目录

0x00 概率

0x01 基本概念

~~大量小学数学概念来袭~~

随机试验： 可在相同条件下重复进行，每次试验的结果可以不止一个且能事先明确所有结果，进行一次试验前并不能确定哪一个结果出现的试验

样本空间： 记作 S，某个随机试验所有可能的结果的集合，其元素即为试验的每个结果（样本点）

基本事件： 由一个样本点组成的单个元素的集合

和事件： 记作 $A\bigcup B$ 或 $A+B$，当且仅当事件 A 和事件 B 至少一个发生时，事件 $A\bigcup B$ 发生

积事件： 记作 $A\bigcap B$ 或 AB，当且仅当事件 A 和事件 B 同时发生时，事件 $A\bigcap B$ 发生

互斥事件： 记作 $A\bigcap B=\varnothing$，事件 A 和事件 B 的交集为空，即不能同时发生

对立事件： $A\bigcup B=S$ 且 $A\bigcap B=\varnothing$，整个样本空间仅有事件 A 和事件 B ，即每次实验必有一个且仅有一个发生

频率： 相同条件下进行 n 次试验，这 n 次试验中，事件 A 发生了 a 次，$\frac{a}{n}$ 即为事件 A 发生的频率

概率： 在大量重复进行同一试验时，试验 A 发生的频率总是在某种意义下接近某个常数，并在他附近摆动，该常数即为事件 A 的概率 $P(A)$

概率的性质：

非负性： 对于任意一个事件 A，$0\leq P(A)\leq 1$

规范性： 对于必然事件 A，$P(A)=1$，对于不可能事件 A，$P(A)=0$

互斥事件可加性： 对于 n 个 **互斥** 的事件，$P(A_1\bigcup A_2\bigcup\ldots\bigcup A_n)=P(A_1)+P(A_2)+\ldots+P(A_n)$

独立事件可乘性： 对于 n 个**对立**的事件：$P(A_1\bigcap A_2\bigcap\ldots\bigcap A_n)=P(A_1)\times P(A_2)\times\cdots\times P(A_n)$

n 重伯努利试验（重复 n 次）： 一次试验中某个事件发生的概率是 p，那么重复 n 次独立试验中这个事件恰好发生 k 次的概率为 $P_n(k)=C_n^k\times p^k\times(1-p)^{n-k}$

0x02 古典概率

古典概率（事前概率）指：随机事件中各种可能发生的结果、出现的次数都可以由演绎、外推法得知，无需经过任何统计实验即可计算各种结果发生的概率。

其有以下几个特点：

- 试验的样本空间**有限**
- 试验中每个结果出现的**可能性相同**
- 试验中所能发生的事件**互斥**

在计算古典概率时，如果在全部可能出现的基本事件范围内构成事件 A 的基本事件有 a 个，不构成事件 A 的事件有 b 个，则出现事件 A 的概率为：$P(A)=\frac{a}{a+b}$

0x03 条件概率

条件概率

举一个形象的例子：让你猜一个人是男是女，直接猜的是女的成功的概率是 50%，那么如果告诉你这个人是长头发，那么猜他是女的概率就大幅提高了，也就是说在长头发这个事件发生的前提下，事件A为这个人是女生的概率也会发生变化。即条件概率的意义为，**当给定条件发生变化后，会导致事件发生的可能性发生变化。**

条件概率定义：事件 A 在 **另外一个事件 B 已经发生条件下** 的发生的概率。条件概率表示为：$P(A|B)$，读作 “在B的条件下A的概率” 。条件概率可以用决策树进行计算。两个事件 A、B 同时发生的概率为 $P(AB)$，则有：

$$P(A|B) = \frac{P(AB)}{P(B)}$$

对于两个**独立事件** A、B 来说（事件A，B互斥），两个事件 A、B 同时发生的概率为 ：

$$P(AB) = P(A) \times P(B)$$

0x03.1 乘法公式

由条件概率推广得到乘法公式：

乘法公式：$P(AB) = P(A|B)P(B) = P(B|A)P(A)$

乘法公式推广得到：

对于任何正整数 $n \geq 2$，当$P(A_1A_2 \ldots A_{n-1}) > 0$ 时，有：

$$P(A_1A_2 \ldots A_{n-1}A_n) = P(A_1)P(A_2|A_1)P(A_3|A_1A_2) \ldots P(A_n|A_1A_2 \ldots A_{n-1})$$

