The 2019 ICPC Asia Shanghai Regional Contest Online Contest

Shanghai University

A - Lightning Routing I

- 题目大意：给定一个边权树，要求支持动态修改边权，询问到某个点最远的点的距离。

- 解法：
- 我们知道，树的直径可以通过两次 dfs() 的方法求得。换句话说，到任意点最远的点，一定是直径的某个端点（反证法）。
- 因此原问题转化为动态维护直径，然后再支持询问两个点的距离，后者可以 dfs 序 + lca + 树状数组。

- 动态维护直径可以用点分治（点分树），具体做法是，考虑过分治中心的最长路径，我们只需要查询分别以分治中心的每个儿子为根，所在子树的最长链，从中再找到最长和次长即可，这个星状图可以用 set 维护。每个子树则可以使用 dfs 序 + 线段树维护。复杂度 O(nlog2n)。对于动态维护直径，存在一个基于树的全 DFS 序的非常巧妙的做法，可以把复杂度改进到 O(nlogn)。具体可以看这里这篇博客：https://www.cnblogs.com/TinyWong/p/11260601.html
- First Blood 中，雅礼中学的做法就是采用这种做法。当然事实上我们可以使用子集动态树直接维护最远的距离，完全不用考虑直径，均摊复杂度为 O(nlogn)，代码也会更短。

B-Light bulbs

- 题意：编号为0~N-1的灯泡，初始都是关闭的，M次操作，每次操作把一个区间的灯泡就像翻转，问最后打开的灯有多少个。
- 签到题。
- 注意复杂度应该是O(MlogM)，把区间进行排序（或者区间端点进行排序），左区间+1, 右区间-1, 然后求前缀和得出每一段的状态。

- https://paste.ubuntu.com/p/92pcwwtvdC/

C - Triple

- 题意：从三个数组中各选一个数，要求这三个数：任意两个数差的绝对值小于等于第三个数。
- 也就是三个数当中最大的数要小于等于另外两个数的和。
- 对于N <= 1000, 使用O(N^2)暴力解法即可。
- 对于N>1000, 用FFT做。枚举其中最大的数，另外两个用FFT进行组合，注意去除重复的情况。
- 分享几个程序作为参考：
  https://paste.ubuntu.com/p/mRM8XKdRwC/

D - Counting Sequences I

- 题意：统计有多少个序列满足和等于积
- 直接用dfs去求解，注意剪枝。
- 先不考虑位置，先求N个数，然后利用N!/(k1!*k2!...)算序列个数。

```
long long dfs(int limit, int pos, int n, int d, int sum, int cnt, long long now) {
    if (n == pos) {
        if (d == sum) {
            return now*rF[cnt]%MOD*F[n]%MOD;
        } else {
            return 0;
        }
    }
    if (d == sum + n - pos) {
        return now*rF[cnt]%MOD * rF[n-pos]%MOD*F[n]%MOD;
    }
    if (limit > n) return 0;
    if (d > sum + n - pos) return 0;
    if (d*limit > sum+limit + (n-pos-1))return 0;
    long long ans = 0;

    ans += dfs(limit, pos+1, n, d * limit, sum + limit, cnt+1, now);
    ans += dfs(limit+1, pos, n, d, sum, 0, now*rF[cnt]%MOD);
    return ans%MOD;
}
```

D - Counting Sequences I

这个题只要搜索剪枝到位，不打表也是可以的。
如果实在巨慢无比，打表也是可行的，反正才3000个数。

E - Counting Sequences II

- 题意：问有多少个长度为n的序列，每个数的范围是1~m, 偶数在序列中出现次数必须是偶数。
- 直接推导公式。至于公式的推导，省略一万行。。。

```
long long n;
long long m;
cin>>n>>m;
long long sum = 0;
long long C = 1;
for (int i = 0; i < m/2+1; i++) {
  sum += C * pow_m(i*2+(m&1), n)%MOD;
  sum %= MOD;
  C *= (m/2-i)*inv(i+1, MOD)%MOD;
  C %= MOD;
}
long long ans = sum * pow_m(inv(2, MOD), m/2) % MOD;
cout<<ans<<endl;
```

枚举偶数数字总个数$2k$,$ans=\sum_{k=0}^{\lfloor \frac{n}{2} \rfloor} C(n,2k)cnt_1^{n-2k}f(2k)$

