概率与期望

zzq

概率与期望

- 概率：某件事情发生的可能性大小。记$P(S)$为S事件发生的概率。例如有一个变量X，$P(X = x)$就是变量X的值为x的概率。显然我们有$\sum_x P(X = x) = 1$。
- 期望：某个随机变量值大小的平均数。记$E(X)$为变量X的期望大小，我们就有$E(X) = \sum_x P(X = x)x$。
- 一个经典的式子是：对于非负实数变量 X 我们有 $E(X) = \int_0^\infty P(X \geq x)dx$。对于离散的情形即 X 是非负整数变量时，我们有 $E(X) = \sum_{x=1}^\infty P(X \geq x)$。
- 当然它也可以推广到可以为负实数的情况：$E(X) = \int_0^\infty P(X \geq x)dx - \int_{-\infty}^0 P(X \leq x)dx$。
- 概率的线性性：$E(A + B) = E(A) + E(B)$，即和的期望等于期望的和。

概率与期望

- 理论部分讲完了，是不是很简单。
- 接下来就全是杂题了。难度与顺序无关。

Little Elephant and Broken Sorting

- 有一个长度为n的排列$p_1, p_2 \dots p_n$，接下来进行m步操作。
- 每步操作给定两个不同的正整数a, b，然后有50%的概率交换p_a和p_b。
- 求最后排列的逆序对数的期望，保留6位小数。
- $1 \leq n, m \leq 1000$。

* Source: Codeforces Round #157

Little Elephant and Broken Sorting

- 考虑记$f_{i,j}$表示$p_i < p_j$的概率，那么由期望的线性性，逆序对数的期望就是$\sum_{i>j} f_{i,j}$。
- 考虑交换p_x和p_y对f的影响。$f_{i,j}$ $(i,j \notin \{x,y\})$ 显然不会受到影响。对于每个$i \notin \{x,y\}$，$f_{i,x}$和$f_{i,y}$都会变成$\frac{f_{i,x}+f_{i,y}}{2}$，$f_{x,i}$和$f_{y,i}$同理。$f_{x,y}$和$f_{y,x}$都会变成$\frac{1}{2}$。注意到变化的f位置只有O(n)个，所以我们每次按如上所述更新f即可。
- $O(nm)$。

Quicksort Strikes Back

```
def quicksort(a,l,r,h):
  if h>1 and l<r:
    m=partition(a,l,r)
    quicksort(a,l,m-1,h-1)
    quicksort(a,m+1,r,h-1)
def partition(a,l,r):
  pivot=a[r]
  i=l
  for j in l..r-1:
    if a[j]<pivot:
      swap(a[i],a[j])
      i+=1
  swap(a[i],a[r])
  return i
```

- 输入n, k，输出对一个$1,2 \ldots n$的随机排列$a_1, a_2 \ldots a_n$调用$quicksort(a, 1, n, k)$后a“几乎被正确排序”的概率，对输入的大质数q取模输出。“几乎被正确排序”定义为可以在删掉序列中至多一个元素之后使得整个序列递增。
- $1 \leq n, k \leq 500$，$10^8 \leq q \leq 10^9$，q为质数。

* Source: hihocoder挑战赛35，有加强

Quicksort Strikes Back

- 显然这就是个计数题，我们只要数“几乎被正确排序”的排列个数。
- 我们称一个序列“差点被正确排序”当且仅当它“几乎被正确排序”且不有序。
- 我们记f[i][j]为长度为j，调用quicksort(a,1,j,i)后有序的序列个数，g[i][j]为长度为j，调用quicksort(a,1,j,i)后差点被正确排序的序列个数。
- 我们有f[1][j]=1，g[1][j]=(j-1)*(j-1)。这是因为我们考虑删除哪个数后能变为有序，这个数本身有j种方案，有j-1种可能放错的位置。考虑有没有可能删去多个数后都能变为有序，容易发现这只有可能是相邻的两个数放反了，共j-1种，所以g[1][j]=j(j-1)-(j-1)=(j-1)*(j-1)。

Quicksort Strikes Back

- 考虑partition干的事，相当于是把≤ a_r的数按原序列的顺序放在左边，其他数按某个顺序放在右边。
- 先考虑最后有序的情况，枚举a_r的值，那么我们有

$$f[i][j] = \sum_{k=1}^{j} C_{j-1}^{k-1} f[i-1][k-1] f[i-1][i-k]$$ 。

- 最后“差点被正确排序”的情况，注意到只有可能是相邻的两个数放反了才会有多种删除一个数后有序的方案，所以我们只需要考虑是哪一边“差点被正确排序”导致了无法有序即可。

