**递归与分治：**

分治法所能解决的问题一般具有以下几个特征：该问题的规模缩小到一定的程度就可以容易地解决；该问题可以分解为若干个规模较小的相同问题，即该问题具有最优子结构性质该问题分解出的子问题的解可以合并为该问题的解；该问题所分解出的各个子问题是相互独立的，即子问题之间不包含公共的子问题。

**二分搜索技术** O(logn) 将n个元素分成大致相同的两半，取a[n/2]与x比较，如果a[n/2]=x，则找到，算法终止；如果a[n/2]>x,则只在数组的左半部继续搜索x；如果a[n/2]<x，则只在数组a的右半部继续搜索x。

**大整数的乘法**O(nlog3)=O(n1.59) 设X和Y都是n位二进制整数，计算XY的乘积，将n位二进制整数X和Y都分为2段，每段的长为n/2位，由X=A2n/2+B，Y=C2n/2+D，这样X和Y的乘积XY = AC2n + (AD+BC) 2n/2 + BD，设T(n)是2个n位整数相乘所需的运算总数，则有

解得T(n)=O(n2)。为改进算法的计算复杂性,必须减少乘法次数，故有XY = AC2n + ((A-B)(D-C)+AC+BD) 2n/2 + BD，则有

解得T(n)=O(nlog3)

**Strassen矩阵乘法** O(n3)—O(nlog7)

**棋盘覆盖**T(n)=O(4k) 当k>0时，将2k×2k棋盘分割为4个2k-1×2k-1 子棋盘。特殊方格必位于4个较小子棋盘之一中，其余3个子棋盘中无特殊方格。为了将这3个无特殊方格的子棋盘转化为特殊棋盘，可以用一个L型骨牌覆盖这3个较小棋盘的会合处，这3个子棋盘上被L型骨牌覆盖的方格就成为该棋盘上的特殊方格，从而将原问题转化为4个较小规模的棋盘覆盖问题。递归地使用这种分割，直至棋盘简化为棋盘1×1。T(k)满足以下递归方程,解此递归方程T(k)= O(4k)

**合并排序** T(n)=O(nlogn) 辅助空间：O(n)，将待排序元素分成大小大致相同的2个子集合，分别对2个子集合进行排序，最终将排好序的子集合合并成为所要求的排好序的集合。

**快速排序**  最坏时间复杂度：O(n2) 平均时间复杂度：O(nlogn) 辅助空间：O(n)或O(logn)

步骤：对于输入的子数组a[p:r]（1）分解：以a[q]为基准元素将a[p:r]划分为3段a[p:q-1],a[q]和a[q+1:r]使a[p:q-1]中任何一个元素<= a[q]，而a[q+1:r]中任何一个元素>= a[q]，下标q在划分过程中确定（2）递归求解：通过递归调用快速排序方法分别对a[p:q-1]和a[q+1:r]排序;(3)合并：由于对a[p:q-1]和a[q+1:r]的排序是就地进行的，所以在a[p:q-1]和a[q+1:r]都已排好的序后，不需要执行任何计算，a[p:r]就已排好序。

**线性时间选择** O(n) 将n个输入元素划分成⎡n/5⎤个组，每组5个元素，只可能有一个组不是5个元素。用任意一种排序算法，将每组中的元素排好序，并取出每组的中位数，共⎡n/5⎤个。递归调用select来找出这⎡n/5⎤个元素的中位数。如果⎡n/5⎤是偶数，就找它的2个中位数中较大的一个，以这个元素作为划分基准。设所有元素互不相同。在这种情况下，找出的基准x至少比3(n-5)/10个元素大，因为在每一组中有2个元素小于本组的中位数，而n/5个中位数中又有(n-5)/10个小于基准x。同理，基准x也至少比3(n-5)/10个元素小。而当n≥75时，3(n-5)/10≥n/4所以按此基准划分所得的2个子数组的长度都至少缩短1/4。

上述算法将每一组的大小定为5，并选取75作为是否作递归调用的分界点。这2点保证了T(n)的递归式中2个自变量之和n/5+3n/4=19n/20=εn(0<ε<1)

**最接近点对** O(nlogn)

**循环赛日程表** O(nlogn) 将所有的选手分为两半，n个选手的比赛日程表就可以通过为n/2个选手设计的比赛日程表来决定。递归地用对选手进行分割，直到只剩下2个选手时，比赛日程表的制定就变得很简单。这时只要让这2个选手进行比赛就可以了。

**动态规划的基本要素（自底向上依赖子问题的解）**:

1.**最优子结构**：矩阵连乘计算次序问题的最优解包含着其子问题的最优解。这种性质称为最优子结构性质。

2.**子问题重叠性**：递归算法求解问题时，每次产生的子问题并不总是新问题，有些子问题被反复计算多次。这种性质称为子问题的重叠性质。动态规划算法，对每一个子问题只解一次，而后将其解保存在一个表格中，当再次需要解此子问题时，只是简单地用常数时间查看一下结果。

**矩阵连乘问题** 穷举法：P(n)=Ω(4n/n3/2) 动规：时间O(n3) 空间 O(n2) 设矩阵连乘乘积AiAi+1…Aj简记为A[i:j].考察A[1:n]的最优计算次序，设这个计算次序在矩阵Ak 和Ak+1 之间将矩阵链断开,1≤k<n，对应完全加括号方式为((A1…Ak)(Ak+1…An))则

(1)**最优子结构性质**：总计算量为A[1:k]的计算量加上A[k+1:n]的计算量，再加上A[1:k]和A[k+1:n]相乘的计算量。计算A[1:n]的最优次序包含计算矩阵子链A[1:k]和A[k+1:n]的次序也是最优的。事实上，如果有一个计算A[1:k]的次序需要的计算量更少，则用此次序替代原来计算A[1:k]的次序，得到计算A[1:n]的计算量将比最优次序所需计算量更少，这是一个矛盾。同理可知，计算A[1:n]的最优次序所包含的计算矩阵子链A[k+1:n]的次序也是最优的。因此，矩阵连乘乘积计算次序问题的最优解包含着其子问题的最优解。

(2)**建立递归关系** 设计算A[i:j]，1≤i≤j≤n，所需要的最少数乘次数m[i,j]，则原问题的最优值为m[1,n] 。当i=j时，A[i:j]=Ai为单一矩阵，无需计算，因此，m[i,i]=0，i=1,2,…,n；当i<j时，m[i,j]=m[i,k]+m[k+1,j]+pi-1pkpj Ai的维数为pi-1\*pi ,k的位置有j-i种可能

