T4.我坚定地迈向黄昏下远方无人问津的阴雨霉湿之地 题解

闲话

经过上一次多校的经验，我深刻的认识到根本没有什么人写T4 .

于是经过深思熟虑， 决定把T4和T3的位置交换一下，这样就会有很多人写T3了?

因此T4就放了一道很有意思，思维含量较高的题。

~~因为感觉并不会有什么人把大把时间放在T4上~~

不过为了丰富比赛体验，还是特地设置了许多部分分。

且这个题一点都不卡常。

题解

记$S=\sum a[i]$

算法1 $n=1$

输出0

算法2 $c_i=1$

如果你拥有过人的数学直觉，就会发现答案是$(n-1)^2$。

证明略。

算法3 $S\le 17$

我们可以暴力把每个颜色的球的个数压起来，但是这样状态数很大。

注意到，我们只要求最后停止的时间，不需要知道是哪种颜色。

也就是说，1个颜色为1的球，2个颜色为2的球，和2个颜色为1的球，1个颜色为2的球是没有区别的。

也就是说，某一个局面其实和具体的颜色是无关的，那么也就是说总状态数是17的分拆数，只有297.

我们可以搜出所有状态，然后列出关于每个状态的dp转移式，高斯消元求解。

这个dp式是类似于经典的随机游走。

可能没有那么好写。

可以参考下面的实现(by Cafard)

```
#include<bits/stdc++.h>
#define ll long long
#define uit unsigned int
using namespace std;
```

```
const int N=1010,mod=1e9+9;
inline int pls(int x,int y){
    return (x+=y)>=mod?x-mod:x;
}
inline int sub(int x,int y){
    return (x-=y)<0?x+mod:x;
}
int ftp(int b,int p){
    int r=1;
    while(p){
        if(p&1) r=1ll*r*b%mod;
        b=1ll*b*b%mod;
        p>>=1;
    }
    return r;
}
int n,bin[N],seg=0,f[N][N],all=0,p[N][N],g[N];
vector<int> vec;
map<vector<int>,int> id;
void Rf(vector<int> &tmp)
{
    sort(tmp.begin(),tmp.end());
}
vector<int> verf(vector<int> tmp)
{
    sort(tmp.begin(),tmp.end());
    return tmp;
}
int Get(vector<int> tmp)
{
    if(id.find(tmp)==id.end()) id[tmp]=++seg;
    return id[tmp];
}
bool check(vector<int> tmp)
{
    for(auto p:tmp){
        if(p==all) return 1;
    }
    return 0;
}
void dfs(vector<int> tmp)
{
    if(id.find(tmp)!=id.end()) return ;
    Get(tmp);
    if(check(tmp)){
        p[id[tmp]][0]=0;
        return ;
    }
    p[id[tmp]][0]=1;int nowid=id[tmp];
    for(int i=0;i<tmp.size();i++){
        if(tmp[i]==0) continue;
        for(int j=0;j<tmp.size();j++){
            if(tmp[j]==0) continue;
            if(i==j&&tmp[i]<2) continue;
            int temp=1ll*tmp[i]*ftp(all,mod-2)%mod;
```

```
                tmp[i]--;temp=1ll*temp*tmp[j]%mod*ftp(all-1,mod-2)%mod;
                tmp[i]+=2;tmp[j]--;
                dfs(verf(tmp));
                p[nowid][id[verf(tmp)]]=pls(p[nowid][id[verf(tmp)]],temp);
                tmp[i]--;tmp[j]++;
            }
        }
}
void gauss(int cnt)
{
    for(int c=1;c<=cnt;c++)
    {
        int now=c;
        for(int i=c+1;i<=cnt;i++)
            if(abs(f[i][c])>abs(f[now][c])) now=i;
        for(int i=1;i<=cnt;i++) swap(f[c][i],f[now][i]);swap(g[c],g[now]);
        for(int i=c+1;i<=cnt;i++)
            if(abs(f[i][c])){
                int t=1ll*f[i][c]*ftp(f[c][c],mod-2)%mod;
                for(int j=1;j<=cnt;j++) f[i][j]=sub(f[i][j],1ll*f[c]
[j]*t%mod);
                g[i]=sub(g[i],1ll*g[c]*t%mod);
            }
    }
    for(int i=cnt;i;i--){
        for(int j=cnt;j>i;j--){
            int t=f[i][j];
            for(int k=1;k<=cnt;k++)
                f[i][k]=sub(f[i][k],1ll*f[j][k]*t%mod);
            g[i]=sub(g[i],1ll*g[j]*t%mod);
        }
        int t=f[i][i];
        for(int j=1;j<=cnt;j++) f[i][j]=1ll*f[i][j]*ftp(t,mod-2)%mod;
        g[i]=1ll*g[i]*ftp(t,mod-2)%mod;
    }
}
void Deal()
{
    for(int i=1;i<=seg;i++)
    {
        g[i]=p[i][0];
        f[i][i]=1;
        for(int j=1;j<=seg;j++) f[i][j]=sub(f[i][j],p[i][j]);
    }
    gauss(seg);
}
int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
scanf("%d",&bin[i]),vec.push_back(bin[i]),all+=bin[i];
    Rf(vec);
    dfs(vec);
    Deal();
    printf("%d\n",g[id[vec]]);
```

```
    return 0;
}
```

