2019牛客暑期多校训练营 第四场

zzq & kcz

Preview

- A 432
- B 33
- C 324
- D 330
- E 23
- F 3
- G 1
- H 1
- I 91
- J 678

K – number

- **可以直接O(300n)dp。**
  - 记f[i][j]为右端点为i的满足mod 300=j的子串个数，可以容易地转移。
- **也有一个简单的O(n)做法。**
  - 注意到是300的倍数也就是既是3的倍数又是100的倍数。3的倍数即各位和为3，100的倍数即最后两位为0（也有可能就是单个0）。
  - 记s_i为前i位的和mod 3的值，那么对于长度大于等于2的区间$[l,r]$，合法条件即$s_{l-1}=s_r$且s_r,s_{r-1}为0。
  - 从小到大枚举r，并同时记录有多少个$0\le x\le r-2$分别满足$s_x=0,1,2$。

J – free

- **最短路。**
- **记f[i][j]表示从起点走到i，至多免费走j条边的花费最小值。**
  - f[i][j]=min(f[i][j-1],min(f[x][j-1],f[x][j]+e) (x和i有边权为e的边))
- **从小到大枚举j进行转移。**
  - 以min(f[i][j-1],f[x][j-1] (x和i有边))作为i号点距离的初始值
  - 使用dijkstra等算法跑最短路
- **时间复杂度O(kmlogn)。**

A – meeting

- **一句话题解：考虑距离最远的两个关键点，设它们的距离为d，d/2上取整即为答案。**
  - 必要性：这两个人要碰面，必然要走至少d/2步。
  - 充分性：我们取两人路径中和一头距离为d/2上取整的一个点，让所有人在这相聚。如果有一个人在d/2时间内到不了，那么它和路径两头中与它远的那一头的距离大于d，与最远的假设矛盾。
- **找到这样最远的一对点类似找树的直径。可以直接dp，也可以采用两遍bfs：从任意一个关键点开始，找到离它最远的关键点x，再从x开始bfs，找到的新的最远点和x形成的就是直径。**
- **当然对着题面直接dp也是可以做的，但是比较难写。**

D – triples I

- 分类讨论。
- 一个二进制位mod 3只可能是1或者2。
- 如果a是3的倍数，那么我们直接取{a}即可。
- 否则如果a的二进制位只有一位或两位，我们根本取不出0以外的三的倍数，所以无解。
- 接下来考虑a至少有三位的情况。

D – triples I

- **若a mod 3=1：**
  - 如果a中的二进制位有至少两个mod 3=1的，设它们为p和q，我们取{a-p,a-q}即可。
  - 如果a中的二进制位有恰好一个mod 3=1的，那么设mod 3=1的这个位为p，mod 3=2的某个位为q，我们取{a-p,p+q}即可。
  - 如果a中的二进制位没有mod 3=1的，那么假设有三个mod 3=2的位p,q,r，我们取{a-p-q,p+q+r}即可。
- **若a mod 3=2只需把上面的讨论中1与2互换即可，是完全对称的。**

C – sequence

- **假设a中的元素互不相同，我们考虑a中的某个元素作为min的时刻。**
  - 对于每个a[i]，我们找到左边第一个比它小的元素a[l]，右边第一个比它小的a[r]
  - 那么左端点在[l+1,i]，右端点在[i,r-1]的区间min就为它。
  - 求l和r可以使用单调栈。
- **考虑如何求某个a[i]作为min的答案。**
  - 如果a[i]>0我们就是要最大化sum(b[l..r])。a[i]<0就是要最小化。
  - 记b的前缀和为s，那么sum(b[l..r])=s[r]-s[l-1]。
  - 所以我们只要查询i..r-1最大的s和l..i-1最小的s，相减即可。
  - 查询区间最小值可以使用st表或线段树等数据结构。
- **也可以直接建立笛卡尔树，然后维护子树中s的最大最小值，从而做到O(n)的复杂度。**

