**《算法设计与分析》——第5章——回溯法作业**

**1. 利用回溯法求解0-1背包问题。**

**【子集树还是排列树】**

**子集树**

**【剪枝策略】**

**从所有的物品中选取若干将它们装进背包，在满足限制条件的情况下，获得最大的收益。用深度优先搜索子集树：对于左孩子，满足限制条件即可；对于右孩子，需当前节点的收益加上还未考察的对象的收益不超过当前最优解的收益。如果超过，则不需要搜索右子树，减去右子树。**

**【算法实现】**

**#include <stdio.h>**

**#include <stdlib.h>**

**#define N 50 //物品最大数量**

**int w[N]; //物品重量**

**int v[N]; //物品价值**

**int x[N]; //物品是是否在背包，1表示放入，0代表不放入**

**int n,c; //n物品数量，c背包容量**

**int CurWeight = 0; //当前放入背包的物品总重量**

**int CurValue = 0; //当前放入背包的物品总价值**

**int BestValue = 0; //最优值；最大价值，初始化为0**

**int BestX[N]; //最优解；1代表物品i放入背包，0代表不放入**

**void input();**

**void backtrack(int t);**

**void output();**

**int main(int argc, char\* argv[])**

**{**

**input();**

**backtrack(1);**

**output();**

**return 0;**

**}**

**void input()**

**{**

**printf("请输入背包的容量：\n");**

**scanf("%d",&c);**

**printf("请输入物品的个数：\n");**

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

**printf("请输入所有物品的重量及价值:\n");**

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

**{**

**printf("\t物品[%d]:",i);**

**scanf("%d%d",&w[i],&v[i]);**

**}**

**}**

**void output()**

**{**

**printf("\n价值最优值是:%d\n",BestValue);**

**printf("选中物品对应为：(");**

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

**printf("%d ",BestX[i]);**

**printf(")\n");**

**//输出物品内容**

**printf("选择物品重量和价值为：[");**

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

**if(BestX[i])**

**printf("(%d,%d) ",w[i],v[i]);**

**}**

**printf("]");**

**}**

**void backtrack(int t)**

**{**

**//叶子节点，输出结果**

**if(t>n)**

**{**

**//找到了更优的解,保存更优的值和解**

**if(CurValue>BestValue)**

**{**

**BestValue = CurValue;**

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

**BestX[i] = x[i];**

**}**

**}**

**else{**

**//遍历当前节点的子节点：0 不放入背包，1放入背包**

**for(int i=0; i<=1; ++i)**

**{**

**x[t]=i;**

**if(i==0) //不放入背包**

**backtrack(t+1);**

**else //放入背包**

**{**

**//约束条件：放的下**

**if((CurWeight+w[t])<=c)**

**{**

**CurWeight += w[t];**

**CurValue += v[t];**

**backtrack(t+1);**

**CurWeight -= w[t];**

**CurValue -= v[t];**

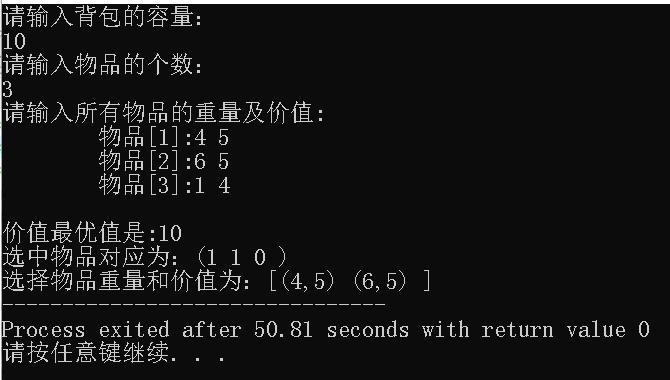
**}**

**}**

**}**

**}**

**}**



**2. 素数环。1到n的整数围成一圈，每相邻两个整数相加结果为素数。输入n，输出解的个数及元素的排列。（假设素数环均从1开始）**

**【子集树还是排列树】**

**排列树**

**【剪枝策略】**

每个环都从1开始，先将数组a[0]赋值1；每选定前一个素数，后一个位置就少一个可选择项，每个节点有2个状态，由此可用一个数组visited[]来标记状态，1表示被占用，0表示没被占用；前一个后一个选定值总和前一个选定值关联，由此可用回溯法(深度优先遍历的方式遍历解答树)。