Quicksort Strikes Back

- 即

$$\mathrm{g}[i][j] = \sum_{k=1}^{j} C_{j-1}^{k-1}(f[i-1][k-1]g[i-1][i-k] + g[i-1][k-1]f[i-1][i-k])$$

Graph Game

- WJMZBMR决定在一个环套树上运行他发明的最新算法：环套树分治。
- 算法流程大约如下：
  - Solve(T): 接受一个联通图T
    - totalCost+=|T|（T的节点数）
    - 在T的节点中随机选取一个点x
    - 在T中删掉x这个点
    - 对于得到的所有非空联通块递归调用Solve
- 对于给定的n个点的环套树，求对它调用Solve后totalCost的期望值，保留六位小数。$n \leq 3000$。

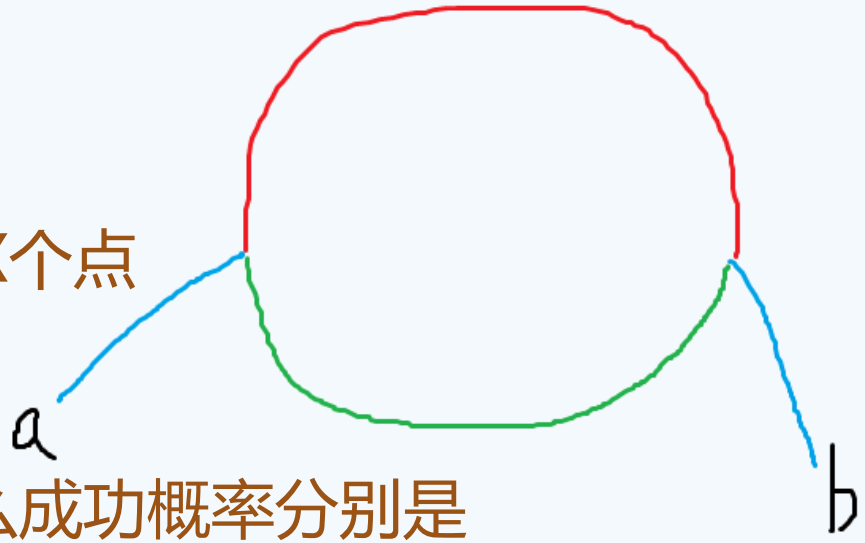
* Source: Codeforces Round #146

Graph Game

- 设$p(a,b)$为删去a时，b与a联通的概率，那么答案就是$\sum_{a,b} p(a,b)$。
- 先考虑树的情况。设a到b的路径上有n个点（包括a和b），我们断言$p(a,b)=\frac{1}{n}$。
- 考虑对a和b所在的联通块大小归纳证明。如果当前联通块就这一条路径，那么确实成立。否则设联通块大小为m。考虑第一步选了啥，如果选的在路径上（概率$\frac{n}{m}$），那么必须得选到a才行（概率$\frac{1}{n}$）。否则（概率$\frac{m-n}{m}$），我们考虑包含a和b的联通块，由归纳假设概率为$\frac{1}{n}$。综上概率就是$\frac{1}{n}$。

Graph Game

- 现在多了个环。如果两个点a, b之间只有一条路径或$a = b$，那么之前的证明仍然成立，即$p(a, b) = \frac{1}{dis(a,b)}$。
- 否则a到b的路径肯定经过了环，如图：
- 我们设a到环的路径和b到环的路径上有X个点
- 环上接点两侧分别有Y和Z个点
- 考虑把它当做两条不同的路径考虑，那么成功概率分别是$\frac{1}{X+Y+1}$和$\frac{1}{X+Z+1}$，但是这样会多算这两条路径都合法的情况
- 都合法的概率就是$\frac{1}{X+Y+Z+1}$，扣掉即可。枚举a,b如上计算，$O(n^2)$。



DZY Loves Games

- 有一个连通的迷宫，你可以理解为一个n个点m条边的无向连通图。有些点可能有陷阱，保证1号点没有陷阱而n号点存在陷阱。一开始你有k条命，每进入一个有陷阱的点你会损失一条命。
- DZY听说这个游戏有一个隐藏关卡：如果进入n号点时你恰好剩余2条命，你会先被扣除一条命，然后进入一个隐藏的极限关卡（在触发关卡之前也可以一次或多次进入n号点）。DZY很想进入这个关卡，但是他游戏水平不行，所以他的游戏策略是从1号点开始，每次随机走到一个与当前点相邻的点。他想知道他能够触发隐藏关卡的概率，保留4位小数。
- $2 \leq n \leq 500$，$1 \leq m \leq 10^5$，$2 \leq k \leq 10^9$，有陷阱的点个数不超过101。