0x03.2 全概率公式

如果事件组 $B_1, B_2, \ldots . B_n$ 满足

- $B_1, B_2, \ldots . B_n$ 两两互斥，即 $B_i \cap B_j = \emptyset$ ，$i \neq j$ ， $i, j = 1, 2, \ldots .$ 且 $P(B_i) > 0, i = 1, 2, \ldots .$
- $B_1 \cup B_2 \cup \ldots . = \Omega$

则称事件组 $B_1, B_2, \ldots . B_n$ 是样本空间 Ω 的一个划分。

设 $B_1, B_2, \ldots . B_n$ 是样本空间 Ω 的一个划分，A为任一事件，则：

$$P(A) = \sum_{i=1}^{\infty} P(B_i) \times P(A|B_i)$$

上式即为全概率公式（formula of total probability)

全概率公式的意义在于，当直接计算 $P(A)$ 较为困难,而 $P(B_i), P(A|B_i)(i = 1, 2, \ldots)$ 的计算较为简单时，可以利用全概率公式计算 $P(A)$ 。其思想就是，将事件 A 分解成几个小事件，通过求小事件的概率，然后相加从而求得事件 A 的概率，而将事件 A 进行分割的时候，不是直接对 A 进行分割，而是先找到样本空间 Ω 的一个个划分 $B_1, B_2, \ldots . B_n$，这样事件A就被事件 $AB_1, AB_2, \ldots AB_n$ 分解成了 n 个互斥的部分，即 $A = AB_1 + AB_2 + \ldots + AB_n$ ， 每一个 B_i 发生都可能导致 A 发生相应的概率是 $P(A|B_i)$ ，由加法公式得

$$
\begin{aligned}
P(A) &= P(AB_1) + P(AB_2) + \ldots + P(AB_n) \\
&= P(A|B_1)P(B_1) + P(A|B_2)P(B_2) + \ldots + P(A|B_n)P(PB_n) \\
&= \sum_{i=1}^{\infty} P(B_i) \times P(A|B_i)
\end{aligned} \tag{1}
$$

Example

某车间用甲、乙、丙三台机床进行生产，各台机床次品率分别为5%，4%，2%，它们各自的产品分别占总量的25%，35%，40%，将它们的产品混在一起，求任取一个产品是次品的概率。

Solution

$$
P(A) = 25\% \times 5\% + 4\% \times 35\% + 2\% \times 40\% = 3.45\% \tag{2}
$$

0x03.3 贝叶斯公式

与全概率公式解决的问题相反，贝叶斯公式是建立在条件概率的基础上寻找事件发生的原因（即大事件 A 已经发生的条件下，分割中的小事件 B_i 的概率），设 $B_1, B_2, \ldots . B_n$ 是样本空间 Ω 的一个划分，则对任一事件A（$P(A) > 0$),有

$$
P(B_i|A) = \frac{P(B_i) \times P(A|B_i)}{\sum_{j=1} P(B_j) \times P(A|B_j)} \tag{3}
$$

上式即为贝叶斯公式（Bayes formula），B_i 常被视为导致试验结果 A 发生的”原因“，$P(B_i)(i = 1, 2, \ldots)$ 表示各种原因发生的可能性大小，故称先验概率；$P(B_i|A)(i = 1, 2\ldots)$ 则反映当试验产生了结果 A 之后，再对各种原因概率的新认识，故称后验概率。

也就是用来计算在 A 事件发生的条件下发生 B_i 事件的概率。

Example

发报台分别以概率0.6和0.4发出信号“∪”和“—”。由于通信系统受到干扰，当发出信号“∪”时，收报台分别以概率0.8和0.2受到信号“∪”和“—”；又当发出信号“—”时，收报台分别以概率0.9和0.1收到信号“—”和“∪”。求当收报台收到信号“∪”时，发报台确系发出“∪”的概率。

