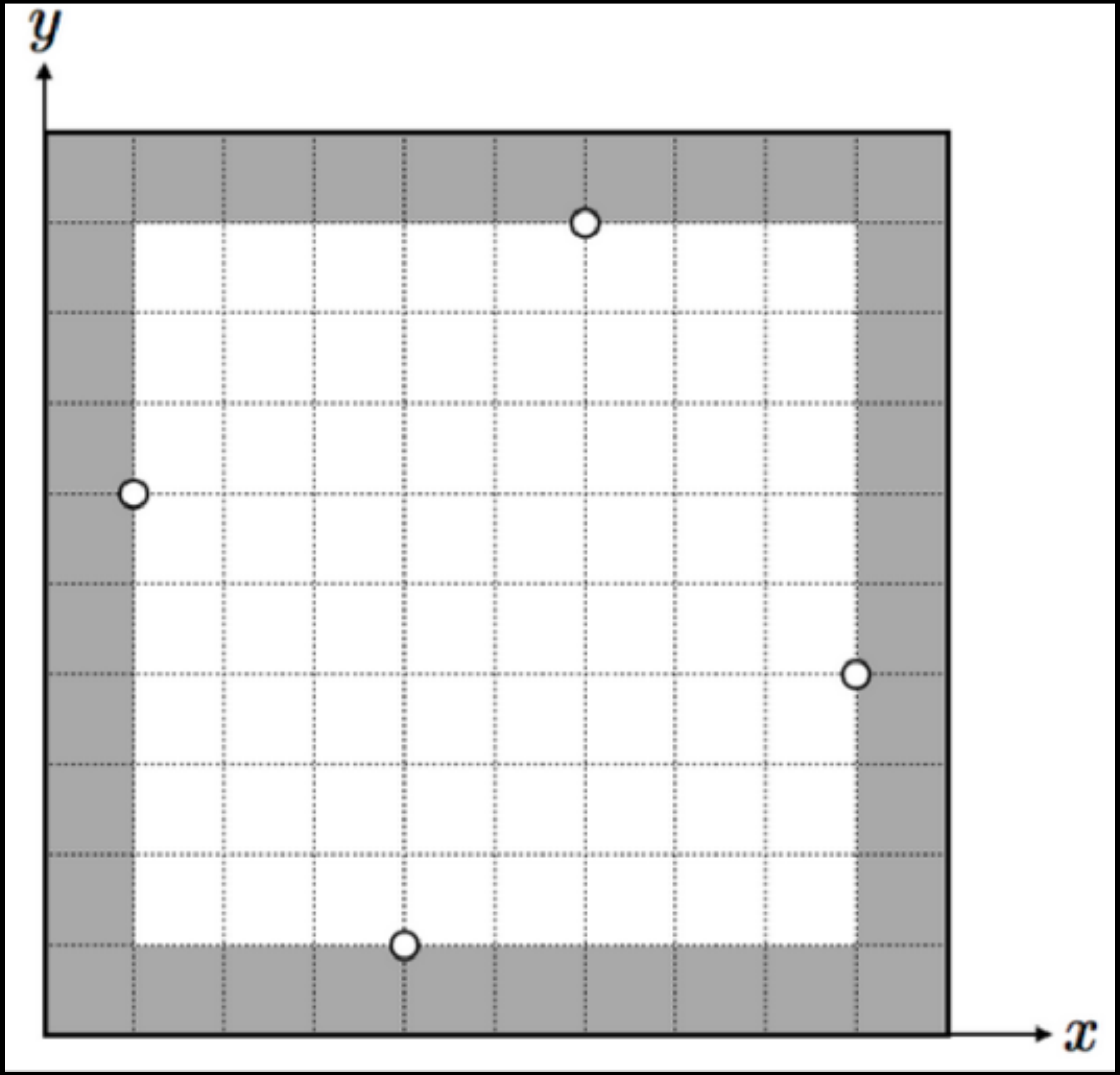
- 这个组合数是不是在哪里看到过?
- 从左下角走到右上角的方案数!
- 对每个食物包(A[i], B[i])在平面上放两个点:(A[i], B[i]), (-A[i], -B[i])
- 然后建立一个源点S,向第三象限的所有点连边.
- 建立一个汇点T，第一象限的点向T连边.
- 每个点向右向上连边.
- 然后S->T的路径数就是答案.
- 时间复杂度O(A[i]^2).
- 代码主体部分不超过10排.

Snuke's Coloring 2

- 在一个平面直角坐标系上,横坐标范围是[0,W],纵坐标是[0,H]
- 在平面上有N个点,第i个点的坐标是(X_i,Y_i)
- 开始的时候整个平面都是白的.
- 对于每个点,你必须做以下4个操作之一:
  - 1.将$X<X_i$的部分全部涂黑.
  - 2.将$X>X_i$的部分全部涂黑.
  - 3.将$Y<Y_i$的部分全部涂黑.
  - 4.将$Y>Y_i$的部分全部涂黑.
- 要求最大化最后白色部分的周长.
- $N<=3*10^5,W,H<=10^9$.
- From ARC063

Sample:

- Input:
  - 10 10 4
  - 1 6
  - 4 1
  - 6 9
  - 9 4
- Output:
  - 32



Analysis

- 转换成找一个周长最大的矩形使得内部没有包含任何一个点。
- 分治?
- 每次考虑经过中线的矩形。
- 扫描线
- 单调栈+线段树即可。
- $T(n) = 2T(n/2)+O(n\log n)$
- $T(n) = O(n\log^2 n)$

