模仿1.4节商人过河问题中的状态转移模型，作下面这个众所周知的智力游戏：

人带着猫、鸡、米过河，船除希望要人计划之外，至多能载猫、鸡、米三者之一，而当人不在场时猫要吃鸡、鸡要吃米，设计一个安全过河方案，并使渡河次数尽量地少.

一、 问题分析

这个实际问题要我们解决：在给定的条件下，把河一边的人、猫、鸡、米送到河对岸。这是我们上课讲过的状态转移模型。可以使用穷举法。用一个四维向量来表示状态的话，那么这个问题就是：在奇数次的变化后，把（1，1，1，1）变成（0，0，0，0），并使这个变换次数尽可能的少。

二、 模型假设

1. 船除希望要人计划之外，至多能载猫、鸡、米三者之一。
2. 而当人不在场时猫要吃鸡、鸡要吃米。
3. 河分为左、右两岸。

三、 模型构成

1. 状态向量

我们把人、猫、鸡、米的状态依次用四维向量中的分量来表示。当一物在左岸时，相应的分量为1，在右岸时为0。如向量（1，0，0，1）表示人和鸡在左岸，猫和米在右岸。由于假设条件，有些状态是允许的，有些则是不允许的。

2. 决策向量

把每一次决策（运载策略）也用一个四维向量来表示，如（1，1，0，0）表示人和猫在船上。不过我们要注意舍弃一些不可取的运载策略。

四、 模型建立

1. 通过穷举法知，可取状态向量一共有10个：

（1，1，1，1）	（0，0，0，0）
（1，1，1，0）	（0，0，0，1）
（1，1，0，1）	（0，0，1，0）
（1，0，1，1）	（0，1，0，0）
（1，0，1，0）	（0，1，0，1）

2. 通过穷举法知，可取的决策向量一共有4个：

（1，0，0，0）	（1，0，0，1）
（1，0，1，0）	（1，1，0，0）

3. 每次决策相当于用状态向量减决策向量。

五、 模型求解

由于问题规模较小，可以采用穷举法求解，我们从起始状态向量（1，1，1，1）开始。

第一步去，左岸起始：

$$（1，1，1，1）\begin{cases}(1,0,1,0)\\(1,1,0,0)\\(1,0,0,1)\\(1,0,0,0)\end{cases}\to\begin{cases}(0,1,0,1)\surd\\(0,0,1,1)\times\\(0,1,1,0)\times\\(0,1,1,1)\times\end{cases}\text{对应右岸状态}\begin{cases}(1,0,1,0)\\(1,1,0,0)\\(1,0,0,1)\\(1,0,0,0)\end{cases}$$

第二步返，右岸起始：

$$（1，0，1，0）\begin{cases}(1,0,1,0)\\(1,1,0,0)\\(1,0,0,1)\\(1,0,0,0)\end{cases}\to\begin{cases}(0,0,0,0)\text{ 重复}\\ \text{无法执行}\\ \text{无法执行}\\(0,0,1,0)\surd\end{cases}\text{对应左岸状态}\begin{cases}(1,1,1,1)\\ \times\\ \times\\(1,1,0,1)\end{cases}$$

第三步去，左岸起始：

$$（1，1，0，1）\begin{cases}(1,0,1,0)\\(1,1,0,0)\\(1,0,0,1)\\(1,0,0,0)\end{cases}\to\begin{cases}\text{无法执行}\\(0,0,0,1)\surd\\(0,1,0,0)\surd\\(0,1,0,1)\text{ 重复}\end{cases}\text{对应右岸状态}\begin{cases}\times\\(1,1,1,0)\\(1,0,1,1)\\(1,0,1,0)\end{cases}$$

由于第三步出现了两种可行状态，所以第四步分为两种情况。

第四步返，右岸起始：

$$（1，1，1，0）\begin{cases}(1,0,1,0)\\(1,1,0,0)\\(1,0,0,1)\\(1,0,0,0)\end{cases}\rightarrow\begin{cases}(0,1,0,0)\surd\\(0,0,1,0)\,重复\\无法执行\\(0,1,1,0)\times\end{cases}\quad 对应左岸状态\begin{cases}(1,0,1,1)\\(1,1,0,1)\\\times\\(1,0,0,1)\end{cases}$$

$$（1，0，1，1）\begin{cases}(1,0,1,0)\\(1,1,0,0)\\(1,0,0,1)\\(1,0,0,0)\end{cases}\rightarrow\begin{cases}(0,0,0,1)\surd\\无法执行\\(0,0,1,0)\,重复\\(0,0,1,1)\times\end{cases}\quad 对应左岸状态\begin{cases}(1,1,1,0)\\\times\\(1,1,0,1)\\(1,1,0,0)\end{cases}$$

同第四步，第五步也要分类讨论。

第五步去，左岸起始：

$$（1，0，1，1）\begin{cases}(1,0,1,0)\\(1,1,0,0)\\(1,0,0,1)\\(1,0,0,0)\end{cases}\rightarrow\begin{cases}(0,0,0,1)\,重复\\无法执行\\(0,0,1,0)\surd\\(0,0,1,1)\times\end{cases}\quad 对应右岸状态\begin{cases}(1,1,1,0)\\\times\\(1,1,0,1)\\(1,1,0,0)\end{cases}$$

$$（1，1，1，0）\begin{cases}(1,0,1,0)\\(1,1,0,0)\\(1,0,0,1)\\(1,0,0,0)\end{cases}\rightarrow\begin{cases}(0,1,0,0)\text{ 重复}\\(0,0,1,0)\surd\\ \textit{无法执行}\\(0,1,1,0)\times\end{cases}\text{ 对应右岸状态}\begin{cases}(1,0,1,1)\\(1,1,0,1)\\ \times\\(1,0,0,1)\end{cases}$$

