一、01 背包问题：有 n 件物品和一个容量为 c 的背包。第 i 件物品的价值是 v[i]，重量是 w[i]。求解将哪些物品装入背包可使价值总和最大。所谓 01 背包，表示每一个物品只有一个，要么装入，要么不装入。而且物品的价值和重量已及背包容量都是实数。假设给定一个问题实例如下：n=4，物品重量为：2，2，6，9；物品价值为 10，10，12，18；背包最大承重为 15。

试设计求解 0-1 背包问题的分枝限界算法（编写伪代码）（6 分），并针对上述实例列出你的算法求解过程中所生成的状态空间树（3 分）；

答：

（1） 分枝限界算法：

在搜索状态空间树时，只要左子节点是一个可行结点，搜索就进入其左子树。对于右子树时，先计算限界函数，以判断是否将其减去， 限界函数 bound():当前价值+剩余容量可容纳的最大价值<=当前最优价值。

w[i]:第 i 个物体的重量；v[i]:第 i 个物体的价值；put[i]:第 i 个物体是否放入背包

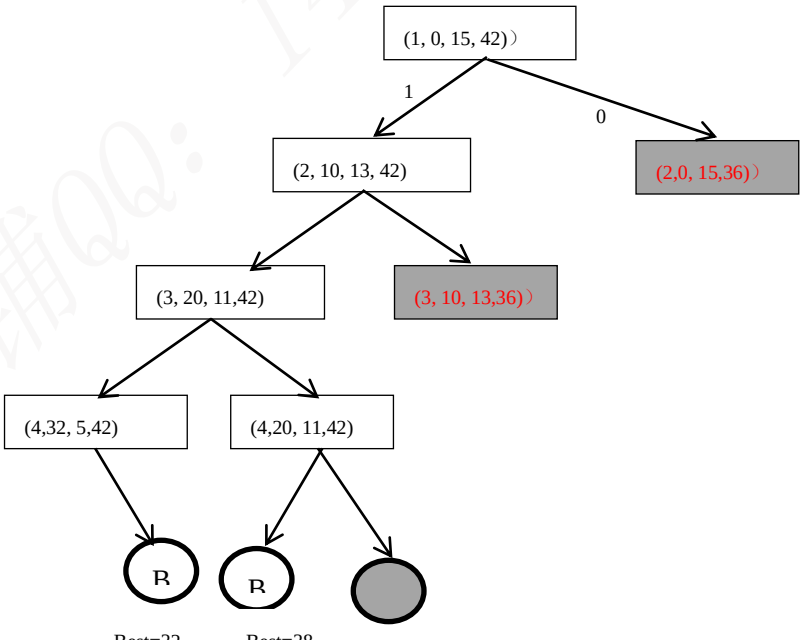
```
double cw = 0.0;//当前背包重量
double cp = 0.0;//当前背包中物品价值
double bestp = 0.0;//当前最优价值        （1 分）
void backtrack(int i)     //  （3 分）
 {
   double bound(int i);//bestp=bound(i);
   if(i>n) //计算到最后一个物体，退出
   {
     bestp = cp;
     return;
   }
   if(cw+w[i]<=c)   //加入当前物体后，当前容量小于背包承受总容量
   {
     cw+=w[i];
     cp+=v[i];
     put[i]=1;
     backtrack(i+1);   //递归调用，判断下一个物体是否放入背包
     cw-=w[i];
     cp-=v[i];
   }
```

```
    if(bound(i+1)>bestp)//符合条件搜索右子树
        backtrack(i+1);
}

//计算限界函数        （2 分）
double bound(int i)
{
    double leftw= c-cw; //剩余背包承重
    double b = cp;    //当前背包价值
//物品还未放完，且下一个物品重量小于背包剩余承重，则循环添加至背包
    while(i<=n&&w[i]<=leftw)
    {
        leftw-=w[i];
        b+=v[i];
        i++;
    }
//如果剩余空间不能放满，但是计算剩余空间的预期价值
    if(i<=n)
        b+=v[i]/w[i]*leftw;
    return b; //预期效益值
}


int main()
{
        Qsort (); //按照单位效益值从大到小排序
        backtrack(1);
         ourput;
}
```

方框中的参数分别表示：考虑的物品编号、已经获得的价值、背包剩余承重、预期价值。



计算说明：

重量 W[i]	2	2	6	9
价值 V[i]	10	10	12	18

（1）初始状态：（1,0,15,42）

当前价值 0，剩余空间 15 中最多可以装预期价值：

10+10+12+（15-（2+2+6））*18/9=32+5*2=42；

（2）第 1 个物体装入：（2,10,13,42）

当前价值 10，预期价值：当前价值+剩余空间 13 中最多价值：

10+10+12+（13-（2+6））*18/9=32+5*2=42；

（3）第 1 个物体不装入：（2,0,15,36）

当前价值 0，预期价值：当前价值+剩余空间 13 中最多价值：

0+10+12+（15-（2+6））*18/9=22+7*2=36；

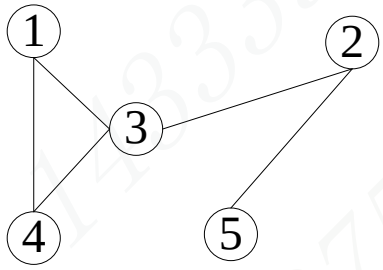
二、最大团问题：给定无向图 G=(V,E)，其中 V 是非空集合，称为顶点集；E 是 V 中元素构成的无序二元组的集合，称为边集，无向图中的边均是顶点的无序对，无序对常用圆括号“( )”表示。

