Guangdong Collegiate Programming Contest 2018

Solution

UESTC

Problem A: Chika's Math Homework

- 题意：计算 $\sum_{i=0}^{n} i^2 \binom{n}{i}$

- 考虑$A(x)=(1+x)^n$，
- 则$B(x)=A'(x)=n(1+x)^{n-1}$，
- $C(x)=B'(x)=n(n-1)(1+x)^{n-2}$

- 另外，$A(x)=\sum_{i=0}^{n} \binom{n}{i} x^i$
- $B(x)=\sum_{i=0}^{n} i \binom{n}{i} x^{i-1}$

$C(x)=\sum_{i=0}^{n} i(i-1) \binom{n}{i} x^{i-2}$

将x=1代入，可得

$$\sum_{i=0}^{n} i^2 \binom{n}{i} = C(1)+B(1)$$

$$= n(n-1)2^{n-2}+n\times 2^{n-1}$$

$$= n(n+1)2^{n-2}$$

快速幂即可解决，

复杂度 $O(T\log n)$

Problem B: Letter Kingdom

- 题意：给定一张图，点数不超过26，且每个点有且只有一条出边。一开始给定你m个人所在的点。让你执行三种操作：对编号在[l,r]中的人，让他们往下一个点走一步；对编号是x倍数的人，让他们往下一个点走一步；询问编号为p的人的当前所在的点。

- 模拟出每个字母的路线，可以发现每个字母的路线都可以形成一个带尾巴的环，用字符串分别记录尾巴和环的模样并统计长度（也有可能没有尾巴，即尾巴长度为0）。
- 对于1号操作，我们可以用线段树或者树状数组来维护每个位置操作的次数，每一次复杂度为O(logm)。

Problem B: Letter Kingdom

- 然后我们应该模拟筛素数的方法预处理出每个数的因数。在1~200000的范围内，每个数的因数不超过160个，为2号操作做准备。
- 对于2号操作，我们直接用数组统计每个数出现的次数，每一次复杂度为O(1)。
- 对于3号操作，我们先计算该位置1号操作和2号操作的操作数和，其中2号操作次数和就直接累加每个因数的操作和，其复杂度为O(logm + K) ,K<=160.计算出总操作数后，如果小于该位置初始字母对应的尾巴长度，则它的新位置在尾巴上；否则则先减去尾巴长度，再模环的长度计算出其在环上的位置，这步操作复杂度为O(1)。总复杂度$O(q(\log m+K)) < O(q\sqrt{m})$

Problem C: Xortree

- 题意：给你一棵树，点带权，多次询问，每次给你一个数x，以及一条路径的两个端点u,v，问你x与这条路径上的任意点的点权异或的最大值是多少。

- 如果在序列上，在可持久化二进制Trie上，利用贪心思想，可以很容易地解决。
- 在树上情况略有不同，建立可持久化Trie的过程中，每个点的可持久化Trie应该从其父节点继承过来，这样如果一个点的Trie，其实存储了从它到根的所有路径的信息。

Problem C: Xortree

- 于是在本题中对于询问，只需要获取到两个端点的Trie，然后再获取到lca的Trie以及fa[lca]的Trie，
- 然后Trie[u].sz+Trie[v].sz-Trie[lca].sz-Trie[fa[lca]].sz，就能知道二进制的某位在链中是否出现过，然后就用从高位到低位的贪心思想就能处理询问。

