# 期望与概率

#### 1.利用期望概念求解

$$E(x) = \sum p_i x_i$$

一般当 $x_i$ 和 $p_i$ 都可以求解时可以这么做，实际上是在求概率.

[Game on Tree]

**题意：**给定一个有根树，每次删去一颗子树，问删完整棵树的期望值.

**做法：**设点 $f_i$ 为 $i$ 被操作的概率，最终期望 $E = \sum_{i=1}^{n} f_i$，$f_i$ 与 $i$ 的祖先有关，如果祖先在它之前没操作了，就已经被删除了，不能产生操作，所以我们需要保证 $i$ 被操作时祖先都没有被操作过，概率 $f_i = \dfrac{(dep_{fa})!}{(dep_i)!} = \dfrac{1}{dep_i}$（这么算是因为我们只关注 $dep_i$ 个点，其余的点不需要我们关心）.

```
#include<bits/stdc++.h>
using namespace std;
typedef double db;
const int maxn = 1e5+20;
int n;
db Ans;
vector<int> E[maxn];

inline int read(){
    int x = 0 , f = 1 ; char c = getchar() ;
    while( c < '0' || c > '9' ) { if( c == '-' ) f = -1 ; c = getchar() ; }
    while( c >= '0' && c <= '9' ) { x = x * 10 + c - '0' ; c = getchar() ; }
    return x * f ;
}

void add(int x,int y){
    E[x].push_back(y);E[y].push_back(x);
}

void dfs(int x,int fa,int dep){
    Ans += 1.0 / dep;
    for(int to : E[x]){
        if(to==fa) continue;
        dfs(to,x,dep+1);
    }
}

int main(){
    n = read();
    for(int i=1;i<n;i++) add(read(),read());
    dfs(1,0,1);printf("%.7f\n",Ans);return 0;
}
```

[CF453A Little Pony and Expected Maximum]

**题意：**有一个 $m$ 面的骰子，投掷 $n$ 次，求最大值的期望.

**做法：**枚举最大值为 $i$，其概率为 $\dfrac{i^n-(i-1)^n}{m^n}$.

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1e5+20;
int n,m;
double Ans,s[maxn];

inline double qu_pow(double base,int k){
    double An = 1.0;
    while(k){
        if(k&1) An *= base;
        base *= base ;k >>= 1;
    }
    return An;
}

