**回溯法**

**1610300c02 夏鹏程**

**总分: 98 / 102**

编程题总分: 98 / 102

7-1部分正确 得分: 6 / 8

有一个背包的最大能承受的重量是 M ，有 n 个物品，每个物品有各自的重量和价值，计算在不超出背包最大承重限制下，背包中物品最大价值可以是多少？

**输入格式:**

第一行输入背包的最大承重量 M 和物品的个数 n (1<M<500，1<n<20)，第二行输入1至n个物品的重量，第三行输入1至n个物品的价值。

**输出格式:**

输出最大可能的价值。

**输入样例:**

在这里给出一组输入。例如：

30 3

5 10 20

50 60 140

**输出样例:**

在这里给出相应的输出。例如：

200

#include<iostream>

#include<algorithm>

using namespace std;

int w[25];

int v[25];

int vt[25][505];

int main()

{ int m,n;

cin>>m>>n;

for(int i=1;i<=n;i++)

{ cin>>w[i];

}

for(int i=1;i<=n;i++)

{ cin>>v[i]; }

vt[0][0]=0;

for(int i=1;i<=n;i++)

{ vt[i][0]=0,vt[0][i]=0; }

for(int i=1;i<=n;i++)

{ for(int j=1;j<=m;j++)

{ if(j>=w[i])

{ vt[i][j]=max(vt[i-1][j],vt[i-1][j-w[i]]+v[i]);

}

else

{ vt[i][j]=vt[i-1][j]; } }

}

cout<<vt[n][m];

return 0; }

| **测试点** | **结果** | **耗时** | **内存** |
| --- | --- | --- | --- |
| 0 | 答案正确 | 3 ms | 248KB |
| 1 | 答案错误 | 2 ms | 252KB |
| 2 | 答案正确 | 3 ms | 252KB |
| 3 | 答案正确 | 3 ms | 248KB |

7-2部分正确 得分: 6 / 8

跳房子是小朋友玩的游戏。地面上画出一连串格子，每个格子里有一个整数，小朋友从外面跳入格子，并继续往前跳，直到跳出所有格子。每次跳跃的规则是，可以跳入下一格或下下格或下下下格。怎么跳能让落脚格子里的数的累加和最小。

**输入格式:**

第一行输入格子数 n (1<n<100)，第二行输入从起点处到终点处每个格子里的数，该数小于10。

**输出格式:**

输出最小累加和。

**输入样例:**

在这里给出一组输入。例如：

4

3 3 2 5

**输出样例:**

在这里给出相应的输出。例如：

2

#include<iostream>

#include<algorithm>

using namespace std;

int a[105];

int n, minx;

void jumphouse(int sum,int steps)

{ if(steps>=n)

{ minx=min(sum,minx);

return ; }

else

{ sum+=a[steps];

jumphouse(sum,steps+1);

jumphouse(sum,steps+2);

jumphouse(sum,steps+3); } }

int main()

{ minx=111111111;

cin>>n;

for(int i=1;i<=n;i++)

{ cin>>a[i]; }

int sum=0;

jumphouse(sum,0);

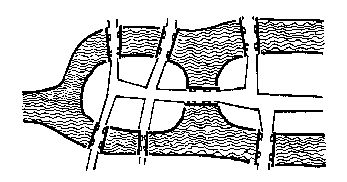
cout<<minx;

return 0; }

| **测试点** | **结果** | **耗时** | **内存** |
| --- | --- | --- | --- |
| 0 | 答案正确 | 3 ms | 248KB |
| 1 | 答案正确 | 3 ms | 252KB |
| 2 | 答案错误 | 3 ms | 256KB |
| 3 | 答案正确 | 3 ms | 248KB |

7-3答案正确 得分: 12 / 12

哥尼斯堡是位于普累格河上的一座城市，它包含两个岛屿及连接它们的七座桥，如下图所示。



可否走过这样的七座桥，而且每桥只走过一次？瑞士数学家欧拉(Leonhard Euler，1707—1783)最终解决了这个问题，并由此创立了拓扑学。

这个问题如今可以描述为判断欧拉回路是否存在的问题。欧拉回路是指不令笔离开纸面，可画过图中每条边仅一次，且可以回到起点的一条回路。现给定一个无向图，问是否存在欧拉回路？

