衡量算法时间效率的方法有哪两种？

答：有事前分析法和事后分析法两种。

事后分析法：先将算法用程序设计语言实现，然后度量程序的运行时间。 事前分析法：算法的时间效率是问题规模的函数，假如，随着问题规模n的增长，算法执行时间的增长率和函数f(n)的增长率相同，则可记作：     T(n)=○(f(n)) 称为算法的时间复杂度。

对下列各组函数f (n) 和g (n)，确定f (n) = O (g (n)) 或f (n) =Ω(g (n))或f(n) =θ(g(n))，并简要说明理由。

(1) f(n)=2^n；        g(n)=n!

f(n) = O(g(n)) 因为g(n)的阶比f(n)的阶高。

(2) f(n)=n；       g (n)=log n^2

f(n) = Ω(g(n)) 因为g(n)的阶比f(n)的阶低。

(3) f(n)=100；       g(n)=log100

f(n) = θ(g(n)) 因为g(n)与f(n)同阶。

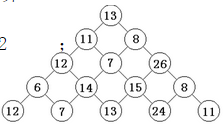
(4) f(n)=n^3；        g(n)= 3^n

f(n) = O(g(n)) 因为g(n)的阶比f(n)的阶高。

(5) f(n)=3^n；       g(n)=2^n

f(n) = Ω(g(n)) 因为g(n)的阶比f(n)的阶低。

数塔问题。从顶部出发，在每一结点可选择向左走或是向右走，一直走到底层，要求找出一条路径，使路径上的值最大。



for(r=n-2;r>=0;r--) //自底向上递归计算

 for(c=0;  c<=r;  c++)

if( t[r+1][c]>t[r+1][c+1])   t[r][c]+=t[r+1][c] ；

else  t[r][c]+=t[r+1][c+1]  ；

Hanoi算法 Hanoi(n,a,b,c)

if （n==1）  move(a,c)  ； else

{ Hanoi(n-1, a, c , b)  ；   Move(a,c)   ；

Hanoi(n-1,b, a, c); }

计算x^m的值的过程 ：

int  power ( x, m )

{//计算x^m的值并返回。

y=**( 1  )**;i=m;

While**(i- - >0)**

y=y\*x;

**(return y)** }

排列问题

Template <class Type>

void perm(Type list[],  int k, int m ) {

//产生[list[k:m]的所有排列

if(**k==m**)

     {  //只剩下一个元素

         for (int i=0;i<=m;i++)  cout<<list[i];          cout<<endl;

else

//还有多个元素待排列，递归产生排列

for (**int i=k; i<=m; i++**)         {

           swap(list[k]，list[i]);

**perm(list,k+1;m)**;

swap(list[k],list[i]);

} }

编写一个算法，可以检测一个字符串是否回文（如：afaddafa，abwba等）。

int fun(char \*A,int n){

int i ,j;

i=0,j=n-1;

while(i<=j){

      if(A[i]!=A[j])break;

 i++;j--;    }

if(i<=j)return 0;

else return 1; }

检测一个字符串中是否包含一个回文子串：

int judge(char \*A,int n0, int n1){

int i,j;

i=n0;j=n1;

while(i<=j){

       if(A[i]!=A[j])break;

i++;j--;    }

   if(i<=j)return 0;

else return 1; }

int fun(char \*A,int n){

int i,j,k;

   for(i=1,i<=n-1;i--){

for(j=0;j<n-i;j++){

         if(judge(&A,j,j+i)==1){output(&A,j,j+i);return 1;}

}    }

   return 0; }

找最大回文：此时最先找到的即为最大回文

int fun(char \*A,int n){

int i,j,k;

for(i=n-1;i>=1;i--){

      for(j=0;j<=n-i-1;j++){

          if(fun(&A,j,i+j)==1){output(&A,j,i+j);return 1;}

 }    }

   return 0; }

在序列A[1..n]中找最大最小元素的问题。一个分治算法描述如下：如果n≤2 就直接求解。否则，将序列等分成两个子序列A[1..n/2]和A[n/2+1..n]，分别找出这两子