算法4 n=2

本质还是上一个做法，只不过因为条件特殊可以不用高斯消元。

设$f[i]$为有i个1颜色的球，期望操作多少次结束。

$f[0]=f[S]=0$

可以列出一个$f[i]=p_1f[i-1]+p_2f[i+1]+c_i$

并且因为$f[0]=0$，所以我们可以得到，$f[1]=p_2f[2]+c_1$

把这个式子代入第二个式子，又可以得到关于只$f[2],f[3]$的式子。

通过归纳我们可以得到，可以把每个数表示成$f[i]=k_if[i+1]+b_i$的形式，并且系数是很好求出的。

然后再倒着代值就好了。

可以参考下面的实现方法。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
const int mod = 1e9+9;
LL Pow(LL a,LL b)
{
    LL res=1;
    while(b)
    {
        if(b&1)res=res*a%mod;
        a=a*a%mod;
        b>>=1;
    }
    return res;
}
int n,m;
const int N = 1e7+7;
int f[N];
int C[N],k[N],b[N];
int main()
{
    cin>>n;
    int u=-1;
    for(int i=1;i<=n;i++)
    {
        int x;
        scanf("%d",&x);
        m+=x;
        if(u==-1) u=x;
    }
    f[0]=0;f[m]=0;
```

```cpp
    int B=1ll*m*(m-1)%mod;
    B=Pow(B,mod-2);
    int P=Pow(2,mod-2);
    for(int i=1;i<m;i++)
    {
        C[i]=2ll*B*i*(m-i)%mod;
        C[i]=Pow(C[i],mod-2);
    }
    k[1]=P;
    b[1]=C[1];
    for(int i=2;i<m;i++)
    {
        k[i]=P;
        b[i]=(C[i]+1ll*b[i-1]*P%mod)%mod;
        int A=2,T=(2-k[i-1]+mod)%mod;
        int v=1ll*A*Pow(T,mod-2)%mod;
        k[i]=1ll*k[i]*v%mod;
        b[i]=1ll*b[i]*v%mod;
    }
    for(int i=m-1;i>=1;i--)
    f[i]=(1ll*k[i]*f[i+1]+b[i])%mod;
    cout<<f[u];
    return 0;
}

```

算法5

~~和前边的毫无关系~~

考虑记一个S_t为操作t次后的局面。

然后是非常神奇的转化，我们考虑找一个势函数$\phi(S_t)$，满足一下条件：

1：$\phi(S_{end})$为常数

2：期望意义下$\phi(S_t)-\phi(S_{t+1})=1$

那么也就是说每做一次操作，势能减少1

那么$\phi(S_{begin})-\phi(S_{end})$

就是答案。

我们只需要构造一个合法的$\phi(S_t)$即可。

考虑$f(a_i)$表示第i种颜色的云有a_i个时的势函数。

我们令$\phi(S_t)=\sum_i f(a_i)$

记$m=\sum a_i$

1.那么我们有$\frac{a_i(a_i-1)}{m(m-1)}$的概率选择两个$i$类云

2:我们有$\frac{a_ia_j}{m(m-1)}$的概率将一个i类云变成j类云

所以

$\phi(S_{t+1}) = \frac{1}{m(m-1)}(\sum_i a_i(a_i-1)\phi(S_t) + \sum_i \sum_{j!=i} a_i a_j(\phi(S_t) + f(a_i+1) - f(a_i) + f(a_j-1) - f(a_j)))$

