# Complete The Graph

- 有一张n个点m条边的简单无向图，每条边上有一个正整数边权，s号点到t号点的最短路为L。
- 由于技术故障，有些边上的边权被擦去了。你需要还原被擦去的这些边权（重新填上一个正整数边权），使得s到t的最短路仍为L。如有多解输出任意一个即可。
- $n, m \leq 500000$

* Source: Codeforces Round #372 Div1 B 加强

# Complete The Graph

- 先把所有边权未定的边的边权设为1，从t开始跑一遍dijkstra，设点x到点t的距离为$dist[x]$。
- 接下来从s开始跑一遍dijkstra，假设现在从u开始增广，遇到了一条边权未定的边u->v。由于是dijkstra，这时候我们已经知道了s到u的距离$du$，我们就把u->v这条边的边权设定为$\max(1, L - du - dist[v])$。如果最后跑出来s到t的距离是L那么显然就给出了一组可行解。我们断言其他情况下无解。

# Complete The Graph

- 考虑证明这个做法的正确性。考虑最后生成的图中每个点x到t的距离，这个距离显然是$\geq dist[x]$ 的，所以我们将边权定为$\max(1, L - du - dist[v])$ 会使得最后s到t的任意包含边权未定的边的路径长度都>=L。而如果存在s到t的不包含边权未定的边的长度<L的路径那么肯定无解。
- 考虑最后一次将边权赋为$L - du - dist[v]$的时候，我们就相当于硬点了一条经过这条边的路径，把它的长度设成L。这肯定是一条满足条件的最短路。
- 如果不存在这样的时候，就说明所有未定边权的边都肯定不在一条长度为L的最短路上，那么也不会对有解性造成影响。

# One Third

- 有一个圆比萨，你想要把它切成n块。每刀是从圆心到圆周的一条线段。由于你技术不好，你只会独立均匀地随机n个角度来切。切完之后你会取出圆上相邻的若干块吃掉。
- 设这个比萨的面积为1，你想要找到面积最接近$1/3$的这若干块，即设你取出的面积为x，你想要找到$|x-1/3|$最小的一种方案。求这个最小值的期望，对$10^9+7$取模。
- $2 \le n \le 10^6$

* Source: AGC032 F

# One Third

- 我们把切的刀标为红线段，把红线段顺时针旋转120°标为蓝线段，逆时针旋转120°标为绿线段，那么|两条红线段的角度差-120°|就相当于两条不同色线段的角度差。所以我们想要询问的就是不同色线段的角度差的期望值。
- 我们把切第一刀的位置看成0°，考虑[0°,120°]这个角度区间，那么每一刀三种颜色的线段恰会有一个落在此区间里，并且我们也只要考虑这个区间内不同色线段的贡献。（如果最小值由一个区间外的线段和一个区间内的线段贡献，那么考虑所夹的0°或120°这条线段，它必然可以贡献一个更小的答案）

# One Third

- 那么问题就转化为了，在一个数轴上，0处有一个红点，1/3处有一个蓝点，接下来重复n-1次，每次在[0,1/3]中随机选择一个位置，随机放置三种颜色之一的点，求不同色点距离最小值的期望。
- 操作结束后数轴上就有n+1个点，考虑相邻的两个点，如果它们的颜色不同就可以贡献答案。
- 考虑恰有k对相邻的点颜色不同的概率，我们首先枚举哪些点和前一个颜色不同，接下来dp这些点的颜色即可。这k段的长度和期望为$\frac{k}{3n}$，而k段长度和为1的线段最小值期望为$\frac{1}{k^2}$（$\int_0^{\frac{1}{k}} P(\min \geq x)dx = \int_0^{\frac{1}{k}}(1-kx)^{k-1}dx = \frac{1}{k}\int_0^1 x^{k-1}dx$），所以贡献期望就为$\frac{1}{3nk}$。

# 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: 集训队作业2019 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$\frac{1}{n-j}$，期望加上当前概率*$\frac{n}{n-j}$（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k-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$C_{i-jk}^{k-1}$。
- 复杂度$O(n^2k)$。