Analysis

- 答案至少是 2 * max(W, H) + 2
- 考虑选择一行或者一列.
- 因此最优解至少会穿过 x = W/2 或 y = H/2中的一条.
- 也就是说分治的时候只要算前两层就好了233。
- 时间复杂度O(NlogN)

Candy Piles

- 有 N 堆糖果，第 i 堆有 A[i] 个糖。
- Snuke 和 Ciel 在玩游戏，Snuke 先手。
- 两人轮流进行,每次必须挑选做两个操作之一:
  - 1.将当前最大的那堆糖果全部吃完
  - 2.将当前存在每堆糖果各吃一个
- 吃完最后一个的人输，问先手是否必胜。
- N <= 10^5, A[i] <= 10^9
- From AGC002

Analysis

- 把所有的A[i]排序,把这张图画成一个不规则轮廓的一个图形.
- 问题可以转化成这样:
  - 假设一开始(0, 0)处有一个棋子.
  - 两个人轮流移动,每个人把棋子向右向上走,走到边缘上的人输.
- SG函数!
- 打表可得SG[i][j] = SG[i + 1][j + 1].
- 这样一个点的SG可以变成计算“角落上”的SG值.
- 边缘上的SG特判即可.
- 时间复杂度O(N)

Zig Zag MST

- 有 N 个点,标号0~N-1,排成一个环,有 M 次操作。
- 每次操作有三个参数A[i], B[i], C[i]
- 它会在这个图上加无穷多条边。
- 具体地可以用一份代码来表示这个过程：AddEdge(x, y, z) 表示 x 向 y 连权值为 z 的边.

```
void Work(int a, int b, int c){
    AddEdge(a % N, b % N, c);
    Work(b, a + 1, c + 1);
}

for(int i = 0; i < M; i ++){
    Work(A[i], B[i], C[i]);
}
```

- 求连完所有边以后的MST大小
- N, M <= 10^5
- From CodeFestival 2016 Final

Analysis

- 考虑它每次的连边:
  - (A, B, C)
  - (B, A + 1, C + 1)
  - (A + 1, B + 1, C + 2)
  - …
- 考虑Kruskal:
  - 连(B, A + 1, C + 1)时(A, B)必然联通,因此这条边等价于(A, A + 1, C + 1)
  - 连(A + 1, B + 1, C + 2)时(B, A + 1)必然联通,因此这条边等价于(B, B + 1, C + 2)
  - …
  - 因此一次操作可以等价于(A, B)之间的连边和相邻点之间的连边!

Analysis

- 相邻点之间的连边可以扫描线解决.
- 这样子边数就O(N+M)了.
- 时间复杂度O((N+M)logN)

Robot and String

- 定义一种字符串的消除方法:
- 每次挑选两个相邻且字符一样的位置,如果有多个,挑选位置最小的那个.将这两个位置上的字符消去,并用字母表中下一个字符替换它们.
- 特殊地,如果你删去的是'z',那么它们将消失
- 'xxyz'->'yyz'->'zz'->''
- 给出一个字符串S,有Q次询问
- 每次询问给出一个区间[l, r], 询问子串S[l : r]能否被消完
- |S|<= 5*10^5, Q<=10^5
- From Mujin Programming Challenge 2017

Sample:

- Input:
    - axxxxza
    - 2
    - 1 7
    - 2 6
- Output:
    - No
    - Yes

axxxxza -> ayxxza -> ayyza -> azza -> aa -> b

xxxxz -> yxxz -> yyz -> zz -> “”

Analysis

- 考虑第一个字符什么时候被消掉。
- 当第一个字符被消掉的时候一定消除了一个前缀，设这个前缀长度为L.
- 然后就变成了i+L什么时候被消掉.

Analysis

- 用F[i][j]表示考虑消除[i..n],i第一次变成j的位置.
- F[i][j] = F[F[i][j - 1]][j - 1]
- F[i][‘z’ + 1]就是i第一次被消除的位置.
- 询问的时候对F[i][‘z’ + 1]倍增即可.
- 时间复杂度O(N * 26 + (Q + N)log N).

Wide Swaps

- 有一个长为N的排列,P[1],P[2]..P[N]
- 你可以做任意次操作，每次挑选两个位置 i, j满足 $i - j >= K$(K是一个给出的常数) 且 $|P[i] - P[j]| = 1$
- 然后交换P[i]和P[j]
- 求出排列P可能的最小字典序.
- $N<=5*10^5$.
- From AGC001

Wide Swaps

- 先给出最后的结论:
  - 令 P 是初始排列，P'是最后的排列
  - 如果 $P[x] < P[y]$ 且 $|x - y| < K$
  - 那么$P'[x] < P'[y]$
- 这个 P' 的充要条件

- From AGC001

Transformation

- 这个结论比较不显然
- 考虑这个问题的另一个等价版本:
  - 令 Q 是一个排列满足 Q[P[i]] = i.
  - 那么一次操作就相当于交换 Q 中相邻且相差至少为 K 的两项
  - 最后让 1 在 Q 尽量靠前，其次 2 在 Q 中尽量靠前。。。。

Transformation

- 令初始的排列是 Q, 最后的排列是 Q’.
- 注意到:
  - 如果 $x < y$ 且 $|Q[x] - Q[y]| < K$, 那么 Q’ 中 Q[x] 和 Q[y] 的相对顺序一定没变.
- 证明:
  - 若相对顺序改变，必然有一步操作是交换了Q[x] 和 Q[y].
  - 而这个操作是不合法的。
- 因此结论成立

Transformation

- 这是 Q’ 的充要条件.
- 因为我们每次在Q中都可以找到相邻的数使得他们在Q’中的相对顺序改变并且交换他们.
- 于是最开始的结论也就成立了.

