NOIplus2016模拟赛Solution

罗哲正

安徽师范大学附属中学

October 15, 2016

幻想

- 拿到这个问题，首先要观察S数列的性质。

幻想

- 拿到这个问题，首先要观察S数列的性质。
- 每次把数列倍长K倍，依次加上$0, 1, \cdots, k-1$。

幻想

- 拿到这个问题，首先要观察S数列的性质。
- 每次把数列倍长K倍，依次加上$0, 1, \cdots, k-1$。
- 如果我们把每次加上的数直接写在原来的数字之前。

幻想

- 拿到这个问题，首先要观察S数列的性质。
- 每次把数列倍长K倍，依次加上$0, 1, \cdots, k-1$。
- 如果我们把每次加上的数直接写在原来的数字之前。
- 例如$K=2$，我们倍长三次，可以得到：$000, 001, 010, 011, 100, 101, 110, 111$。

幻想

- 拿到这个问题，首先要观察S数列的性质。
- 每次把数列倍长K倍，依次加上$0, 1, \cdots, k-1$。
- 如果我们把每次加上的数直接写在原来的数字之前。
- 例如$K = 2$，我们倍长三次，可以得到：$000, 001, 010, 011, 100, 101, 110, 111$。
- 可以发现就是自然数序列，那么S_i其实就是i在K进制下的数位和模K的值。

幻想

- 注意到$m = R - L + 1$的总和不是很大，考虑暴力。

幻想

- 注意到$m = R - L + 1$的总和不是很大，考虑暴力。
- 暴力求S_i是$\log_K i$的，不能接受。

幻想

- 注意到$m = R - L + 1$的总和不是很大，考虑暴力。
- 暴力求S_i是$\log_K i$的，不能接受。
- 我们从L开始每次加一，维护K进制高精度，同时维护S_i。

幻想

- 注意到$m = R - L + 1$的总和不是很大，考虑暴力。
- 暴力求S_i是$\log_K i$的，不能接受。
- 我们从L开始每次加一，维护K进制高精度，同时维护S_i。
- 计算复杂度：对于第t位，改变的次数是$\left\lfloor \frac{m}{K^t} \right\rfloor + O(1)$。

幻想

- 注意到$m = R - L + 1$的总和不是很大，考虑暴力。
- 暴力求S_i是$\log_K i$的，不能接受。
- 我们从L开始每次加一，维护K进制高精度，同时维护S_i。
- 计算复杂度：对于第t位，改变的次数是$\left\lfloor \frac{m}{K^t} \right\rfloor + O(1)$。
- 对t求和就是$O(m + \log_K R)$。

幻想

- 注意到$m = R - L + 1$的总和不是很大，考虑暴力。
- 暴力求S_i是$\log_K i$的，不能接受。
- 我们从L开始每次加一，维护K进制高精度，同时维护S_i。
- 计算复杂度：对于第t位，改变的次数是$\left\lfloor \frac{m}{K^t} \right\rfloor + O(1)$。
- 对t求和就是$O(m + \log_K R)$。
- 于是这样暴力就可以过了，时间复杂度$O(T \log_K R + \sum R - L)$

幻想

- 首先我们要注意到，排列中每个元素都是等价的。

幻想

- 首先我们要注意到，排列中每个元素都是等价的。
- 如果我们把每一步的排列都乘上一个置换，其发生的概率不变。

幻想

- 首先我们要注意到，排列中每个元素都是等价的。
- 如果我们把每一步的排列都乘上一个置换，其发生的概率不变。
- 例如从$(1,3,2)$，变到$(3,2,1)$，和从(2,3,1)变到$(1,2,3)$，的概率是一样的。

幻想

- 首先我们要注意到，排列中每个元素都是等价的。
- 如果我们把每一步的排列都乘上一个置换，其发生的概率不变。
- 例如从$(1,3,2)$，变到$(3,2,1)$，和从(2,3,1)变到$(1,2,3)$，的概率是一样的。
- 那么不妨把A,B同时乘以B的逆置换，从而问题变成把A变成单位置换。

- 接下来考虑如何记录当前置换，$n!$种排列全部记录显然不现实，我们考虑哪些东西会对每次的操作及变化产生影响。

- 接下来考虑如何记录当前置换，$n!$种排列全部记录显然不现实，我们考虑哪些东西会对每次的操作及变化产生影响。
- 注意到如果两个置换拥有同构的轮换，那么从中随机选择三个元素而得到所有可能的置换，其结果的轮换也必定是同构的。