（3）**计算最优值**:依据递归式自底向上的方式计算，输入参数{p0,p1…pn}存储于数组p中，除了输出最优值数组m外，还输出记录最优断开位置的数组s。首先计算出m[i][i]=0,i=1,2…n;根据递归式按矩阵链长递增的方式计算m[i][i+1], i=1,2…n-1（矩阵链长度为2）；m[i][i+2]，i=1,2…n-2（矩阵链长度为3）;……在计算m[i][j]时，只用到已计算出的m[i][k]和m[k+1][j]。

(4**)构造最优解:**s[i][j]中的数表明，计算矩阵链A[i:j]的最佳方式应在矩阵Ak和Ak+1之间断开，即最优的加括号方式应为(A[i:k])(A[k+1:j])。因此，从s[1][n]记录的信息可知计算A[1:n]的最优加括号方式为(A[1:s[1][n]])(A[s[1][n]+1:n])。而A[1:s[1][n]]的最优加括号方式为(A[1:s[1][s[1][n]]])(A[s[1][s[1][n]]+1:s[1][s[1][n]]])。同理可以确定A[s[1][n]+1:n]的最优加括号方式在s[s[1][n]+1][n]处断开……照此递推下去，最终可以确定A[1:n]的最优完全加括号方式，即构造出问题的一个最优解。

**最长公共子序列** O(mn)

(1)**最优子结构性质**:设序列X={x1,x2,…,xm}和Y={y1,y2,…,yn}的最长公共子序列为Z={z1,z2,…,zk} ，则(a)若xm=yn，则zk=xm=yn，且Zk-1是Xm-1和Yn-1的最长公共子序列。(b)若xm≠yn且zk≠xm，则Z是Xm-1和Y的最长公共子序列。(c)若xm≠yn且zk≠yn，则Z是X和Yn-1的最长公共子序列。

**证明**:(1)用反证法。若zk≠xm,则{z1,z2...,zk,zm}是X和Y的长度为k+1的公共子序列。这与Z是X和Y的一个最长公共子序列矛盾。因此,必有zk=xm=yn。由此可知Zk-1是Xm-1和Yn-1的一个长度为k-1的公共子序列。若Xm-1和Yn-1有一个长度大于k-1的公共子序列W,则将xm加在其尾部产生X和Y的一个长度大于k的公共子序列.此为矛盾。故Zk-1是Xm-1和Yn-1的一个最长公共子序列。

(2)由于zk≠xm,Z是Xm-1和Y的一个公共子序列。若Xm-1和Y有一个长度大于k的公共子序列W,则W也是X和Y的一个长度大于k的公共子序列。这与Z是X和Y的一个最长公共子序列矛盾。由此即知,Z是Xm-1和Y的一个最长公共子序列。

(3)证明与(2)类似。

上述性质告诉我们,两个序列的最长公共子序列包含了这两个序列的前缀的最长公共子

序列。因此,最长公共子序列问题具有最优子结构性质。

(2)**子问题的递归结构**：当Xm=Yn时，找出Xm-1和Yn-1的最长公共子序列，然后在其尾部加上xm(=yn)即可得到X和Y的最长公共子序列。当XmYn时，必须解两个子问题，即找出Xm-1和Y的一个最长公共子序列及X和Yn-1的一个最长公共子序列。这两个公共子序列中较长者即为X和Y的最长公共子序列。用c[i][j]记录序列Xi和Yj 的最长公共子序列的长度。其中，Xi={x1,x2,…,xi};Yj={y1,y2,…,yj}。当i=0或j=0时，空序列是Xi和Yj的最长公共子序列。故此时c[i][j]=0。递归关系如下：

(3)**计算最优值** c[i][j]存储Xi和Yj 的最长公共子序列的长度，b[i][j]记录c[i][j]的值由哪个子问题得到，问题的最优值，即X和Y的最长公共子序列的长度记录于c[m][n]中。

(4)**构造最长公共子序列**，首先从b[m][n]开始，依其值在数组b中搜索。当在b[i][j]=1时，表示Xi和Yj的最长公共子序列是由Xi-1和Yj-1的最长公共子序列在尾部加上xi所得到的子序列。当b[i][j]=2 时，表示Xi和Yj的最长公共子序列与Xi-1与Yj的最长公共子序列相同。当b[i][j]=3时，表示Xi和Yj的最长公共子序列与Xi和Yj-1的最长公共子序列相同。

**最大子段和** 分治法O(nlogn) 动规：空间和时间 O(n)

**凸多边形的最优三角剖分**(与矩阵连乘问题类似) 时间O(n3) 空间 O(n2)

(1)**最优子结构性质**：若凸(n+1)边形P={v0,v1,…,vn-1}的最优三角剖分T包含三角形v0vkvn，1≤k≤n-1，则T的权为3个部分权的和：三角形v0vkvn的权，子多边形{v0,v1,…,vk}和{vk,vk+1,…,vn}的权之和。可以断言，由T所确定的这2个子多边形的三角剖分也是最优的。因为若有{v0,v1,…,vk}或{vk,vk+1,…,vn}的更小权的三角剖分将导致T不是最优三角剖分的矛盾。

(2)**递归结构**：t[i][j]，1≤i<j≤n为凸子多边形{vi-1,vi,…,vj}的最优三角剖分所对应的权函数值，即其为最优值(t[i][j]记录的是vi-1 - vj 的权值和)，t[i][j]可递归的定义为：

其中，i≤k≤j-1,k的位置有j-i个。

(3)**计算最优值** (4)**构造最优值**，在计算每一个凸子多边形{vi-1,vi,…,vj}的最优值时，用数组s记录了最优三角剖分中所有三角形信息。s[i][j]记录了与vi-1和vj一起构成三角形的第三个顶点位置，据此，用O(n)时间就可构造出最优三角剖分中的所有三角形。

**多边形游戏时间O(n3)**

多边形游戏是一个单人玩的游戏，开始时有一个由n个顶点构成的多边形。每个顶点被赋予一个整数值，每条边被赋予一个运算符“+”或“\*”。所有边依次用整数从1到n编号。游戏第1步，将一条边删除。随后n-1步按以下方式操作：(1)选择一条边E以及由E连接着的2个顶点V1和V2；(2)用一个新的顶点取代边E以及由E连接着的2个顶点V1和V2。将由顶点V1和V2的整数值通过边E上的运算得到的结果赋予新顶点。最后，所有边都被删除，游戏结束。游戏的得分就是所剩顶点上的整数值。问题:对于给定的多边形，计算最高得分。

