《算法分析与设计》期末复习

西南石油大学理学院-谭兵

一、选择题

1.应用 Johnson 法则的流水作业调度采用的算法是（D）

A. 贪心算法　　B. 分支限界法　　C.分治法　　**D. 动态规划算法**

3.动态规划算法的基本要素为（C）

A. 最优子结构性质与贪心选择性质

B. 重叠子问题性质与贪心选择性质

C. 最优子结构性质与重叠子问题性质

D. 预排序与递归调用

4.算法分

析中，记号O表示（B），记号Ω表示（A），记号Θ表示（若 f(n)=O(g(n))且 f(n)= Ω(g(n))，则 f(n)= Θ(g(n))， f(n)和 g(n)同阶。）。

A. 渐进下界 B.渐进上界 C.非紧上界 D.紧渐进界 E.非紧下界

备注：

O 定义： 若∃ c>0,N>0 满足 0≤f(n)≤cg(n) ∀ n≥N，则称函数 f(n)在 n 充分大时有上界，且 g(n)是 f(n)的一个上界，记为 f(n)=O(g(n))。g(n)是 f(n)的渐进上界。

Ω 定义：若∃ c>0,N>0 满足 0≤cg(n)≤f(n) ∀ n≥N，则 f(n)= Ω(g(n))，g(n)是 f(n)的渐进下界。

Θ 定义：若 f(n)=O(g(n))且 f(n)= Ω(g(n))，则 f(n)= Θ(g(n))， f(n)和 g(n)同阶。

5. 以下关于渐进记号的性质是正确的有：(A)

