Q:为什么只有A□B的题解呢

A:当然因为我太菜了，2个小时就做出来两道题

~~再给我两个小时估计也只能做出来这两道题~~

A. Hilbert’s Hotel

通过简单的证明，我们可知，只需验证$\{x|x=(a_i+i)mod\; n\}(0\le i< n)$是否就是集合$\{1,2,\ldots,n\}$即可

代码：

```
#include <stdio.h>
#include <algorithm>
using namespace std;
bool cmp(int a,int b)
{
    return a<b;
 }
int a[200001];
int main()
{
    int T;
    scanf("%d",&T);
    while (T--)
    {
        int n;
        scanf("%d",&n);
        for (int i=0;i<n;i++)
        {
        scanf("%d",&a[i]);
        a[i]+=i;
        a[i]%=n;
        a[i]+=n;
        a[i]%=n;
        }
         sort(a,a+n,cmp);
         bool check=true;
         for (int i=1;i<n;i++)
         if (a[i]-a[i-1]!=1) check=false;
         if (check)
         printf("YES\n");
         else printf("NO\n");
     }
    return 0;
}
```

B. Monopole Magnets

假如存在一种放置满足题意，那么每个连通块只需放置一个N级磁铁即可，答案ans=连通块的个数

这道题的关键是如何判断是否存在，不存在的情况有两种：

对于某一行或某一列的方格，存在下面情况：在某一处存在一块黑色方格，紧接着几个是白色的，后面又出现了一块黑色的方格。我们可以用反证法（下面写的不太严谨），记这两个黑色方格为方格A和方格B假如存在一种放置方法满足题意，那么经过一系列操作后，方格A可以有一块N级磁铁，由于该行或该列必须有一块S级磁体。所以这块磁铁必须放置在方格A的上方，对方格B用同样的方法，可以推出这块磁铁必须放置在方格B的下方。所以得出矛盾。

有一行或多行全是白色方格，但所有的列均含有黑色方格；或者有一列或多列全是白色方格，但所有的行均含黑色方格。（就不证了）

代码:

```
#include <stdio.h>
int map[1001][1001];
int n,m;
int dx[4]={1,-1,0,0};
int dy[4]={0,0,-1,1};
inline bool check(int x,int y)
{
    return (x>=1&&x<=n&&y>=1&&y<=m);
}
void dfs(int x,int y)
{
    for (int i=0;i<4;i++)
    if (check(x+dx[i],y+dy[i])&&map[x+dx[i]][y+dy[i]])
    {
        map[x+dx[i]][y+dy[i]]=0;
        dfs(x+dx[i],y+dy[i]);
    }
}
int main()
{
    scanf("%d %d\n",&n,&m);
    char c;
    for (int i=1;i<=n;i++)
    {
    for (int j=1;j<=m;j++)
    {
        scanf("%c",&c);
        if (c=='#')
        map[i][j]=1;
        else map[i][j]=0;
    }
    scanf("%c",&c);
```

```
    }
    bool check=true;
    bool checks=true;
    bool check1=true,check2=true;
    for (int i=1;i<=n;i++)
    {
        int jugde=0;
        bool check1s=true;
        for (int j=1;j<=m;j++)
        {
            if (map[i][j]==1)
            check1s=false,checks=false;
        if (!jugde&&map[i][j]==1)
            jugde=1;
        if (jugde==1&&map[i][j]==0)
        jugde=2;
        if (jugde==2&&map[i][j]==1)
        check=false;
        }
        if (check1s) check1=false;
    }
    for (int j=1;j<=m;j++)
    {
        int jugde=0;
        bool check2s=true;
        for (int i=1;i<=n;i++)
        {
            if (map[i][j]==1)
            check2s=false;
        if (!jugde&&map[i][j]==1)
            jugde=1;
        if (jugde==1&&map[i][j]==0)
        jugde=2;
        if (jugde==2&&map[i][j]==1)
        check=false;
        }
        if (check2s) check2=false;
    }
    if ((!check||(check1^check2))&&!checks)
    {
    printf("-1");return 0;}
    int ans=0;
    for (int i=1;i<=n;i++)
    {
    for (int j=1;j<=m;j++)

    if (map[i][j])
    {
        ans++;
        dfs(i,j);
    }
```

```
        }
    printf("%d",ans);
    return 0;
}
```