(1)**最优子结构性质**：在所给多边形中,从顶点i(1≤i≤n)开始,长度为j(链中有j个顶点)的顺时针链p(i，j) 可表示为v[i]，op[i+1]，…，v[i+j-1]。如果这条链的最后一次合并运算在op[i+s]处发生(1≤s≤j-1)，则可在op[i+s]处将链分割为2个子链p(i,s)和p(i+s,j-s)。设m1是对子链p(i，s)的任意一种合并方式得到的值，而a和b分别是在所有可能的合并中得到的最小值和最大值。m2是p(i+s,j-s)的任意一种合并方式得到的值，而c和d分别是在所有可能的合并中得到的最小值和最大值。依此定义有a≤m1≤b，c≤m2≤d,(1)当op[i+s]='+'时，显然有a+c≤m≤b+d，(2)当op[i+s]='\*'时，有min{ac,ad,bc,bd}≤m≤max{ac,ad,bc,bd};换句话说，主链的最大值和最小值可由子链的最大值和最小值得到。故多边形游戏问题满足最优子结构性质。 (2)**递归求解**

**流水作业调度时间O(nlogn)空间O(n)**

n个作业{1，2，…，n}要在由2台机器M1和M2组成的流水线上完成加工。每个作业加工的顺序都是先在M1上加工，然后在M2上加工。M1和M2加工作业i所需的时间分别为ai和bi。流水作业调度问题要求确定这n个作业的最优加工顺序，使得从第一个作业在机器M1上开始加工，到最后一个作业在机器M2上加工完成所需的时间最少。

(1)**最优子结构性质**：设π是所给n个流水作业的一个最优调度，它所需的加工时间为 +T’。其中T’是在机器M2的等待时间为时，安排作业π (2)，…，π (n)所需的时间。记S=N-{π (1)}，则有T’=T(S,)。事实上，由T的定义知T’≥T(S,)。若T’>T(S,)，设π’是作业集S在机器M2的等待时间为情况下的一个最优调度。则π (1)， π’(2)，…， π’(n)是N的一个调度，且该调度所需的时间为+T(S,)<+T’。这与π是N的一个最优调度矛盾。故T’≤T(S,)。从而T’=T(S,)。这就证明了流水作业调度问题具有最优子结构的性质。

(2)**递归计算最优值**: 由流水作业调度问题的最优子结构性质可知，,推广到一般形式有，按照递归式，可设计出解流水作业调度问题的动态规划算法。

**电路布线 O(n2)** 在一块电路板的上、下2端分别有n个接线柱。根据电路设计，要求用导线(i,π(i))将上端接线柱与下端接线柱相连。其中π(i)是{1,2,…,n}的一个排列。导线(i,π(i))称为该电路板上的第i条连线。对于任何1≤i<j≤n，第i条连线和第j条连线相交的充分且必要的条件是π(i)>π(j)。电路布线问题要确定将哪些连线安排在第一层上，使得该层上有尽可能多的连线。换句话说，该问题要求确定导线集Nets={(i,π(i)),1≤i≤n}的最大不相交子集。

(1)**最优子结构性质**：记,N(i,j)的最大不相交子集为MNS(i,j)。Size(i,j)=|MNS(i,j)|。

(1)当i=1时，

(2)当i>1时，① j<π(i)。此时， 。故在这种情况下，N(i,j)=N(i-1,j)，从而Size(i,j)=Size(i-1,j)。②j≥π(i)，若(i,π(i))∈MNS(i,j).则对任意(t,π(t)) ∈MNS(i,j)有t<i且π(t)<π(i)。在这种情况下MNS(i,j)-{(i,π(i))}是N(i-1,π(i)-1)的最大不相交子集.③若 ，则对任意(t,π(t)) ∈MNS(i,j)有t<i。从而 。因此，Size(i,j)≤Size(i-1,j)。另一方面，故又有Size(i,j)≥Size(i-1,j),从而Size(i,j)=Size(i-1,j)。综上可知，电路布线问题具有最优子结构性质。

(2)**递归计算最优值**:电路布线问题的最优值为Size(n,n),由该问题的最优子结构性质可知：(1)当i=1时, (2)当i>1时,

**图像压缩 O(n)**

**0-1背包问题** O(nc) 改进O(min{nc,2n})

(1)**最优子结构性质** 0-1背包问题具有最优子结构性质。设(y1,y2,..,yn)是所给0-1背包问题的一个最优解，则(y2,..,yn)是下面相应子问题的一个最优解：

因若不然，设(z2,..,zn)是上述子问题的一个最优解，而(y2,..,yn)不是它的最优解。由此可知，且。因此 ，,这说明(y1,z2,..,zn)是所给0-1背包问题的一个更优解，从而(y1,y2,..,yn)不是所给0-1背包问题的最优解。此为矛盾。

(2)**递归关系** 设所给0-1背包问题的子问题

的最优值为m(i，j)，即m(i，j)是背包容量为j，可选择物品为i，i+1，…，n时0-1背包问题的最优值。由0-1背包问题的最优子结构性质，可以建立计算m(i，j)的递归式如下。

（3）**算法描述** 当wi(1<=i<=n)为正整数时，用二维数组m[][]来存储m(i,j)的相应值，m[1][c]给出所要求的0-1背包问题的最优值。若m[1][c]=m[2][c]，则x1 =0，否则x1=1.当x1=0时，由m[2][c]继续构造最优解。当x1=1时，由m[2][c-w1]继续构造最优解。以此类推，可构造出相应的最优解(x1,x2,…,xn).