# 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。

# Traffic Blights

- 有一条街上有n个红绿灯，第i个红绿灯在位置$v_i$，有两个非负整数参数$r_i$和$g_i$，这个红绿灯从0时刻开始以$r_i + g_i$为周期亮灯，每个周期开始$r_i$秒是红的，后面$g_i$秒是绿的。
- 有一辆车从$[0,2019!]$的一个随机实数时刻从位置0出发，每秒会向右移动1的距离，碰到第一个红绿灯就停下，你需要输出它在每一个红绿灯处停下的概率，保留6位小数。
- $1 \le n \le 500$，$1 \le v_i \le 10^5$，$v_1 < v_2 < \cdots < v_n$，$1 \le r_i + g_i \le 100$

* Source: ICPC World Final 2019

# Traffic Blights

- 设通过前i个红绿灯的概率为$s_i$，那么在第i个红绿灯处停下的概率就是$s_{i-1} - s_i$。考虑如何求出s。
- 设出发时刻下取整的值为t，那么它到第i个红绿灯的时间下取整就是$t + v_i$，如果$(t + v_i) \bmod (r_i + g_i) < r_i$灯就是红的，否则是绿的。
- 设$p_i = r_i + g_i$为第i个红绿灯的周期，那么整个红绿灯系统就以$P = \mathrm{lcm}(p_1, p_2 \cdots p_n)$为周期循环。由于$p_i \in [1,100]$，我们有$P | 2019!$，所以t相当于是在mod P下的剩余系随机的。

# Traffic Blights

- 不妨先考虑一些简单的情形。如果$p_i$均为质数，那么由中国剩余定理，对于每个红绿灯，通过前i个红绿灯的概率对于每个质数是独立的。我们可以对于每个质数$p_0$，对于每个前缀求出能通过前i个红绿灯中$p = p_0$的红绿灯的概率，即要求合法的$t \bmod p_0$个数，容易归纳出这是一个模$p_0$意义下的区间，直接维护即可。求出通过每种$p$的灯的概率之后乘起来就行了。

# Traffic Blights

- 对于一般的情形，考虑把它转化到这种情况。我们选择一个常数X，使得对于每个$p_i$，$\frac{p_i}{\gcd(X,p_i)}$都为1或者质数。我们枚举答案mod X的余数a，即设$t = a + kX$，那么$(a + kX) \bmod p_i$的值就只由k mod $\frac{p_i}{\gcd(X,p_i)}$决定。由于$\frac{p_i}{\gcd(X,p_i)}$全是质数或者1，我们和简化版的问题一样，求出通过每个不同$\frac{p_i}{\gcd(X,p_i)}$的概率相乘即可。
- 人工讨论或枚举知最小的合法X为$2^3 \times 3^2 \times 5 \times 7 = 2520$，时间复杂度$O(Xnp)$。

# Mouse

- 交互库有一个$1,2\cdots n$ 的排列$p_1,p_2\cdots p_n$ ，你每次可以给出一个$1,2\cdots n$的排列$q_1,q_2\cdots q_n$ ，交互库会给出相同的位置个数（即满足$p_i=q_i$的$i$的个数），你需要还原$p_1,p_2\cdots p_n$。
- $n\leq 256$，你可以使用不超过2400次询问。

