**实验报告**

**课程名称：算法设计与分析**

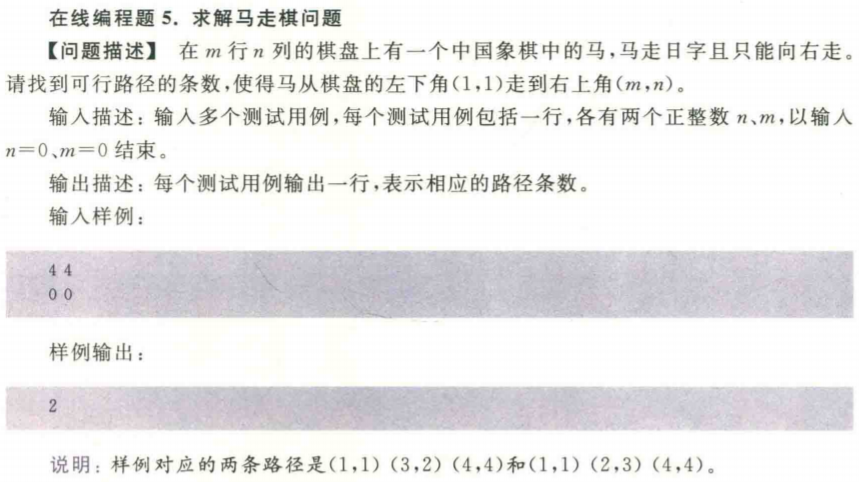
**专业班级：大数据182**

**学号姓名：3180439031陈佳婧**

**实验日期：2020年11月9日**

**第五章在线编程题5： 求解马走棋问题**

1. **实验目标和要求：**



**2. 实验环境：（操作系统、语言、编译工具…）WIN10、C++、Clion**

**3. 关键问题及解决思路：**

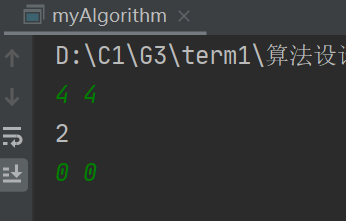
**Dfs回溯求解。**

**4. 程序流程：**

**5. 完整程序代码：**

#include <iostream>  
using namespace std;  
int n=1,m=1;  
int dep=0;  
void dfs(int i,int j,int flag){  
 if(i<1||j<1||i>n||j>m)return;  
 if(i==n&&j==m){  
 dep++;  
 return;  
 }  
 if(flag){ //flag=1 ,表示前一步为直走  
 flag=0;  
 dfs(i+1,j+1,flag);  
 }  
 else { //flag=0,前一步为斜走  
 flag=1;  
 dfs(i+1,j,flag); //向右直走  
 flag=1;  
 dfs(i,j+1,flag); //向上直走  
 }  
}  
  
int main() {  
 while(cin>>n>>m&&n!=0&&m!=0){  
 dep=0;  
 dfs(1,1,0);  
 cout<<dep<<endl;  
 }  
 return 0;  
}