**最优二叉搜索树**

**贪心算法（自顶向下）：** 贪心算法是指，在对问题求解时，总是做出在当前看来是最好的选择。也就是说，不从整体最优上加以考虑，他所做出的是在某种意义上的局部最优解。贪心算法不是对所有问题都能得到整体最优解，即使贪心算法不能得到整体最优解，但其最终结果却是最优解的很好的近似解。贪心策略的选择，选择的贪心策略必须具备的特点是：某个状态以前的过程不会影响以后的状态，只与当前状态有关。可用贪心算法求解问题具备的两个性质：贪心选择性质（所求问题的整体最优解可以通过一系列局部最优的选择即贪心选择达到）和最优子结构性质（问题的最优解包含其子问题的最优解）

**最优装载：**要使用贪心算法解决问题，我们必须先证明：（1）该问题具备贪心选择性质；（2）**该问题具备最优子结构性质**。首先先证明贪心选择性质：设集装箱已依其重量从小到大排序，（,,.......,）是最优装载问题的一个最优解。又设k=min{i|=1}{1in}.易知，如果给定的最优装载问题有解，则1kn;

**证明**：当k=1时，（,,…,）是满足贪心选择性质的最优解。当k1时，取,,,1in,ik,则=+ (x1=0,xk=1),因此（,,…,）是所给最优装载问题的可行解，又因为知，（,,…,）是一个满足贪心性质的最优解，所以最优装载问题具有贪心选择性质.其次，证明该问题具备最优子结构性质：设（,,.......,）是最优装载的满足贪心选择性质的最优解，易知，=1,（,,.......,）是轮船载重量为c-，待装船集装箱为{2,3，.....n}时相应的最优装载问题的最优解。

**单源最短路径Dijkstra算法：算法思想**：设置一个顶点集合S并不断地作贪心选择来扩充这个集合。一个顶点属于集合S当且仅当从源到该顶点的最短路径长度已知。初始时，S中仅含有源。设u是G的某个顶点，从源到u且中间只经过S中顶点的路称为从源到u的特殊路径，并用数组dist来记录当前每个顶点所对应的最短特殊路径长度。该算法每次从V-S中取出具有最短特殊路长度的顶点u，将u添加到S中，同时对数组dist作必要的修改。S包含所有V中顶点时，dist就记录了从源到所有其他顶点的最短路径。

**贪心选择性质：**它所做的贪心选择是从V-S中选择具有最短特殊路径的顶点u，从而确定从源到u的最短路径长度dist[u]。这种贪心选择能导致最优解，是因为，如果存在一条从源到u且长度比dist[u]更短的路，设这条路初次走出S之外到达的顶点为x∈V-S，然后徘徊于S内外若干次，最后离开S到达u。在这条路径上，分别记d(v，x），d（x，u）和d（v，u）为顶点v到顶点x，顶点x到顶点u和顶点v到顶点u的路长，那么，dist[x]≤d（v，x），d（v，x）+ d（x，u）=d（v，u）<dist[u]。利用边权的非负性，可知d（x，u）≥0，从而推得dist[x]<dist[u]。此为矛盾。这就证明了dist[u]是从源到顶点u的最短路径长度。

**最优子结构性质：**该性质描述为：如果P(i,j)={Vi....Vk..Vs...Vj}是从顶点i到j的最短路径，k和s是这条路径上的一个中间顶点，那么P(k,s)必定是从k到s的最短路径。下面证明该性质的正确性。证明：假设P(i,j)={Vi....Vk..Vs...Vj}是从顶点i到j的最短路径，则有P(i,j)=P(i,k)+P(k,s)+P(s,j)。而P(k,s)不是从k到s的最短距离，那么必定存在另一条从k到s的最短路径P'(k,s)，那么P'(i,j)=P(i,k)+P'(k,s)+P(s,j)<P(i,j)。则与P(i,j)是从i到j的最短路径相矛盾。因此该性质得证。时间复杂度：O(n2)

**最小生成树**Prim算法(点) 设G=(V,E)是连通带权图，V={1,2,…,n}。构造G的最小生成树的Prim算法的基本思想是：首先置S={1}，然后，只要S是V的真子集，就作如下的贪心选择：选取满足条件iS，jV-S，且c[i][j]最小的边，将顶点j添加到S中。这个过程一直进行到S=V时为止。在这个过程中选取到的所有边恰好构成G的一棵最小生成树。 利用最小生成树性质和数学归纳法容易证明，上述算法中的边集合T始终包含G的某棵最小生成树中的边。因此，在算法结束时，T中的所有边构成G的一棵最小生成树，Prim算法所需要的计算时间为O(n2)。

**最小生成树Kruskal算法（边）**设G=(V,E)是无向连通带权图，V={1,2,…,n},该算法构造G的最小生成树的基本思想是，首先将G的n个顶点看成n个孤立的连通分支。将所有的边按权从小到大排序。然后从第一条边开始，依边权递增的顺序查看每一条边，并按下述方法连接2个不同的连通分支：当查看到第k条边(v,w)时，如果端点v和w分别是当前2个不同的连通分支T1和T2中的顶点时，就用边(v,w)将T1和T2连接成一个连通分支，然后继续查看第k+1条边；如果端点v和w在当前的同一个连通分支中，就直接再查看第k+1条边。这个过程一直进行到只剩下一个连通分支时为止

**回溯算法基本概念:**回溯法在问题的解空间树中，按深度优先策略，从根结点出发搜索解空间树。算法搜索至解空间树的任意一点时，先判断该结点是否包含问题的解。如果肯定不包含，则跳过对该结点为根的子树的搜索，逐层向其祖先结点回溯；否则，进入该子树，继续按深度优先策略搜索

**回溯法基本思想:** (1)针对所给问题，定义问题的解空间（排列树和子集树）；(2)确定易于搜索的解空间结构；(3)以深度优先方式搜索解空间，并在搜索过程中用剪枝函数避免无效搜索。常用剪枝函数：用约束函数在扩展结点处剪去不满足约束的子树；用限界函数剪去得不到最优解的子树

**装载问题O(2n):**有一批共n个集装箱要装上2艘载重量分别为c1和c2的轮船，其中集箱i的重量为wi，且装载问题要求确定是否有一个合理的装载方案可将这个集装箱装上这2艘轮船。如果有，找出一种装载方案。容易证明，如果一个给定装载问题有解，则采用下面的策略可得到最优装载方案。(1)首先将第一艘轮船尽可能装满；(2)将剩余的集装箱装上第二艘轮船。

**算法思想(回溯法)**:将第一艘轮船尽可能装满等价于选取全体集装箱的一个子集，使该子集中集装箱重量之和最接近。由此可知，装载问题等价于以下特殊的0-1背包问题.

1)**算法设计:**解空间：子集树，可行性约束函数(选择当前元素)可剪去不满足约束条件 的子树，在子集树的第j+1层结点Z处，用cw记当前载重量，即，当cw>c1时，以结点Z为根的子树中所有节点都不满足约束条件，因而该子树中的解均为不可行解，故可将该子树剪去。