**输入格式:**

输入第一行给出两个正整数，分别是节点数*N* (1≤*N*≤1000)和边数*M*；随后的*M*行对应*M*条边，每行给出一对正整数，分别是该条边直接连通的两个节点的编号（节点从1到*N*编号）。

**输出格式:**

若欧拉回路存在则输出1，否则输出0。

**输入样例1:**

6 10

1 2

2 3

3 1

4 5

5 6

6 4

1 4

1 6

3 4

3 6

**输出样例1:**

1

**输入样例2:**

5 8

1 2

1 3

2 3

2 4

2 5

5 3

5 4

3 4

**输出样例2:**

0

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define inf 999999999

int n,m,flag = 0;

int e[1001][1001];

int vis[1001];

int degree[1001];

void dfs(int s,int c)

{ if(flag)return;

if(c == n)

{ flag = 1;

return; }

for(int i = 1;i <= n;i ++)

{ if(vis[i] || e[s][i] == 0)continue;

vis[i] = 1;

dfs(i,c + 1);

vis[i] = 0; } }

int check()

{ for(int i = 1;i <= n;i ++)

if(degree[i]%2)return 0;

return 1; }

int main()

{ int a,b;

scanf("%d%d",&n,&m);

for(int i = 0;i < m;i ++)

{ scanf("%d%d",&a,&b);

e[a][b] = e[b][a] = 1;

degree[a] ++;

degree[b] ++;

}

vis[1] = 1;

dfs(1,1);

if(flag&&check())printf("1");

else printf("0"); }

| **测试点** | **结果** | **耗时** | **内存** |
| --- | --- | --- | --- |
| 0 | 答案正确 | 1 ms | 128KB |
| 1 | 答案正确 | 1 ms | 128KB |
| 2 | 答案正确 | 1 ms | 128KB |
| 3 | 答案正确 | 69 ms | 4092KB |
| 4 | 答案正确 | 68 ms | 4120KB |

7-4答案正确 得分: 12 / 12

将1-9九个数不重复地赋给不同的9个元素 ，实现形如a/bc+d/ef=f/hi 的形式。例：1/26+5/78=4/39 1/32+5/96=7/84 （注意：1/26+5/78=4/39 和5/78+1/26=4/39 只能算一种解），共有多少种不同的解。

**输入格式:**

无

**输出格式:**

输出整数代表解的个数。

**输入样例:**

在这里给出一组输入。例如：

无

**输出样例:**

在这里给出相应的输出。例如：

略

#include<iostream>

#include<algorithm>

#include<string.h>

#include<list>

using namespace std;

int lcd(int a,int b)

{

int r=a%b;

while(r!=0)

{

a=b;

b=r;

r=a%b;

}

return b;

}

int main()

{

int coun=0;

int a[10];

for(int i=0;i<9;i++)

{

a[i]=i+1;

}

int warn\_g[100];

int warn\_hm[200];

while(next\_permutation(a,a+9))

{

int b,c,d,e,f,g,h,m,n,nn;

b=a[0];

c=a[1]\*10+a[2];

d=a[3];

e=a[4]\*10+a[5];

f=a[6];

g=a[7]\*10+a[8];

h=e\*c;

m=b\*e+c\*d;

n=lcd(h,m);

nn=lcd(f,g);

if((m/n)==(f/nn)&&(h/n)==(g/nn))

{

coun++;

}

}

cout<<coun/2<<endl;

return 0;

}

| **测试点** | **结果** | **耗时** | **内存** |
| --- | --- | --- | --- |
| 0 | 答案正确 | 31 ms | 420KB |