$f(2k)$为用cnt_0种球放$2k$个盒子，且每种球个数为偶数的方案数

考虑$f(n)$为用m种球放n个盒子的方案：

则方案数为$\sum_{x_1+x_2+..x_m=n,x_i\text{为偶数}} \frac{n!}{x_1!x_2!...x_m!}$

构造指数型生成函数$g(x)=1+\frac{x^2}{2!}+\frac{x^4}{4!}+\ldots=\frac{e^x+e^{-x}}{2}$

则$f(n)=g(x)^m[x^n]*n!$

$g(x)^m=\frac{\sum_{i=0}^{m} C(m,i)e^{(m-2i)x}}{2^m}$

所以$g(x)^m[x^n]*n!=\frac{\sum_{i=0}^{m} C(m,i)(m-2i)^n}{2^m}$

$ans=\sum_{k=0}^{\lfloor \frac{n}{2} \rfloor} C(n,2k)cnt_1^n invcnt_1^{2k}\frac{\sum_{i=0}^{cnt_0} C(cnt_0,i)(cnt_0-2i)^{2k}}{2^{cnt_0}}$

$ans=\frac{cnt_1^n}{2^{cnt_0}}\sum_{i=0}^{cnt_0} C(cnt_0,i)\sum_{k=0}^{\lfloor \frac{n}{2} \rfloor} C(n,2k)(invcnt_1(cnt_0-2i))^{2k}$

$ans=\frac{cnt_1^n}{2^{cnt_0}}\sum_{i=0}^{cnt_0} C(cnt_0,i)\frac{(1+invcnt_1(cnt_0-2i))^n+(1-invcnt_1(cnt_0-2i))^n}{2}$

$ans=\frac{1}{2^{cnt_0}}\sum_{i=0}^{cnt_0} C(cnt_0,i)\frac{(cnt_1+cnt_0-2i)^n+(cnt_1-cnt_0+2i)^n}{2}$

$ans=\frac{1}{2^{cnt_0}}\sum_{i=0}^{cnt_0} C(cnt_0,i)\frac{(cnt_1+cnt_0-2(cnt_0-i))^n+(cnt_1-cnt_0+2i)^n}{2}$

$ans=\frac{1}{2^{cnt_0}}\sum_{i=0}^{cnt_0} C(cnt_0,i)(cnt_1-cnt_0+2i)^n$

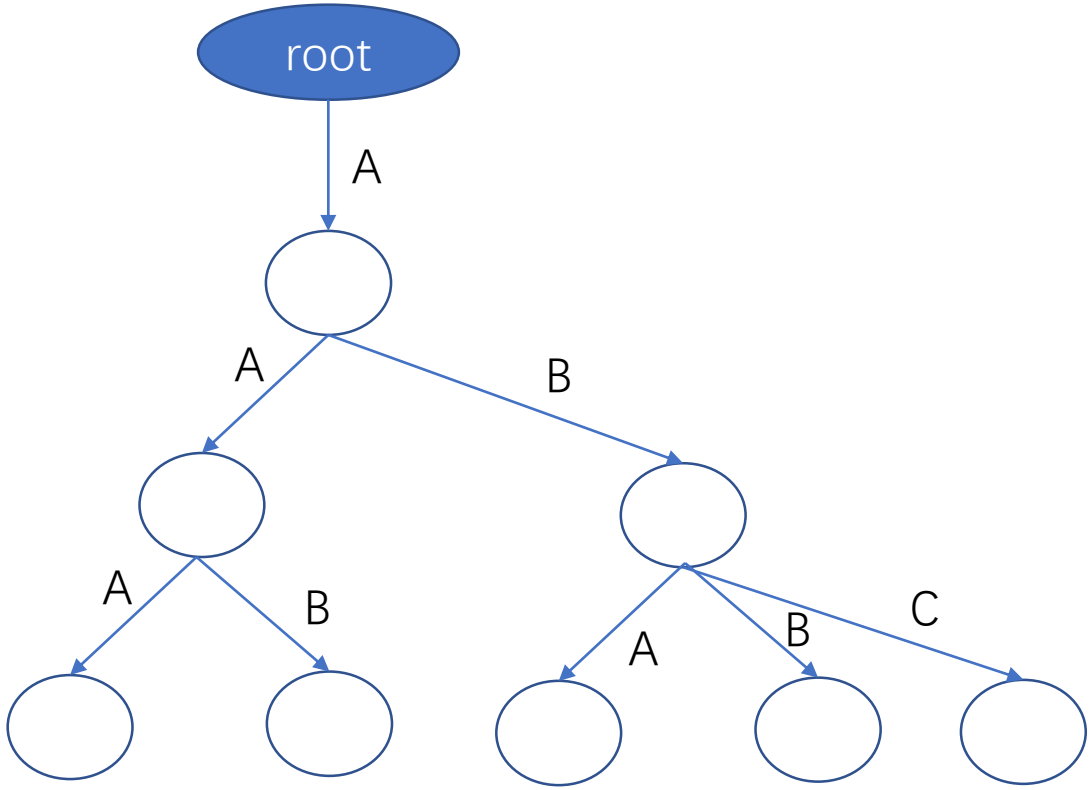
F - Rhyme scheme

- 题意：输出长度为n的第k个Rhyme scheme.
- Rhyme scheme: 长度为n的个数是bell number的例子。
- 如果不懂bell number也没有关系。
- Rhyme scheme就是第i个字母最多是前面出现过的字母+1.
- 比如：AAC是不可以的，ABCAAD是可以的，ABBD也是不可以的