Solution

$$
P(B_1|A) = \frac{0.6 \times 0.8}{0.6 \times 0.8 + 0.4 \times 0.1} = 0.923
$$

上述内容摘自：全概率公式、贝叶斯公式推导过程

全概率公式和贝叶斯公式更多性质、证明详见：浅谈全概率公式和贝叶斯公式

1.两点分布

即只先进行一次伯努利试验，该事件发生的概率为 p，不发生的概率为 $1-p$。这是一个最简单的分布，任何一个只有两种结果的随机现象都服从 $0-1$分布。

期望：$E = p$
方差: $D = p * (1-p)^2 + (1-p) * (0-p)^2 = p * (1-p)$

2.二项分布

做 n 次伯努利实验，有 k 次实验成功

$$
P\{X = k\} = C_n^k p^k (1-p)^{n-k} \tag{4}
$$

期望：$E = np$
方差：$D = np(1-p)$

3.几何分布

在 n 次伯努利实验中，第 k 次实验才得到第一次成功的概率分布，可以理解为前 $k-1$ 次皆失败，第 k 次成功的概率。
几何分布是帕斯卡分布当 $r=1$ 时的特例。其中：$P(k)=(1-p)^{k-1}*p$

期望：$E=\frac{1}{p}$
方差：$D=\frac{1-p}{p^2}$

4.泊松分布 $(x\sim p(\lambda))$

描述单位时间/面积内，随机事件发生的次数。

$$P(X=k)=\frac{\lambda^k}{k!}e^{-\lambda},k=0,1,\cdots \tag{5}$$

期望：$E=\lambda$
方差：$E=\lambda$

5.均匀分布 $(x\sim u(a,b))$

对于随机变量 x 的概率密度函数：

$$f(x)=\begin{cases}\frac{1}{b-a} & \text{a<x<b}\\ 0 & \text{otherwise}\end{cases} \tag{6}$$

则称随机变量 X 服从区间 $[a,b]$ 上的均匀分布。

期望：$E=0.5(a+b)$
方差：$D=\frac{(b-a)^2}{12}$

6.指数分布 $X\sim E(\lambda)$

期望：$E=\frac{1}{\lambda}$
方差：$D=\frac{1}{\lambda^2}$

7.正态分布 $(X\sim N(\mu,\ \sigma^2))$

期望：$E=\mu$
方差：$D=\sigma^2$

0x10 期望

在概率论和统计学中，一个离散型随机变量的数学期望是试验中每次可能结果的概率乘以其结果的总和。

我们一般讨论的均为离散型随机变量。

如果 X 是一个离散的随机变量，输出值是 $x_1,x_2,\ldots,x_n$，输出值对应的概率是 $p_1,p_2,\ldots,p_n$（概率和为 1 ），那么期望值为：$E(X)=\sum_i x_ip_i$

Example

投掷一枚骰子，X表示掷出的点数，$P(X=1),P(X=2),\ldots P(X=6)$ 均为 $\frac{1}{6}$，则：

$$E(X)=1\times\frac{1}{6}+2\times\frac{1}{6}+\cdots+6\times\frac{1}{6}=\frac{1+2+3+4+5+6}{6}=3.5 \tag{7}$$

0x11 期望的线性性质

期望是“随机变量的期望”。

随机变量是定义在概率空间上的函数。随机试验的结果不同，随机变量的取值不同。

不同的基本结果可能导致随机变量取到相同的数值。

- 对于任意随机变量 X 和 Y 以及常量 a 和 b，有

$$E(aX+bY)=aE(X)+bE(Y)$$

- 当两个随机变量 X 和 Y 独立且各自都有一个已定义的期望时，有

$$E(XY)=E(X)E(Y)$$

条件期望

$p_{ij}=P(X=x_i,Y=y_i)(i,j,=1,2,3,\cdots)$

当 $X=x_i$ 时，随机变量 Y 的条件数学期望以 $E(Y|X=x_i)$ 表示。

全期望公式

$$E(Y)=E(E(Y|X))=\sum_i P(X=x_i)E(Y|X=x_i) \tag{8}$$

Example

一项工作由甲一个人完成，平均需要 4 小时，而乙有0.4的概率来帮
忙，两个人完成平均只需要3小时。若用X表示完成这项工作的人数，而Y表
示完成的这项工作的期望时间（单位小时），由于这项工作要么由一个人完成， 要么由两个人完成，那么这项工作完成的期望时间：

$$E(Y)=P(X=1)E(Y|X=1)+P(X=2)E(Y|X=2)=(1-0.4)\times4-0.4\times3=3.6$$

下面的内容摘自 概率与期望总结:

1.推式子.
这个方法一般是用到了等比数列求和，极限等思想来解决问题。
2.递推或动态规划.
这是求解概率，期望问题的最常见的套路。其重要的是确定好思考的方向，不要将每个状态独立起来，而考虑对整体的影响。不然对于一些东西，求解并不方便。更要考虑优化动态规划，来达到更优的复杂度。
3.迭代.
动态规划要求问题无后效性， 而如果问题有不可避免的后效性， 动态规划就无能为力了。 这时我们可以采用迭代的方法来进行计算。在题目中没有出现极大或极小的概率且收敛较快时，可以使用这个方法，对于一些题目可以做到较优秀的复杂度。对于可能出现无限（也就是环）情况的，迭代至达到所求解的精度为止。此外， 迭代法也未必是要解决有后效性的问题， 只要问题有收敛性， 迭代都可以起到一定的作用。
4.高斯消元.
对于出现无限（环）的情况时，且精度要求较高，可以考虑列出期望-概率系统（概率—期望系统是一个带权的有向图，可以存在环），运用高斯消元来求解。但是对于环之间满足一个偏序时，可以用等比数列求和来求解，得到更优秀的复杂度。

问题分析:
对于一般的有限状态的问题，可以通过一般的递推，动态规划来求解。如果单纯的动态规划复杂度太高，且收敛较快，可以尝试使用迭代+动态规划来求解。
对于出现环的题目，尝试对问题建图，运用高斯消元来求解。
当然，如果概率比较难求解时，不妨用期望来间接求解。
P(x)=E(x)/E(all).

对于期望DP一般是逆推，记忆化搜索的写法可以很清楚的明白为什么。而概率DP一般是正着推。

0x12 利用递推或动态规划解决

~~毕竟有一种DP叫做期望DP~~

对于一类期望问题，我们不需要将所有的可能情况一一列举出来计算，而是可以根据已经求出的期望退出其他状态的期望，或者根据一些特点与待求结果相同的情况，求出其概率。

A. P4206 [NOI2005] 聪聪与可可

Weblink

https://www.luogu.com.cn/problem/P4206

Problem

在一个魔法森林里，住着一只聪明的小猫聪聪和一只可爱的小老鼠可可。

整个森林可以认为是一个无向图，图中有 N 个美丽的景点，景点从 1 至 N 编号。在景点之间有一些路连接。
可可正在景点 $M(M \leq N)$ 处。以后的每个时间单位，可可都会选择去相邻的景点（可能有多个）中的一个或停留在原景点不动。而去这些地方所发生的概率是相等的。
聪聪是很聪明的，所以，当她在景点 C 时，她会选一个更靠近可可的景点，
如果这样的景点有多个，她会选一个标号最小的景点。如果走完第一步以后仍然没吃到可可，她还可以在本段时间内再向可可走近一步。
在每个时间单位，假设聪聪先走，可可后走。在某一时刻，若聪聪和可可位于同一个景点，则可可就被吃掉了。
问平均情况下，聪聪几步就可能吃到可可。

Solution

根据题目要求聪聪会向可可不断靠近，且边无权，所以先设边权为 1 ，对于每一个点进行一次 SPFA ，预处理出 $p[i,j]$ ，表示顶点 i 到顶点 j 的最短路上与顶点 i 相邻且编号最小的顶点编号。即聪聪在景点 i，可可在景点 j 时，聪聪第 1 步会走到的景点编号。

设 $f[i,j]$ 来表示聪聪在顶点 i，可可在顶点 j 时聪聪抓住可可的平均步数，即期望步数。令 $w[i,j]$ 表示与顶点 i 相邻的 j 个点编号，而用 $d[i]$ 表示顶点 i 的度数。

显然聪聪下一步所在的顶点即为 $p[p[i,j],j]$，可可下一步在顶点 $w[i,j]$ 的概率为 $\frac{1}{d[i]+1}$，下一步这个情况下的期望 $f[p[p[i,j],j],w[i,j]]$ 已经计算出，那么就是比 $f[p[p[i,j],j],w[i,j]]$ 多出一步。可可在原地停留的情况则类似。