I – string

- 不等价大约是让a和rev(a)只算一次？
- **考虑rev(s)和s中的不同子串个数，那么这样s和rev(s)就会大约恰好被算两次！**
- **其实并不是所有s都是这样，如果s本身是回文的，那么s=rev(s)。**
- **求出rev(s)和s中的不同子串个数p：**
  - 可以直接使用广义后缀自动机
  - 也可以对s$rev(s)建立后缀数组，求出不同的子串个数，减去包含$的串个数即可，这些串显然是两两不同的
- **求出s中的不同回文串个数q：**
  - 使用回文树即可，也可以使用manacher+后缀数组，但是较为繁琐
- **答案即为(p+q)/2。**

B – xor

- **考虑每个集合形成的线性空间，那么我们要求的就是这些线性空间的交中包含x。**
- **如何刻画线性空间的交?**
  - 两个线性空间的交仍然是线性空间。那么仍然可以用一组线性基来描述。
- **如何求两个线性基的交?**
  - 若V1,V2是线性空间，B1,B2分别是他们的一组基，令$W = B_2 \cap V_1$，若$B_1 \cup (B_2 \setminus W)$线性无关，则W是$V_1 \cap V_2$的一组基。
  - 我们从低到高考虑B_2中的元素，同时维护由B_1和B_2中已经插入的向量构成的线性基，并记录这个线性基中每个元素由V_1贡献的部分。如果这个元素不能被旧线性基表出就直接加入线性基，否则把它由V_1贡献的部分加入答案。
  - 时间复杂度$O(\log^2)$。

B – xor

- **考虑如何处理区间询问。**
  - 我们直接建立线段树，每个点存下对应集合的线性基的交。
  - 建立线段树的时候需要求$O(n)$次交，询问的时候需要求$O(q\log(n))$次交。
  - 事实上，我们询问的时候并不需要真的求交，在$\log(n)$个线性基上依次查询即可。
  - 时间复杂度$O((n+q)\log^2)$。

E – triples II

- 如果a中为1的二进制位在b中也都为1，我们称a是b的'子集'。

- **考虑如果只要求最后的答案是a的'子集'，那么我们只需要保证每次选的数都是a的'子集'即可。**
  - 考虑计数是a的'子集'并且是3的倍数的数，我们只需要简单地进行dp，记f[i][j]为考虑a的前i位的'子集'并且当前mod 3的值为j的数的个数即可。
- **回到原问题，我们只需要枚举最后实际或出来的数是a的哪一'子集'，进行容斥即可。**
- **直接枚举a的'子集'复杂度难以接受，但是我们注意到每一个子集的贡献只和这一'子集'中mod 3=1和2的二进制位分别有多少个有关。我们枚举分别有多少个，组合数算出这一种'子集'有多少个，再进行dp即可。**
- **不需要真的枚举一个子集进行dp，而是预处理每种'子集'的答案，每组数据只需重新求n次方。**
- **复杂度O(Tlog^2(a)log(n))。**

F – merge

- 不知道大家是否注意到了数据范围中l=1。

- **首先考虑merge函数：**
  - 例如对于3 0 1和2 5，我们会merge成2 3 0 1 5
  - 考虑把序列分成一段一段的，每一段开头为这一前缀的最大值，例如3 0 1就分为[3 0 1]一段，2 5分为[2] [5]，那么我们实际上merge就是把这些段按开头排序。

    这是因为一旦我们把一个段的开头放了进来，那么这一段一定会接着放完，因为剩下的数比开头小

- **接下来我们的问题就变为了维护这些段。**

F – merge

- **考虑一次操作1 m r的影响：**
  - 首先我们把r所在的段分裂，假设为$[l_0, r_0]$，我们分成$[l_0, r]$和$[r+1, r_0]$这样两段。
  - 接下来把m所在的段分裂，假设为$[l_1, r_1]$，我们先分成$[l_1, m]$一段，再把$[m+1, r_1]$按定义分拆成若干段。
  - 接下来我们需要把$[1, r]$这些段按开头排序，之后把$[r+1, r_0]$接到最后一段的末尾。
  - （m和r可能同在一段，不过只要按照这个实现就没有问题）