递归剪枝:若当前值没被选中过且符合条件,则选定该值,跳到下一层,否则返回；最后一层结束后执行输出，然后从后往前还原每个节点状态；首层又从第2个值开始重复操作。

**【算法实现】**

**#include<stdio.h>**

**#include<math.h>**

**#define N 20 //最大整数**

**int a[20]; //存放n个整数**

**int visited[20]={0}; //1表示被占用，0表示未被占用**

**int n,count; //n为实际整数个数，count为符合条件的排列**

**int isPrime(int a){**

**if(a<=1) return 0;**

**if(a==2||a==3) return 1;**

**for(int i=2;i<=sqrt(a);i++){**

**if(a%i==0)**

**return 0;**

**}**

**return 1;**

**}**

**//递归输出所有素数环**

**void DFS(int step){**

**//若step到底，输出**

**if(step==n&&isPrime(a[0]+a[n-1])){**

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

**printf("%d ",a[i]);**

**count++;**

**printf("\n");**

**return ;**

**}**

**//递归选定每一层的整数**

**else for (int i=2;i<= n;i++){**

**if(!visited[i]&&isPrime(i+a[step-1])){ //当前值i没被使用,且与前一个选定值之和为素数**

**a[step]=i; //选i为当前项值**

**visited[i]=1; //改变状态**

**DFS(step+1); //进入下一层**

**visited[i]=0; //状态还原**

**}**

**}**

**}**

**int main(){**

**printf("请输入整数n:\n");**

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

**for(int i=0;i<20;i++) //输入要排列的整数**

**a[i]=i+1;**

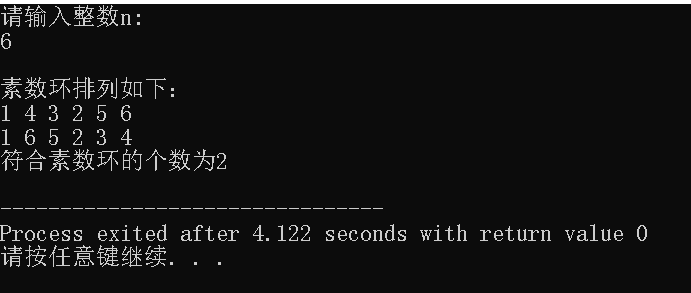
**printf("\n素数环排列如下：\n");**

**DFS(1); //回溯法遍历解答树,输出所有素数环**

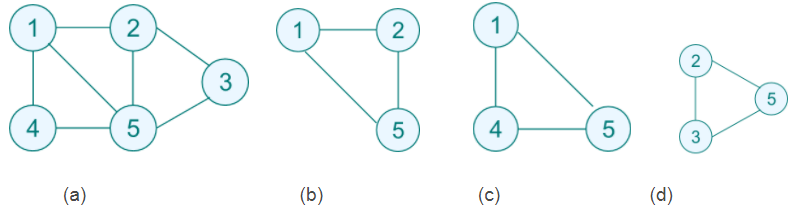
**printf("符合素数环的个数为%d\n",count);**

**return 0;**

**}**



**3. 最大团问题。团就是最大完全子图，最大团就是最大完全子图。如图1所示，图(a)有3个最大团，分别为图(b)(c)(d)。对一个图求其最大团（输出顶点编号），并统计其个数。**

****

**【子集树还是排列树】**

**子集树**

**【剪枝策略】**

最大团是从无向图的顶点集中选出k个并且k个顶点之间任意两点之间都相邻（完全图）；最大的k就是最大团x[i]存储问题的解。b[n][n]存储图的邻接矩阵，b[i][j]=1表示有相连，0表示无连接。

-约束函数 顶点i到已选入的顶点集中每一个顶点都有边相连。

-限界函数 有足够多的可选择顶点使得算法有可能在右子树中找到更大的团

-约束条件：①xi!=xj (0<=i,j<=n,i!=j）②b[1][n]=1。

**【算法实现】**

**#include<stdio.h>**

**#include<stdlib.h>**

**#define N 20 //最大节点数**

**int n; //实际节点数**

**int x[20]; //1代表第i个节点被占用，0未被占用**

**int visited[20]; //最大团里的节点选择情况**

**int b[20][20]; //邻接矩阵**

**int count=0,bestcount=0; //最大团里的节点个数以及最优解**

**void input(){**

**printf("请输入点数个数（默认记为1-n）:\n");**

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

**printf("请输入连接矩阵（i点和j点相连为1，不相连为0）：\n");**

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

**for(int j=1;j<=n;j++)**

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

**}**

**void output(){**

**printf("\n最大团个数为：%d\n节点包括：[",bestcount);**

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

**{**

**if(visited[i]==1)**

**{**

**printf("%d ",i);**

**}**

**}**

**printf("]\n");**

**}**

**void DFS(int step)**

**{**

**if(step>n) //抵达叶子节点输出**

**{**

**bestcount=count;**

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

**visited[i]=x[i];**

**return;**

**}**

**else**

**{**

**int sign=1; //标志位，sign=1代表遍历左子树(i加入已选择的点集)**

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

**{**

**if((x[i]==1)&&(b[step][i]==0))**

**{**

**sign=0;**

**break;**

**}**

**}**

**if(sign==1)//说明第i个点可以被加进去,遍历左子树**

**{**

**x[step]=1;**

**count++;**

**DFS(step+1);**

**count--;//回溯到原点**

**x[step]=0;**

**}**

**else //说明第i个点不能被加进去,遍历右子树**

**{**

**x[step]=0;**

**if(count+n-step>bestcount)**

**DFS(step+1);**

**}**

**}**

**}**

**int main()**

**{**

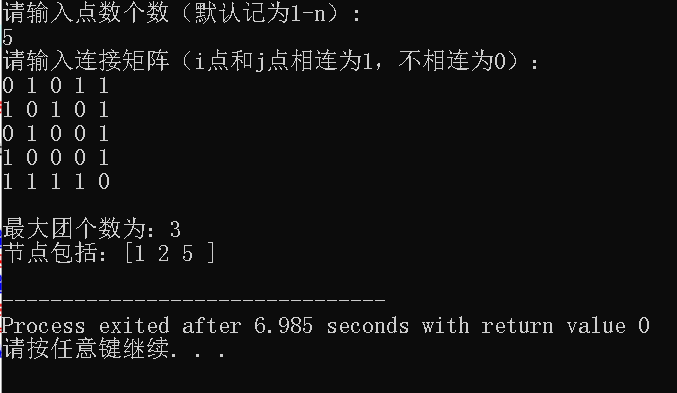
**input();**

**DFS(1);**

**output();**

**return 0;**

**}**

****