有转移方程：

$$
\begin{aligned}
f[i,j] &= \frac{(\sum_{k=1}^{t[i]} f[p[p[i,j],j],w[j,k]]+1)+f[p[p[i,j],j],j]+1}{d[i]+1}+1 \\
&= \frac{(\sum_{k=1}^{t[i]} f[p[p[i,j],j],w[j,k]])+f[p[p[i,j],j],j]+d[i]+1}{d[i]+1} \\
&= \frac{(\sum_{k=1}^{t[i]} f[p[p[i,j],j],w[j,k]])+f[p[p[i,j],j],j]}{d[i]+1}+1
\end{aligned} \tag{9}
$$

- 若 $i==j$，则 $f[i,i]=0$ 。
- 若 $p[p[i,j],j]=j$ 或 $p[i,j]=j$ 则说明在这一步聪聪即可吃掉可可，那么 $f[i,j]=1$。

记忆化搜索即可。

Code

```
1 // Problem: P4206 [NOI2005] 聪聪与可可
```

```cpp
// Contest: Luogu
// URL: https://www.luogu.com.cn/problem/P4206
// Memory Limit: 125 MB
// Time Limit: 1000 ms
//
// Powered by CP Editor (https://cpeditor.org)

#include <bits/stdc++.h>

using namespace std;

const int N = 1e3 + 7, M = 2e3 + 7, INF = 0x3f3f3f3f;

int n, m, s, t;
int head[N], edge[M], ver[M], nex[M], tot;
bool vis[N];
int p[N][N];
int dist[N][N];
double f[N][N];
int d[N];

void add(int x, int y, int z)
{
    ver[tot] = y;
    edge[tot] = z;
    nex[tot] = head[x];
    head[x] = tot ++ ;
    d[x] ++ ;
}

void SPFA(int s)
{
    memset(dist, -1, sizeof dist);
    vis[s] = 1;
    dist[s][s] = 0;
    queue<int> q;
    q.push(s);
    while(q.size()) {
        int x = q.front();
        q.pop();
        vis[x] = 0;
        for(int i = head[x]; ~i; i = nex[i]) {
            int y = ver[i];
            int z = edge[i];
            if(dist[s][y] == -1 || (dist[s][y] == dist[s][x] + z && p[s][x] < p[s][y])) {
                dist[s][y] = dist[s][x] + z;
                if(p[s][x]) {
                    p[s][y] = p[s][x];
                }
                else p[s][y] = y;
                if(vis[y] == 0) {
                    vis[y] = 1;
                    q.push(y);
                }
            }
```

```
        }
    }
}

//记忆化搜索
double dfs(int x, int y)
{
    if(f[x][y]) {
        return f[x][y];
    }
    if(x == y){
        return f[x][y] = 0.0;
    }
    if(p[x][y] == y || p[p[x][y]][y] == y) {
        return f[x][y] = 1.0;
    }
    for(int i = head[y]; ~i; i = nex[i]) {
        int nex_node = ver[i];
        f[x][y] += dfs(p[p[x][y]][y], nex_node);
    }
    f[x][y] = (f[x][y] + dfs(p[p[x][y]][y], y)) / (d[y] + 1) + 1;
    return f[x][y];
}

double ans;

int main()
{
    memset(head, -1, sizeof head);
    scanf("%d%d", &n, &m);
    scanf("%d%d", &s, &t);
    for(int i = 1; i ≤ m; ++ i) {
        int x, y;
        scanf("%d%d", &x, &y);
        add(x, y, 1);
        add(y, x, 1);
    }
    for(int i = 1; i ≤ n; ++ i)
        SPFA(i);
    ans = dfs(s, t);
    printf("%.3f\n", ans);
    return 0;
}
```

B. P4745 [CERC2017]Gambling Guide（期望DP + 最短路实现）

Weblink

https://www.luogu.com.cn/problem/P4745

Problem

一个在邻国的铁路系统是由n个城市（编号从1到n），和m条连接两个不同城市的双向铁路组成的。铁路票只能在安装在每个城市的自动售票机购买。不幸的是，黑客们已经篡改了这些售票机，现在它们有下面的规则： 当a市的售票机有一个硬币投入时，机器会发一张从a市到随机一个邻市的单程票。更精确地来说，目的地城市是被统一的、随机的从所有由出发城市为起点的铁路的终点中选取的。

一个研究计算机科学的学生需要从城市1（她生活在那里）到城市n（那里正举行一个编程比赛）。她知道机器是怎么工作的（但当然她不能预测随机的目的地）并且有一份铁路系统的地图。在每一个城市，当她买了一张票时，她可以选择立即使用它后到达目的地，或者是丢掉它并买一张新票。她可以无限制的购买的票。当她一到达n城市，旅行就会结束。