int main(){
    scanf("%d%d",&m,&n);
    for(int i=1;i<=m;i++){
        s[i] = qu_pow(i/(double)m,n);
        Ans += i * (s[i] - s[i-1]);
    }
    printf("%.6f\n",Ans);return 0;
}
```

[CF1925D]

**做法：**考虑单独计算每对朋友，枚举一对朋友被选择了 $i$ 次，计算 $p_i$，可得答案为 $Ans=\sum_{i=1}^{m}\sum_{j=1}^{k}p_j*v_{i,j}$.

显然复杂度不允许暴力计算，考虑优化.

$$v_{i,j}=(2*f_i+j-1)*j\,/\,2$$
$$p_j=(M-1)^{k-j}*C(k,j)/M^k$$
$$p_j*v_{i,j}=(M-1)^{k-j}*C(k,j)*(2*f_i+j-1)*j\,/\,2\,/M^k$$

在这里可以看出一些眉目，这里只有 $f_i$ 是与 $i$ 有关的，所以剩下的可以预处理出来求和，可以省去第二层枚举，很容易算出答案.

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
const int maxn = 2e5+20;
const LL mod = 1e9+7;
LL n,k,m,M;
LL A[maxn],B[maxn],fact[maxn],inv[maxn];

inline int read(){
    int x = 0 , f = 1 ; char c = getchar() ;
    while( c < '0' || c > '9' ) { if( c == '-' ) f = -1 ; c = getchar() ; }
    while( c >= '0' && c <= '9' ) { x = x * 10 + c - '0' ; c = getchar() ; }
```

```
    return x * f ;
}

inline LL qu_pow(LL base,LL k){
    base %= mod;LL Ans = 1;
    while(k){
        if(k&1) Ans = Ans * base % mod;
        base = base * base % mod;k >>= 1;
    }
    return Ans;
}

inline LL C(int N,int c){
    return fact[N] * inv[c] % mod * inv[N-c] % mod;
}

void init(int N){
    for(int i=fact[0]=1;i<N;i++) fact[i] = fact[i-1] * i % mod;
    inv[N-1] = qu_pow(fact[N-1],mod-2);
    for(int i=N-2;i>=0;i--) inv[i] = inv[i+1] * (i+1) % mod;
}

int main(){
    init(maxn);
    int T = read();
    while(T--){
        n = read();m = read();k = read();M = n * (n - 1) / 2;
        for(int i=1;i<=k;i++){
            A[i] = (A[i-1] + qu_pow(M-1,k-i) * C(k,i) % mod * qu_pow(qu_pow(M,k),mod-2) % mod * i * 2 % mod) % mod;
            B[i] = (B[i-1] + qu_pow(M-1,k-i) * C(k,i) % mod * qu_pow(qu_pow(M,k),mod-2) % mod * (i-1) % mod * i % mod) % mod;
        }
        LL Ans = 0;
        for(int i=1;i<=m;i++){
            LL a = read(),b = read(),c = read();
            Ans = (Ans + A[k] * c % mod) % mod;
            Ans = (Ans + B[k]) % mod;
        }
        Ans = Ans * qu_pow(2,mod-2) % mod;
        printf("%lld\n",Ans);
    }
    return 0;
}
```

**2.顺序的期望DP**

大家都学过期望公式

$$E(X+Y)=E(X)+E(Y)$$

根据此写出状态转移方程进行DP.

[P1654 OSU!](P1654 OSU!)

$E(X)$ 表示 $x^3$ 的期望值，考虑由 $x^3 \to (x+1)^3$ 的过程.

$dp_i$ 表示到第 $i$ 个时 $x^3$ 的期望，$g_i$ 表示 $x^2$ 的期望，$f_i$ 表示 $x$ 的期望

$$f_i = (f_{i-1} + 1) * p$$
$$g_i = (g_{i-1} + 2 * f_{i-1} + 1) * p$$
$$(x+1)^3 = x^3 + 3x^2 + 3x + 1$$
$$dp_{i+1} = dp_i * (1-p) + (dp_i + 3 * f_i + 3 * g_i + 1) * p$$
$$dp_{i+1} = dp_i + (3 * f_i + 3 * g_i + 1) * p$$

```
#include<cstdio>
int n; double prob, d, x2, x;
int main() {
    scanf("%d", &n);
    while (n--) {
        scanf("%lf", &prob);
        d = d + prob * (3 * x2 + 3 * x + 1);
        x2 = prob * (x2 + 2 * x + 1);
        x = prob * (x + 1);
    }
    printf("%.1lf", d);
}
```

[CF235B Let's Play Osu!]

相同的题，甚至是弱化版.

> $(x+1)^2 = x^2 + 2x + 1$

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1e5+20;
int n;
double temp[maxn],dp[maxn];
int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        double p;
        scanf("%lf",&p);
        temp[i] = (temp[i-1]+1)*p;
        dp[i] = dp[i-1]+(2*temp[i-1]+1)*p;
    }
    printf("%.6f\n",dp[n]);
    return 0;
}
```

三倍经验

P1365 WJMZBMR打osu! / Easy

这里3个字符分别对应三个概率值

> $'x' == 1;' o' == 0;' ?' == 0.5$