Problem D: Bipartite Coloring

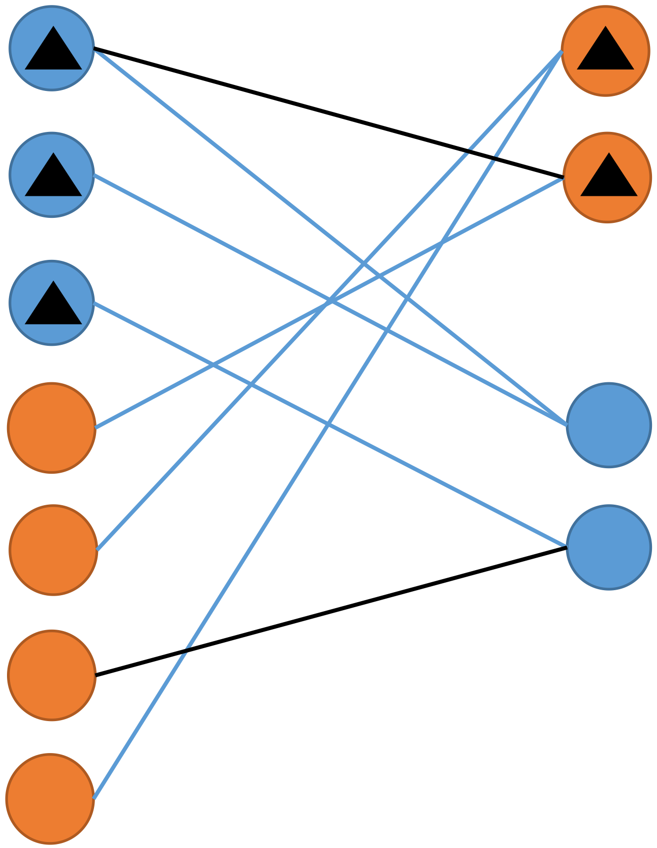
由于G是二分图，我们一定可以得到G的一个二染色$f: V \to 1, 2$，使得任意的边(u, v)都有$f(u) \neq f(v)$，但是f可能不满足B_1, B_2的约束条件。

令$B_1^* = f^{-1}(1)$，$B_2^* = f^{-1}(2)$。再令$X = (B_1 \cap B_1^*) \cup (B_2 \cap B_2^*)$，$Y = (B_1 \cap B_2^*) \cup (B_2 \cap B_1^*)$，下面我们证明删去一个点集$S$之后能够使剩下的图存在满足$B_1, B_2$约束条件的二染色当且仅当删去$S$之后$X$中的点和$Y$中的点不连通。

对于X中的一个点，要么它被删去，要么它的颜色不发生改变；而对于Y中的一个点，要么它被删去，要么它的颜色必须发生改变。因为f是一个二染色，对于一个连通分量而言，若其中一个点要改变颜色，则整个连通分量的点都必须改变颜色，所以显然X中的点和Y中的点不能处于一个连通分量。而若X和Y中的点不连通的话，那么我们一定可以通过修改f来得到一个满足B_1, B_2约束条件的二染色。

所以我们只需要求X和Y的最小点割集，这里X和Y中的点也是可以删的。这个问题用最大流就能求解。

Problem D: Bipartite Coloring



带有三角形的点：B_1中的点
不带三角形的点：B_2中的点
左侧的点：$B^*{}_1$中的点
右侧的点：$B^*{}_2$中的点
蓝色：X中的点
橙色：Y中的点
黑色：非法的边

Problem E: Lottery

- 题意：题目要求大于等于p的最小的n，使得存在正整数m，使得
- 2*m*(m-1) = n*(n-1)

- 原式 = 1+2+...+m-1 + 1+2+...+m-1 = 1+2+...+n-1
- 将左边的 1+2+..+m-1 与右边的 1+2+...+n-1中的前m-1项对应相减
- 得到1+2+...+m-1 = m + m+1 + ... + n-1

Problem E: Lottery

- 再将左边的1+2+...+m-1中的后n-m项与右边每一项对应相减
- 即
- 2m-n + 2m-n+1 + ... + m-1 与 m + m+1 + ... + n-1 对应相减
- 得到原式变为
- 1+2+...2m-n-1 = n-m + n-m + n-m + ... n-m = (n-m)*(n-m)
- 左边 = 1+2+...+2m-n-1 = (2m-n)(2m-n-1)/2
- 因为(2m-n)和(2m-n-1)互质，故要么(2m-n)和(2m-n-1)/2都是完全平方数①，要么(2m-n)/2和(2m-n-1)都是完全平方数②。

Problem E: Lottery

- 以情况①为例
- 枚举一个i作为sqrt(2m-n)，去check(2m-n-1)/2是不是完全平方数。
- 如果是的话，就对应一个解，可以列二元方程组解出对应的n和m：

$$\begin{cases} 2m-n=i^2 \\ n-m=sqrt(\frac{i^2(i^2-1)}{2}) \end{cases}$$

若n>=P了，则是原题要求的解

情况②同理

复杂度O(sqrt(P))