* Source: Codeforces Round #255

DZY Loves Games

- 设$w(u,v)$表示从u开始瞎走，第一次撞到的陷阱（出发的那一刻不算）是v的概率。
- 假设我们求出了w，我们就是要枚举一个撞陷阱序列$A_0, A_1 \dots A_{k-1}$，其中$A_0 = 1$，$A_{k-1} = n$，这种序列的形成概率就是$\prod_{i=0}^{k-2} w(A_i, A_{i+1})$。这个过程就相当于是在求矩阵乘法，每次把当前点进行转移，那么我们使用矩阵快速幂即可，复杂度就是$O(a^3 \log(k))$（a是陷阱个数）。
- 我们记$h(u,v)$表示从u开始瞎走，第一次撞到的陷阱（如果u是陷阱也算）是v的概率。那么不难用h推出w：枚举第一步的走法，这个复杂度是$O(am)$。接下来我们就只需要求出h。

DZY Loves Games

- 如果给定u，求$h(u,v)$十分容易：记a_u为每个时刻处在u的概率之和，那么有$a_u = \sum_{v \notin traps,(v,u) \in E} \frac{a_v}{\deg(v)} + [u\text{为起点}]$，即 $a_u - \sum_{v \notin traps,(v,u) \in E} \frac{a_v}{\deg(v)} = [u\text{为起点}]$。高斯消元解，单次复杂度$O(n^3)$。
- 注意到每次高斯消元的矩阵都长得一样，而只有等式右边的常数不一样。
- 高斯消元相当于是对于给定的矩阵A和向量B，求向量X使$AX = B$。考虑用高斯消元直接求出A的逆矩阵A^{-1}，我们就有$X = A^{-1}B$，而矩阵乘向量的复杂度是$O(n^2)$的，所以总复杂度就变成$O(n^3)$的了。实际上也可以把求逆矩阵理解为记录下消元过程，最后再带入右侧的值。

Expected Tree Degrees

- 有一棵n个点的树，由如下方式生成：
- 根为1，第i（$i > 1$）个点的父亲为$[1, i-1]$的均匀随机整数。
- 我们定义一棵树的权值为$\sum_{i=1}^{n} \deg(i)^2$（deg是度数），求这棵树权值的期望，保留六位小数。
- $1 \leq n \leq 10^6$。

* Source: CSAcademy Round #18

Expected Tree Degrees

- 考虑设$f(i,j)$表示i个点的树，度数为j的点的期望个数，那么考虑i的父亲的度数，原来是x就会变成x+1，所以我们有：
- $f(i,j)=f(i-1,j)+\frac{f(i-1,j-1)}{i-1}-\frac{f(i-1,j)}{i-1}+[j=1]$（中间两项是要考虑选到度数为j-1和度数为j的点的贡献）。
- 这样dp显然太慢了。注意到这棵树是随机的，且输出只需要保留六位小数，可以猜测过大的j没有意义。事实上跑一下会发现$f(10^6,50)$大约是10^{-14}级别，所以我们只需要跑$j\le 50$的状态。
- 复杂度$O\left(n\log\left(\frac{1}{\epsilon}\right)\right)$。

Combining Slimes

- 有一个初始为空的序列，每次有$\frac{p}{10^9}$的概率在末尾添加一个1，有$1-\frac{p}{10^9}$的概率在末尾添加一个2。
- 如果序列末尾两个数为同一个数x，那么它们会合并为$x+1$。
- 当序列有n个数且无法合并时结束。
- 求结束时序列中所有数的和的期望，保留六位小数输出。
- $1 \le n \le 10^9, 1 \le p < 10^9$。

* Source: Wunder Fund Round 2016

Combining Slimes

- 超过50的数出现的概率非常小，对答案的贡献可以忽略不计。
- $a(i,j)$表示序列长度限制为i，最左侧出现j的概率。
- $a(i,j) = a(i,j-1)a(i-1,j-1)$
- $b(i,j)$表示序列长度限制为i，第一次出现的数为2，最左侧出现j的概率。
- $b(i,j) = b(i,j-1)a(i-1,j-1)$