```
#include<bits/stdc++.h>
using namespace std;
typedef long double ld;
const int maxn = 3*1e5+20;
int n;
double dp[maxn],temp[maxn];
int main(){
    scanf("%d",&n);
    for(int i=1;i<=n;i++){
        char t;double p = 1;cin >> t;
        if(t=='?') p = 0.5;
        else if(t=='x') p = 0;
        temp[i] = (temp[i-1]+1)*p;
        dp[i] = dp[i-1]+(2*temp[i-1]+1)*p;
    }
    printf("%.4lf",dp[n]);
    return 0;
}
```

P6835 [Cnoi2020]线形生物

**题意：**有 $n$ 个关卡，要么进入下一个关卡，或者回到之前某些特定的关卡，问期望的经过关卡次数是多少

**做法：**首先设 $E_{i,j}$ 表示从 $i$ 走到 $j$ 的期望次数，存在等式 $E_{i,j} = \sum_{k=i}^{j-1} E_{k,k+1}$。

设 $d_i$ 表示第 $i$ 关的出边，存在转移方程为

$E_{i,i+1} = \frac{1}{d} + \frac{1}{d}\sum_{to}(E_{to,i+1} + 1) = 1 + \frac{1}{d}\sum_{to} E_{to,i+1} = 1 + \frac{1}{d}\sum_{to}\sum_{k=to}^{i} E_{k,k+1}$.

设 $f_i = E_{i,i+1}$，移项得到 $\frac{1}{d}f_i = 1 + \frac{1}{d}\sum_{to}\sum_{k=to}^{i-1} f_k$，后面可以用到前缀和优化，整理得 $f_i = d + \sum_{to} s_{i-1} - s_{to}$，最终答案为 $s_n$.

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1e6+20;
const int mod = 998244353;
int n,m;
int du[maxn],f[maxn],s[maxn];
vector<int> edge[maxn];

inline int read(){
    int x = 0 , f = 1 ; char c = getchar() ;
    while( c < '0' || c > '9' ) { if( c == '-' ) f = -1 ; c = getchar() ; }
    while( c >= '0' && c <= '9' ) { x = x * 10 + c - '0' ; c = getchar() ; }
    return x * f ;
}

int main(){
```

```
    n = read();n = read();m = read();
    for(int i=1;i<=m;i++){
        int a = read(),b = read();
        edge[a].push_back(b);du[a]++;
    }
    for(int i=1;i<=n;i++){
        f[i] = du[i] + 1;
        for(int to : edge[i])
            f[i] = (f[i] + (s[i-1] - s[to-1] + mod) % mod) % mod;
        s[i] = (s[i-1] + f[i]) % mod;
    }
    printf("%d\n",s[n]);return 0;
}
```

一般顺序求概率DP

OIwiki

**3.逆序的期望DP**

**如果没有特殊说明均是完全随机**

这里强调逆序，至于为什么逆序来看这里.