Transformation

- 最后变成了一个图论问题:
  - 如果 $|x - y| < K$, 且 $P[x] < P[y]$
  - 那么 x 向 y 连一条有向边.
  - 那么这张图的所有拓扑序列都是合法的P’
  - 求最小字典序的拓扑序列.

Analysis

- 费了很大的力气,终于转成一个比较传统的问题了。。。
- 最小字典序的拓扑排列只需每次挑选入度为 0 且编号最小的点就行了.
- 这个可以用优先队列来实现.
- 时间复杂度O((N+M)logN).

Analysis

- 等等。。这张图的边数，似乎是O(N^2)的?
- 用树套树大力优化边数!
- 复杂度O(Nlog^3N)

Analysis

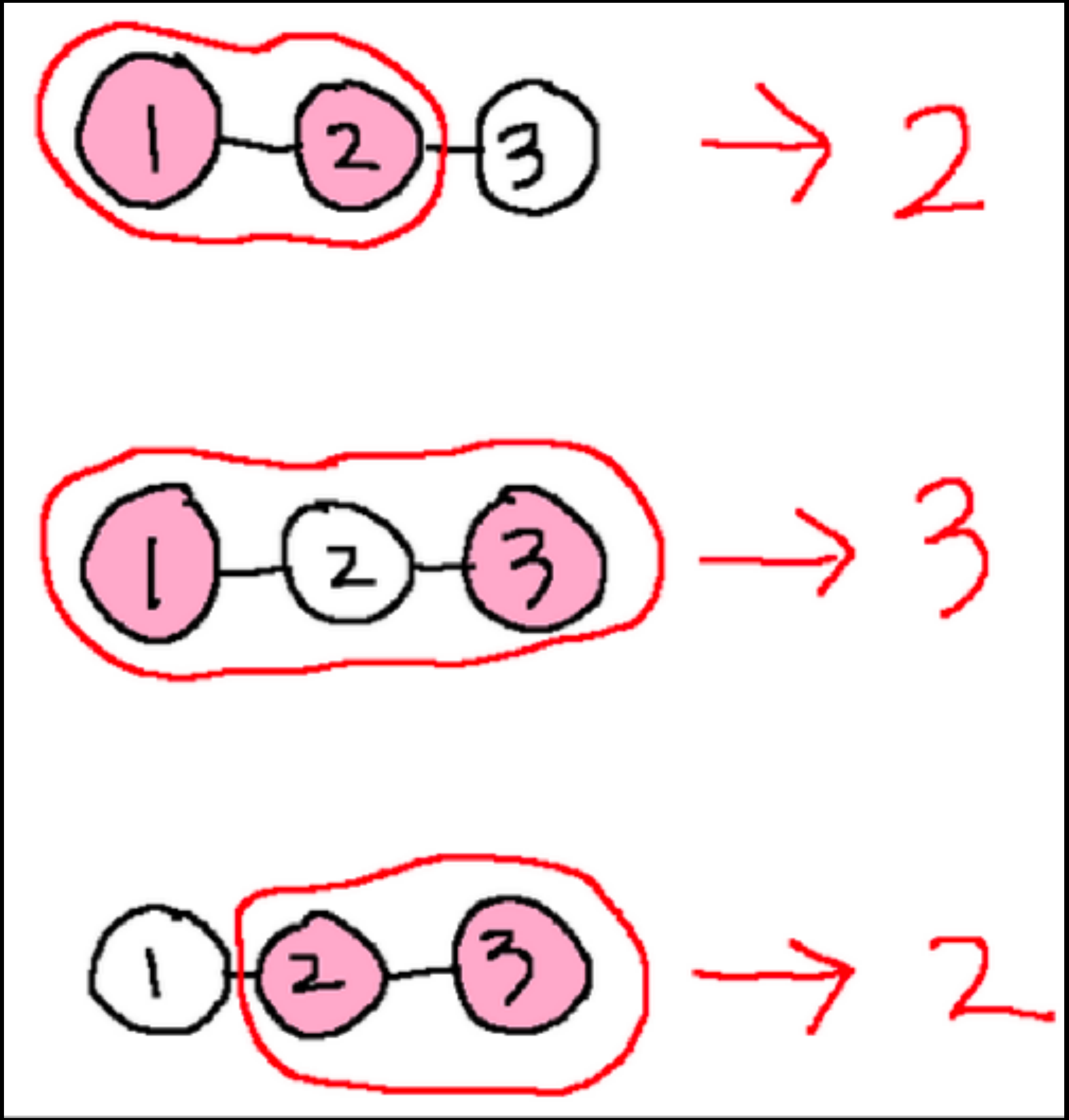
- 其实边数是可以优化到 O(N) 的.
- 对于一个 x，只需考虑 [x - K + 1, x - 1] 中 P[x] 的后继以及 [x + 1, x + K - 1] 中 P[x] 的后继就好了.
- 因为剩下的边会由它们帮你连好.
- 复杂度O(NlogN).

Many Easy Problems

- 有一个N个点的树.
- 你在树上随机了K个点,然后找出了包含这K个点的最小联通子图的大小(点数)X.
- 求 X 的期望乘上C(n,k)的结果
- 可以证明是个整数,对924844033(可以NTT)取模.
- 对每个1 <= K <= N都输出答案.
- N <= 200000
- From AGC005

Samples:

- Input:
  - 3
  - 1 2
  - 2 3
- Output:
  - 3
  - 7
  - 3



Analysis

- 根据期望的线性性，考虑某个点被包含的方案数.
- 用 F(x, K) 表示 x 被包含的方案数.
- 这个不太好算，可以转成不被包含的方案数。
- 也就是删掉这个点后每个联通块中选K个点的方案数的和.

$$F(x,K)=C(N,K)-\sum_{i=1}^{tot}C(sz[i],K)$$

Analysis

- 用 G[i] 表示 i 在组合数上方出现时的系数和.
- 一个树上的节点 x 对 G 的影响就是:
  - G[N] = G[N] + 1
  - G[sz[i]] = G[sz[i]] - 1
- $Ans[k] = \sum_{i=k}^{n} G[i] * C(i,k)$
- $Ans[k] * k! = \sum_{i=k}^{n} G[i] * i! / (i-k)!$
- 直接FFT即可.