7-5答案正确 得分: 14 / 14

从整数 1 至 n 中选出 m 个数，按字典序排列输出。

**输入格式:**

输入 n 和 m， 1<n, m<10。

**输出格式:**

每行输出一种组合情况，且数字由小到大排列 。每种情况不重复地按字典序输出。最后一行输出组合的总数。

**输入样例:**

在这里给出一组输入。例如：

3 2

**输出样例:**

在这里给出相应的输出。例如：

1 2

1 3

2 3

3

#include<iostream>

#include<algorithm>

#include<string.h>

using namespace std;

int n,m,num;

int a[10];

void dfs(int k,int s)

{

if(k==m)

{

cout<<a[0];

for(int i=1;i<m;i++)

{

cout<<" "<<a[i];

}

cout<<endl;

num++;

return ;

}

else

{

if(s>n)

{

return ;

}

for(int i=1;i>=0;i--)

{

if(i==1)

{

a[k]=s;

dfs(k+1,s+1);

}

else

{

dfs(k,s+1);

}

}

}

}

int main()

{

memset(a,0,sizeof(a));

num=0;

cin>>n>>m;

dfs(0,1);

cout<<num;

return 0;

}

| **测试点** | **结果** | **耗时** | **内存** |
| --- | --- | --- | --- |
| 0 | 答案正确 | 2 ms | 252KB |
| 1 | 答案正确 | 3 ms | 180KB |
| 2 | 答案正确 | 3 ms | 252KB |
| 3 | 答案正确 | 2 ms | 180KB |

7-6答案正确 得分: 16 / 16

（原题来自蓝桥杯训练题）给定一个nxn的棋盘，棋盘中有一些位置不能放皇后。现在要向棋盘中放入n个黑皇后和n个白皇后，使任意的两个黑皇后都不在同一行、同一列或同一条对角线上，任意的两个白皇后都不在同一行、同一列或同一条对角线上。问总共有多少种放法？

**输入格式:**

输入的第一行为一个整数n，1<n<9，表示棋盘的大小。接下来n行，每行n个0或1的整数，如果一个整数为1，表示对应的位置可以放皇后，如果一个整数为0，表示对应的位置不可以放皇后。

**输出格式:**

输出一个整数，表示总共有多少种放法。

**输入样例1:**

在这里给出一组输入。例如：

4

1 1 1 1

1 1 1 1

1 1 1 1

1 1 1 1

**输出样例1:**

在这里给出相应的输出。例如：

2

**输入样例2:**

在这里给出一组输入。例如：

4

1 0 1 1

1 1 1 1

1 1 1 1

1 1 1 1

**输出样例2:**

在这里给出相应的输出。例如：

0

#include<stdio.h>

#include<string.h>

#include<math.h>

#define MAXSIZE 1000

int bqueen[MAXSIZE];//黑皇后

int wqueen[MAXSIZE];//白皇后

int chessboard[MAXSIZE][MAXSIZE];//1:能放 0:不能放

int count =0;

int n;

int place(int k);

int BlackQueen(int k)

{

int i;

int j;

for(i =0; i < k -1; i++)

{

int judge = bqueen[i]- bqueen[k -1];

if(judge ==0|| fabs(k -1- i)== fabs(judge))

return 0;

}

if(k == n)

{

count++;

return 0;

}

for( j =0; j < n; j++)

{

if(j != wqueen[k]&& chessboard[k][j])

{

bqueen[k]= j;

BlackQueen(k +1);

}

}

}

int WhiteQueen(int k)

{

int i;

int j;

for( i =0; i < k -1; i++)

{

int judge = wqueen[i]- wqueen[k -1];

if(judge ==0|| fabs(k -1- i)== fabs(judge))

return 0;

}

if(k == n)

{

BlackQueen(0);

return 0;

}

for( j =0; j < n; j++)

{

if(chessboard[k][j])

{

wqueen[k]= j;

WhiteQueen(k +1);

}

}

}

int main(void)

{ int i;

int j;

scanf("%d",&n);

for(i =0; i < n; i++)

for(j =0; j < n; j++)

scanf("%d",&chessboard[i][j]);

WhiteQueen(0);

printf("%d\n",count);

return 0;

