解题报告

周镇东

November 16, 2019

1 Digits of Number Pi

CF585F

1.1 题目大意

给定一个数字串 s，可能存在前导零。

定义一个长度为 d 的数字串 t 是“半出现的”，当且仅当存在一个长度为 $\lfloor \frac{d}{2} \rfloor$ 的 s 和 t 的公共子串。

给定两个无前导零的 d 位的数字 $x, y(x < y)$，问 $[x, y]$ 中所有的整数所代表的数字串中，有多少是“半出现的”。

答案对 $10^9 + 7$ 取模。

1.2 数据范围

- $1 \leq |s| \leq 1000$
- $2 \leq d \leq 50$

1.3 解题过程

在这里，我们认为字符串的下标从 0 开始。

首先，将字符串 s 的所有长度为 $\lfloor \frac{d}{2} \rfloor$ 的子串取出，并对这些串建立 AC 自动机。

于是,我们可以从 AC 自动机的空串节点开始,依次经过转移边 $t_0, t_1, \cdots, t_{|t|-1}$，并记录这个过程中匹配长度是否到过 $\lfloor \frac{d}{2} \rfloor$，来判断 t 是否是“半出现的”（若达到过，则是“半出现的”）。

定义 AC 自动机中代表长度为 $\lfloor \frac{d}{2} \rfloor$ 的串的节点集合为 S，为了方便，我们将所有属于 S 的节点的所有转移都设为自己。我们在执行第二段所述的转移操作时，一旦到达 S 中的节点，t 就一定是“半出现的”，上述修改转移的操作，就是为了到达 S 中节点后永远留在 S 中节点。于是，我们就不再需要记录匹配长度是否达到过 $\lfloor \frac{d}{2} \rfloor$，只需要判定最后转移到的节点是否属于 S 即可。

注意，之后提到的 AC 自动机都代表我们已经修改过的 AC 自动机。

我们可以采用数位 DP 的方法来解决剩下的问题。

令 i 表示在给定范围内确定数字串的前 i 位，j 表示转移到了 AC 自动机的状态 j ，a 表示数字串的前 i 位是否与 x 的前 i 位相同，$a=1$ 表示相同，$a=0$ 表示不同，b 表示数字串的前 i 位是否与 y 的前 i 位相同，取值方式和 a 一样。那么定义 $dp_{i,j,a,b}$ 表示 i,j,a,b 分别满足上述定义时的方案数，我们就可以得到相应的转移。

定义 lb,ub 分别表示第 $i+1$ 位可以选的数的下界和上界。具体地，当 $a=1$ 时，$lb=x_i$；否则 $lb=0$。当 $b=1$ 时，$ub=y_i$；否则 $ub=9$。

假设在 AC 自动机上节点 X 走转移边 Y 后到达的节点为 $Next_{X,Y}$ ，那么 $dp_{i,j,a,b}$ 会对如下 DP 值做出贡献，即状态 $dp_{i,j,a,b}$ 的状态可以转移到下述 DP 状态:

$$\forall k\in[lb,ub]\cap\mathbb{N},dp_{i+1,Next_{j,k},a\wedge[k=lb],b\wedge[k=ub]}$$

最后统计 AC 自动机上集合 S 中点的答案即可。

建 AC 自动机的时间复杂度为 $O(d\cdot|S|)$，DP 的时间复杂度为 $O(d^2\cdot|S|\cdot\alpha)$，其中 $\alpha=10$，表示数字字符集大小。所以，总时间复杂度为 $O(d^2\cdot|S|\cdot\alpha)$ ，可以通过此题。

2 Incorrect Flow

CF708D

2.1 题目大意

给定一个不一定正确的残量网络 $G(V,E)$，令 $n=|V|$，表示节点总数；$m=|E|$，表示边的总数。在这个网络中，节点 1 是源点，节点 n 是汇点。