2)**上界函数(不选择当前元素)**：设Z是解空间树第i层上的当前扩展点，cw是当前载重量，bestw是当前最优载重量，r是剩余集装箱的重量，定义上界函数为cw+r。在以Z为根的子树中任一叶结点所相应的载重量不超过cw+r。因此，当cw+ rbestw时，可将Z的右子树剪去。

3)**构造最优解**，x用于记录从根至当前结点的路径；bestx记录当前最优解。算法搜索到达叶节点处，就修正bestx的值。

**批处理作业调度O(n!):** 给定n个作业的集合{J1,J2,…,Jn}。每个作业必须先由机器1处理，然后由机器2处理。作业Ji需要机器j的处理时间为tji。对于一个确定的作业调度，设Fji是作业i在机器j上完成处理的时间。所有作业在机器2上完成处理的时间和称为该作业调度的完成时间和f=。批处理作业调度问题要求对于给定的n个作业，制定最佳作业调度方案，使其完成时间和达到最小。

**算法设计**:设开始时x=[1,2…,n]是所给的n个作业，则相应的排列树由x[1:n]的所有排列构成。i>n时算法搜索至叶节点，得到一个新的作业调度方案，此时算法适时更新当前最优值和最佳的作业调度。i<n时，当前扩展结点位于排列树的i-1层，此时算法选择下一个要安排的作业，以深度优先的方式递归的对相应子树进行搜索，对于不满足上界约束的结点则减去相应的子树。

**连续邮资问题:**假设某国家发行了n种不同面值的邮票，并且规定每张信封上最多只允许贴m张邮票。连续邮资问题要求对于给定的n和m，给出邮票面值的最佳设计.

**算法设计**:对于连续邮资问题，用n元组x[1:n]表示n种不同的邮票面值，并约定它们从小到大排列。x[1]= 1是唯一的选择。此时的最大连续邮资区间是[1:m]。接下来，x[2]的可取值范围是[2:m + 1].在一般情况下,已选定x[1:i- 1],最大连续邮资区间是[1:r],接下来x[i]的可取值范围是[x[i- 1]+1:r+1]。由此可以看出,在用回溯法解连续邮资问题时，可以用树表示其解空间。该解空间树中各结点的度随x的不同取值而变化。当i>n时,算法搜索至叶结点,得到新的邮票面值设计方案x[1:n].如果该方案能贴出的最大连续邮资区间大于当前已找到的最大连续邮资区间maxvalue,则更新当前最优值maxvalue和相应的最优解bestx。当i≤n时，当前扩展结点Z是解空间中的内部结点。在该结点处x[1:i- 1]能贴出的最大连续邮资区间为r-1。因此，在结点Z处,x[i]的可取值范围是[x[i-1]+1:r],从而,结点Z有r-x[i-1]个儿子结点。算法对当前扩展结点Z的每一个儿子结点，以深度优先的方式递归地对相应子树进行搜索。

**分支限界的基本思想**: 分支限界法常以广度优先或以最小耗费（最大效益）优先的方式搜索问题的解空间树.在分支限界法中，每一个活结点只有一次机会成为扩展结点。活结点一旦成为扩展结点，就一次性产生其所有儿子结点。在这些儿子结点中，导致不可行解或导致非最优解的儿子结点被舍弃，其余儿子结点被加入活结点表中此后，从活结点表中取下一结点成为当前扩展结点，并重复上述结点扩展过程。这个过程一直持续到找到所需的解或活结点表为空时为止.

**随机化算法：**

**线性同余法(产生随机数的算法)：**是产生伪随机数的最常用的方法。由线性同余法产生的随机序列a0,a1,…,an满足=(b)mod m n=1,2,…

其中b0，c0，dm。d称为该随机序列的种子。如何选取该方法中的常数b、c和m直接关系到所产生的随机序列的随机性能。这是随机性理论研究的内容，已超出本书讨论的范围。从直观上看，m应取得充分大，因此可取m为机器大数，另外应取gcd(m,b)=1，因此可取b为一素数

**舍伍德算法(使随机算法稳定在均匀时间附近):**设A是一个确定性算法，当它的输入实例为x时所需的计算时间记为。设Xn是算法A的输入规模为n的实例的全体，则当问题的输入规模为n时，算法A所需的平均时间为=,这显然不能排除存在x∈Xn使得  的可能性。希望获得一个概率算法B，使得对问题的输入规模为n的每一个实例均有这就是舍伍德算法设计的基本思想。当s(n)与相比可忽略时，舍伍德算法可获得很好的平均性能。

**跳跃表：**提高有序链表效率的一个技巧是在有序链表的部分结点处增设附加指针以提高其搜索性能。在增设附加指针的有序链表中搜索一个元素时，可借助于附加指针跳过链表中若干结点，加快搜索速度。这种增加了向前附加指针的有序链表称为跳跃表。在一般情况下，给定一个含有n个元素的有序链表，可以将它改造成一个完全跳跃表，使得每一个k级结点含有k+1个指针，分别跳过-1，-1，…，-1个中间结点。第i个k级结点安排在跳跃表的位置i处，i0。这样就可以在时间O(logn)内完成集合成员的搜索运算。在一个完全跳跃表中，最高级的结点是logn 级结点。为了在动态变化中维持跳跃表中附加指针的平衡性，必须使跳跃表中k级结点数维持在总结点数的一定比例范围内。注意到在一个完全跳跃表中，50%的指针是0级指针；25%的指针是1级指针；…；()%的指针是k级指针。因此，在插入一个元素时，以概率1/2引入一个0级结点，以概率1/4引入一个1级结点，…，以概率引入一个k级结点。另一方面，一个i级结点指向下一个同级或更高级的结点，它所跳过的结点数不再准确地维持在-1。经过这样的修改，就可以在插入或删除一个元素时，通过对跳跃表的局部修改来维持其平衡性。

**拉斯维加斯算法（需要反复调用直到给出一个正确解）:**拉斯维加斯算法的一个显著特征是它所作的随机性决策有可能导致算法找不到所需的解。拉斯维加斯算法不会得到不正确的解。一旦用拉斯维加斯算法找到一个解，这个解就一定是正确解。但有时用拉斯维加斯算法找不到解。与蒙特卡罗算法类似，拉斯维加斯算法找到正确解的概率随着它所用的计算时间的增加而提高。对于所求解问题的任一实例，用同一拉斯维加斯算法反复对该实例求解足够多次，可使求解失败的概率任意小。

void **obstinate**(Object x, Object y)