Leftmost Ball

- 你有N*K个球,它们有N种颜色,每种K个.
- 颜色的标号是1~N.
- 你可以将它们任意排列.
- 在排列完以后把每种颜色的球最靠左的那个涂成第0种颜色.
- 问最后会有多少种不同的方案.
- 对10^9+7取模.
- N,K<=2000
- From AGC001

Sample:

- Input:
  - 2 2
- Output:
  - 4

(1, 1, 2, 2) -> (0, 1, 0, 2)
(2, 2, 1, 1) -> (0, 2, 0, 1)
(2, 1, 2, 1) -> (0, 0, 2, 1)
(1, 2, 2, 1) -> (0, 0, 2, 1)
(1, 2, 1, 2) -> (0, 0, 1, 2)
(2, 1, 1, 2) -> (0, 0, 1, 2)

Analysis

- K = 1时答案就是1.
- 考虑K>1的情况

Analysis

- 考虑合法方案的充要条件:
  - 如果第i个位置是0，那么[i, n*k]中出现了K-1次颜色的种类数应不少于0的个数.
- 用F[i][j]表示出现了i个0,j个颜色出现了K-1个数字的方案数
- F[i][j] -> F[i + 1][j] (if i < j)
- F[i][j] * C(i + K * j, K - 1) -> F[i][j + 1]
- 时间复杂度O(N^2)

123 Pairs

- 有2N个数字,1~2N.你要将他们分成N个Pair,使得每个数字恰好在一个Pair中.
- 要求如果 i, j 在一个Pair中,那么|i - j| <= 3
- 并且有恰好有A个Pair满足 |i - j| = 1
- 并且有恰好有B个Pair满足 |i - j| = 2
- 并且有恰好有C个Pair满足 |i - j| = 3
- 计算有多少种不同的方案.
- 对10^9+7取模.
- N, A, B, C <= 5000, A + B + C = N
- From Code Festival Grand Final

Analysis

- $C = 0$ 怎么做?
- $C \neq 0$ 怎么做?

Analysis : |i - j| <= 2

- 用 L1 表示标号差为 1 的连接, L2, L3 类似.
- 考虑 1 与谁相连
- 如果1 - 2相连,就变成 n - 1 对点的子问题.
- 如果1 - 3相连,那么2 - 4必须连, 就变成n - 2对点的子问题.
- 也就是每次消耗 L1 * 1 或者 L2 * 1.

Analysis : |i - j| <= 3

- 考虑第一个位置的连接情况.
- 考虑怎么连接才能填满一个前缀.
- L1 * 1 : 1 - 2
- L3 * 1 + L1 * 1 : 1 - 4, 2 - 3
- L3 * 3 : 1 - 4, 2 - 5, 3 - 6
- L2 * 2 + L3 * v(v >= 0) : 1 - 3, 2 - 5, 4 - 7…, 2v - (2v + 2)

Analysis

- 设这 4 种各消耗了 x1, x2, x3, x4 个。
- 并设第 4 种中 v 的和是 S.
- 枚举 x1, x3
- 那么 x2 = A - x1, x4 = B / 2, S = C - x3 * 3 - x2
- 方案数可以用带标号组合计算.

Analysis

- 具体地: $Ans = \frac{(x_1 + x_2 + x_3 + x_4)!}{x_1 ! x_2 ! x_3 !} f(x_4, S)$
- f(x4, S) 表示:
  - a1 + a2 + a3 + … + an = x4
  - a1 + 2 * a2 + 3 * a3 + … + n * an = S
  - 1 / (a1! * a2! * a3! * … * an!) 的和
- 用生成函数大力推一发以后发现f(x4, S)是 $\frac{\binom{x_4 + S - 1}{S}}{x_4 !}$

Painting Graphs With AtCoDeer

- 有一个 N 个点, M 条边的无向图,保证没有重边或自环.
- 你要对每条边染上颜色,颜色有 K 种.
- 问有多少种本质不同的方案(mod 10^9+7)
- 本质相同的具体定义如下:
    - 考虑两个染色方案 A, B .
    - 你每次可以挑选一个没有重复点的环,设这个环按照顺序是 e1, e2, e3, …, ek.
    - 然后col[e2] = col[e1], col[e3] = col[e2], …, col[ek] = col[e1]
    - 也就是做了一个轮换.
    - 如果 A 能在有限步之内变成 B.那么 A 和 B 是本质相同的.
- N, M, K <= 100 (N, M <= 10^5?)
- From ARC062