第六步返，右岸起始：

$$（1，1，0，1）\begin{cases}(1,0,1,0)\\(1,1,0,0)\\(1,0,0,1)\\(1,0,0,0)\end{cases}\rightarrow\begin{cases}\textit{无法执行}\\(0,0,0,1)\textit{ 重复}\\(0,1,0,0)\textit{ 重复}\\(0,1,0,1)\surd\end{cases}\text{ 对应左岸状态}\begin{cases}\times\\(1,1,1,0)\\(1,0,1,1)\\(1,0,1,0)\end{cases}$$

第七步去，左岸起始：

$$（1，0，1，0）\begin{cases}(1,0,1,0)\\(1,1,0,0)\\(1,0,0,1)\\(1,0,0,0)\end{cases}\rightarrow\begin{cases}(0,0,0,0)\\(0,1,1,0)\times\\(0,0,1,1)\times\\(0,0,1,0)\textit{ 重复}\end{cases}\text{ 对应右岸状态}\begin{cases}(1,1,1,1)\\(1,0,0,1)\\(1,1,0,0)\\(1,1,0,1)\end{cases}$$

第七步是左岸出现了(0, 0, 0, 0)状态，说明七次决策即可完成问题他的具体过程是：

$\xrightarrow{去}$（人，鸡）$\xrightarrow{回}$（人）$\xrightarrow{去}$（人，猫）$\xrightarrow{回}$（人，鸡）$\xrightarrow{去}$（人，米）$\xrightarrow{回}$（人）$\xrightarrow{去}$（人，鸡）

或

$\xrightarrow{去}$（人，鸡）$\xrightarrow{回}$（人）$\xrightarrow{去}$（人，米）$\xrightarrow{回}$（人，鸡）$\xrightarrow{去}$（人，猫）$\xrightarrow{回}$（人）$\xrightarrow{去}$（人，鸡）

六、 模型评价

1. 优点

模型简单，符合实际，易于理解。

2. 缺点

当情况变多的时候操作繁琐。

七、 代码(C 语言描述)

```
#include<stdio.h>
int initial[4]= {1,1,1,1};                //储存初始向量
int chose[4][4]=
{{1,0,0,0},{1,1,0,0},{1,0,1,0},{1,0,0,1}}; //储存决策
int used[16]= {0};                        //记录状态的重复
int ans[10]= {0};                         //记录答案
int count = 1;                            //记录次数
void display(int count) {                 //输出函数
   int i=0,temp=0;
   for(i=1; i<count; i++) {
      temp=ans[i+1]-ans[i];
      if(temp<0) {
         printf("第%d 次：去：",i);
         if(temp==-8)
            printf("人独自");
```

```
        if(temp==-12)
          printf("人和猫 ");
        if(temp==-10)
          printf("人和鸡 ");
        if(temp==-9)
          printf("人和米 ");
        printf("\n");
      } else {
        printf("第%d 次: 返: ",i);
        if(temp==8)
          printf("人独自");
        if(temp==12)
          printf("人和猫 ");
        if(temp==10)
          printf("人和鸡 ");
        if(temp==9)
          printf("人和米 ");
        printf("\n");
      }
    }
    printf("\n");
}
```

```
int judge(int i,int count,int a[]) {        //判断能不能走
    int j=0;
    if(count%2==1) {
        for(j=0; j<4; j++) {
            if(chose[i][j]==1&&a[j]==0)
                return 0;
        }
    } else {
        for(j=0; j<4; j++) {
            if(chose[i][j]==1&&a[j]==1)
                return 0;
        }
    }
    return true;
}
void dfs(int a[],int b[],int x,int count) {        //深搜
    int last[4]= {0};
    int i=0,j=0,k=0;
    int _this[4]= {0},pre[16]= {0};
    int flag = 0;
    for(i=0; i<4; i++)
        _this[i]=a[i];
```

```
for(i=0; i<16; i++)
    pre[i]=b[i];
x=_this[0]*8+_this[1]*4+_this[2]*2+_this[3];
ans[count]=x;
if(x==0) {
    display(count);
    return;
}
pre[x]=1;
for(i=0; i<4; i++) {
    if(judge(i,count,_this)==0)
        continue;

k=chose[i][0]*8+chose[i][1]*4+chose[i][2]*2+chose[i][3]
;
    if(count%2==0) {
        flag=1;
    } else flag=-1;

if(!(x+flag*k==3||x+flag*k==6||x+flag*k==7||x+flag*k==8
||x+flag*k==9||x+flag*k==12)&&pre[x+flag*k]!=1) {
        last[0]=(_this[0]+chose[i][0])%2;
```

```
            last[1]=(_this[1]+chose[i][1])%2;
            last[2]=(_this[2]+chose[i][2])%2;
            last[3]=(_this[3]+chose[i][3])%2;
            dfs(last,pre,x+flag*k,count+1);
        }
    }
}
int main() {
    int n=15,flag=1;
    dfs(initial,used,n,count);
    return 0;
}
```