   {// 反复调用拉斯维加斯算法LV(x,y)，直到找到问题的一个解y

      bool success= false; while (!success) success=lv(x,y);  }

设p(x)是对输入x调用拉斯维加斯算法获得问题的一个解的概率。一个正确的拉斯维加斯算法应该对所有输入x均有p(x)>0。设t(x)是算法**obstinate**找到具体实例x的一个解所需的平均时间 ,s(x)和e(x)分别是算法对于具体实例x求解成功或求解失败所需的平均时间，则有：t(x)=p(x)\*s(x)+(1-p(x))(e(x)+t(x));解此方程可得： t(x)=s(x)+e(x)

**N后问题**：在棋盘上相继的各行中随机地放置皇后（任何2个皇后不放在同一行或同一列或同一斜线上），并注意使新放置的皇后与已放置的皇后互不攻击，直至n个皇后均已相容地放置好，或已没有下一个皇后的可放置位置时为止；如果将上述随机放置策略与回溯法相结合，可能会获得更好的效果。可以先在棋盘的若干行中随机地放置皇后，然后在后继行中用回溯法继续放置，直至找到一个解或宣告失败。随机放置的皇后越多，后继回溯搜索所需的时间就越少，但失败的概率也就越大。

**整数因子分解：**设n>1是一个整数。关于整数n的因子分解问题是找出n的如下形式的唯一分解式:n=,其中，p1<p2<…<pk是k个素数，m1,m2,…,mk是k个正整数。如果n是一个合数，则n必有一个非平凡因子x，1<x<n，使得x可以整除n。给定一个合数n，求n的一个非平凡因子的问题称为整数n的因子分割问题。

Pollard算法：在开始时选取0～n-1范围内的随机数x1，然后递归地由,产生无穷序列对于i=2k ,k=0,1,…，以及2k<j≤2k+1，算法计算出xj-xi与n的最大公因子d=gcd(xj-xi，n)。如果d是n的非平凡因子，则实现对n的一次分割，算法输出n的因子d。对Pollard算法更深入的分析可知，执行算法的while循环约次后，Pollard算法会输出n的一个因子p。由于n的最小素因子p≤，故Pollard算法可在O(n1/4)时间内找到n的一个素因子。

**蒙特卡洛算法（反复调用给出概率大的作为问题的解）:** 在实际应用中常会遇到一些问题，不论采用确定性算法或概率算法都无法保证每次都能得到正确的解答。蒙特卡罗算法则在一般情况下可以保证对问题的所有实例都以高概率给出正确解，但是通常无法判定一个具体解是否正确。设p是一个实数，且1/2<p<1。如果一个蒙特卡罗算法对于问题的任一实例得到正确解的概率不小于p，则称该蒙特卡罗算法是p正确的，且称p-1/2是该算法的优势。如果对于同一实例，蒙特卡罗算法不会给出2个不同的正确解答，则称该蒙特卡罗算法是一致的。有些蒙特卡罗算法除了具有描述问题实例的输入参数外，还具有描述错误解可接受概率的参数。这类算法的计算时间复杂性通常由问题的实例规模以及错误解可接受概率的函数来描述.对于一个一致的p正确蒙特卡罗算法，要提高获得正确解的概率，只要执行该算法若干次，并选择出现频次最高的解即可。如果重复调用一个一致的(1/2+ε)正确的蒙特卡罗算法2m-1次，得到正确解的概率至少为1-δ，其中 对于一个解所给问题的蒙特卡罗算法MC(x)，如果存在问题实例的子集X使得：(1)当xX时，MC(x)返回的解是正确的；(2)当xX时，正确解是y0，但MC(x)返回的解未必是y0。称上述算法MC(x)是偏y0的算法

**主元素问题：**设T[1:n]是一个含有n个元素的数组。当|{i|T[i]=x}|>n/2时，称元素x是数组T的主元素。 对于任何给定的ε>0，算法**majorityMC**重复调用⎡log(1/ε)⎤ 次算法**majority**。它是一个偏真蒙特卡罗算法，且其错误概率小于ε。算法**majorityMC**所需的计算时间显然是O(nlog(1/ ε))

**素数测试**Wilson定理：对于给定正整数n，判定n是素数的充要条件是(n-1)! ≡ -1(mod n)。费尔马小定理：如果p是一个素数，且0<a<p，则 ≡ 1(mod p)。 表示（）余1二次探测定理：如果p是一个素数，且0<x<p，则方程≡1(mod p)的解为x=1，p-1。

**现行规划问题和单纯性算法**：**约束标准型：**每个等式约束中至少有一个变量系数为正，且这个变量只在该约束中出现即基本变量。

**单纯形算法的第1步：**选出使目标函数增加的非基本变量作为**入基变量**

**单纯形算法的第2步：**选取离基变量 在单纯形表中考察由第1步选出的入基变量所相应的列。在一个基本变量变为负值之前，入基变量可以增到多大如果入基变量所在的列与基本变量所在行交叉处的表元素为负数，那么该元素将不受任何限制，相应的基本变量只会越变越大。如果入基变量所在列的所有元素都是负值，则目标函数无界，已经得到了问题的无界解。如果选出的列中有一个或多个元素为正数，要弄清是哪个数限制了入基变量值的增加。受限的增加量可以用入基变量所在列的元素（称为主元素）来除主元素所在行的“常数列”（最左边的列）中元素而得到。所得到数值越小说明受到限制越多。应该选取受到限制最多的基本变量作为离基变量，才能保证将入基变量与离基变量互调位置后，仍满足约束条件。