序列的最大最小元素x1,y1 和x2,y2；然后据此求出A[1..n]的最大元素x=max{x1,x2}及最小元素y=min{y1,y2}。请给出该算法计算时间T(n)满足的递归方程，并解方程来确定算法的时间复杂度。

假定n=2k（k 为正整数）。

算法时间复杂度满足如下递归方程：

*T*(*n*)=2*T*(*n*/2)+2（*n*>2）；*T*(2)=1。

因为 *n*=2 *k*（*k* 为正整数），所以，

*T*(*n*)= *T*(2 *k*)= 2T(2 *k-1*)+2= 22T(2 *k-2*)+ 22+2

⋯

= 2k-1T(2)+ 2k-2+⋯+23+22+2

= 2k-1+⋯+23+22+2。因此，*T*(*n*)=(*n*)。

写出maxmin算法对下列实例中找最大数和最小数的过程。 数组 A=(48,12,61,3,5,19,32,7)

参考解答：写出maxmin算法对下列实例中找最大数和最小数的过程。

数组 A=()

1、 48,12,61,3,       5,19,32,7

2、 48,12    61,3    5,19     32,7

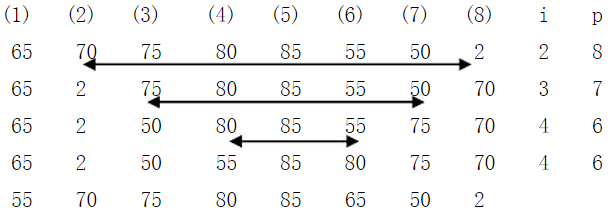
3、 48～61, 12～3   19～32，5～7

4、 61～32    3～5

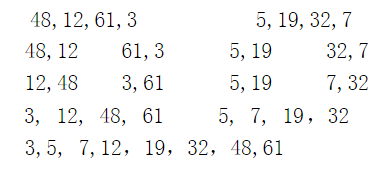
5、 61   3

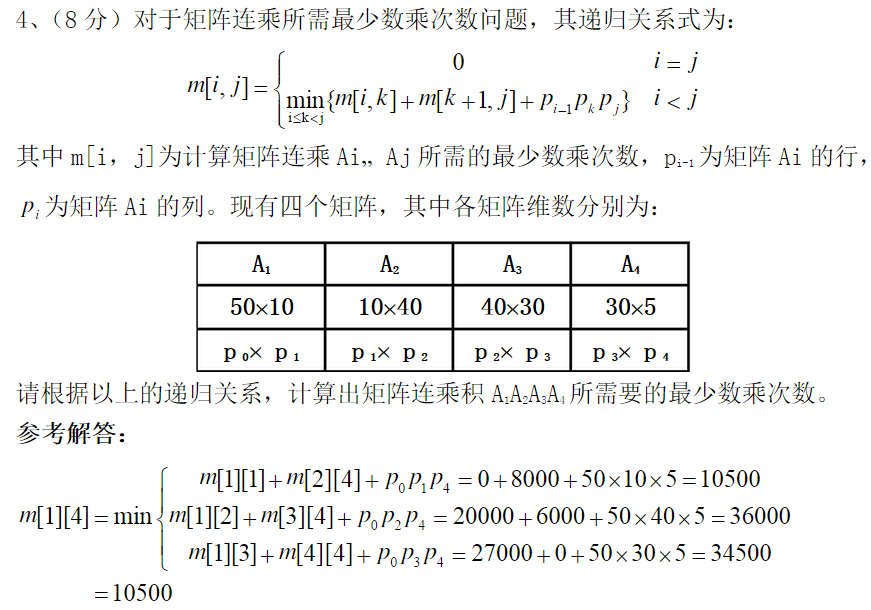
速排序算法对下列实例排序，算法执行过程中，写出数组A第一次被分割的过程。  A=(65,70,75,80,85,55,50,2)

答：第一个分割元素为65



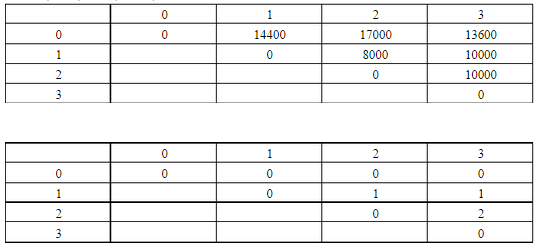
归并排序算法对下列实例排序，写出算法执行过程。        A=(48,12,61,3,5,19,32,7)