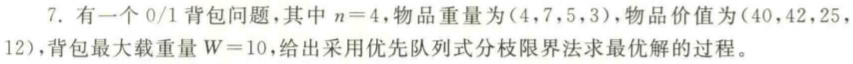
**6. 运行结果展示：**



**7. 实验体会（可选）：**

**第六章练习题7：**

**1.实验目标和要求：**



**2. 实验环境：（操作系统、语言、编译工具…）WIN10、C++、Clion**

**3. 关键问题及解决思路：**

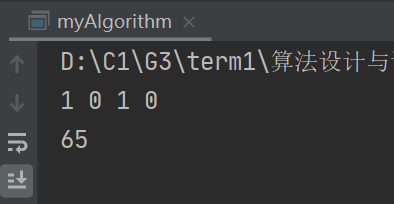
**优先队列bfs。**

**4. 程序流程：**

**5. 完整程序代码：**

#include <iostream>  
#include <queue>  
using namespace std;  
#define MAXN 20  
#define INF 0x3f3f3f3f  
int n=4,W=10;  
int w[]={0,4,7,5,3};  
int v[]={0,40,42,25,12};  
int maxv=-9999; //最大价值  
int bestx[MAXN]; //最优解  
int total=1;  
struct NodeType{  
 int no; //结点编号  
 int i; //当前结点在搜索空间中的层次  
 int w;  
 int v;  
 int x[MAXN];  
 double ub; //上界  
 bool operator<(const NodeType &s) const{ //重载<关系函数  
 return ub<s.ub; //ub越大越优先出局  
 }  
};  
void bound(NodeType &e){ //计算结点上界  
 int i=e.i+1;  
 int sumw=e.w;  
 double sumv=e.v;  
 while((sumw+w[i]<=W)&&i<=n){  
 sumw+=w[i];  
 sumv+=v[i];  
 i++;  
 }  
 if(i<=n){  
 e.ub=sumv+1.0\*(W-sumw)\*v[i]/w[i]; //计算上界  
 }else{  
 e.ub=sumv;  
 }  
}  
void EnQueue(NodeType e,priority\_queue<NodeType> &qu){  
 if(e.i==n){  
 if(e.v>maxv){  
 maxv=e.v;  
 for(int j=1;j<n;j++){  
 bestx[j]=e.x[j];  
 }  
 }  
 }else{  
 qu.push(e);  
 }  
}  
void bfs(){  
 int j;  
 NodeType e{},e1{},e2{};  
 priority\_queue<NodeType> qu;  
 e.i=0;  
 e.w=0;  
 e.v=0;  
 e.no=total++;  
 for(j=1;j<=n;j++){  
 e.x[j]=0;  
 }  
 bound(e);  
 qu.push(e); //根结点进队  
 while(!qu.empty()){  
 e=qu.top();  
 qu.pop();  
 if(e.w+w[e.i+1]<=W){ //检查左孩子结点  
 e1.no=total++;  
 e1.i=e.i+1;  
 e1.w=e.w+w[e1.i];  
 e1.v=e.v+v[e1.i];  
 for(j=1;j<=n;j++){  
 e1.x[j]=e.x[j];  
 }  
 e1.x[e1.i]=1;  
 bound(e1);  
 EnQueue(e1,qu);  
 }  
 e2.no=total++;  
 e2.i=e.i+1;  
 e2.w=e.w;  
 e2.v=e.v;  
 for(j=1;j<=n;j++){  
 e2.x[j]=e.x[j];  
 }  
 e2.x[e2.i]=0;  
 bound(e2);  
 if(e2.ub>maxv)  
 EnQueue(e2,qu);  
 }  
}  
  
int main() {  
 bfs();  
 for(int i=1;i<=n;i++){  
 cout<<bestx[i]<<" ";  
 }cout<<endl;  
 cout<<maxv<<endl;  
 return 0;  
}

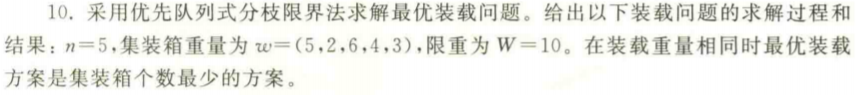
**6. 运行结果展示：**



**7. 实验体会（可选）：**

**第六章练习题10：**

**1.实验目标和要求：**



**2. 实验环境：（操作系统、语言、编译工具…）WIN10、C++、Clion**

**3. 关键问题及解决思路：**

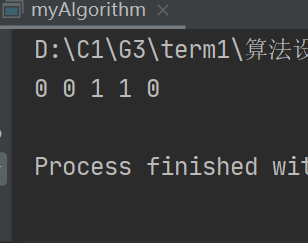
**优先队列bfs。**

**4. 程序流程：**

**5. 完整程序代码：**

#include <iostream>  
#include <queue>  
using namespace std;  
#define MAXN 20  
#define INF 0x3f3f3f3f  
int n=5,W=10;  
int w[]={0,5,2,6,4,3};  
int bestx[MAXN]; //最优解  
int total=1;  
int maxw=-9999;  
int minc=6;  
struct NodeType{  
 int no; //结点编号  
 int i; //当前结点在搜索空间中的层次  
 int w;  
 int x[MAXN];  
   
 double ub; //上界  
 bool operator<(const NodeType &s) const{ //重载<关系函数  
 return ub<s.ub; //ub越大越优先出局  
 }  
};  
void bound(NodeType &e){ //计算结点上界  
 int i=e.i+1;  
 int sumw=e.w;  
 while((sumw+w[i]<=W)&&i<=n){  
 sumw+=w[i];  
 i++;  
 }  
 e.ub=sumw;  
}  
void EnQueue(NodeType e,priority\_queue<NodeType> &qu){  
 if(e.i==n){  
 int cnt=0;  
 for(int j=1;j<n;j++){  
 if(e.x[j])cnt++;  
 }  
 if(e.w>maxw&&cnt<minc){  
 maxw=e.w;  
 for(int j=1;j<n;j++){  
 bestx[j]=e.x[j];  
 }  
 }  
 }else{  
 qu.push(e);  
 }  
}  
void bfs(){  
 int j;  
 NodeType e{},e1{},e2{};  
 priority\_queue<NodeType> qu;  
 e.i=0;  
 e.w=0;  
 e.no=total++;  
 for(j=1;j<=n;j++){  
 e.x[j]=0;  
 }  
 bound(e);  
 qu.push(e); //根结点进队  
 while(!qu.empty()){  
 e=qu.top();  
 qu.pop();  
 if(e.w+w[e.i+1]<=W){ //检查左孩子结点  
 e1.no=total++;  
 e1.i=e.i+1;  
 e1.w=e.w+w[e1.i];  
 for(j=1;j<=n;j++){  
 e1.x[j]=e.x[j];  
 }  
 e1.x[e1.i]=1;  
 bound(e1);  
 EnQueue(e1,qu); //左孩子进队  
 }  
 e2.no=total++;  
 e2.i=e.i+1;  
 e2.w=e.w;  
 for(j=1;j<=n;j++){  
 e2.x[j]=e.x[j];  
 }  
 e2.x[e2.i]=0;  
 bound(e2);  
 if(e2.ub>=maxw)  
 //cout<<e2.ub<<endl;  
 EnQueue(e2,qu); //右孩子进队  
 }  
}  
  