illustration

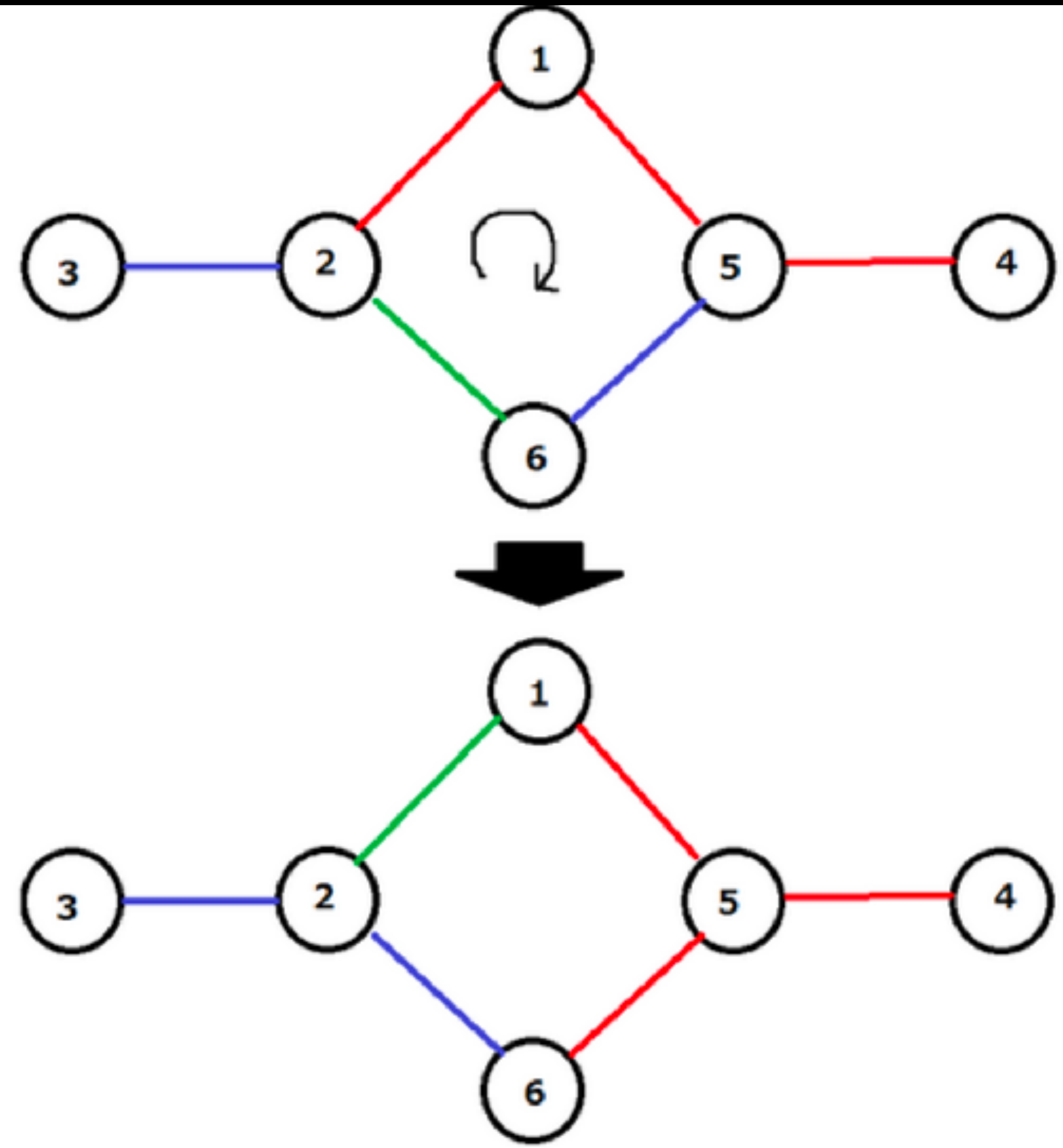


Figure 1: An example of a circular shift

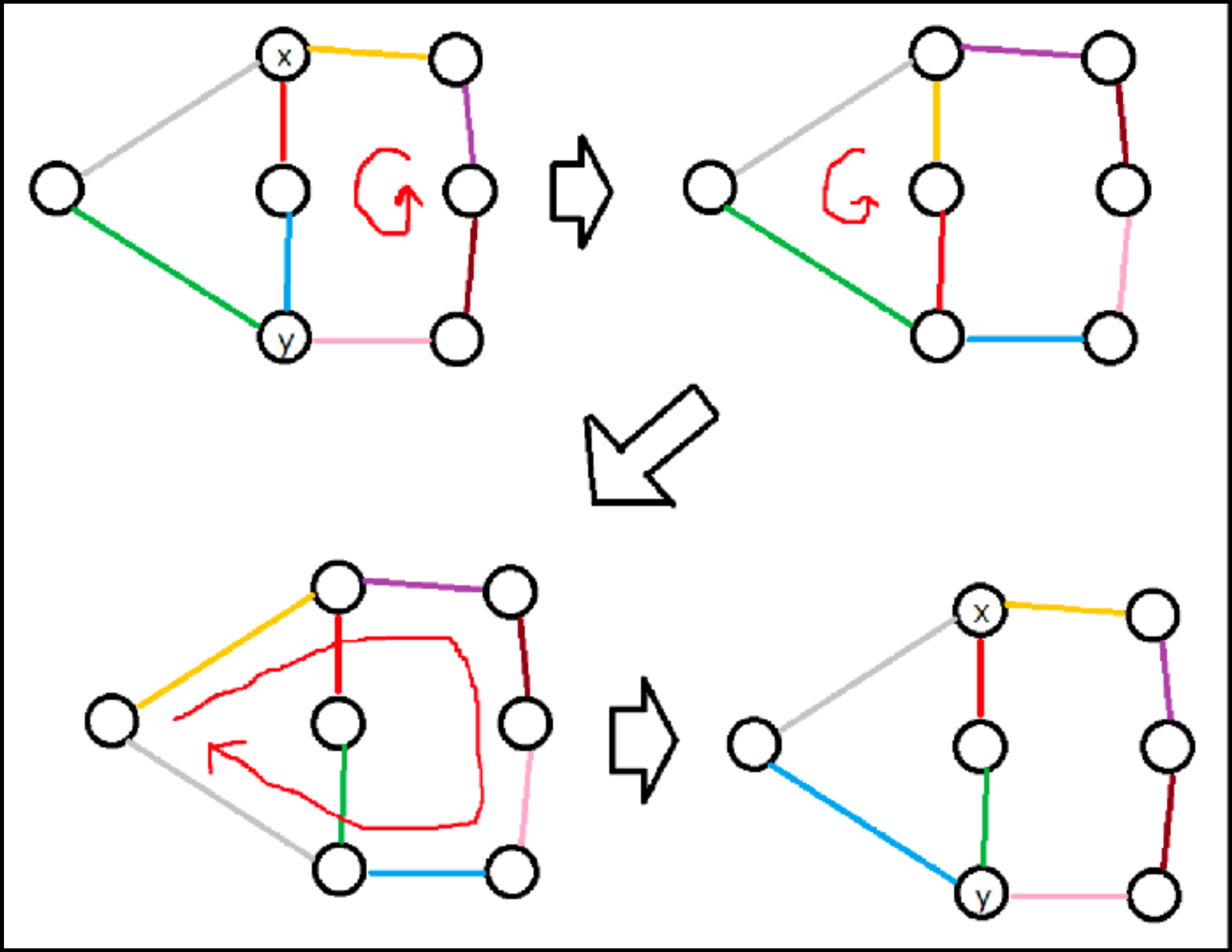
Analysis

- 每个点双显然是独立的.
- 对于一个点双,它有三种情况:
  - 1.两个点一条边,方案显然是一种
  - 2.一个环,可以直接polya.
  - 3.otherwise,这是我们要分析的地方.

Analysis:otherwise

- 这种情况下,他一定有一个边的子集是一个包含点双所有点的一个大环,并且中间用若干个点割出一条弦
- 可以用这组边构造出所有的置换.

Analysis:otherwise



Analysis:otherwise

- 可以用上述操作来构造出交换度数为3的点相邻两条边.
- 然后通过对左右两个环的旋转构造出交换任意相邻两条边的置换
- 于是就可以构造出所有置换
- 因此本质不同的方案只和每种颜色边的数量有关.
- 也就是 $x1+x2+x3+x4+\ldots+xk = E$ 的解数.
- 这个就是$C(E+k-1,k-1)$
- 时间复杂度$O(N+M)$.

Next or Nextnext

- 有一个长为 N 的数组 A.
- 问有多少长为 N 的排列 P 使得
- P[i] = A[i] 或 P[P[i]] = A[i]
- 答案对 10^9+7 取模.
- N <= 10^5.
- From AGC008

Analysis