$= \frac{1}{m(m-1)}(\sum_i (a_i(a_i-1) + a_i(m-a_i))\phi(S_t) + \sum_i a_i(m-a_i)(f(a_i+1) - 2\times f(a_i) + f(a_i-1))$

$= \frac{1}{m(m-1)}(\sum_i (a_i m - a_i)\phi(S_t) + \sum_i a_i(m-a_i)(f(a_i+1) - 2\times f(a_i) + f(a_i-1))$

$= \frac{1}{m(m-1)}(m(m-1)\phi(S_t) + \sum_i a_i(m-a_i)(f(a_i+1) - 2\times f(a_i) + f(a_i-1))$

$= S_t + \frac{1}{m(m-1)}(\sum_i a_i(m-a_i)(f(a_i+1) - 2\times f(a_i) + f(a_i-1))$

因为我们要$\phi(S_{t+1}) = \phi(S_t) - 1$

因此，$\frac{1}{m(m-1)}(\sum_i a_i(m-a_i)(f(a_i+1) - 2\times f(a_i) + f(a_i-1)) = -1$

$1 + \frac{1}{m(m-1)}(\sum_i a_i(m-a_i)(f(a_i+1) - 2\times f(a_i) + f(a_i-1)) = 0\$\$$

$m + \frac{1}{m-1}(\sum_i a_i(m-a_i)(f(a_i+1) - 2\times f(a_i) + f(a_i-1)) = 0$

$\sum_i a_i(1 + \frac{m-a_i}{m-1}(f(a_i+1) - 2\times f(a_i) + f(a_i-1))) = 0$

因为我们是构造，所以要求该式在任何情况下都必须成立，因此，就是要求

$1 + \frac{m-x}{m-1}(f(x+1) - 2f(x) + f(x-1)) = 0$对任意的$x$恒成立

就是

$f(x+1) - 2f(x) + f(x-1) = -\frac{m-1}{m-x}$

这个形式很像一个差分的形式。

设$g(x) = f(x) - f(x-1)$

那么

$g(x+1) - g(x) = -\frac{m-1}{m-x}$

即$g(x+1) = g(x) + \frac{1-m}{m-x}$

$g(x) = g(0) + \sum_{i=0}^{x-1} \frac{1-m}{m-i}$

$f(x) = f(0) + \sum_{i=1}^{x} g(i)$

$= f(0) + x\times g(0) - \sum_{i=0}^{x-1} \frac{(m-1)(x-i)}{m-i}$

$= f(0) + xg(0) - (m-1)\sum_{i=0}^{x-1} \frac{x-i}{m-i}$

$= f(0) + xg(0) - (m-1)\sum_{i=0}^{x-1} \frac{x-i-m+m}{m-i}$

$= f(0) + xg(0) - (m-1)\sum_{i=0}^{x-1} \frac{x-m+m-i}{m-i}$

$= f(0) + xg(0) - (m-1)\sum_{i=0}^{x-1} (\frac{x-m}{m-i} + 1)$

$= f(0) + xg(0) - (m-1)\sum_{i=0}^{x-1} \frac{x-m}{m-i} - (m-1)x\$\$$

$= f(0) + xg(0) + (m-x)(m-1)\sum_{i=0}^{x-1} \frac{1}{m-i} - (m-1)x$

$= f(0) + x(g(0) - (m-1)) + (m-x)(m-1)\sum_{i=0}^{x-1} \frac{1}{m-i}$

在保证$\phi(S_{end})$为常数的情况下，我们可以让$f(0)$和$g(0)$取任意值。

这里为了方便化简与计算，取$f(0)=0,g(0)=m-1$

那么

$f(x)=(m-x)(m-1)\sum_{i=0}^{x-1}\frac{1}{m-i}$

易证得此时$\phi(S_{end})=0$为常数

线性求逆元可以做到$O(n+\max c_i)$，使用快速幂求逆元就是一个log，均可通过。