有4个矩阵连乘积ABCD，设它们的维数分别为A:45\*8,B:8\*40,C:40\*25,D:25\*10,请求出它们的最优计算次序和计算量

解： P0=45,P1=8,P2=40,P3=25,P4=10



m[0,1]=min{m[0,0]+m[1,1]+P0P1P2}=14400

m[1,2]=8000

m[2,3]=10000

m[0,2]=min{m[0,0]+m[1,2]+P0P1P3,m[0,1]+m[2,2]+P0P2P3}=17000

m[1,3]=min{m[1,1]+m[2,3]+P1P2P4,m[1,2]+m[3,3]+P1P2P4}=10000

m[0,3]=min{m[0,0]+m[1,3]+P0P1P4,m[0,1]+m[2,3]+P0P2P4,m[0,2]+m[3,3]+P0P3P4}=13600

最优计算次序：A(B(CD)) 计算量为：13600

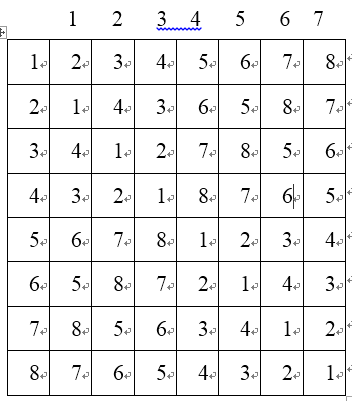
设有n=2k个运动员要进行循环赛，现设计一个满足以下要求的比赛日程表：

1. 每个选手必须与其他n-1名选手比赛各一次；
2. 每个选手一天至多只能赛一次；

③循环赛要在最短时间内完成。

（1）如果n=2^k，循环赛最少需要进行几天？ (n-1)天

（2）当n=8时，请画出循环赛日程表。



使用动态规划方法求解:0-1背包数据如下表，

求：能够放入背包的最有价值的物品集合。

|  |  |  |  |
| --- | --- | --- | --- |
| 物品  i | 重量 wi | 价值 vi | 承重量 W |
| 1 | w1=2 | v1=12 | W=5 |
| 2 | w2=1 | v2=10 |
| 3 | w3=3 | v3=20 |
| 4 | w4=2 | v4=15 |