对于每一条边 $e \in E$，我们定义容量函数 $c(e)$ 和流量函数 $f(e)$，分别表示这条边的容量和当前流量。

如果以下条件成立，那么该残量网络就是正确的：

1. 对于每条边 $e \in E$，其流量非负且不超过 $c(e)$，即 $0 \le f(e) \le c(e)$。
2. 对于每个节点 $v \in V$，且 v 不是源点或汇点 $(v \neq 1, v \neq n)$，则 v 的所有入边流量之和等于其所有出边流量之和。换句话说，节点 v 流量平衡。

但是 G 不一定是正确的残量网络，现在，你可以修改容量函数 $c(e)$ 和流量函数 $f(e)$，使得修改后的图时正确的残量网络。假设修改后的函数分别是 $c'(e)$ 和 $f'(e)$，那么这种修改方式的代价是 $\sum_{e \in E} |f(e)-f'(e)| + |c(e)-c'(e)|$。

问代价最小是多少。

2.2 数据范围

- $2 \le n \le 100, 0 \le m \le 100$
- 设第 i 条边从 u_i 到 v_i，容量为 c_i，当前流量为 f_i，$1 \le u_i, v_i \le n, u_i \neq v_i, 0 \le c_i, f_i \le 1000000$

2.3 解题过程

我们采用最小费用最大流来解决这个问题。

我将要建立一张新的费用流图。接下来的描述中，我们用 (a,b,c,d) 来表示一条由 a 到 b、容量为 c 、费用为 d 的边。

首先，考虑如何处理每一条边，以第 i 条边为例：

记 c_i', f_i' 分别表示修改后的流量和容量。

若 $c_i \geq f_i$，那么，我们分 3 类进行讨论：

1. $0 \leq f_i' \leq f_i$：花费为 $f_i - f_i'$，在费用流图上加入边 $(v_i, u_i, f_i, 1)$，表示流量每减一需要 1 的花费，且最多只能减 f_i 次；

2. $f_i < f_i' \leq c_i$：花费为 $f_i' - f_i$，在费用流图上加入边 $(u_i, v_i, c_i - f_i, 1)$，表示流量每加一需要 1 的花费，且最多只能加 f_i 次；

3. $f_i' > c_i$：花费为 $2(f_i' - c_i) + (c_i - f_i)$，当 f_i' 超过 c_i 时，c_i 也得随之增大，于是在费用流图上加入边 $(u_i, v_i, \infty, 2)$，表示流量每加一，容量也加一，需要 2 的花费，且可以无限加。

由于我们求的是最小费用最大流，所以，当边 $(v_i, u_i, f_i, 1)$ 有流量时，其余边必然没有流量，因为无论是 $(v_i, u_i, f_i, 1)$ 和 $(u_i, v_i, c_i - f_i, 1)$，还是 $(v_i, u_i, f_i, 1)$ 和 $(u_i, v_i, \infty, 2)$，都可以通过同时将两条边的流量减一的方法在不改变其他因素的情况下减小花费。类似地，当我们要使用第 3 类边时，第二类边必然满流，而第二类边的花费恰好是 $c_i - f_i$，再加上第三类边的花费，得到 $2(f_i' - c_i) + (c_i - f_i)$ ，而当 $f_i' > c_i$ 时，$f_i' = c_i'$，所以 $2(f_i' - c_i) + (c_i - f_i) = (f_i' - f_i) + (c_i' - c_i)$，符合定义。

若 $c_i < f_i$，由于至少花费 $f_i - c_i$，所以我们先将答案预加 $f_i - c_i$。剩下的也分 3 类讨论：

1. 若 $f_i' > f_i$，那么，我们可以先花费预加的 $f_i - c_i$ 的代价将 c_i 调节到 f_i，在这之后，每加一单位流量的同时容量也加一，故一单位花费为 2，可以无限加。所以加边 $(u_i, v_i, \infty, 2)$；

2. 若 $c_i \leq f_i' \leq f_i$，那么，我们可以花费预加的 $f_i - c_i$ 的代价来将流量和容量都调整到任意的 f_i' ，因为 $|f_i' - c_i| + |f_i - f_i'| = f_i - c_i$ 。所以加边 $(v_i, u_i, f_i - c_i, 0)$；

