

# 2019第一中学信息奥赛小组模测 10月22号题解

## 魔法部落

### 题解

本题使用等比数列求和公式即可得到以下答案

$$\text{ans} = \frac{3^n - 1}{2}$$

之后用矩阵快速幂+逆元即可在log的复杂度内解决本题。

### 标准代码

C++

```
#include<stdio.h>
#define mod 1000000007
typedef long long ll;

ll mod_pow(ll x,ll n)
{
    ll res = 1;
    while(n > 0)
    {
        if(n & 1)
            res = res * x % mod;
        x = x * x % mod;
        n >>= 1;
    }
    return res;
}

int main()
{
    ll n, ans;
    scanf("%lld", &n);
    n++;
    ans = (mod_pow(3, n) - 1) * 500000004 % mod;
    printf("%lld", ans);
    return 0;
}
```

# 圆盘

## 题解

首先求出关键点两两之间的距离，存到一个数组中。

由于圆盘可以旋转，那么只要两个数组通过旋转后相同即可。但暴力比较2个数组是平方的，因为需要旋转 $n$ 次。如果枚举2对做比较，是 $O(n^4)$ 。

这里用最小表示法来做，即设计一个滚动的Hash算法，然后对每个数组求所有旋转方案的Hash，这样可以 $O(n)$ 算出一个数组的 $n$ 次旋转后的Hash值。我们只保留其中Hash值最小的。即最小表示。这样可以在 $O(n^2)$ 求出所有最小表示，然后可以 $O(n)$ 统计有多少对相同。

## 标准代码

C++

```
#include <cstdio>
#include <set>
#include <cstring>
#include <algorithm>
using namespace std;

typedef long long ll;

const int N = 510;

int n, m, p, mp[N][N], sa[N][N], fa[N], num[N];

void init() {
    for (int i = 1; i <= n; i++)
        fa[i] = i;
    for (int i = 1; i <= n; i++)
    {
        sort(sa[i], sa[i] + m);
        for (int j = 0; j < m - 1; j++) {
            mp[i][j] = sa[i][j + 1] - sa[i][j];
        }
        mp[i][m - 1] = p - sa[i][m - 1] + sa[i][0];
    }
}

int find(int x) {
    if (fa[x] == x)
        return x;
    return fa[x] = find(fa[x]);
}

bool ok(int a, int b) {
    for (int i = 0; i < m; i++)
    {
        bool is = true;
```

```

        for (int k = 0; k < m; k++) {
            if (mp[a][k] != mp[b][(i + k) % m])
                { is = false;
                  break;
                }
        }
        if (is)
            return true;
    }
    return false;
}

void solve() {
    init();
    for (int i = 2; i <= n; i++)
        { set<int> s;
          for (int j = 1; j < i; j++)
              { if (s.count(find(j)))
                  continue;
                if (ok(i, j)) {
                    fa[i] = fa[j];
                    break;
                }
                }else
                    s.insert(fa[i]);
            }
        }
    memset(num, 0, sizeof(num));
    for (int i = 1; i <= n; i++)
        find(i);
    for (int i = 1; i <= n; i++)
        num[fa[i]]++;
    ll ans = 0;
    for (int i = 1; i <= n; i++) {
        ans += 1LL * num[i] * (num[i] - 1) / 2;
    }
    printf("%lld\n", ans);
}

int main() {
    while (~scanf("%d%d%d", &n, &m, &p))
        { for (int i = 1; i <= n; i++) {
            for (int j = 0; j < m; j++)
                { scanf("%d", &sa[i][j]);
                  }
            }
        solve();
    }
    return 0;
}

```

## 棋盘行走

## 题解

在同颜色相邻的点之间连上边。我们考虑逐个把边加入进来，每次加边时考虑连边的2个点是否已经在同一个集合里了，如果在，则表示存在一个环。如果不在则合并。这是用并查集判断是否存在环的经典例题，由于边的数量是 $O(n)$ 的，因此总的复杂度为 $O(n \times \alpha(n))$ 。

继续优化：除了并查集判环，直接使用DFS也是可以判断是否存在环的，与并查集的方法类似，在同颜色相邻的点之间连上边。DFS过程中对访问过的点进行标注，假如DFS到了某个已经被标注过的点，则认为有环。这样做的复杂度是 $O(n)$ 。

## 标准代码

C++

```
#include <bits/stdc++.h>
using namespace std;

int n,m,k;
const int N=64;
char arr[N][N];
bool flag = false;
bool visit[N][N];
int dirx[]={1,0,-1,0};
int diry[]={0,1,0,-1};
void dfs(int i,int j,int pi,int pj)
{
    if(visit[i][j])
    {
        flag = true;
        return;
    }
    visit[i][j]=true;
    for(int k=0;k<4;k++)
    {
        int nextx = i+dirx[k];
        int nexty = j+diry[k];
        if(nextx>=0&&nextx<n&&nexty>=0&&nexty<m&&arr[i][j]==arr[nextx][nexty])
        {
            if(!(nextx==pi&&nexty==pj))
                dfs(nextx,nexty,i,j);
        }
    }
}
int main()
{
    while(scanf("%d%d", &n, &m)==2)
    {
        flag = false;
        for(int i=0;i<n;i++)
            scanf("%s", arr[i]);
        for(int i=0;i<n;i++)
            for(int j=0;j<m;j++)
```

```

        {
            memset(visit,0,sizeof(visit));
            dfs(i,j,-1,-1);
            if(flag) goto FINISH;
        }
    FINISH:
    if(flag)
        cout<<"Yes"<<endl;
    else
        cout<<"No"<<endl;
}
return 0;
}

```

# 走方格

## 题解

首先分别预处理奇数位偶数位的前缀和 和 后缀和。

之后  $O(n)$  枚举位置并检查即可。

## 标准代码

C++

```

#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
#define rep(i,l,r) for(int i=l;i<=r;++i)
const int N=2e5+5;
ll a[N],s[N][2];

int main()
{
    //freopen("7.in","r",stdin);
    //freopen("7.out","w",stdout);
    int n;
    cin>>n;
    rep(i,1,n)scanf("%lld",a+i);
    rep(i,1,n)
    rep(j,0,1)s[i][j]=s[i-1][j]+(i%2==j)*a[i];
    ll ans=0;
    rep(i,1,n)
    if(s[i-1][1]-s[i-1][0]-(s[n][1]-s[i][1]-s[n][0]+s[i][0])==0)++ans;
    cout<<ans;
}

```