Combining Slimes

- $f(i,j)$表示序列长度限制为i，一开始最左侧为j，结束后最左侧的j保持不动的和的期望。
- $f(i,1)=1+\frac{\sum_{k=2}^{50} f(i-1,k)b(i-1,k)\left(1-a(i-2,k)\right)}{\sum_{k=2}^{50} b(i-1,k)\left(1-a(i-2,k)\right)}$，最左侧为1时只需第一次出2
- $f(i,\mathrm{j})=j+\frac{\sum_{k=1}^{j-1} f(i-1,k)a(i-1,k)\left(1-a(i-2,k)\right)}{\sum_{k=1}^{j-1} a(i-1,k)\left(1-a(i-2,k)\right)}\ (j>1)$ ，最左侧为j时右边的数必须<j，否则合出的过程就会与j合并。
- 答案就是$\sum_{i=1}^{50} f(n,i)a(n,i)(1-a(n-1,i))$。
- 直接递推到n太慢了，但我们可以发现$a(n,k)\approx a(50,k)\ (n>50)$，所以我们可以把$n>50$的$a$和$b$直接用$a(50)$和$b(50)$近似，然后用矩阵快速幂优化。

期望

- 有n个实数变量，第i个变量在$[l_i, r_i]$独立均匀随机，求和的k次方的期望 $\bmod 10^9 + 7$。
- $1 \le n \le 12$，$1 \le k \le 10^9$，$0 \le l_i < r_i \le 10^7$。

* Source: ysy题改编

期望

- 记$f_i(j)$表示前i个变量和>=j的概率，那么我们有

$$f_i(j) = \frac{1}{r_i - l_i} \int_{j-r_i}^{j-l_i} f_{i-1}(t)dt$$

- 考虑维护f_i，容易发现f_i是一个分段多项式函数，f_i的断点就是f_{i-1}的断点加上l_i或r_i，那么f_i的段数就是$O(2^n)$级别的，可以暴力地维护这个积分。

期望

- 我们欲求的是$E(x^k)$。如果求的是$E(x)$，我们只需要算$\int_{j=0}^{\infty} f_n(j)dj$。
- 我们对期望的计算式进行推广，容易得到：
- $E(f(X)) = \int_0^{\infty} f'(X)P(X \geq x)dx$
- 所以我们只需求$\int_{j=0}^{\infty} kj^{k-1} f_n(j)dj$，直接暴力计算即可。

随机数生成器

- 有一个长度为n的整数列$a_1, a_2 \dots a_n$，每个元素在$[1, x]$中的整数中均匀随机生成。
- 有q个询问，第i个询问的结果是下标在$[l_i, r_i]$的元素的最小值。
- 求这q个询问结果的最大值的期望，mod 666623333。
- $1 \leq n, q \leq 2000$，$1 \leq x \leq 10^7$。

* Source: luogu3600，有加强

随机数生成器

- 先假装$1 \le x \le 2000$。
- 继续用期望公式，$E(ans) = \sum_{i=0}^{\infty} P(ans > i) = \sum_{i=0}^{\infty}(1 - P(ans \le i))$。
- $ans \le t$就是每个询问结果都$\le t$。
- 考虑对前缀进行dp，但是区间关系较为复杂不容易dp。
- 注意到如果一个区间包含了另一个区间，那么这个区间就无法贡献答案，因为它的结果不大于它包含的那个区间的结果。所以我们可以先去掉互相包含的区间，现在我们就可以假设没有区间两两包含。

随机数生成器

- 记f_i为右端点不超过i的区间询问结果都$\leq t$的概率。
- 若不存在右端点$= i$的区间，$f_i = f_{i-1}$。
- 否则记右端点为i的区间左端点为j，枚举区间内最后一个$\leq t$的数的位置l，那么我们有$f_i = \sum_{l=j}^{i} f_{l-1} \frac{t}{x}\left(1 - \frac{t}{x}\right)^{i-l}$。前缀和优化即可。
- 这样我们就在$O(xn)$的时间内解决了问题。
- 观察转移方程发现答案其实是一个关于p的$O(n)$次多项式，也就是关于i的$O(n)$次多项式。所以它的前缀和仍然是一个$O(n)$次多项式，我们算出前$O(n)$项和前缀和，插值出x处的前缀和即可。