**单纯形算法的第3步：**转轴变换转轴变换的目的是将入基变量与离基变量互调位置。给入基变量一个增值，使之成为基本变量；修改离基变量，让入基变量所在列中，离基变量所在行的元素值减为零，而使之成为非基本变量。

**单纯形算法的第4步：**转回并重复第1步，进一步改进目标函数值。不断重复上述过程，直到z行的所有非基本变量系数都变成负值为止。这表明目标函数不可能再增加了

**增广路算法:**设P是网络G中联结源s和汇t的一条路。定义路的方向是从s到t。将路P上的边分成2类：一类边的方向与路的方向一致，称为**向前边**。向前边的全体记为P+。另一类边的方向与路的方向相反，称为**向后边**。向后边的全体记为P-。设flow是一个可行流，P是从s到t的一条路，若P满足下列条件：（1）在P的所有向前边(v,w)上，flow(v,w)<cap(v,w)，即P+中的每一条边都是非饱和边；（2）在P的所有向后边(v,w)上，flow(v,w)>0，即P-中的每一条边都是非零流边。则称P为关于可行流flow的一条可增广路，可增广路是残流网络中一条容量大于0的路。将具有上述特征的路P称为可增广路是因为可以通过修正路P上所有边流量flow(v,w),将当前可行流改进成一个流值更大的可行流。增流的具体做法是：（1）不属于可增广路P的边(v,w)上的流量保持不变；（2）可增广路P上的所有边(v,w)上的流量按下述规则变化：在向前边(v,w)上flow(v,w)=flow(v,w)+d；在向后边(v,w)上，flow(v,w)=flow(v,w)-d；不属于p的边flow(v,w)= flow(v,w)。其中d称为可增广量，可按下述原则确定：d取得尽量大，又要使变化后的流仍为可行流。按照这个原则，d既不能超过每条向前边(v,w)的cap(v,w)-flow(v,w)，也不能超过每向后边(v,w)的flow(v,w)。因此d应该等于向前边上的cap(v,w)-flow(v,w)与向后边上的flow(v,w)的最小值。就是残流网络中P的最大容量。**增广路定理：**设flow是网络G的一个可行流，如果不存在从s到t关于flow的可增广路P，则flow是G的一个最大流。

**图灵机:**一台图灵机由一个有限状态控制器和K条读写带组成，这些读写带的右端无无限，每条带从左到右划分为一个方格，每个方格可以存放一个带符号，带符号的总数是有限的。每条带上都有一个由有限状态控制器操纵的读写头（带头），他可以对这K条带头进行读写操作，有限状态就控制器在某一时刻处于某种状态。且状态总数是有限的。根据有限状态控制器的当前状态及每个读写头读到的带符号，图灵机的一个计算步可实现下面3个操作之一或全部  (1)改变有限状态控制器中的状态 (2)清除当前读写头下的方格中原有带符号并写上新的带符号。  (3)独立地将任何一个或所有读写头，向左移动一个方格(L)或向右移动一个方格(R)或停在当前单元不动(S)。k带图灵机可形式化地描述为一个7元组(Q，T，I，δ，b，q0，qf)，其中:(1)Q是有限个状态的集合。  (2)T是有限个带符号的集合。(3)I是输入符号的集合，I⊆T。(4)b是唯一的空白符，b∈T-I。(5)q0是初始状态。 (6)qf是终止(或接受)状态。 (7)δ是移动函数。它是从QxTk的某一子集映射到Qx (Tx{L，R，S})k的函数

**p类与NP类问题：**

**P类和NP类语言的定义：**    P={L|L是一个能在**多项式时间内**被一台**DTM**所接受的语言}     NP={L|L是一个能在**多项式时间**内被一台**NDTM**所接受的语言}由于一台确定性图灵机可看作是非确定性图灵机的特例，所以可在多项式时间内被确定性图灵机接受的语言也可在多项式时间内被非确定性图灵机接受。故**PNP**

**定理1：**VP=NP 证明；先证明VP⊆NP。对于任意L∈VP,设p是一个多项式,A是一个多项式时间验证算法，则下面的非确定性算法接受语言L:(1)对于输入X,非确定性地产生一字符串Y∈∑\*;(2)当A(X,Y)= 1时接受X。该算法的步骤(1)与团问题的第2阶段的非确定性算法一\*样， 至多在O( |X|)时间内完成。步骤(2)的计算时间是|X|和|Y|的多项式，而|Y|≤p(|X|).因此,它也是|X|的多项式。整个算法可在多项式时间内完成。因此,L∈NP,VP⊆NP。反之,设L∈NP,L∈∑\*，且非确定性图灵机M在多项式时间p内接受语言L。设M在任何情况下只有不超过d个的下一动作选择，则对于输人串X,M的任一动作序列可用{0, 1..,d-1}的长度不超过p(|X|)的字符串来编码。不失一般性,设|∑|≥d。验证算法A(X,Y)用于验证“Y是M上关于输人X的一条接受计算路径的编码”。即当Y是这样一个编码时,A(X,Y)=1。A(X,Y) 显然可在多项式时间内确定性地进行验证,且L= {X|存在Y使得|Y|≤p(|X|)且A(X,Y)= 1}因此L∈VP,VP⊇NP. 综上即知,VP==NP.

**多项式时间变换:**设 是2个语言。所谓语言L1能在多项式时间内变换为语言L2(简记为L1 ∝p L2)是指存在映身f: .且f满足:(1)有一个计算f的多项式时间确定性图灵机；(2)对于所有x∈，x∈L1，当且仅当f(x)∈L2。**定理1**:语言L是NP完全的当且仅当(1)L∈NP；(2)对于所有L’∈NP有L’ ∝p L。如果有一个语言L满足上述性质(2)，但不一定满足性质(1)，则称该语言是NP难的。所有NP完全语言构成的语言类称为NP完全语言类，记为NPC。

**定理9-2：**设L是NP完全的，则(1)L∈P当且仅当P＝NP；(2)若L∝p L1，且L1∈NP，则L1是NP完全的

证明:(1)若P=NP,则显然L∈P。反之,设L∈P,而L1∈NP,则L可在多项式时间p1内被确定性图灵机M所接受。又由L的NP完全性知L1∝p L,即存在映射f,使L=f(L1)。设N是在多项式时间p2内计算f的确定性图灵机。用图灵机M和N构造识别语言L1的算法A如下:1、对于输入x,用N在p2(|x|)时间内计算出f(x);2、在时间|f(x)|内将读写头移到f(x)的第一个符号处;3、用M在时间p(f|x|)内判定f(x)∈L。若f(x)∈L,则接受x,否则拒绝x。上述算法显然可接受语言L1 ,其计算时间为p2(|x|)+|f(x)|+p1(f|x|)。由于图灵机一次只能在一个方格中写人一个符号,故|f(x)|≤|x|+p2(|x|)。因此，存在多项式r使得P2(|x|)+ |f(x)|+p1(f|x|)≤r(x)。因此，L1∈P。由L1的任意性即知P=NP.