在做了一些计算之后，她制定了一个拥有以下的目标的旅行计划：

- 旅行最终到达终点的概率为1
- 预期花在旅行上的硬币越少越好

找到这个预期的她要花在旅途上的硬币数

输入格式：

第一行包含两个整数n和m(1≤n,m≤300000)（城市的数量和铁道的数量）。

接下来m行每行包含了两个不同的整数a和b(1≤a,b≤n)，描述了一条连接a市和b市的双向铁路

两个城市之间最多只会有一条铁路，输入保证有一条从城市1到n的路径。

输出格式：

输出一个数——预期的要花在旅途上的硬币数。此输出只要与正解的相对差或绝对差小于10^−6就可以通过。

Solution

显然考虑期望 dp 。因为只有一个终点 n，且状态已知，考虑逆推。

设 $f[x]$ 表示点 x 到 n 的期望花费，或者可以说是从 n 到点 x 的期望花费。 $d[x]$ 表示 x 点的度数，则有

$$f[x]=\frac{\sum \min\{f[x],f[y]\}}{d[x]}+1$$

即，花费一枚硬币，选择原地不动，或者从 y 走到 x ，到达点 x 一共有 $d[x]$ 条路，即一共有 $d[x]$ 种情况，所以概率期望就是$\frac{1}{d[x]}$，花费的期望就为 总花费乘上 $\frac{1}{d[x]}$。

观察公式显然可以发现对于一个点 x ，所有与 x 相邻的点 y，能对 x 产生贡献，当且仅当 $f[y]<f[x]$ 。

假设对于点 x 而言，满足上述条件（$f[y]<f[x]$）的相邻点 y 点一共有 $cnt[x]$ 个，则有

$$\begin{aligned}f[x]&=\frac{(d[x]-cnt[x])\times f[x]+\sum f[y]+cnt[x]}{d[x]}\\&=\frac{\sum f[y]+d[x]}{cnt[x]}\end{aligned}$$

显然 $\sum f[y]$ 可以直接把所有小于 $f[x]$ 的都加起来，可以用最短路去实现，因为我们每次从队里拿出来的一定是最小的 $f[y]$，那么所有与他相邻的点都可以被他更新，我们就可以对于每一个 x 能到达的点 y ，求和，即 $sum[y]=\sum f[x]$。然后对于每一个点 y ，直接根据公式更新 $f[y]$ 即可。

更新的时候，是从 n 出发，走到 1 ，即每次从 x 走到 y ，那么公式中的 x 就是 y ，也就是用所有 的 x 去更新 y，每次更新的是 $f[y]$ 而不是公式里的 $f[x]$。

Code

```cpp
int n, m, t;
int head[N], ver[M], edge[M], tot, nex[N];
double d[N], cnt[N], sum[N];
double dist[N];
bool vis[N];

void add(itn x, int y)
{
    ver[tot] = y;
    nex[tot] = head[x];
    head[x] = tot ++ ;
}

void dijkstra()
{
    memset(dist, 0x3f, sizeof dist);
    dist[n] = 0;
    priority_queue<PDI, vector<PDI>, greater<PDI> > q;
    q.push({0.0, n});
    while(q.size()) {
        itn x = q.top().second;
        q.pop();
        if(vis[x]) continue;
        vis[x] = 1;
        for(int i = head[x]; ~i; i = nex[i]) {
            itn y = ver[i];
            if(vis[y]) continue;
            cnt[y] ++ ;
            sum[y] += dist[x];
            dist[y] = (d[y] + sum[y]) / cnt[y];
            q.push({dist[y], y});
        }
    }
}

int main()
{
    memset(head, -1, sizeof head);
    scanf("%d%d", &n, &m);
    for(int i = 1; i ≤ m; ++ i) {
        itn x, y;
        scanf("%d%d", &x, &y);
        add(x, y);
        add(y, x);
        d[x] ++ ;
        d[y] ++ ;
    }
    dijkstra();

    printf("%.8f\n", dist[1]);
    return 0;
}
```

0x13 建立线性方程组解决

待更...

参考资料：

- 《浅析竞赛中一类数学期望问题的解决方法》汤可因 IOI国家队2009论文
- 数学期望公式大全
- ACM 概率&期望