F - Rhyme scheme

- 如右图，Rhyme scheme可以形成一个字典树。（图为n=3）
- 可以用DP求出，dp[n][i][j]表示长度为n的Rhyme scheme, 在第i层，前面出现的字母最大是j有多少个。
- 如果询问n,k的时候，只要在字典树上从上往下走即可。

- B(26)会超出long long, 可以用两个long long或者__int128搞一搞。



F - Rhyme scheme

- https://paste.ubuntu.com/p/x9vsTPhqd7/
- https://paste.ubuntu.com/p/4n3XBVk3kr/

- (NOTE: Rhyme scheme其实是Bell number的经典例子，如果你不懂Bell number, 认真读题也是可以的，所以本题没有详细介绍Bell number是啥)

G - Substring

- 题意：给了一个母串S, 每次循环给了一个模板串，问模板串在母串中“匹配”了多少次？“匹配”的意思就是首字母和尾字母一样，中间字母顺序可以换。
- 首先如何快速判断两个字符串是否“匹配”？
- 长度必须一样，首尾字母，中间字母只要出现次数一样的就可以。
- 因为只包含小写字母，只要用长度为26的数组统计每个字母出现的次数即可。
- 这样如果询问只有一个，用滑动窗口在O(n)时间内就可以的出来了。

G - Substring

- 但是因为询问有20000个，O(NM)肯定超时的。
- 所以对于长度一样的询问可以弄在一起搞，因为窗口大小是一样的。
- 所有询问的总长不超过100000，不同长度的个数最多也是sqrt(100000)级别的（最坏情况是长度为2+3+4+…）。

G - Substring

- 解法就是:
- 1) 把M个询问的字符串的Hash值丢入一个unordered_map
- 2) 按照询问的不同长度，对母串循环sqrt(100000)次，滑动窗口求出Hash值，累加到unordered_map中。
- 3) 输出答案即可。

- (有人吐槽这题卡时限和内存了，std没有用特殊优化，时限大概是标程的两倍，内存是标程的三倍，是不是很良心 ☺)

H - Luhhy's Matrix

- 题意：维护一个队列，元素是16*16的01矩阵，每次入队或出队操作后输出队列中所有矩阵的乘积，矩阵乘法是在模2意义下并且是从队尾到队首的顺序，强制在线。

  题解：设一个分界线把队列分成左右两段，左半段记录所有位置到分界线的后缀信息，即每个后缀的矩阵乘积，右半段只维护所有矩阵的乘积，每次询问就将左右两边乘起来。入队直接在右半段添加，出队操作时，如果左半段非空，则直接弹掉队首，如果已经空了，则暴力地将右半段中所有的矩阵拿到左边，并重新计算出所有后缀信息即可，这样每个矩阵在维护的过程中最多只会被做2次乘法，用bitset加速矩阵乘法的话，时间复杂度为O(16^2 N) 。

I - Debug

- 原题 bzoj 4428，加强了数据范围，但是做法完全不一样。如果用dp的想法最多只能做到 1e9 的数据范围。

- 首先，枚举 run 的次数，不会超过 logn 次，因为每次至少二分。然后我们希望能够根据固定 run 的次数来直接得到最少 print 次数。显然每次 print 都是等分的，因此 (p1 + 1) * (p2 + 1) * …… * (pk + 1) >= n，其中 p1, p2 ……pk 为每次 print 的次数。

- 我们希望 sum(pi) 最小，由均值不等式可以得知所有 p 越接近越好，但是由于 p 是整数，因此需要花 log 的复杂度枚举边界。总复杂度 O((logn)^2)

J - Stone Game

- 对所有石子从大到小排序，进行dp。

- 我们考虑取出的那一堆石子，f[i][j] 表示该堆石子里最小的石子为 i，总价值为 j 的方案数，这个通过dp来算。

- 对于所有最小石子为 i 的方案，可以求出其左右边界，那么对应答案加上改区间内的 f[i][l] ~ f[i][r]。其中 i 这一维可以略去。

K - Peekaboo

- 很套路的题，由于是整点，自然而然想到勾股数。对于 a 和 b 分别枚举勾股数之后 check 是否可行。

- 如果同时枚举所有旋转对称的情况可能会T，因此需要合并旋转对称的情况。

- 注意三点共线的情况，旋转翻折后可能会有重合。

L - Digit sum

- 签到题。
- 直接b进制分解求$S_b(n)$，先预处理出dp[N][b]表示1~N的digit sum的总和。
- 对于每一组数据O(1)输出即可。

- 感谢您参加本次网络赛。
- 如对本次比赛有任何建议，欢迎填写满意度调查问卷：
- https://tp.wjx.top/jq/45765616.aspx