如设： V(i, j) —— 前 i 个物品中能够装入承重量 j 的背包中的最大总价值。请将如下递推式填写完整：

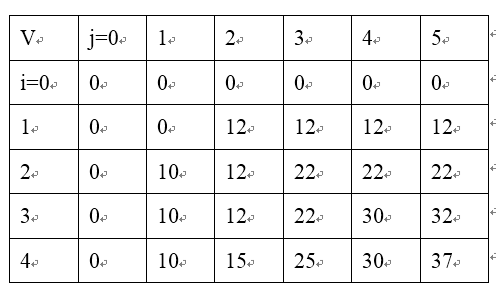
V(0, j) = 0（0个物品），V(i, 0) = 0（承重量0）

V(i, j) = V(i-1, j) 第 i 个物品不能装入,

j < wi （超重）

V(i, j) = max { vi + V(i-1,j-wj)(i在最优子集中), V(i-1, j) (i不在最优子集中)} j > wi （不超重）

自底向上：按行或列填写下表。



动态规划算法与贪心算法的异同。

共同点：

都需要最优子结构性质， 都用来求有优化问题。

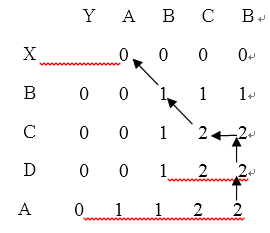
不同点：

动态规划：每一步作一个选择—依赖于子问题的解。    贪心方法：每一步作一个选择—不依赖于子问题的解。

动态规划方法的条件：子问题的重叠性质。 可用贪心方法的条件：最优子结构性质；贪心选择性质。

动态规划：自底向上求解；     贪心方法： 自顶向下求解。

可用贪心法时，动态规划方法可能不适用；  可用动态规划方法时，贪心法可能不适用。

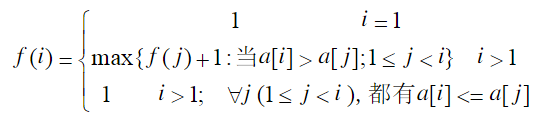


最长公共子序列：｛ＢＣ｝

【最长上升子序列问题】动态规划算法

对于给定的一个序列(a1,a2,…,an),1≤n≤1000，我们可以得到一些递增上升的子序列(ai1,ai2,…,aik),对于给定的序列，求出最长上升子序列的长度。要求写出你设计的算法思想及递推函数的公式表达。.

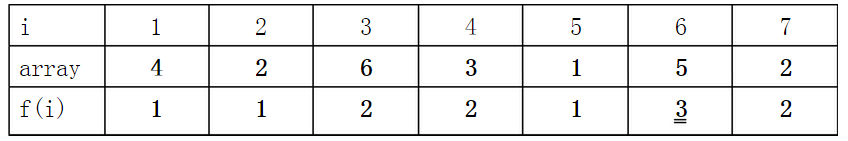
参考解答：设f(i)表示：从左向右扫描过来直到以a[i]元素结尾的序列，获得的最长上升子序列的长度，且子序列包含a[i]元素（1≤i≤n）。



即，f(i)是从f(1)，f(2)„„到f(i-1)中找最大的一个值，再加1。或者就是1。主要是看a[i]这个元素能否加入到之前已经获得的最长上升子序列，如果能加入，是之前已获得的最长上升子序列长度加一；如果不能加入，就取这最后一个元素作为一个单独子序列，长度为1。

最后，所要求的整个序列的最长公共子序列长度为max{f(i): 1≤i≤n}

例如，对于序列：4  2  6  3  1  5  2



有这样一类特殊0-1背包问题：可选物品重量越轻的物品价值越高。 n=6，c=20，P=（4，8，15，1，6，3），W=（5，3，2，10，4，8）。 其中n为物品个数，c为背包载重量，P表示物品的价值，W表示物品的重量。请问对于此0-1背包问题，如何选择放进去的物品，使放进背包的物品总价值最大，能获得的最大总价值多少？

答：因为该0－1背包问题比较特殊，恰好重量越轻的物品价值越高，所以优先取重量轻的物品放进背包。最终可以把重量分别为2，3，4，5的三个物品放进背包，得到的价值和为15 + 8 + 6 + 4 = 33，为最大值。

【Gray码构造问题】分治递归算法：“格雷码”是一个长度为n2的序列，满足：

（a）每个元素都是长度为n比特的串

（b）序列中无相同元素

（c）连续的两个元素恰好只有1个比特不同 例如：n=2时，格雷码为{00，01，11，10}。

Gray码是一种编码，这种编码可以避免在读取时，因各数据位时序上的差异造成的误读。格雷码在工程上有广泛应用。但格雷码不便于运算，请你设计一种构造方法，输入长度序列n，输出格雷码（你只要做出一种构造方案即可，格雷码并不唯一）。

答：此题可用分治法解决。

当n＝1时，输出格雷码{0, 1}

当n>1时，格雷码的长度为n2，即共有n2个码序列。此时，将问题一分为二，即上半部分和下半部分。上半部分最高位设为0，下半部分最高位设为1。剩下n-1位的格雷码的构造采用递归的思路。

某体育馆有一羽毛球场出租，现在总共有10位客户申请租用此羽毛球场，每个客户所租用的时间单元如下表所示，s(i)表示开始租用时刻，f(i)表示结束租用时刻，10个客户的申请如下表所示：

i  1 2 3 4 5 6 7 8 9 10

s(i)  0 3 1 5 3 5 11 8 8 6

f(i)  6 5 4 9 8 7 13 12 11 10

同一时刻，该羽毛球场只能租借给一位客户，请设计一个租用安排方案，在这10位客户里面，使得体育馆能尽可能满足多位客户的需求，并算出针对上表的10个客户申请，最多可以安排几位客户申请。