3. 若 $0 \leq f_i' < c_i$，那么，我们首先花费预加的 $f_i - c_i$ 的代价将流量调节到 c_i，接下来流量每减一需要花费 1 的代价，只能减 c_i 次，所以加边 $(v_i, u_i, c_i, 1)$。

同样地，还是由于是求最小费用最大流，第 1 类边和第 2、3 类边不可能共存，原因和之前提到的一样。

最后，我们需要解决节点流量的平衡问题。这个问题和无源汇上下界可行流中要解决的流量平衡问题类似，令 in_i 表示节点 i 的入流总量，out_i 表示节点 i 的出流总量，建立超级源 S 和超级汇 T，对于每个 i，建边 $(S, i, in_i, 0)$、$(i, T, out_i, 0)$。由于原图源点 1 和汇点 n 不要求流量平衡，所以我们要加一条边 $(n, 1, \infty, 0)$，加

这条边的作用和于有源汇上下界可行流中解决源汇流量不平衡时加的边的作用一样。

由于本题必然有解，所以最大流中连接超级源汇的边必然满流，即流量必然平衡。我们只需要使用任意时空复杂度可行的费用流算法求流量最大时的最小费用即可解决问题。

3 Permutation Tree

ARC095F

3.1 题目大意

给定 n，设 p 为 $(1,2,\ldots,n)$ 的一个排列，定义函数 $T(p)$ 表示以以下方式生成的一棵无根树：

首先，设这棵树有 n 个节点，编号为 $1,2\ldots,n$ ，对于所有 $i=1,2,\ldots,n$ ，执行以下操作：

- 如果 $p_i=1$ ，什么都不做。
- 如果 $p_i\neq 1$ ，找到一个最大的满足 $p_j<p_i$ 的 j，在节点 i 和节点 j 之间连一条边。

现在给定一棵 n 个节点的无根树 t，问是否存在一个排列 p，满足 $T(p)$ 与 t 同构；如果存在，请求出字典序最小的 p。

3.2 数据范围

- $2\leq n\leq 10^5$

3.3 解题报告

首先，我们对建树的过程做一些探讨。由于对于每一个点要执行的操作是独立的，所以我们可以以任何顺序执行每一个点的操作。定义排列 q 为排列 p 的复合逆，即 q 满足 $q_{p_i}=i$ 。由于 $T(p)$ 的节点编号信息不影响是否与 t 同构，所以我们可以将 $T(p)$ 的编号映射到任意排列，在这里，我们将节点编号映射到排列 p。

于是，我们可以修改对操作的描述，得到以下等价描述：

对于所有 $i=1,2,\ldots,n$ ，执行以下操作：

- 如果 $i=1$ ，什么都不做。
- 如果 $i\neq 1$ ，找到一个满足 $j<i$ 的最大的 q_j，在节点 i 和节点 j 之间连一条边。

于是，我们可以更明确地指出：如果我们称 q_x 为节点 x 的权值，那么对于每一个节点 i，我们都会加入一条连接 i 和编号小于 i 的节点中权值最大的点的一条边。

我们记 $\max_i = \max_{j=1}^{i}(q_i)$。

定义链 L 为一个有限数列，L 中的元素对应的节点依次连成一条链，我们将不断在这个数列的末端加入元素。它代表的意义在接下来的分类讨论中会说明。

我们将依次对 $i = 1, 2, \ldots, n$ 分以下几类讨论：

1. $i = 1$：节点 1 不向外连任何边，它将作为链 L 的第一个元素。
2. $i \neq 1, q_i < \max_{i-1}$：首先，这个节点会连向权值为 $\max_{i-1}$ 的节点，但是由于当 $j \geq i$ 时，$\max_j \geq \max_{i-1} > q_i$（根据定义），所以任何 $j(j > i)$ 都不会有边连向节点 i，故节点 i 只有一条边相连，是一个叶子节点。
3. $i \neq 1, q_i > \max_{i-1}$：首先，这个节点会连向权值为 $\max_{i-1}$ 的节点。当 $j \geq i$ 时，$\max_j \geq q_i > \max_{i-1}$，所以任何 $j(j > i)$ 都不会有边连向权值为 $\max_{i-1}$ 的节点，故这个节点将不再会有新边。我们将节点 i 加入到链 L 末端。
4. $i \neq 1, q_i = \max_{i-1}$：由于 q 是个排列，所以不可能有任意两个数一样。$\max_{i-1} = q_i$ 表明存在一个 $j < i$，使得 $q_i = q_j$，矛盾，所以这种情况不存在。