CF1042E Vasya and Magic Matrix

**题意：**二维带权矩阵，给定起点，每次随机向权值小的位置移动，移动一次的贡献是欧几里得距离的平方，求期望得分.

**做法：**首先和位置没有关系，按照权值从小到大排序，考虑倒着推，$f_i$ 表示从 $i$ 开始移动的得分，很明显 $f_[1] = 0$.（$1$ 指最小权值）.

$f_{a_i} = \dfrac{1}{c}\sum\limits_{a_j<a_i}(f_{a_j} + (x_i - x_j)^2 + (y_i - y_j)^2)$，然后大力维护前缀和，$f_i, x_i^2, y_i^2, x_i, y_i$，然后计算答案即可.

```
#include<bits/stdc++.h>
#define pii pair<int,int>
#define pb push_back
#define mp make_pair
using namespace std;
const int mod = 998244353;
const int maxn = 1e6+20;
int n,m,N,X,Y;
int A[1020][1020],B[maxn],inv[maxn];
vector< pii > E[maxn];
struct Node{
    int f,x,y,x2,y2,c;
}S[maxn];

inline int read(){
    int x = 0 , f = 1 ; char c = getchar() ;
```

```
    while( c < '0' || c > '9' ) { if( c == '-' ) f = -1 ; c = getchar() ; }
    while( c >= '0' && c <= '9' ) { x = x * 10 + c - '0' ; c = getchar() ; }
    return x * f ;
}

inline int qu_pow(int base,int k){
    int An = 1;
    while(k){
        if(k&1) An = 1ll * An * base % mod;
        base = 1ll * base * base % mod;k >>= 1;
    }
    return An;
}

int main(){
    n = read();m = read();
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
            A[i][j] = B[++N] = read();
    X = read();Y = read();
    sort(B+1,B+1+N);N = unique(B+1,B+1+N) - B - 1;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++){
            A[i][j] = lower_bound(B+1,B+1+N,A[i][j]) - B;
            int t = A[i][j];E[t].pb(mp(i,j));S[t].c++;
            S[t].x += i;(S[t].x2 += i * i) %= mod;
            S[t].y += j;(S[t].y2 += j * j) %= mod;
            if(i==X&&j==Y&&t==1) return 0 * puts("0");
        }
    for(int i=1;i<=N;i++){
        S[i].c += S[i-1].c;(S[i].x += S[i-1].x) %= mod;(S[i].y += S[i-1].y) %= mod;
        (S[i].x2 += S[i-1].x2) %= mod;(S[i].y2 += S[i-1].y2) %= mod;
    }
    for(int i=2;i<=N;i++){
        int inv = qu_pow(S[i-1].c,mod-2);
        for(pii T : E[i]){
            int t = T.first * T.first + T.second * T.second;
            (t += 1ll * S[i-1].x2 * inv % mod) %= mod;
            (t += 1ll * S[i-1].y2 * inv % mod) %= mod;
            (t += 1ll * S[i-1].f * inv % mod) %= mod;
            (t += (mod - 1ll * S[i-1].x * 2 % mod * inv % mod * T.first % mod)) %= mod;
            (t += (mod - 1ll * S[i-1].y * 2 % mod * inv % mod * T.second % mod)) %= mod;
            (S[i].f += t) %= mod;
            if(T.first==X&&T.second==Y) return 0 * printf("%d\n",t);
        }
        (S[i].f += S[i-1].f) %= mod;
    }
}
```

P4550 收集邮票

**题意**：有 $n$ 种邮票，第 $k$ 次买需要付 $k$ 元，问卖完 $n$ 种的期望花费是多少

**做法**：设 $f_i$ 表示已经买了 $i$ 种，还需要买 $n-i$ 种的期望次数，有转移方程 $f_i=\frac{i}{n}f_i+\frac{n-i}{n}f_{i+1}+1$，化简得 $f_i=f_{i+1}+\frac{n}{n-i}$。

设 $g_i$ 表示已经买了 $i$ 种，还需要买 $n-i$ 种的期望花费，转移为 $g_i=\frac{i}{n}(g_i+f_i+1)+\frac{n-i}{n}(g_{i+1}+f_{i+1}+1)$，化简得到 $g_i=g_{i+1}+f_{i+1}+\frac{i*f(i)+n}{n-i}$。

初始值很好定义 $f_n=g_n=0$.

```
#include<bits/stdc++.h>
using namespace std;
typedef double db;
const int maxn = 10020;
int n;
db f[maxn],g[maxn];

int main(){
    scanf("%d",&n);
    for(int i=n-1;i>=0;i--){
        f[i] = f[i+1] + n * (1.0 / (n-i));
        g[i] = g[i+1] + f[i+1] + (i * f[i] + n) / (1.0 * (n-i));
    }
    printf("%.2f\n",g[0]);return 0;
}
```