* Source: Info(1) Cup 2019

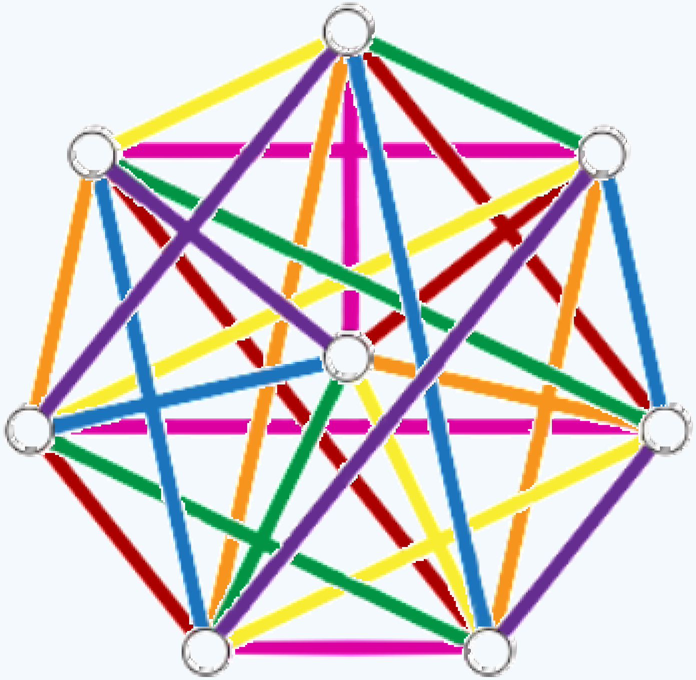
# Mouse

- 假设n>1。我们首先进行随机若干个排列进行询问，直到找到一个排列q和p的各位均不相同（即返回结果为0）。期望需要随机的次数是$\frac{n!}{1..n\text{的错排个数}}$，趋近于$e$。
- 如果$p_i = q_j$，我们把i和j连一条有向边，那么我们就得到了一张由若干个有向环组成的有向图（每个点出入度恰好为1）。
- 考虑交换q中的两个元素$q_i$和$q_j$，如果得到的排列与q有相同元素，那么说明i和j在这张有向图中有至少一条边。假设我们对于每对$i < j$都进行了询问，那么我们就可以还原出这张有向图去掉方向和重边的结果。

# Mouse

- 对于大小为2的联通块，显然原来就是一个大小为2的环。对于大小大于2的联通块，我们可以枚举这个环的方向并询问确认。这样我们就能还原出这张有向图，从而得到排列p。
- 设$f(i,j)$为交换$q_i$和$q_j$的询问结果，我们就是要找到所有满足$f(i,j) > 0$的$(i,j)$。这样的$(i,j)$对数不超过n。注意到如果$(i_1,j_1),(i_2,j_2)\cdots(i_t,j_t)$这$2t$个数互不相同，我们事实上可以把每一对q的对应位置进行交换后一起询问，从而求出$f(i_1,j_1)+f(i_2,j_2)+\cdots f(i_t,j_t)$的值。
- 接下来问题就转化成了把所有$i<j$划分成若干个集合，每个集合两两不交。当n为偶数时，我们有以下的经典构造：

# Mouse



# Mouse

- 当n为奇数时，我们只需将n加一，并忽略与n有关的询问。
- 划分出$O(n)$个集合之后，对于每个f值之和非零的集合，我们可以每次二分出最长的f值之和非零的前缀，这个前缀的终止点就是一对(i,j)，删去这个前缀重复这个过程即可。询问次数$O(n\log(n))$。