参考解答：将这10位客户的申请按照结束时间f(i)递增排序，如下表：

i  1 2 3 4 5 6 7 8 9 10

s(i) 1 3 0 5 3 5 6 8 8 11

f(i) 4 5 6 7 8 9 10 11 12 13

(1)选择申请1（1,4）

⑵依次检查后续客户申请，只要与已选择的申请相容不冲突，则选择该申请。直到所有申请检查完毕。申请4（5,7）、申请8（8,11）、申请10（11,13） ⑶最后，可以满足：申请1（1,4）、申请4（5,7）、申请8（8,11）、申请10（11,13）共4个客户申请。这已经是可以满足的最大客户人数。

通过键盘输入一个高精度的正整数n(n的有效位数≤240)，去掉其中任意s个数字后，剩下的数字按原左右次序将组成一个新的正整数。编程对给定的n 和s，寻找一种方案，使得剩下的数字组成的新数最小。

【样例输入】 178543 S=4 【样例输出】 13

为了尽可能地逼近目标，我们选取的贪心策略为：每一步总是选择一个使剩下的数最小的数字删去，即按高位到低位的顺序搜索，若各位数字递增，则删除最后一个数字，否则删除第一个递减区间的首字符。然后回到串首，按上述规则再删除下一个数字。重复以上过程s次，剩下的数字串便是问题的解了。

具体算法如下：

 输入s, n;

while（ s > 0 ）

{  i=1;  //从串首开始找

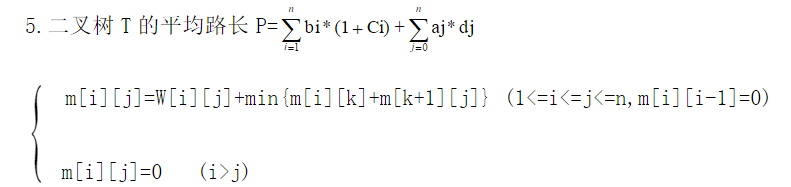
while (i < length(n)) && (n[i]<n[i+1])  {i++;}

delete(n,i,1); //删除字符串n的第i个字符 s--; }

while (length(n)>1)&& (n[1]=„0‟)

delete(n,1,1); //删去串首可能产生的无用零 输出n;

设S=｛X1，X2，···，Xn｝是严格递增的有序集，利用二叉树的结点来存储S中的元素，在表示S的二叉搜索树中搜索一个元素X，返回的结果有两种情形，（1）在二叉搜索树的内结点中找到X=Xi，其概率为bi。（2）在二叉搜索树的叶结点中确定X∈（Xi，Xi+1），其概率为ai。在表示S的二叉搜索树T中，设存储元素Xi的结点深度为Ci；叶结点（Xi，Xi+1）的结点深度为di，则二叉搜索树T的平均路长p为多少？假设二叉搜索树T[i][j]=｛Xi，Xi+1，···，Xj｝最优值为m[i][j]，W[i][j]= ai-1+bi+···+bj+aj，则m[i][j](1<=i<=j<=n)递归关系表达式是什么？



设计**找零问题**的贪心算法?       并分析其时间复杂度？

答：void  greedy\_zhaoling ( float GZ, int B[ ], int S[ ] )

//GZ应发工资                       {

B[ j]初始化排序； //为了贪心选择，依次选最大币种

for( j=1, j<=6;j++)

   {  S[ j]=0；                     //初始化S[ j]

A=GZ/B[ j];                  //A表示对应j币种张数

S[ j]=A;                //S[ j]存放对应j币种总张数

GZ=GZ-A\*B[ j];  }

//每求出一种面额所需的张数后， 一定要把这部分金额减去：        for(i=1;i<=6;i++)

        print( B[ i], “----”, S[ i]);     //输出币种和对应张数   }

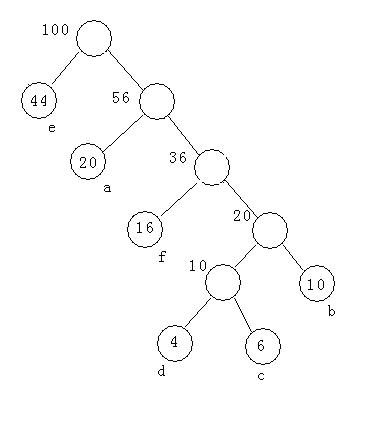
考虑用哈夫曼算法来找字符a,b,c,d,e,f 的最优编码。这些字符出现在文件中的频数之比为 20:10:6:4:44:16。