P2473 [SCOI2008] 奖励关

**题意**：有 $k$ 次机会，有 $n$ 个物品，每次机会随机跳出一个物品，你可以选择是否获得该物品以及其分数，求最优策略下的分数.

**做法**：状压 + 期望 $DP$,设 $f[i][S]$ 表示前 $i-1$ 次机会拿取的物品集合为 $S$，第 $i$ 次到第 $k$ 次的得分为 $f$.

$$f_{i,S}=\max(f_{i+1,S},f_{i+1,(S|(1<<j))}+v_j)$$

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
typedef long double LD;
const int maxn = 102;
int n,k;
int g[15],va[15];
LD f[maxn][1<<15];
bool vis[maxn][1<<15];

inline int read(){
    int x = 0 , f = 1 ; char c = getchar() ;
    while( c < '0' || c > '9' ) { if( c == '-' ) f = -1 ; c = getchar() ; }
    while( c >= '0' && c <= '9' ) { x = x * 10 + c - '0' ; c = getchar() ; }
    return x * f ;
}
```

```
int main(){
    k = read();n = read();
    for(int i=0;i<n;i++){
        va[i] = read();int x = read();
        while(x) g[i] |= (1<<(x-1)),x = read();
    }
    for(int i=k;i>=1;i--){
        for(int s=0;s<(1<<n);s++){
            for(int j=0;j<n;j++){
                if((s&g[j])==g[j]) f[i][s] += max(f[i+1][s],f[i+1][s|(1<<j)]
+ va[j]);
                else f[i][s] += f[i+1][s];
            }
            f[i][s] /= n;
        }
    }
    printf("%.6Lf\n",f[1][0]);
    return 0;
}
```

**4.有后效性的期望DP**

P3232 [HNOI2013]游走

**题意：**有一个无向连通图，每次随机选一个边走过去，权值加上边的标号，走到 $n$ 结束。给每条边标号，使得期望值最小.

**做法：**每条边的权值计算独立，可以算每条边经过的期望次数，把概率排序，然后贪心编号。

设 $f_x$ 为经过 $x$ 的期望次数，那么经过一条边的期望可以表示为 $\frac{f_x}{d_x}+\frac{f_y}{d_y}$，只需要考虑 $f_x$ 的转移。

$f_i=\sum_{to}\frac{f_{to}}{d_{to}}+[i==1]$，由于直接做会出现环，采用高斯消元，可以解出来每个概率

```
#include<bits/stdc++.h>
#define pii pair<int,int>
#define pb push_back
#define mp make_pair
using namespace std;
typedef double db;
const int maxn = 505;
int n,m;
int d[maxn];
db Ans,f[maxn][maxn],g[maxn*maxn];
vector< pii > E[maxn];

inline int read(){
    int x = 0 , f = 1 ; char c = getchar() ;
    while( c < '0' || c > '9' ) { if( c == '-' ) f = -1 ; c = getchar() ; }
    while( c >= '0' && c <= '9' ) { x = x * 10 + c - '0' ; c = getchar() ; }
    return x * f ;
}

```

```
void add(int x,int y,int id){
    E[x].pb(mp(y,id));E[y].pb(mp(x,id));
    d[x]++;d[y]++;
}

void Gauss(int N){
    for(int i=1;i<=N;i++){
        for(int j=N+1;j>=i;j--) f[i][j] /= f[i][i];
        for(int j=1;j<=N;j++){
            if(i==j) continue;
            for(int k=N+1;k>=i;k--)
                f[j][k] -= f[j][i] * f[i][k];
        }
    }
}

int main(){
    n = read();m = read();
    for(int i=1;i<=m;i++) add(read(),read(),i);
    for(int i=1;i<n;i++){
        f[i][i] = 1.0;
        for(pii T : E[i]){
            int to = T.first;if(to==n) continue;
            f[i][to] = -1.0 / d[to];
        }
    }
    f[1][n] = 1;
    Gauss(n-1);
    for(int i=1;i<n;i++)
        for(pii T : E[i]){
            int to = T.first;
            g[T.second] += f[i][n] / d[i];
        }
    sort(g+1,g+1+m);
    for(int i=1;i<=m;i++) Ans += g[i] * (m-i+1);
    printf("%.3f\n",Ans);return 0;
}
```