- 如果 a 是一个排列, 怎么做?
- 更一般的做法?

Analysis : permutation

- 考虑怎么从 P 得到 A。
- 考虑一个有向图, i 向 P[i] 连边.
- 那么每个弱联通块都是环
- 然后每个点选择出边不变或者把出边连向 P[P[i]].

Analysis : permutation

- 因为 A 是一个排列
- 所以重新选择出边后,每个弱联通块依旧是一个环.
- 因此对于 P 中的每个环,环上的点要么都不改变出边，要么都改变出边.

Analysis : permutation

- 如果都不改变出边，那么环不变。
- 如果都改变出边:
  - 一个长为奇数的环会改变它的顺序。
  - 一个长为偶数的环会分裂成两个大小一样的环。

Analysis : permutation

- 也就是说，对于 A 中的一个环，有 3 种情况:
  - 1.它可能是 P 中的一个环。
  - 2.它可能是 P 中某个偶环分裂出的环
  - 3.如果这是一个奇环它还可能是 P 中某个奇环改变顺序得到的.
- 将环按照环长分类，分别计算。

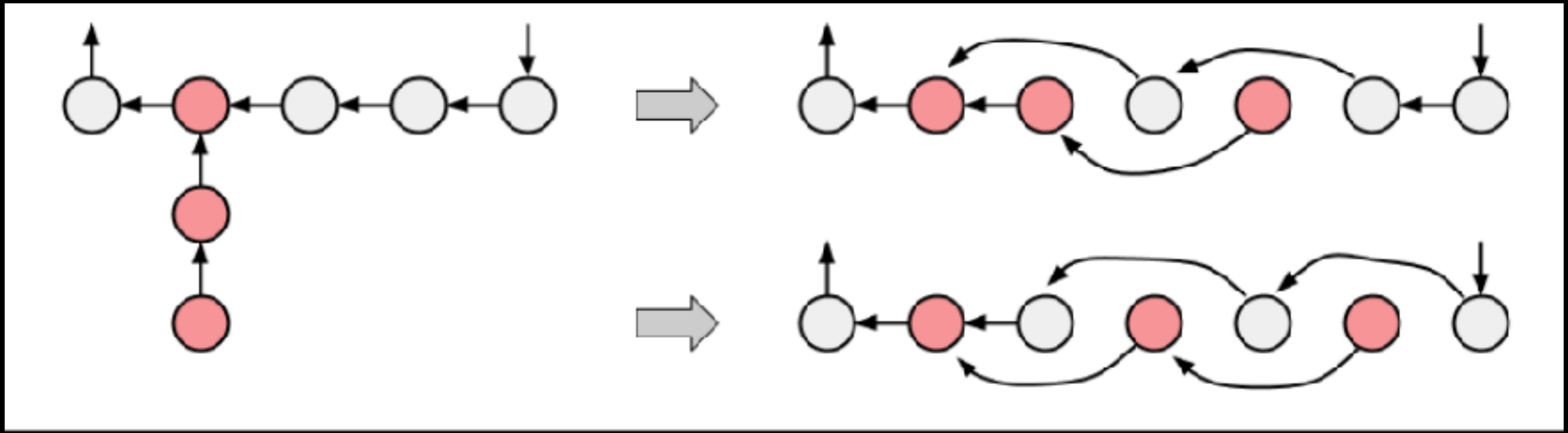
Analysis : permutation

- 具体地:
- 对于一个奇数的环长 L， 设这种环有 tot 个:
    - 枚举有多少个是分裂出来的,设有 2x 个.
    - C(tot, 2x) * 2^(tot-2x) * L ^ x * (2n - 1)!!
- 偶数环长也类似:
    - C(tot, 2x) * L ^ x * (2n - 1)!!