int main() {  
 bfs();  
 for(int i=1;i<=n;i++){  
 cout<<bestx[i]<<" ";  
 }cout<<endl;  
 return 0;  
}

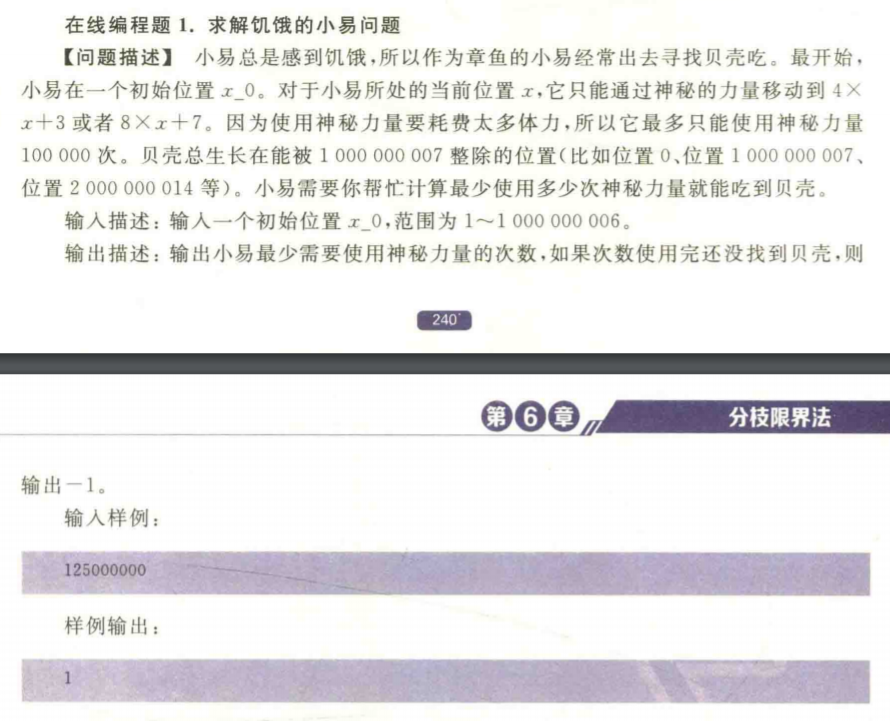
**6. 运行结果展示：**



**7. 实验体会（可选）：**

**第六章在线编程题1： 求解饥饿的小易问题**

**1.实验目标和要求：**



**2. 实验环境：（操作系统、语言、编译工具…）WIN10、C++、Clion**

**3. 关键问题及解决思路：**

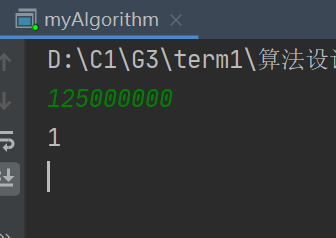
**直接上bfs**

**4. 程序流程：**

**5. 完整程序代码：**

#include <cstdio>  
#include <map>  
#include <queue>  
using namespace std;  
const long long mod=1000000007;  
struct node{  
 long long x,step;  
 node(long long x,long long step):x(x),step(step){}  
};  
int main(){  
 long long x;  
 while(scanf("%lld",&x)!=EOF){  
 queue<node> Q;  
 map<long long,int> book;  
 Q.push(node(x,0)),book[x]=1;  
 long long res=-1;  
 while(!Q.empty()){  
 node head=Q.front();Q.pop();  
 if(head.x%mod==0){  
 res=head.step;  
 break;  
 }  
 if(head.step>100000) break;  
 long long x1=(4\*head.x+3)%mod,x2=(8\*head.x+7)%mod;  
 if(book.count(x1)==0){  
 book[x1]=1;  
 Q.push(node(x1,head.step+1));  
 }  
 if(book.count(x2)==0){  
 book[x2]=1;  
 Q.push(node(x2,head.step+1));  
 }  
 }  
 printf("%lld\n",res);  
 }  
}

**6. 运行结果展示：**



**7. 实验体会（可选）：**