}

| **测试点** | **结果** | **耗时** | **内存** |
| --- | --- | --- | --- |
| 1 | 答案正确 | 2 ms | 128KB |
| 2 | 答案正确 | 2 ms | 256KB |
| 3 | 答案正确 | 2 ms | 128KB |
| 4 | 答案正确 | 2 ms | 132KB |
| 5 | 答案正确 | 3 ms | 128KB |
| 6 | 答案正确 | 14 ms | 128KB |
| 7 | 答案正确 | 2 ms | 128KB |
| 8 | 答案正确 | 1 ms | 128KB |

7-7答案正确 得分: 16 / 16

农夫带着一只狼、一只羊和一筐菜划船过河。农夫一次只能载一种货物过河。为了提高效率，不允许把货物A载过河后又立刻把货物A载回来。问有几种过河方案。

**输入格式:**

无

**输出格式:**

输出一个整数代表过河的方案数

**输入样例:**

在这里给出一组输入。例如：

无

**输出样例:**

在这里给出相应的输出。例如：

略

#include<iostream>

#include<algorithm>

using namespace std;

int main()

{

cout<<2;

return 0;

}

| **测试点** | **结果** | **耗时** | **内存** |
| --- | --- | --- | --- |
| 0 | 答案正确 | 2 ms | 256KB |

7-8答案正确 得分: 16 / 16

输入 n 个大于0的整数，这些整数采用任一顺序，用加减乘除四则运算符连接起来形成一个表达式。假设这四种运算符的优先级相同，即表达式从左向右计算，判读是否存在某个表达式的计算结果为0。

**输入格式:**

第一行输入 n ，1<n<10。第二行输入 n 个整数

**输出格式:**

如果存在计算结果为0的表达式，则输出"yes"，否则输出"no"。

**输入样例:**

在这里给出一组输入。例如：

3

2 3 2

**输出样例:**

因为 2-2\*3=0 （注意，假设四则运算符优先级相同），所以存在结果为0的表达式，因此输出

yes

#include <cstdio>

#include<algorithm>

#include<cstring>

#include<cmath>

#include<queue>

#include<vector>

#include<iostream>

using namespace std;

typedef long long ll;

#define inf 0x3f3f3f3f

int n;

int a[20];

const int maxn = 1e6+4;

int visit[20];

vector<int> v;

vector<char> p;

int s[]={'+','-','\*'};

int cal()

{

int ans = v[0];

for( int i = 0 ; i < 9 ; i++ )

{

switch(p[i])

{

case '+':

ans = ans+v[i+1];

break;

case '-':

ans = ans-v[i+1];

break;

case '\*':

ans = ans\*v[i+1];

break;

}

}

return ans;

}

bool judge( int k )

{

if( k == n-1 )

{

int t = cal();

if( t == 0 )

return true;

return false;

}

for( int i = 0 ; i < 3 ; i++ )

{

p.push\_back(s[i]);

bool flag = judge(k+1);

p.pop\_back();

if( flag )

return true;

}

return false;

}

bool dfs( int k )

{

if( k == n )

{

if( judge(0) )

return true;

return false;

}

for( int i = 1 ; i <= n ; i++ )

{

if( visit[i] == 0 )

{

v.push\_back(a[i]);

visit[i] = 1;

bool flag = dfs(k+1);

visit[i] = 0;

v.pop\_back();

if( flag )

return true;

}

}

return false;

}

int main()

{

cin>>n;

for( int i =1 ; i <= n ; i++ )

cin>>a[i];

memset(visit,0,sizeof visit);

if( dfs(0) )

printf("yes\n");

else printf("no\n");

return 0;

}

| **测试点** | **结果** | **耗时** | **内存** |
| --- | --- | --- | --- |
| 0 | 答案正确 | 2 ms | 248KB |
| 1 | 答案正确 | 3 ms | 256KB |
| 2 | 答案正确 | 3 ms | 288KB |
| 3 | 答案正确 | 3 ms | 252KB |
| 4 | 答案正确 | 3 ms | 252KB |
| 5 | 答案正确 | 3 ms | 288KB |