Analysis : general condition

- 情况会更复杂些。
- A 的每个弱联通块可能是一个环加外向树。
- 但是这个环加外向树支出去的子树都是链.
- 并且环加外向树和其他弱联通分量是不可以合并的.
- 直接计算即可.

Analysis : general condition



Eternal Average

- 在黑板上有 N 个 0 和 M 个 1.
- 每次你可以挑 K 个数字将他们全部删去.
- 并写上它们的平均数.
- 保证 (K - 1) 是 (N + M - 1) 的约数.
- 最后一定会剩下一个数字，问那个数字有多少种不同的情况.
- N, M, K <= 2000
- From AGC009

Analysis

- 把这个过程用一个 K 叉的有根树表示出来.
- 也就是一个 K 叉的有根树,有N+M个叶子,有M个叶子标为1,且每个非叶节点恰好有K个儿子.
- 然后这个树的权值 $\sum_{lab[x]=1} K^{-dep[x]}$
- 统计不同的权值个数.

Analysis

- 用 A[i] 表示所有标号为 0 的叶子的深度.
- B[i] 表示所有标号为 1 的叶子的深度.
- 一组 (A, B) 是合法的,当且仅当:

$$\sum K^{-A[i]} + \sum K^{-B[i]} = 1$$

Analysis

- 考虑一个有理数 z 能成为答案的条件:
  - z 在 K 进制下是有限小数.
  - z 能被 M 个 K^{-xi} 表示出来.
    - ⇔ z 每个小数位的和 mod (K - 1) ≡ M
  - 1 - z 能被 N 个 K^{-xi} 表示出来
    - ⇔ (1 - z) 每个小数位的和 mod (K - 1) ≡ N

A Simple O(N^2) DP

- 最后,问题就转变成了这个样子:
- 找一个序列 x1,x2,x3..xt满足:
  - $0 <= x_i <= K-1$
  - $x_t > 0$
  - $(x_1+x_2+x_3+\ldots+x_t) \bmod (K-1) \equiv M$
  - $(K-1) * t - (x_1+x_2+x_3+\ldots+x_t) \bmod (K-1) \equiv N-1$
  - $x_1+x_2+x_3+\ldots+x_t <= M$
  - $(K-1) * t - (x_1+x_2+x_3+\ldots+x_t) <= N-1$

AB = C Problem

- Snuke 有两个 N * N 的 01 矩阵 A, B
- 然后 Snuke 做了一个 C = A * B(一般的矩阵乘法)
- 然后让 C[i][j] = C[i][j] & 1.
- 有一天 Snuke 发现矩阵 A, B 被他吃掉了.
- 他只有矩阵 C.
- 问有多少种不同可能的 (A, B).
- 答案对10^9+7取模.
- N <= 300 (N <= 1000 ?)
- From Code Festival Grand Final

Analysis

- 什么样的矩阵才能成为 A 或 B ?
- 如果 B 确定, 那么 A 的方案数有几种? 与 B 的什么属性有关?
- 如何计算答案?

Analysis

- 设 rank(B) = x, rank(C) = R
- 考虑矩阵 B.
- C 的一个行向量可以看成 B 的行向量的线性表示.
- 表示方法是 A 的一个行向量.
- 能成为 B 的矩阵满足能线性表示出 C 的所有行向量.
- 这个条件是充要的.

Analysis

- 矩阵 B 的行向量表出一个向量的方案数是 2^(n - x).
- 因此 A 的方案数是 2 ^ (n * (n - x))

Analysis

- 现在就是要对每个 $1 <= x <= N$ 都计算:
  - $rank(B) = x$
  - B 的行向量可以线性表出 C 的每个行向量.
- 这样的 B 的个数.

Approach I

- 用 f[i][j][k] 表示这样的方案数.：
  - 考虑了 B 的前 i 行,
  - 有 j 个线性基。
  - 与 C 的线性基生成的空间的交大小是 2 ^ k.
- f[i][j][k] * (2 ^ j - 2 ^ k) -> f[i + 1][j][k + 1]
- f[i][j][k] * (2 ^ k) -> f[i + 1][j][k]
- f[i][j][k] * (2 ^ R - 2 ^ k) -> f[i + 1][j + 1][k + 1]
- f[i][j][k] * (2 ^ N - 2 ^ R - 2 ^ j + 2 ^ k) -> f[i + 1][j + 1][k]
- 答案就是 f[n][i][R] * 2 ^ (n * (n - i))的和.

Approach II

- 注意到答案只和 rank(C) 有关.
- 不妨统计所有 rank(C) = R 的 C 矩阵的答案和.
- 然后除掉 rank(C) = R 的矩阵个数就好了.