喂鸽子

- 有n只鸽子，每秒小Z会等概率选择一只鸽子并给他一粒玉米。一只鸽子饱了当且仅当它吃了的玉米粒数量$\geq k$。 小Z想要你告诉他，期望多少秒之后所有的鸽子都饱了。
- 你只需要输出这个值mod 998244353。
- $1 \leq n \leq 100$，$1 \leq k \leq 5000$。

* Source: uoj449，有加强

喂鸽子

- 我们称喂给没饱的鸽子食物为有效的喂食，我们把每次有效的喂食喂到的鸽子写成一个序列，那么这个序列长度就是nk，考虑进行dp。
- 我们记$f_{i,j}$为定了序列前i个，喂饱了鸽子$1\sim j$，定下了序列中这喂饱的j只鸽子部分的概率和。注意这里状态设计时相当于序列里这个前缀只填上了$0\sim j$，0的部分要由后面饱的鸽子补足。同样地，$g_{i,j}$为定了序列前i个，喂饱了鸽子$1\sim j$，定下了序列中这喂饱的j只鸽子部分的概率和*期望实际长度，这里实际长度就是加上喂给已经饱了的鸽子的部分的。

喂鸽子

- 转移就比较容易了，如果是加一个0就直接概率乘上$\frac{1}{n-j}$，期望加上概率$*\frac{n}{n-j}$（每次有$\frac{n-j}{n}$的概率是有效喂食，所以期望要喂$\frac{n}{n-j}$次）。如果是喂饱一只鸽子就只要多选出来m-1个0，改成现在这个i+1就行了，即转移(i,j)时方案数乘上C_{i-jm}^{m-1}。
- 复杂度$O(n^2m)$。
- 不是很好理解，没听懂的可以参见 http://uoj.ac/submission/309764

Expected Value

- 有一个n个点m条边的连通图，你从1号点开始，每次会随机走到一个相邻点，问期望要走多少步才能第一次走到n号点。对998244353取模输出。
- $2 \leq n \leq 3000$，$n-1 \leq m \leq 5000$，数据较为随机。

* Source: ptz camp 2019

Expected Value

- 这题需要的前置知识是Berlekamp-Massey算法，它可以求出一个数列的最短递推式。
- 考虑算出p_i表示i步后还没到达n号点的概率。记$f_{i,j}$表示当前走了i步，在第j个点，没到达过n号点的概率，那么如果x->y有边，$f_{i,x}$会转移到$f_{i+1,y}$。注意到转移与i无关，那么就是f_i这一向量乘了一个矩阵M得到了f_{i+1}，所以$f_i = f_0 M^i$。注意到M的特征多项式次数不超过n，所以f_i的最短递推式长度不超过n，而$p_i = \sum_j f_{i,j}$最短递推式长度也不超过n。我们可以求出$p_0, p_1 \dots p_{n+n+n}$，进而使用Berlekamp-Massey算法求出它的最短递推式。

Expected Value

- 考虑一个递推序列a的生成函数。我们设当$i \geq i_0$时$a_i = \sum_{j=1}^{k} c_j a_{i-j}$，a和c的生成函数分别为A和C，那么我们有$A = AC + A_0$，$A_0$由a的前几项决定。
- 回到原题，我们要求的就是$\sum_{i=0}^{\infty} p_i$，由于p递推，我们求出最短递推式后求出如上所述的C和A_0，我们就可以求出p的生成函数$\frac{A_0}{1-C}$。$\sum_{i=0}^{\infty} p_i$就是把生成函数带入$x = 1$后的值，我们把分子分母都带入$x = 1$除一下即可。由于数据随机分母不为0。

Mergesort Strikes Back

```
def mergesort(a,l,r,h):
  b=[]
  if l<=r:
    if h<=1:
      for i=l..r
        b.append(a[i])
    else:
      m=(l+r)/2
      c=mergesort(a,l,m,h-1)
      d=mergesort(a,m+1,r,h-1)
      b=merge(c,d)
  return b
```

```
def merge(a,b):
  c=[]
  while a.size() and b.size():
    if a[0]<b[0]:
      c.append(a[0])
      a.pop_front()
    else:
      c.append(b[0])
      b.pop_front()
  return c+a+b
```

- 输入n, k，输出对一个$1,2 \dots n$的随机排列$a_1, a_2 \dots a_n$调用$mergesort(a, 1, n, k)$后a的期望逆序对个数，对输入的大质数q取模输出。
- $1 \le n, k \le 10^5$，$10^8 \le q \le 10^9$，q为质数。