如果给定 U⊆V，且对任意两个顶点 u，v∈U 有(u,v)∈E，则称 U 是 G 的完全子图。G 的完全子图 U 是 G 的团当且仅当 U 不包含在 G 的更大的完全子图中。G 的最大团是指 G 中所含顶点数最多的团。

（1）请设计一个回溯算法（伪代码即可）来求解最大团问题；（10 分）

（2）你设计的算法的解向量如何表示？时间复杂度是多少？（5 分）

（3）假设有如下下图所示的问题实例，



试采用你设计的算法，把求解最大团的搜索过程详细写出来。（5 分）

参考答案：注意, 本题会有多种解法, 参考答案仅仅是一种

（1）假设解向量采用等长的二进制编码（$x_1,x_2,\ldots,x_n$），其中 n 为图中顶点的个数，回溯算法的递归版本代码如下：

Input: An undirected graph G=(V,E).

Output: A solution vector x[1,2,…,n].

1. **for** *k*←*1* **to** *n*

2. y[k]=x*[k]*←*-1*

3. **end for**

4. *mcl*←-1

5. maxcl(1,0,n)

6. **output** y　　//y：最大团的顶点集合

7. **output** mcl　//mcl：是否找到最大团

// maxcl 函数说明:递归函数，判断第 k 个顶点加入或不加入最大团的计算；

函数参数说明：k:第 k 个点，r：目前的最大团顶点数；l：剩余的顶点数；

Procedure maxcl (k,r,l)

1. **x(k) = 0**　　**//不取 K 点加入**
2. **if** k=n **then**
3. 　　**if** r>mcl **then** {mcl = r, y=x}　**endif**
4. **else if** r+l >mcl **then** maxcl(k+1,r,l-1)
5. **endif**
6. **x(k) =1**　　**//取 K 点加入**
7. **if** 节点 k 与前面取值为 1 的节点均有边相连 **then**
8. 　　**if** k=n **then**
9. 　　**if** r>mcl **then** {mcl = r, y=x} **endif**
10. 　　**else** maxcl(k+1,r+1,l-1)
11. **endif**
12. **end if**

----------得分 10 分

（2）解向量采用等长的二进制编码（$x_1,x_2,\ldots,x_n$），其中 n 为图中顶点的个数。

----------得分 2 分

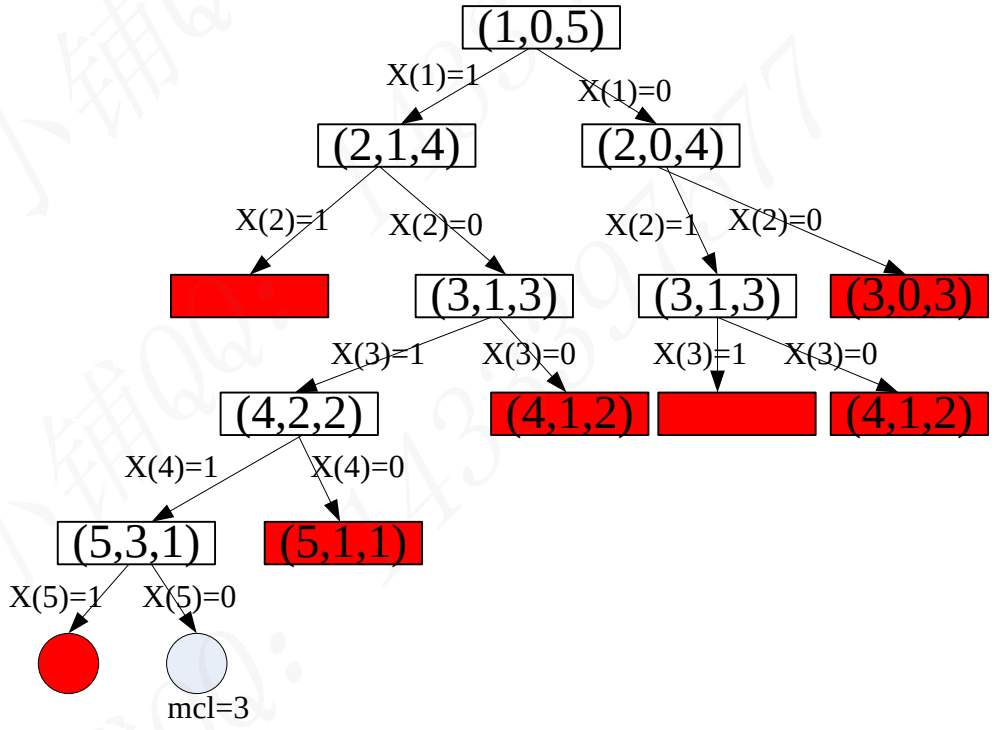
1）图中 n 个顶点，每个顶点选择加还是不加，最多做 2^n 次判断，计算次数为 2^n ；

2）每次判断需要当前点与其他所有的点进行判断，是否连接，计算量为 n；

所以，算法的时间复杂度为 $O(n2^n)$。

----------得分 3 分

（3）求解过程如下图所示（由于先选取 x(k)=0 的节点先生成，本实例造成的树太大，所以我们先生成 x(k)=1 的节点，不管那种做法，答案都算对），其中红色无字方框是不满足要求的中间节点，红色有字方框为被限界的中间节点，红色圆形为不满足要求的解，灰色圆形为满足要求的解。



----------得分 5 分