(1)简述使用哈夫曼算法构造最优编码的基本步骤；

哈夫曼算法是构造最优编码树的贪心算法。其基本思想是，首先所有字符对应n 棵树构成的森林，每棵树只有一个结点，根权为对应字符的频率。然后，重复下列过程n-1 次：将森林中的根权最小的两棵树进行合并产生一个新树，该新树根的两个子树分别是参与合并的两棵子树，根权为两个子树根权之和。

(2)构造对应的哈夫曼树，并据此给出a,b,c,d,e,f 的一种最优编码。

构造哈夫曼树如下图所示。



由此可以得出 a,b,c,d,e,f 的一组最优编码：01,0000,00010,00011, 1,001。

Dijkstra算法求单源最短路径

d[u]:s到u的距离   p[u]:记录前一节点信息

Init-single-source(G,s)

for each vertex v∈V[G]

do { d[v]=∞;   p[v]=NIL  }

d[s]=0  Relax(u,v,w)

if d[v]>d[u]+w(u,v)

then { d[v]=d[u]+w[u,v];

  p[v]=u    }

dijkstra(G,w,s)

1. Init-single-source(G,s)

2. S=Φ     3. Q=V[G]

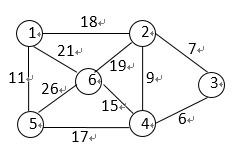
4.while Q<> Φ

do u=min(Q)            S=S∪{u}

         for each vertex    v∈adj[u]

do    Relax(u,v,w)

对下图所示的连通网络G，用克鲁斯卡尔(Kruskal)算法求G的最小生成树T,请写出在算法执行过程中，依次加入T的边集TE中的边。说明该算法的贪心策略和算法的基本思想，并简要分析算法的时间复杂度。



TE={(3,4), (2,3),(1,5),（4,6）（4,5）}

贪心策略是每次都在连接两个不同连通分量的边中选权值最小的边。

基本思想：首先将图中所有顶点都放到生成树中，然后每次都在连接两个不同连通分量的边中选权值最小的边，将其放入生成树中，直到生成树中有n-1条边。时间复杂度为：O(eloge)

n后问题回溯算法

(1)用二维数组A[N][N]存储皇后位置,若第i行第j列放有皇后,则A[i][j]为非0值,否则值为0。

(2)分别用一维数组M[N]、L[2\*N-1]、R[2\*N-1]表示竖列、左斜线、右斜线是否放有棋子，有则值为1,否则值为0。

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

 if(  !M[j]&&!L[i+j]&&!R[i-j+N]      )  /\*安全检查\*/

 { A[i][j]=i+1;    /\*放皇后\*/

   M[j]=L[i+j]=R[i-j+N]=1;

      if(i==N-1)   输出结果；

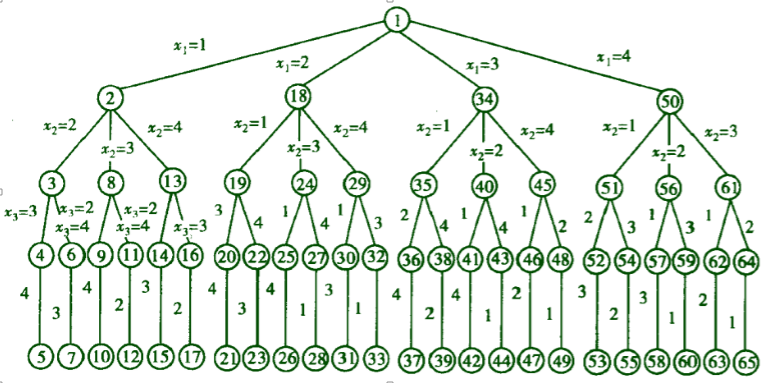
      else  try(i+1,M,L,R,A) ；;  /\*试探下一行\*/