# 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。
- $2 \le n \le 2000$，$1 \le a < b \le 100$。

* 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

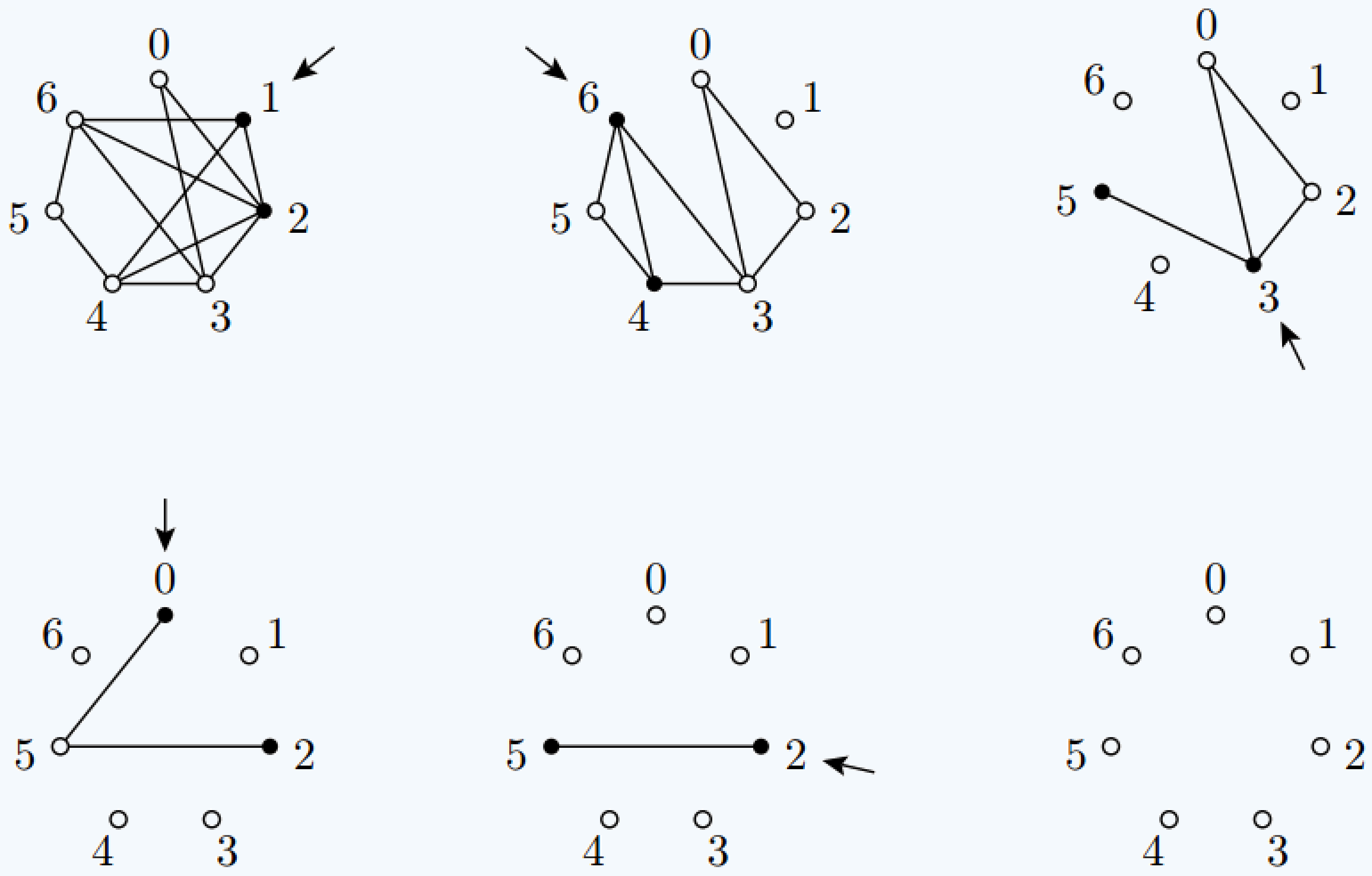
- 考虑如何求strong(s)。这个较为简单，我们只需要容斥掉存在一些被吊打的强连通网友的方案数即可。
- $strong(s) = 1 - \sum_{i=1}^{s-1} strong(i) cp(s,i)$。
- 接下来考虑如何求cp(t,s)，考虑编号为t的人，如果他属于败者组，他就要被其他的t-s个人吊着打，否则他就需要吊打前面的s个人。
- $cp(t,s) = cp(t-1,s)(1-p)^s + cp(t-1,s-1)p^{t-s}$。
- 复杂度$O(n^2)$。

# Keep clicking, keep flipping

- 我们在一个n个点的无向图上玩一个名叫翻转的游戏。每个点有一个黑或白的颜色，两点之间可能有边或没有边。
- 每次你可以选定一个黑点，并把它和与它有连边的所有点颜色反转。接下来我们考虑这个点和与它有连边的所有点形成的子图，把这个子图内的边翻转（原来有边的两个点现在没边，否则现在有边）。
- 游戏的目标是使得点两两不相邻且均为白色。输出一种操作序列或输出无解。
- $n \leq 600$，1s。