(2)只要证明对任意的L'∈NP,有L'∝pL1.由于L是NP完全的,故存在一个多项式时间变换f使L= f(L')。 又由于L∝p L1,故存在一个多项式时间变换g使L1=g(L)。因此，若取f和g的和复合函数h=g(f),则L1=h(L')。易知h为一多项式。因此L'∝p L1 。由L'的任意性即知 L1∈NPC。从定理9-2的(1)可知，如果任一NP完全问题可在多项式时间内求解，则所有NP中的问题都可在多项式时间内求解。反之,若P≠NP,则所有NP完全问题都不可能在多项式时间内求解。定理9-2的(2)实际上是证明问题的NP完全性的有力工具。一旦建立了问题L的NP完全性后,对于L1∈NP,只要证明问题L可在多项式时间内变换为L1,即L∝p L1,就可证明L1也是NP完全的。

**典型的NP完全问题**：**近似算法的性能:** 若一个最优化问题的最优值为c\*，求解该问题的一个近似算法求得的近似最优解相应的目标函数值为c，则将该近似算法的性能比定义为η=max。在通常情况下，该性能比是问题输入规模n的一个函数ρ(n)，即max ≤ρ(n).

**近似算法的相对误差**定义为= 。若对问题的输入规模n，有一函数ε(n)使得  ≤ε(n)，则称ε(n)为该**近似算法的相对误差界**。近似算法的性能比ρ(n)与相对误差界ε(n)之间显然有如下关系：**ε(n)≤ρ(n)-1**。

**旅行商售货问题近似算法:**问题描述：给定一个完全无向图G=(V,E)，其每一边(u,v)∈E有一非负整数费用c(u,v)。要找出G的最小费用哈密顿回路,比如费用函数c往往具有三角不等式性质，即对任意的3个顶点u,v,w∈V，有：c(u,w)≤c(u,v)+c(v,w)。当图G中的顶点就是平面上的点，任意2顶点间的费用就是这2点间的欧氏距离时，费用函数c就具有三角不等式性质。对于给定的无向图G，可以利用找图G的最小生成树的算法设计找近似最优的旅行售货员回路的算法。

void approxTSP (Graph g){   (1)选择g的任一顶点r；   (2)用Prim算法找出带权图g的一棵以r为根的最小生成树T；   (3)前序遍历树T得到的顶点表L；   (4)将r加到表L的末尾，按表L中顶点次序组成回路H，作为计算结果返回；}

当费用函数满足三角不等式时，算法找出的旅行售货员回路的费用不会超过最优旅行售货员回路费用的2倍

**一般旅行售货问题**：在费用函数不一定满足三角不等式的一般情况下，不存在具有常数性能比的解TSP问题的多项式时间近似算法，除非P=NP。换句话说，若P≠NP，则对任意常数ρ>1，不存在性能比为ρ的解旅行售货员问题的多项式时间近似算法

**集合覆盖问题的近似算法：**问题描述：给定一个完全无向图G=(V,E)，其每一边(u,v)∈E有一非负整数费用c(u,v)。要找出G的最小费用哈密顿回路。利用找图G的最小生成树的算法，(1)选择g的任一顶点r； (2)用Prim算法找出带权图g的一棵以r为根的最小生成树T；(3)前序遍历树T得到的顶点表L；(4)将r加到表L的末尾，按表L中顶点次序组成回路H，作为计算结果返回。集合覆盖问题的一个实例〈X,F〉由一个有限集X及X的一个子集族F组成。子集族F覆盖了有限集X。也就是说X中每一元素至少属于F中的一个子集，即X=。对于F中的一个子集CF，若C中的X的子集覆盖了X，即X=，则称C覆盖了X。集合覆盖问题就是要找出F中覆盖X的最小子集C\*，使得 |C\*|=min{|C||CF且C覆盖X}

**集合覆盖问题近似算法——贪心算法**Set **greedySetCover**(X,F){U=X； C=； while (U !=) { 选择F中使|S∩U|最大的子集S；U=U-S；C=C∪{S}； }return C；}

算法的循环体最多执行min{|X|，|F|}次。而循环体内的计算显然可在O(|X||F|)时间内完成。因此，算法的计算时间为O(|X||F|min{|X|，|F|})。由此即知，该算法是一个多项式时间算法。

**递归与分治：**

**二分搜索技术、大整数的乘法、Strassen矩阵乘法、棋盘覆盖、排序、线性时间选择、最接近点对、循环赛日程表、**

**动态规划**

**矩阵连乘问题、最长公共子序列、最大子段和、凸多边形的最优三角剖分、多边形游戏时间、流水作业调度、电路布线、背包问题**

**最优二叉搜索树**

**贪心算法、最优装载、单源最短路径Dijkstra算法、最小生成树**

**回溯算法**

**装载问题、批处理作业调度、连续邮资问题、分支限界**

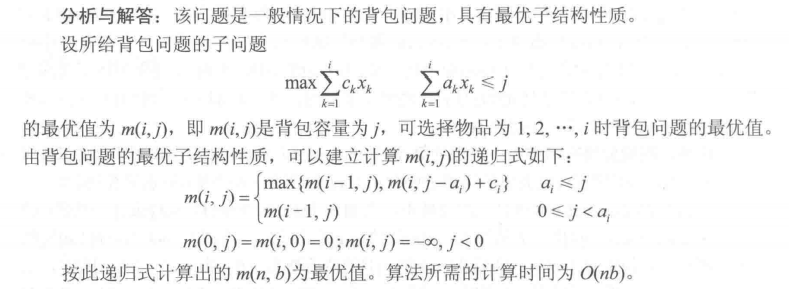
**随机化算法**

**线性同余法、舍伍德算法、跳跃表、拉斯维加斯算法、N后问题、整数因子分解、蒙特卡洛算法、主元素问题、素数测试、现行规划问题和单纯性算法、增广路算法:**

**图灵机、p类与NP类问题、多项式时间变换、NP完全问题、旅行商售货问题、集合覆盖问题、集合覆盖问题**

矩阵连乘strassen：:将n阶矩阵分块为m×m的矩阵.用传统方法求两个m阶矩阵的乘积需要计算O(M)次2个2\*×2\*矩阵的乘积.用Strassen矩阵乘法计算两个2\*×2\*矩阵的乘积需要的计算时间为O(7\*)，因此算法的计算时间为o(M)。

大整数乘法：三等分，计算u012v012的值，计算abcde值，时间复杂度O(nlog35).

整数线性规划

霍夫曼树：树深为n-1，第一个字符编码长度n-1，自底而上第i个根节点中的数为求和fk，利用数学归纳法易证<fk+2，保证树的结构。

最优前缀码：任何最优前缀码所相应的编码二叉树是一棵完全二叉树，有n个叶结点和n-1个内结点.用1位表示1个结点的类型，1表示内结点，0表示叶结点，共需2n-1位.对编码树的前序遍历可以唯一表示该编码树结构.