Approach II

- dp[i][j] 表示 N * i 的矩阵, 线性基有 j 个的方案数.
- dp[i][j] * 2 ^ j -> dp[i + 1][j]
- dp[i][j] * (2 ^ N - 2 ^ j) -> dp[i + 1][j + 1]

Approach II

- 枚举 B 的秩, 答案就是:

$$\frac{\sum_{i=R}^{n} dp[n][i] * 2^{n(n-i)} * dp[i][R]}{dp[n][R]}$$

フィボナッチ数の総和

- 有一个数列a[]满足:
  - 1.a[1] = 1, a[2] = 1
  - 2.a[i] = a[i - 1] + a[i - 2]
- 有一个矩阵D[][]满足.
  - D[1][i] = a[i]
  - D[i][j] = D[i - 1][j] + D[i][j - 1] (i > 1)
- 给出 N, M, 求 D[N][M] % 998244353.
- N <= 200000, M <= 10^18 (N = 30000 ? N <= 2000000?)
- From Square869120Contest #3

Approach I

- 考虑生成函数
- $Fib(x) = \dfrac{x}{1 - x - x^2}$
- 做了 N - 1 次前缀和： $A(x) = \dfrac{1}{(1-x)^{N-1}}$
- 因此最后要计算的就是： $\dfrac{x}{(1 - x - x^2)(1-x)^{n-1}}[x^m]$
- 这个东西的系数是一个 2 * (N - 1) 阶的常系数递推.
- 时间复杂度O(NlogNlogM)
- 应该可以做 N = 30000

Approach II

$$d_{1,i} = a_i$$

$$d_{2,i} = a_{i+2} - 1$$

$$d_{3,i} = a_{i+4} - (i + 3)$$

$$d_{4,i} = a_{i+6} - \frac{1}{2}(i^2 + 7i + 16)$$

$$d_{5,i} = a_{i+8} - \frac{1}{6}(i^3 + 12i^2 + 59i + 126)$$

Approach II

- 可以证明 $D(n, m) = Fib[m + 2n - 2] - P(x)$
- 其中 $deg(P(x)) <= N - 2$.
- 拉格朗日插值即可.
- 时间复杂度$O(N+logM)$.

Approach III

- 考虑 $\dfrac{x}{(1-x-x^2)(1-x)^{n-1}}[x^m]$
- 设他为 $\dfrac{\sum_{i=0}^{n-2} w_i x^i}{(1-x)^{n-1}}+\dfrac{A+Bx}{1-x-x^2}$
- 那么答案就是: $A*Fib(m)+B*Fib(m-1)+\sum_{i=0}^{n-2} w_i*C(n-2+m-i,n-2)$
- 用通分解方程。

Approach III

- x^0 : w0 + A = 0
- x^1 : w1 - w0 - C(k - 1, 1) * A + B = 1
- x^2 : w2 - w1 - w0 + C(k - 1, 2) * A - C(k - 1, 1) * B = 0
- x^3 : w3 - w2 - w1 - C(k - 1, 3) * A + C(k - 1, 2) * B = 0
- …
- x^{k - 1} : - w(k-3) - w(k-2) + (-1)^{k - 1} * A + (-1)^{k -2} * (k - 1) * B = 0
- x^{k} : -w(k - 2) + (-1)^{k - 1} * B = 0

Approach III

- w[i] = a[i] * A + b[i] * B + C
- 其中 a[], b[], c[] 是可以 O(N) 递推的常数.
- 用最后两个方程把 A, B 解出来.
- 然后就可以把 w[i] 解出来.
- 枚举分子的每项计算贡献即可.
- 时间复杂度O(N + logM).

Shik and Copying String

- 有一个字符串S,有一个目标串T.
- 考虑一种字符串序列A的构造方式:
  - 1.初始A[0]=S
  - 2.|A[i]| = |S|
  - 3.A[i][j] = A[i][j - 1] 或者 A[i - 1][j]
- 求最小的i使得存在一种方案让A[i]=T.
- |S|=|T|<=10^6
- From AGC007

Sample:

- Input:
  - 4
  - acaa
  - aaca
- Output:
  - 2   (acaa -> acca -> aaca)

Analysis

- 考虑一个贪心做法.
- 首先如果S = T, 那么答案为 0
- 否则至少存在一个位置 i 使得 S[i] != T[i]

Analysis

- 接下来给每个置 i,在 S 中寻找覆盖他的地方.
- 这个只需找 S[1:i] 中最大的等于 T[i] 的位置, 设他为 F[i]
- 因为 S ≠ T, 因此必然会出现一堆连续的 i 使得 F[i] 相等
- 对于这样的一堆 i 我们只需考虑最靠左的就好了.因为剩下的可以用一次向右操作完成.
- 更进一步地,即使 T[i] = T[i - 1] 而 F[i] != F[i - 1],我们让 i 这个位置从 i - 1 覆盖过来也不会影响答案.

Analysis

- 我们把所有满足存在一个 j 使得 F[j] = i的 i 称为关键点.
- 每次贪心将关键点覆盖到要覆盖的位置 或者 下一个关键点之前的位置 - 1.
- 时间复杂度O(N^2)

Analysis

- 考虑加速这个过程.
- 也就是考虑一个关键点几步之内可以完成覆盖.
- 设这个位置是 x，他要覆盖到的位置是 y.
- 设 [x : y] 中有 K 个关键点,且[y + 1 : n] 中第 K 个非关键点为 z.
- 那么[x : z] 中的关键点都要往右移一次.
- 步数就是 [x : z] 中的点数.
- 所有的步数取max + 1即可.

Analysis

- 最后要询问 [x : n] 中第 y 个非关键点的位置.
- 这个可以直接预处理所有非关键点的位置.
- 时间复杂度 O(N).

GL & HF