* Source: Avito Cool Challenge 2018

Mergesort Strikes Back

- 先考虑这个merge在干啥。把序列分成一段一段的，每个段的开头都是前缀max，例如1 2 4 3 5就分成1|2|4 3|5这样。那么merge的时候我们就是要把两个序列的段按照开头排序，然后接在一起。这是因为我们一旦放过了一个段的开头，就会把段内剩下的元素也输出去，因为它小于段的开头。
- 接下来回到这个mergesort，我们发现这个mergesort就是把序列隔成了若干个区间（不再递归的区间）。长度为l的区间内部期望就有$\frac{l(l-1)}{4}$个逆序对，块内的元素merge的时候相对位置不变，所以块内的逆序对就只有这些。

Mergesort Strikes Back

- 接下来考虑来自不同区间的逆序对。假设两个区间长度为l_1和l_2，考虑第一个区间的第i个元素和第二个区间的第j个元素构成逆序对的概率。
- 考虑第一个区间中的前i个元素和第二个区间中的前j个元素，这两个元素所在的块开头就分别是这两个前缀的最大值。如果这$i+j$个元素的最大值是第一个区间中的前i个元素或第二个区间中的前j个元素，那么无论如何都构不成逆序对。否则，考虑这个最大值开头的块，它一定会排在第一个区间的这个前缀后面。那么我们交换第一个区间中的前i个元素和第二个区间中的前j个元素，原来有逆序对的现在就顺序了，否则现在就逆序了。所以，它们构成逆序对的概率就是$\frac{i+j-2}{2(i+j)} = \frac{1}{2} - \frac{1}{\mathrm{i+j}}$。

Mergesort Strikes Back

- 那么如果给定两个区间的长度a和b，期望形成的逆序对个数就是$\sum_{i=1}^{a}\sum_{j=1}^{b}(\frac{1}{2}-\frac{1}{i+j})$，我们枚举i，预处理倒数的前缀和，就可以$O(a)$地求出这个值。
- 现在区间的个数可能有$O(n)$个，但注意到区间的种数并不多。事实上我们容易证明区间的长度只有两种，所以我们就可以$O(n)$求出答案。
- 证明：
  - 对k归纳。假设深度为k-1时只有长度为a和a+1两种线段。
  - 如果a是偶数，设a=2t，深度为k时就只有t和t+1两种线段。
  - 如果a是奇数，设a=2t+1，深度为k时还是只有t和t+1两种线段。

Strongly Connected Tournament

- 有n个人在比赛。比赛规则如下：
  - 一开始每两个人打一局。
  - 建出一个有向图。对于每一局，假设a赢了b，就把a向b连边。
  - 把给定的有向图缩强连通分量。每个强连通分量内的人继续递归用这个过程比赛。
- 对于两个编号为i的人和编号为j的人，不妨设i<j，那么i打赢j的概率为 $p = \frac{a}{b}$。输出n, a, b，输出期望要打几局mod 998244353。
- $2 \leq n \leq 2000$，$1 \leq a < b \leq 100$。

* Source: Hello 2018

Strongly Connected Tournament

- 记ans[t]为n=t时的答案，考虑枚举最菜的强连通分量，那么这个强连通分量本身必须连通，并且这些人被其他人吊着打，那么假设它的大小为s，我们就已经打了$s(t-s)+\frac{s(s-1)}{2}$局了，这s个人还要接着打，剩下的$t-s$个人还没考虑，所以我们有：

$$ans(t)=\sum_{s=1}^{t} strong(s)cp(t,s)\left(s(t-s)+\frac{s(s-1)}{2}+ans(s)+ans(t-s)\right)$$

- 其中strong(s)表示s个人形成的图强连通的方案数，cp(t,s)为t个人中有s个人被其他人吊着打的方案数。注意这里转移是成环的，当s=t时ans(s)会从自己转移，需要移项一下。

Strongly Connected Tournament

- 考虑如何求strong(s)。这个较为简单，我们只需要容斥掉存在一些被吊打的强连通网友的方案数即可。
- $strong(s) = 1 - \sum_{i=1}^{s-1} strong(i)cp(s,i)$。
- 接下来考虑如何求cp(t,s)，考虑编号为t的人，如果他属于败者组，他就要被其他的t-s个人吊着打，否则他就需要吊打前面的s个人。
- $cp(t,s) = cp(t-1,s)(1-p)^s + cp(t-1,s-1)p^{t-s}$。
- 复杂度$O(n^2)$。