F – merge

- **考虑如何把段按开头排序。**
  - 注意到一开始段就是按开头排好序的，所以任意时刻段都是按开头排好序的。所以我们只需要用线段树或平衡树存下所有段的开头元素进行维护即可。
- **我们还需要维护每一个段内具体有什么元素。**
  - 由于需要支持分裂，所以我们可以把每个段用一棵平衡树进行维护。
- **查询的时候只需要先在维护段的数据结构中查询到对应的段，接下来在对应的平衡树上查询。**
- **时间复杂度**
  - 唯一不确定的地方在于把$[m+1, r_1]$按定义分拆成若干段。
  - 但是注意到这样分拆段数会+1，而总的段数不可能超过n。
  - 时间复杂度$O((n+q)\log(n))$。

G – tree

- **点数不超过12的本质不同的有根树只有8000个不到，考虑都搜出来，然后树形dp。**
- **搜出所有树并预处理转移：**
  - 按点数从小到大处理，点数为i的树可以通过在点数为i-1的树上加一个叶子得到，因此可以枚举每个点数为i-1的树，然后枚举叶子加在哪个点下面，用树hash去重。
  - 对于每对大小之和不超过12的树A,B，预处理将B的根加在A的根的下面得到的树是哪个树，只需要树hash+map即可。这样的对数只有 10000 多个。
- **对于给你的树A，dp[i][j]表示以点i为根的生成子图和树j同构的方案数，然后暴力转移即可。**
- **对于每次询问的树B，枚举哪个点当根，然后树hash得到树对应的标号并去重。然后就是对一些树j求$\sum_{i=1}^{n} dp[i][j]$，对于每个j预处理即可。**
- **时间复杂度$O\left(10000n + tm^2\log(m)\right)$。**

H – RNGs

- **背景：**
  - RNG0就是windows下最常见的rand的实现，rand实际返回的是高16位，而这里返回只有第28位。
  - RNG1是著名的梅森旋转(Mersenne Twister)算法。
  - RNG2是著名的xorshift64算法，它的32位版本xorshift32曾经在WC2017【挑战】中作为数据生成器出现。
  - 当然这些背景知识并不是解决本题必须的~~，而且也确实没有什么帮助~~。
- **还需要注意的一点是u32和u64的运算均为自然溢出，即在mod 2^32意义下运算。**

H – RNGs

- **分辨RNG1相对困难，我们分辨出来RNG0和RNG2，然后分辨不出来的就是RNG1。**
- **RNG0**
  - 由于u32是自然溢出，我们实际上做的就是每次$x \leftarrow (214013x + 2531011) \bmod 2^{29}$，然后取出$2^{28}$这一位。
  - 容易发现x实际上是一整个环，即我们从某个x开始重复2^{29}次这个操作会回到本身，并且经过的x互不相同。
  - 虽然我们并不知道x具体是多少，而只知道2^{28}这一位，但是我们可以考虑相邻的60个x，如果输入中有一个子串等于这个对应的01串，我们就认为它是由RNG0生成的。

H – RNGs

- **RNG0**
  - 考虑一个类似bsgs的过程。注意到$x \leftarrow (214013x + 2531011) \bmod 2^{29}$操作实际上是线性的，所以进行若干次这个操作之后实际的变化也是形如$x \leftarrow (ax + b) \bmod 2^{29}$的形式。
  - 考虑设$S = 1600$，然后在环上每次跳S步，把经过的x后面的60位记下来。对于询问串中每个长度为60的子串匹配即可。
  - 1600+60+60<2000。

H – RNGs

- **RNG2**
  - 我们把x的各位当做mod 2意义下的64维向量，然后我们每次就是把这个向量乘上一个矩阵（仍然是mod 2），然后取出某一维输出。
  - 考虑矩阵的特征多项式，那么它就对应了关于x的一个线性递推数列，所以x有一个长度不超过64的线性递推式。
  - 我们使用高斯消元或BM算法在本机求出线性递推式并对于给定的输入验证即可。
- **如上所述，分辨不出的我们就认为是RNG1即可。这个算法正确率接近100%（估计是100%，没验证过）。**

Thanks