- 接下来考虑如何记录当前置换，$n!$种排列全部记录显然不现实，我们考虑哪些东西会对每次的操作及变化产生影响。
- 注意到如果两个置换拥有同构的轮换，那么从中随机选择三个元素而得到所有可能的置换，其结果的轮换也必定是同构的。
- 于是，如果我们只关心置换的轮换情况，那么我们只需要记录轮换大小的分布就可以了，例如$(2,4,5,1,3,6)$，其轮换就是$(2,4,1),(5,3),(6)$，记录为$<1,2,3>$表示有大小为$1,2,3$的轮换各一个。

- 接下来考虑如何记录当前置换，$n!$种排列全部记录显然不现实，我们考虑哪些东西会对每次的操作及变化产生影响。
- 注意到如果两个置换拥有同构的轮换，那么从中随机选择三个元素而得到所有可能的置换，其结果的轮换也必定是同构的。
- 于是，如果我们只关心置换的轮换情况，那么我们只需要记录轮换大小的分布就可以了，例如$(2,4,5,1,3,6)$，其轮换就是$(2,4,1),(5,3),(6)$，记录为$<1,2,3>$表示有大小为$1,2,3$的轮换各一个。
- 这样记录的方案数是多少呢？是n的划分数D_n，在$n \leq 14$的情况下是不超过150的。

- 那么我们使用$f_{i,j}$表示进行i次操作得到轮换结构为j的排列的概率，每次枚举操作就可以简单的转移了，当然这个转移可以预处理成一个矩阵，复杂度$O(mn^3)$，预处理转移复杂度为$O(mn^2)$。

- 那么我们使用$f_{i,j}$表示进行i次操作得到轮换结构为j的排列的概率，每次枚举操作就可以简单的转移了，当然这个转移可以预处理成一个矩阵，复杂度$O(mn^3)$，预处理转移复杂度为$O(mn^2)$。
- 显然初始时只有$f_{i,A}$为1，其余都是0。

- 那么我们使用$f_{i,j}$表示进行i次操作得到轮换结构为j的排列的概率，每次枚举操作就可以简单的转移了，当然这个转移可以预处理成一个矩阵，复杂度$O(mn^3)$，预处理转移复杂度为$O(mn^2)$。
- 显然初始时只有$f_{i,A}$为1，其余都是0。
- 注意到我们只是合并了轮换结构相同的排列的概率，而单位置换的轮换结构是n个1是唯一的，所以我们得到的单位置换的概率是可以直接使用的。

- 那么我们使用$f_{i,j}$表示进行i次操作得到轮换结构为j的排列的概率，每次枚举操作就可以简单的转移了，当然这个转移可以预处理成一个矩阵，复杂度$O(mn^3)$，预处理转移复杂度为$O(mn^2)$。
- 显然初始时只有$f_{i,A}$为1，其余都是0。
- 注意到我们只是合并了轮换结构相同的排列的概率，而单位置换的轮换结构是n个1是唯一的，所以我们得到的单位置换的概率是可以直接使用的。
- 使用矩阵乘法加速转移，复杂度$O(n^3 + D_n^3 \log m)$。

现实

- 问题：给一个有向图，问对于那些点，删除它和它的相邻边之后，图是一个DAG。

现实

- 问题：给一个有向图，问对于那些点，删除它和它的相邻边之后，图是一个DAG。
- 转化：对于图中存在的所有回路，求交。

现实

- 问题：给一个有向图，问对于那些点，删除它和它的相邻边之后，图是一个DAG。
- 转化：对于图中存在的所有回路，求交。
- 两个特判：

现实

- 问题：给一个有向图，问对于那些点，删除它和它的相邻边之后，图是一个DAG。
- 转化：对于图中存在的所有回路，求交。
- 两个特判：
    1. 如果图本来就是DAG，输出所有点。

现实

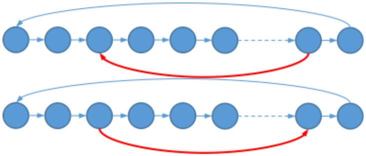
- 问题：给一个有向图，问对于那些点，删除它和它的相邻边之后，图是一个DAG。
- 转化：对于图中存在的所有回路，求交。
- 两个特判：
  1. 如果图本来就是DAG，输出所有点。
  2. 找到图中一个环删去，若图中仍存在一个环，输出0个点。

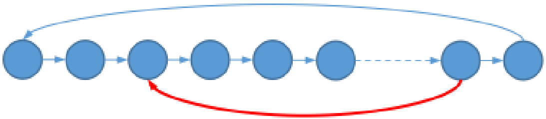
- 我们接下来解决一般情况：

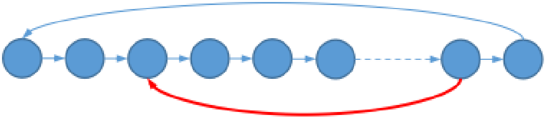
- 我们接下来解决一般情况：
- 首先我们把每个点拆成入点和出点，入点像出点连一条边，每条边从u的出点连向v的入点，然后环交从点集变成了边集，这样易于处理。