于是我们得到了链 L，它是 $T(p)$ 中的一条链。当且仅当存在一个 i ，使得 $\max_i = q_v$ 时，v 会在链 L 中出现。因为根据构造 L 的定义可得：

- 当 $i = 1$ 时，$\max_1 = q_1$，i 会被加入。
- 当 $i \neq 1, q_i > \max_{i-1}$ 时，$\max_i = \max(\max_{i-1}, q_i) = q_i$，$i$ 会被加入。

同样根据定义可得除了链 L 以外其他点都属于第一次分类讨论中的第 2 类，是叶子。

于是我们可以得知 $T(p)$ 的中必然存在一条链，使得其他节点都是连接该链上节点的叶子。

接下来我们要通过给出合法的构造方案，证明任何 n 个点的满足这种性质的树都能被构造出来。

首先，我们在 $T(p)$ 中寻找链 L ，只需要删除所有叶子，并将链 L 两端的任一叶子加到 L 的两端。

接下来我们将以这个被叶子为队头 L 看做一个可以支持前后删除插入的双端队列，并讲述构造方式：

1. 我们的目的是构造排列 q，等价于构造 p。首先令 $q=(1)$，并使 q 中有 i 个元素时 q 是 $(1,2,\ldots,i)$ 的一个排列。

2. 记 L 当前的第一个元素为节点 x，并假设我们将要插入 q 中的第 i 个元素。接下来我们将分两类来描述。

   A. 如果 x 连接的叶子节点个数不够，那么我们令 $q_i=\max_{i-1}$，并对于 $j<i$ 且 $q_j=q_i$，令 q_j 加一。随后更新 max 数组。

   B. 如果够了，那么我们令 $q_i=i$，并弹出 L 的第一个元素。

   这些操作不会改变 q 的前 $i-1$ 个元素的大小关系，但是可以使得 q_i 取得合适的值，从而使节点 i 连出正确的边。这两种类型分别对应之前描述的第二类和第三类的讨论。

最后，我们需要找到使排列 p 字典序最小的构造方案。

由于链 L 有两个端点可以加叶子，所以我们分别求出这两个方向的最小字典序排列之后取最优解即可。

事实上，在确定链 L 之后，上述构造方案是最优的。可以发现，上述构造方案有一个特点：在执行操作 B 之后，对于所有 $j<i$，q_j 都不会再变，因为 $q_j<\max_i$，对应到 p 上就是我们可以在此之后不改变前 $i-1$ 个元素的值并得到解。于是我们可以证明 A 操作是最最优的。因为如果 A 操作中 q_i 取更小的值，对应到排列 p 上就是将一个极大值插入到排在前 $i-1$ 位的某个位置上，由于这个值比前 $i-1$ 个数都要大，将其插入显然更劣，所以原方案中 A 操作是最优的。由于 q 的前 $i-1$ 个位置上出现了 $1,2,\ldots,i-1$ 各一次，所以 $\max_{i-1}=i-1$，$i\geq q_i>\max_{i-1}=i-1\Longrightarrow q_i=i$，于是 B 操作唯一，所以最优。

至此，我们证明了构造方法的最优性。

注意到我们在构造的过程中每一次每向 q 中插入一个元素都需要 $O(1)$ 的时间复杂度，需要插入 $n-1$ 次，时间复杂度 $O(n)$；而寻找链 L 的过程也可以在 $O(n)$ 的时间复杂度内解决，所以总时间复杂度为 $O(n)$。