Every Day is a Holiday

- Byteland中，一年有恰好d天。一年中的有些日子是假日，定义如下：
  - 皇帝出生的日子是假日
  - 如果一天的前一天和后一天都是假日，那么它也是假日
- 一开始没有假日，接下来会上任若干皇帝。假设每个皇帝的生日在一年中均匀随机，期望要多少任皇帝才能让一年中的每一天都是假日呢？
- 输出答案mod 998244353，$1 \leq d \leq 10^5$。

* Source: Project Euler 645，有加强

Every Day is a Holiday

- 让每一天都是假日就等价于没有连续的两个没有皇帝出生的日子。
- 考虑令 $D(n)$ 为有恰好n个不同生日的皇帝出生后每一天都是假日的概率。记相邻两个皇帝的间隔依次为$x_1, x_2 \dots x_n$，那么总方案数就是$x_1 + x_2 + \cdots + x_n = d$的方案数，即$C_{d-1}^{n-1}$，而合法的方案还需要满足$x_i \leq 2$，那么=2的就需要恰好有d-n个，所以方案数就是$C_n^{d-n}$。所以$D(n) = \frac{C_n^{d-n}}{C_{d-1}^{n-1}}$。
- 注意到已经出生了k个不同生日的皇帝后下一个期望要过$\frac{d}{d-k}$个才是不同生日的，我们枚举已经出生了几个不同生日的皇帝后仍未满足条件，我们就有答案为 $\sum_{i=0}^{d-1} \frac{d}{d-i}(1 - D(i))$，直接计算即可。

Frank

- 有一张n个点m条边的连通无向图，可能有重边和自环。
- 随机游走指的是从某个点开始，每次等概率随机找一条以该点为端点的边，走到该边的另一端点。
- 对每个$i \neq j$求出从i开始随机游走走到j需要的期望步数，保留8位小数。
- $3 \leq n \leq 400$，$1 \leq m \leq 4 \times 10^5$。

* Source: 2018 Nanjing Regional，有简化

感谢wxh同学提供题解

Frank

- 记$f(i,j)$表示从i走到j的期望步数，$g(i)$表示从i走一圈回到i的期望步数，$e(i,j)=\frac{1}{\deg(i)}$，那么我们有$f(i,j)=1+\sum_k e(i,k)f(k,j)-[i=j]g(i)$。
- 我们记I为单位矩阵，J为全1矩阵，G为$G_{i,i}=g_i$的矩阵，F为答案矩阵，P为转移矩阵（即$P_{i,j}=e(i,j)$），那么我们有$F=J-G+PF$，所以我们只需要解$(I-P)F=J-G$。

Frank

- 考虑如何求出g，我们引入稳定分布p，稳定分布满足$\sum_{i=1}^{n} p_i = 1$，$pP = p$，即若在某一时刻位置的概率分布为p，之后任意时刻概率分布仍为p。
- 可以证明$g(i) = \frac{1}{p(i)}$。
- 这是因为$G = PF + J - F$，所以$pG = pPF + pJ - pF$，所以$pG = pJ$。令$A = pG = pJ$，那么我们有 $A_i = p_i g_i = \sum_{j=1}^{n} p_j = 1$。

Frank

- 注意到$I-P$的秩为n-1，我们并没有办法直接求出$I-P$的逆矩阵，但是我们有$f(i,i)=0$，而去掉该条件所有的解相当于只是调整了$f(i,i)$，所以我们可以先令$Y_{n,i}=0$解出一个解Y，然后解出$X_{i,j}=Y_{i,j}-Y_{i,i}$。

Encrypted romance

- Alice和Bob选取了一个质数p和一个秘钥$k \in [0, p)$，他们用这个秘钥生成了n个子秘钥来加密文件，其中第i个子秘钥k_i满足$k_i = \left((a_i k + b_i) \bmod p\right) \bmod m_i$。
- Eve打算破解这个加密，但是Eve也很蠢，他的破解方法是随机一个$k' \in [0, p)$，然后同样生成n个子秘钥$k'_i = \left((a_i k' + b_i) \bmod p\right) \bmod m_i$尝试解密。Eve要求不高，他只希望解密出任意一个文件，即存在一个i使得$k_i = k'_i$。你想知道Eve有多少种可能的k'能达成计划，**使用科学计数法输出。**
- t组数据，每组数据输入p, k, n和三个长度为n的数组a_i, b_i, m_i。$1 \le n \le 100, 0 \le k, b_i < p, 1 \le a_i < p, 1 \le m_i \le p$，p为质数。
- 分为两个子任务。需要注意的是ipsc的赛制是提交答案题，比赛时间5h。
- 子任务1：$\mathrm{t} = 400$，$p \le 10^9$，$m_i \le 1000$，答案**相对误差**不超过0.1即视为正确。
- 子任务2：$\mathrm{t} = 100$，$p \le 10^{100}$，答案**相对误差**不超过0.01即视为正确。