Problem F: Find the Number

- 题意：有数字1到n，裁判选一个数字，两个选手轮流将当前的数字从小到大排序后分成左右两堆，裁判会留下包含自己所选数字的那堆。若轮到某个人时只剩下一个数字，则其获胜，问两个人都按最优策略（选择胜率最大的方案）的情况下先手的胜率。

- 令$f[i]$表示剩下i个数字时先手胜的胜率，则有$f[i]=\max\{j/i*(1-f[j])+(i-j)/i*(1-f[i-j])\}(0<j<i)$，其中$f[1]=1$。
- 令$g[i]=i*f[i]$，则$g[i]=\max\{i-g[j]-g[i-j]\}$，打表可以发现g的增量以4为循环节，进而推出：
- $g[i]=i/2+2(i\%4=0)$、$(i-2)/2(i\%4=2)$、$(i+1)/2(i\%2=1)$，可通过归纳法证明。

Problem G: Commemorate

- 题意：N个点M条边的无向图，多次询问保留图中编号在[l,r]的边的时候图中的联通块个数。

- 首先每个连通块只要保留一棵生成树的边就可以保证连通了
- 把每条边的编号当做边权
- 我们把每条边按顺序加入，维护一个每个连通块的最大生成树
- 每次替换树上路径的最小边
- 把它替换的边的编号记录到一个数组a[i]中，如果连通了两个连通块，a[i]=0
- 这个显然是可以用LCT解决的

Problem G: Commemorate

- 得到a[i]后，对于每个询问[l,r]，统计区间[l,r]中有多少个数小于l，用n去减即可得答案
- 因为我们维护的是最大生成树，如果[l,r]中有一条边x替换的边y是小于l的，那么没有编号大于等于的l边使这个连通块连通，就说明该边使两个连通块连通了，连通块数量-1
- 这个用主席树搞一搞即可，也可以离线询问，用线段树解决。
- 复杂度O(klogm)

Problem H: Number String

- 题意：给定一个只由0~9组成的串，你一次操作可以删除任意一个数，或者选择相邻的和不超过9的两个数，将它们替换为它们的和。你可以进行任意次操作，问你最终所能获得的字典序最大的串，以及所对应的最少的操作次数。

- 因为要使字典序尽量大，所以要前面要有尽量多的9，8，7，以此类推。因此，对原串扫10次，前面组成尽量多的9，然后在最后一个9之后组成尽量多的8……

Problem H: Number String

- 对于每一次扫描(以9为例)，记录之前的未使用过的数字所有可能的组合情况，如123，他们的和可以组成1，2，3，4，5，6，这时假设我添加进去一位3，就可以组成一个9，这时贪心的选择9，然后把之前保留的结果清零，从下一位开始重新统计。
- 9扫描完毕后，从最后一个组成9的数字开始扫描8，以此类推。至于操作数，无论是1操作还是2操作，每一次减少的位数都是1，这时只要用原来的位数减去答案的位数就行了。

Problem I: Convex Hull

- 题意：给定一个凸包，问你所有满足如下条件的点P的轨迹围成
- 的图形的面积：$\sum_{i=0}^{n-1} S_{\triangle PA_iA_{(i+1)\%n}} = w.$
- 首先因为w>S,所以P点在凸包外面
- 定义状态f(i,j)表示仅有$\triangle PA_iA_{i+1}, \triangle PA_{i+1}A_{i+2}, \cdots, \triangle PA_{j-1}A_j$这些三角形与凸包的面积交为0,其它三角形与凸包的面积交不为0
- 可以证明,所有的f状态包含了P点所有的位置,同时,在一个f(i,j)状态内,P点的轨迹是一个线段