   A[i][j]=0  ；      /\*去皇后\*/

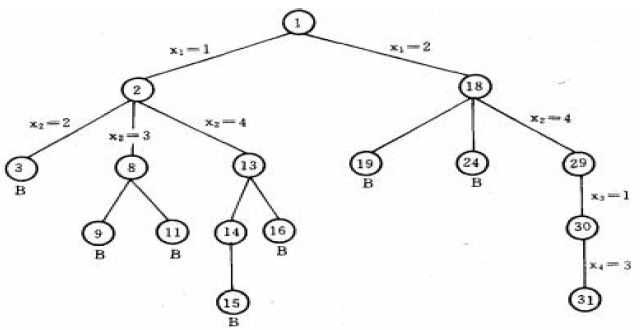
   M[j]=L[i+j]=R[i-j+N]=0 ；;      }

**请画出**用回溯法解**4皇后问题的解空间树和搜索空间树：**

解空间树：



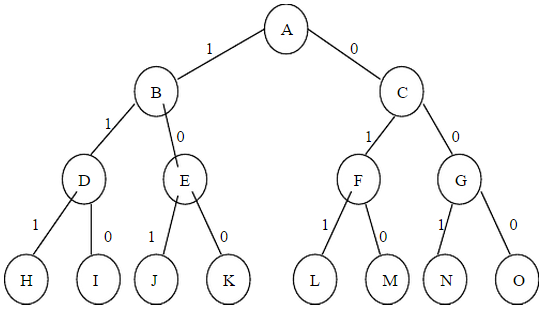
用回溯法的搜索空间树：



使用回溯法解0/1背包问题：n=3，C=9，V={6,10,3}，W={3,4,4},其解空间有长度为3的0-1向量组成，要求用一棵完全二叉树表示其解空间（从根出发，左1右0），并画出其解空间树，计算其最优值及最优解。

解空间为：{(0,0,0),(0,1,0),(0,0,1),(1,0,0),(0,1,1),(1,0,1), (1,1,0),(1,1,1)}

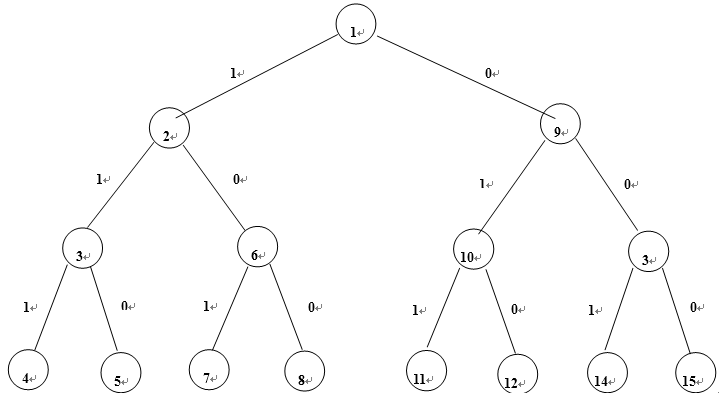
解空间树为：



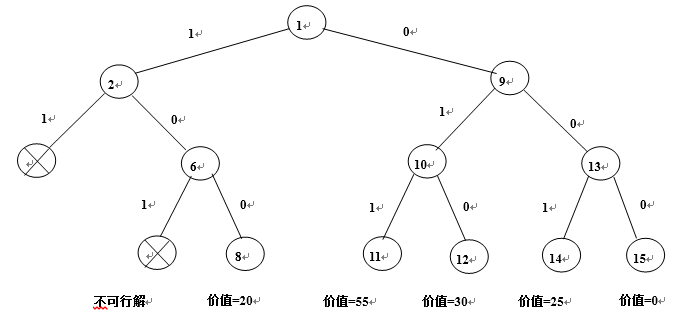
该问题的最优值为：16     最优解为：（1，1，0）

请画出用回溯法解n=3的0-1背包问题的解空间树和当三个物品的重量为{20, 15, 10}，价值为{20, 30, 25}，背包容量为25时搜索空间树。

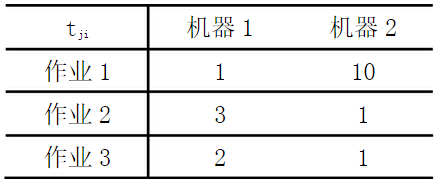
解空间树:



搜索空间树：

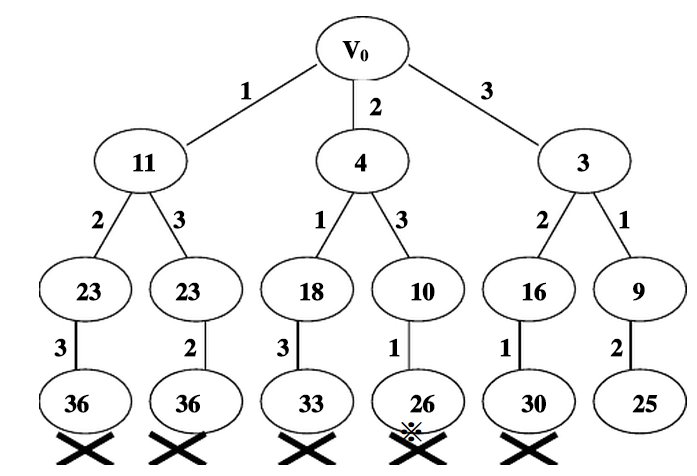


考虑n=3的批处理作业调度实例：



其中tji是作业Ji需要在机器j上处理的时间。对于给定的3个作业，制定一个最佳作业调度方案，使其完成时间和达到最小。 要求：

1. 画出该问题的解空间树；



1. 写出该问题的剪枝策略(即限界条件),要求只保留第一个最优解;

若当前代价f >= 当前最优解代价bestf，则剪枝

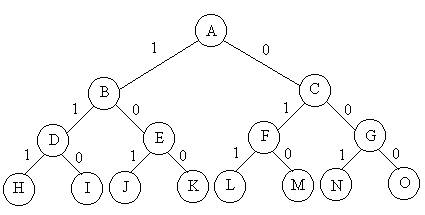
（3）按优先队列式分支限界法搜索解空间树，并用剪枝策略对解空间树中该剪枝的位置打×；

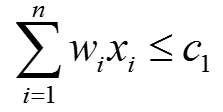
（4）给出最优解及最优值。

最优解为{3，1，2}，最优值为25。

**用分支限界法解0-1背包问题:**给定n种物品和一背包。物品i的重量是wi，其价值为vi，背包的容量为C。问应如何选择装入背包的物品，使得装入背包中物品的总价值最大?示例：n=3, C=30, w={16, 15, 15}, v={45, 25, 25}

**求：**1、问题的解空间树

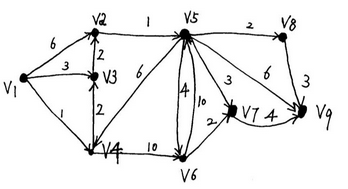


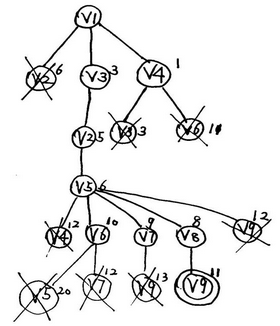
2、约束条件

3、如何剪枝？

设r是当前尚未考虑的剩余物品价值总和；Cv是当前价值；bestv是当前最优价值。当r＋Cv≤bestv时，可剪去右子树。

根据优先队列式分支限界法，求下图中从v1点到v9点的单源最短路径，请画出求得最优解的解空间树。要求中间被舍弃的结点用×标记，获得中间解的结点用单圆圈○框起，最优解用双圆圈◎框起。





1. 优先队列可用什么数据结构实现？ 堆
2. 优先队列插入算法基本思想？

在小根堆中，将元素x插入到堆的末尾， 然后将元素x的关键字与其双亲的关键字比较， 若元素x的关键字小于其双亲的关键字， 则将元素x与其双亲交换，然后再将元素x与其新双亲的关键字相比，直到元素x的关键字大于双亲的关键字，或元素x到根为止。

1. 优先队列插入算法时间复杂度？ O( log n)