* Source: IPSC2011

# Keep clicking, keep flipping



# Keep clicking, keep flipping

- 如果有一个至少两个点的全是白点组成的联通块，那么一定无解，因为这些点无论如何也无法进行消除。我们称这样的联通块为坏联通块，那么我们就要尽量避免产生坏联通块。
- 事实上我们可以证明没有坏联通块时一定有解。考虑以下这个贪心策略：我们每次选择能使操作后黑点个数最多的黑点。我们只需要证明它不会产生坏联通块，我们就可以对单点的白联通块施归纳来证明。

# Keep clicking, keep flipping

- 一个点x对黑点个数的贡献就是x周围的白点个数-x周围的黑点个数-1，我们记x的价值为x周围的白点个数-x周围的黑点个数。
- 考虑反证法，如果u是一个价值最大的点，而删去它产生了一个坏联通块。由于原图没有坏联通块，坏联通块中一定有一个与u相邻的黑点v。
- 考虑u的白色邻接点，这些点必须也是v的白色邻接点，否则这些点就会变成黑色并与v相连。类似地，考虑v的黑色邻接点，这些点必须也是u的黑色邻接点，否则它们仍然是v的黑色邻接点。

# Keep clicking, keep flipping

- 那么我们就有v的白邻居个数>=u的白邻居个数，v的黑邻居个数<=u的黑邻居个数，所以v的分数不低于u，那么只能等于u的分数，此时我们就有u,v的邻居完全相同，那么v就会成为一个孤立的白点，矛盾。
- 直接模拟这个过程，复杂度$O(n^3)$，也可以手写bitset优化到$O(\frac{n^3}{w})$。

# 转盘

- 有n个随机数，第i个在$[0,t_i]$中独立均匀随机，求这些实数之和的m次方的期望。
- $1 \le n,m \le 10^5$，2s。

* Source: 原创

# 转盘

- 设第i次随机出了$x_i$，那么我们就要求$E((\sum_i x_i)^m)$。
- 首先我们注意到$x_i^m$的期望为$\frac{r_i^m}{m+1}$，这是因为$\int_0^1 x^m\,dx = \frac{1}{m+1}$。
- 考虑$E\left((\sum_{i=1}^n x_i)^m\right) = \sum_{i=0}^m C_m^i E\left(x_n^i\right)E\left(\left(\sum_{i=1}^{n-1} x_i\right)^{m-i}\right)$，这就是一个指数生成函数的形式，我们设$F_t(x) = \sum_{i=0}^m E\left(x_t^i\right)\frac{x^i}{i!} = \sum_{i=0}^m \frac{r_t^i x^i}{(i+1)!}$，答案就是$m!\,[x^m]\prod_{i=1}^n F_i(x)$。
- 设$F(x) = \sum_{i=0}^m \frac{x^i}{(i+1)!}$，那么$F_t(x) = F(r_t x)$，$\prod_{i=1}^n F_i(x) = \exp\left(\sum_{i=1}^n \ln\left(F_i(x)\right)\right)$。

# 转盘

- 我们求出$\ln(F(x)) = \sum_{i=0}^{m} p_i x^i$，那么$\sum_{i=1}^{n} \ln(F_i(x)) = \sum_{i=1}^{n} \sum_{j=0}^{m} p_j (xr_i)^j = \sum_{j=0}^{m} p_j x^j (\sum_{i=1}^{n} r_i^j)$
- 那么我们就只需要对于每个$j \in [0, m]$求出$\sum_{i=1}^{n} r_i^j$即可。
- 一种简单易懂的求法是注意到$\frac{1}{1-xr_i} = \sum_{j=0}^{\infty} x^j r_i^j$，所以我们只需要求出$\sum_{i=1}^{n} \frac{1}{1-xr_i}$。
- 考虑进行暴力通分（即$\frac{a}{b} + \frac{c}{d} = \frac{ad+bc}{bd}$）。直接一项一项通分肯定复杂度不对，但是我们可以分治进行通分，复杂度就对了。算出来整个的结果之后算出分子*分母的逆元即可。

# 谢谢大家

祝大家省选顺利