* Source: IPSC 2018

Encrypted romance

- 子任务1：$\mathrm{t}=400$，$p \leq 10^9$，$m_i \leq 1000$，答案**相对误差**不超过0.1即视为正确。
- 相对误差不超过0.1听起来真牛逼，来写个随机撒点吧。
- 对于每组数据随机3×10^5个左右的k'并测试是否满足题意即可。
- 什么？你说你不信？

Encrypted romance

- 我们来讲讲道理。
- 首先我们对答案的大小进行一个估计。题意相当于是有n个哈希表，每个大小为m_i，选取了一些简单的哈希函数，用来存放$[0,p)$的正整数。问一个随机数与一个已知数在任意一个哈希表中碰撞的概率。
- 由于p是个质数，哈希表顶多也就能做到大约$1-\frac{1}{m_i}$的不碰撞率，所以答案至少约为$p-p(1-\frac{1}{m_1})$，因为$m_i \le 1000$，这个值至少是$\frac{p}{1000}$。

Encrypted romance

- 假设共有pt个合法的k'，我们每次就是在生成一个随机变量，它有t的概率为1，有$1-t$的概率为0。假设我们生成了m个随机变量$x_1, x_2 \dots x_m$，设$\sum_{i=1}^{n} x_i = X$，我们就把$\frac{X}{m}$当作t。设$\mu = mt$。
- 根据Multiplicative Chernoff Bound，我们有

$$P(X \geq (1+\epsilon)\mu) \leq e^{-\frac{\epsilon^2 \mu}{2}} \ (\epsilon \in [0,1])$$

- 所以我们有大约$2e^{-\frac{\epsilon^2 \mu}{2}}$的错误率。为了达到单组99.75%（即期望400次错一次）的正确率，我们需要有$\epsilon^2 tm = \epsilon^2 \mu \geq 20$。子任务1中$\epsilon = 0.1$，$t \geq \frac{1}{1000}$，所以取$m = 2 \times 10^5$即可。

* 可参见https://en.wikipedia.org/wiki/Chernoff_bound#Multiplicative_form_(relative_error)

Encrypted romance

- 子任务2：$\mathrm{t}=100$，$p \leq 10^{100}$，答案**相对误差**不超过0.01即视为正确。
- 首先写个高精度是免不了的，但是随机撒点似乎行不通了。仍然带入$\epsilon^2 tm = \epsilon^2 \mu \geq 20$，子任务2中$\epsilon = 0.01$，$t \geq \frac{1}{10^{100}}$，所以取$m = 2 \times 10^{104}$即可……这可不像五个小时内能跑得出来的样子。
- 我们考虑换一个思路。假设我们有一个$p \times n$的表格，第i行第j列为1当且仅当$k' = i$时$k_j = k'_j$，那么我们就是要问存在1的行的个数。

Encrypted romance

- 表格中总共的1的个数就是对于每个i，$\left((a_i k' + b_i) \bmod p\right) \bmod m_i = k_i$ 的解数之和。$s \bmod m_i = k$显然有$\lfloor\frac{p+m_i-k}{m_i}\rfloor$个$[0, p)$的解，而由于$a_i \in [1, p)$，$k'$与$s$一一对应。
- 现在我们会求表格中总共的1的个数了，要问存在1的行的个数。那么我们只需要乘上一个1是这行第一个1的概率就可以了！
- 我们知道了每个 i 的解数，如上抽样出表格中的一个1，求出它的所在行，即它所对应的k'，然后就可以判断它是不是这行第一个1了。

Encrypted romance

- 由于一共只有n列，一个1是这行第一个的概率至少是$\frac{1}{n}$。我们带入$\epsilon^2 tm = \epsilon^2 \mu \geq 20$，现在就有$\epsilon = 0.01$，$t \geq \frac{1}{n} \geq \frac{1}{100}$，所以取$m = 2 \times 10^6$即可。所以只要抽样$2 \times 10^6$次就可以通过了。

讲完了

谢谢大家