- 我们接下来解决一般情况：
- 首先我们把每个点拆成入点和出点，入点像出点连一条边，每条边从u的出点连向v的入点，然后环交从点集变成了边集，这样易于处理。
- 首先用dfs等方法找到一个环，那么环交必定是这个环的子集。

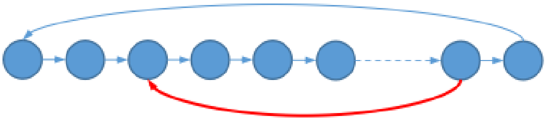
- 我们接下来解决一般情况:
- 首先我们把每个点拆成入点和出点，入点像出点连一条边，每条边从u的出点连向v的入点，然后环交从点集变成了边集，这样易于处理。
- 首先用dfs等方法找到一个环，那么环交必定是这个环的子集。
- 接下来我们把环断成链，考虑以下两种路径:

- 我们接下来解决一般情况：
- 首先我们把每个点拆成入点和出点，入点像出点连一条边，每条边从u的出点连向v的入点，然后环交从点集变成了边集，这样易于处理。
- 首先用dfs等方法找到一个环，那么环交必定是这个环的子集。
- 接下来我们把环断成链，考虑以下两种路径：
- 



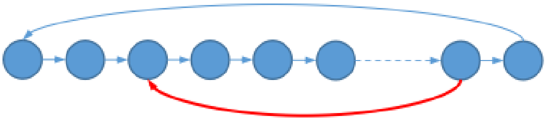


-
- 对于第一种路径，我们在非环边组成的DAG上DP出每个点顺着非环边能走到的最靠前的环上的点，再求出逆着环边能走出的最靠后的环上的点。

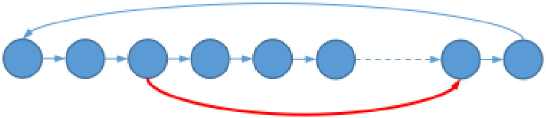


-

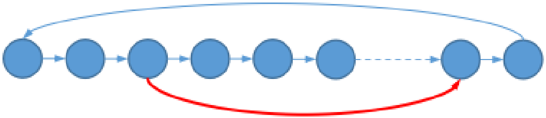
- 对于第一种路径，我们在非环边组成的DAG上DP出每个点顺着非环边能走到的最靠前的环上的点，再求出逆着环边能走出的最靠后的环上的点。
- 那么对于一个点x，若x能通过非环边走到x之前的点，那么环上x之后的边都不在环交中。



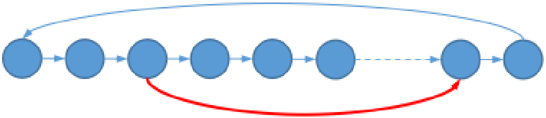
-
- 对于第一种路径，我们在非环边组成的DAG上DP出每个点顺着非环边能走到的最靠前的环上的点，再求出逆着环边能走出的最靠后的环上的点。
- 那么对于一个点x，若x能通过非环边走到x之前的点，那么环上x之后的边都不在环交中。
- 同理对于一个点x，若x能逆着非环边走到x之后的点，那么环上x之前的边都不在环交中。



-



-
- 对于第二种路径，我们在非环边组成的DAG上DP出每个点顺着环边能走到的最靠后的环上的点、



- 对于第二种路径，我们在非环边组成的DAG上DP出每个点顺着环边能走到的最靠后的环上的点、
- 对于一个点x，若x能通过非环边走环上最靠后的点y在x之后，那么环上(x, y)区间内的边一定不在环交上，同时由于y是能走到的最靠后的点，那么不会出现把非环交的边算入环交的边的情况。

- 那么闲拓扑排序特判，接着找环，再做三次DAG上的DP，然后扫一遍环上的点区间打删除标记，最后扫一遍环输出答案。

- 那么闲拓扑排序特判，接着找环，再做三次DAG上的DP，然后扫一遍环上的点区间打删除标记，最后扫一遍环输出答案。
- 区间打标记可以使用线段树，也可以使用并查集+暴力，今天复杂度是$O(1)$的。

- 那么闲拓扑排序特判，接着找环，再做三次DAG上的DP，然后扫一遍环上的点区间打删除标记，最后扫一遍环输出答案。
- 区间打标记可以使用线段树，也可以使用并查集+暴力，今天复杂度是$O(1)$的。
- 由于只需要拓扑排序+DP+打标记，时间复杂度$O(n+m)$。