Problem I: Convex Hull

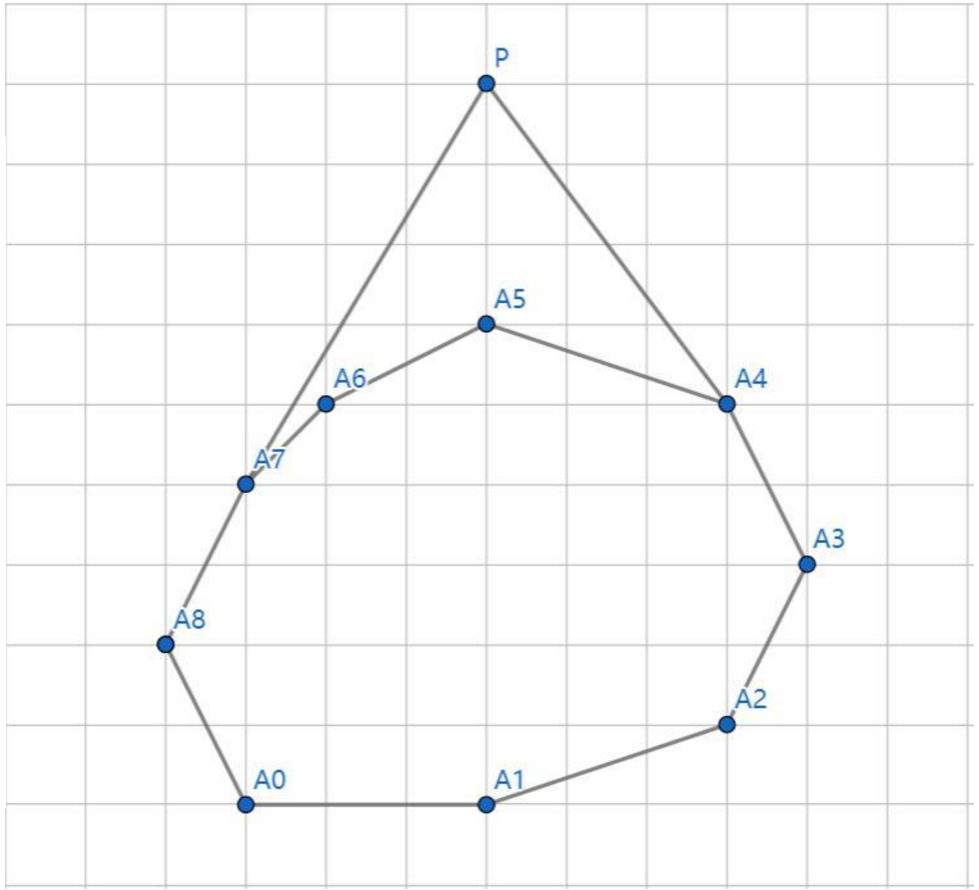
- 换句话说,答案图形是一个凸包,那么一个f(i,j)状态表示的是凸包上的一条边。例如,当n=9时,f(4,7)如下:

这种情况下三角形的面积和可以表示为

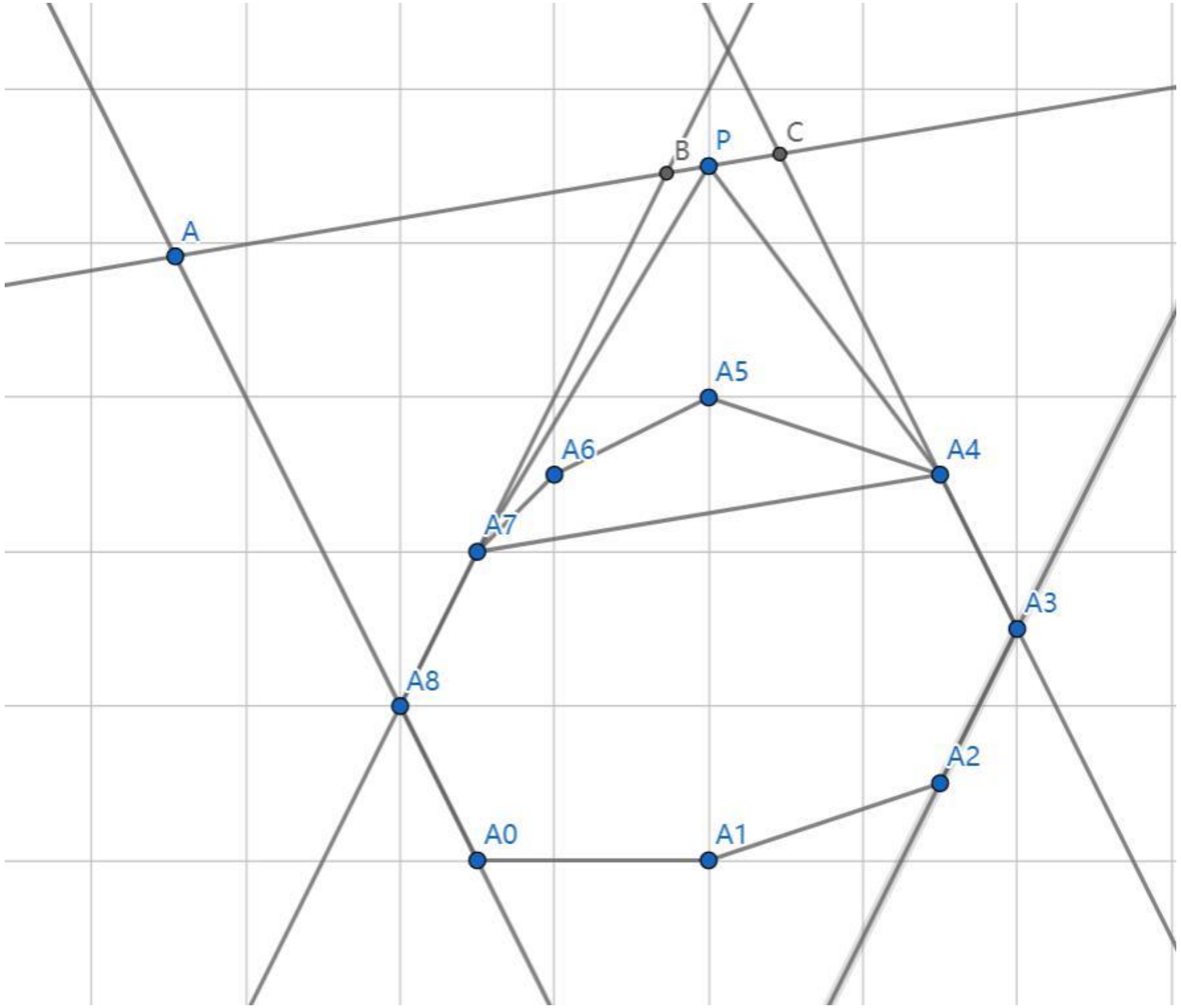
$S_{convex}+2*\ S_{\triangle PA4A7}-2*\ S_{A4A5A6A7}$,

P点的轨迹将是一条平行于A_4A_7的线段

这下我们可以利用三角形面积公式很容易求出P的轨迹所在的直线,但是我们还需要求出P点轨迹的端点.



Problem I: Convex Hull



BC即为f(4,7)下对应的线段。

于是我们就可以枚举i,j, $O(n^2)$解决这个问题了。

Problem J: Maximum × Minimum

- 题意：给定一个序列，求所有区间最大值和最小值之积的和。
- 将所有数排序后，从小到大依次插回原位置，对于每个最小值（下标为i）都能求出一段区间[L,R]，使得左端点在[L,i]、右端点在[i,R]的区间即所有最小值为a[i]的区间。
- 每次枚举[L,i]和[i,R]中较小的一边，枚举总复杂度为O(nlogn)（可以通过DP检验数量级在1e5时总枚举次数约为8e5）。
- 这样求出一边的最大值后，对于另一边，可以发现从i开始最大值只会出现在数值比之前都大的位置上，考虑倍增处理。

Problem J: Maximum × Minimum

- 可以定义从i往左或右的下一步就是往左或右第一个数值大于i的位置，倍增预处理从i往左及往右第2^j步所在的位置。
- 这样枚举一边时，假设是左边，右边可以倍增+二分地找出最大值比左边大的第一个位置，以及最后一个会出现最大值的位置，并将右边分成三段。
- 对于第一和第三段都容易处理，对第二段倍增维护区间每个前缀最大值之和即可。

Thanks for watching.