$A.f(n)=\Theta(g(n)),g(n)=\Theta(h(n))\Leftrightarrow f(n)=\Theta(h(n))$

$B.f(n)=O(g(n)),g(n)=O(h(n))\Leftrightarrow h(n)=O(f(n))$

$C.O(f(n))+O(g(n))=O(\min\{f(n),g(n)\})$

$D.f(n)=O(g(n))\Leftrightarrow g(n)=O(f(n))$

6.能采用贪心算法求最优解的问题，一般具有的重要性质为：（A）

A. 最优子结构性质与贪心选择性质

B. 重叠子问题性质与贪心选择性质

C. 最优子结构性质与重叠子问题性质

D. 预排序与递归调用

7.回溯法在问题的解空间树中，按（D）策略，从根结点出发搜索解空间树。

A.广度优先 B.活结点优先 C.扩展结点优先 **D.深度优先**

8.分支限界法在问题的解空间树中，按(A)策略，从根结点出发搜索解空间树。

A．广度优先 B. 活结点优先 C.扩展结点优先 D. 深度优先

10.回溯法的效率不依赖于以下哪一个因素？（C ）

A. 产生 x[k]的时间；

B. 满足显约束的 x[k]值的个数；

C. 问题的解空间的形式；

D. 计算上界函数 bound 的时间；

E. 满足约束函数和上界函数约束的所有 x[k]的个数。

F. 计算约束函数 constraint 的时间；

11. 常见的两种分支限界法为（ **D** ）

A. 广度优先分支限界法与深度优先分支限界法

B. 队列式（FIFO）分支限界法与堆栈式分支限界法

C. 排列树法与子集树法

D. 队列式（FIFO）分支限界法与优先队列式分支限界法

12. NP 类语言在图灵机下的定义为（**B** ）

A. NP={L|L 是一个能在非多项式时间内被一台 NDTM 所接受的语言}；

B. NP={L|L 是一个能在多项式时间内被一台 NDTM 所接受的语言}；

C. NP={L|L 是一个能在非多项式时间内被一台 DTM 所接受的语言}；

D. NP={L|L 是一个能在多项式时间内被一台 DTM 所接受的语言}；

13.记号 O 的定义正确的是（**A**）。

A. O(g(n)) = { f(n) | 存在正常数 c 和 n0 使得对所有 $n \geq n_0$ 有：$0 \leq$ f(n) $\leq$ cg(n) }；

B. O(g(n)) = { f(n) | 存在正常数 c 和 n0 使得对所有 $n \geq n0$ 有：$0 \leq$ cg(n) $\leq$ f(n) }；

C. O(g(n)) = { f(n) | 对于任何正常数 c>0，存在正数和 $n_0 > 0$ 使得对所有 $n \geq n_0$ 有：$0 \leq f(n) < cg(n)$ }；

D. O(g(n)) = { f(n) | 对于任何正常数 c>0，存在正数和 $n_0 > 0$ 使得对所有 $n \geq n_0$ 有：$0 \leq cg(n) < f(n)$ }；

14. 记号Ω的定义正确的是(A)

$A.f(n)=\Omega(g(n)),\exists c>0,N>0$满足$0\le cg(n)\le f(n),\forall n\ge N$

$B.g(n)=\Omega(f(n)),\exists c>0,N>0$满足$0\le cg(n)\le f(n),\forall n\ge N$

$C.f(n)=\Omega(g(n)),\forall c>0,N>0$满足$0\le cg(n)\le f(n),\forall n\ge N$

$D.g(n)=\Omega(f(n)),\forall c>0,N>0$满足$0\le cg(n)\le f(n),\forall n\ge N$

15.计算机算法的正确描述是：(D)

A. 一个算法是求特定问题的运算序列

B. 算法是一个有穷规则的集合，规定了一个解决某一特定类型的问题的运算序列

C. 算法是一个对任一有效输入能够停机的图灵机

D. 一个算法，它是满足 5 个特性(有限性、确定性、可行性、输入、输出)的程序

16.影响程序执行时间的因素有哪些？(ABCD)

A. 算法设计的策略 B. 问题的规模

C. 编译程序产生的机器代码质量 D. 计算机执行指令的速度

17.用数量级形式表示的算法执行时间称为算法的(A)

A. 时间复杂度 B. 空间复杂度 C. 处理器复杂度 D. 通信复杂度

18.时间复杂性为多项式界的算法有：(C)

A. 快速排序算法 B. n-后问题 **C. 计算 pi 值** D. prim 算法

19.对于并行算法与串行算法的关系，正确的理解是：（ ）

A. 高效的串行算法不一定是能导出高效的并行算法

B. 高效的串行算法不一定隐含并行性

C. 串行算法经适当的改造有些可以变化成并行算法

D. 用串行方法设计和实现的并行算法未必有效

20.衡量近似算法性能的重要标准有：(AC)

A. 算法复杂度 B. 问题复杂度 **C. 解的最优近似度** D. 算法的策略

21.分治法的适用条件是，所解决的问题一般具有这些特征：（**ABCD** ）

A. 该问题的规模缩小到一定的程度就可以容易地解决；

B. 该问题可以分解为若干个规模较小的相同问题；

C. 利用该问题分解出的子问题的解可以合并为该问题的解

D. 该问题所分解出的各个子问题是相互独立的。

22.具有最优子结构的算法有：（**D** ）

A. 概率算法 B. 回溯法 C. 分支限界法 D. 动态规划法

23.下列哪些问题是典型的 NP 完全问题：（ CD）

A. 排序问题 B. n-后问题 C. m-着色问题 D. 旅行商问题

24.适于递归实现的算法有：（ ）

A. 并行算法 B. 近似算法 C. 分治法 D. 回溯法

二、填空题

1.下面程序段的所需要的计算时间为（$O(n^2)$）。

```
int MaxSum(int n, int *a, int &besti, int &bestj)
{
 int sum=0;
 for(int i=1;i<=n;i++){
      int thissum=0;
      for(int j=i;j<=n;j++){
           thissum+=a[j];
          if(thissum>sum)  {
               sum=thissum;
               besti=i;
               bestj=j;
          }
      }
 }
      return sum;
}
```

2.回溯框架与（排列树）算法框架。

3.下面 **Matlab** 代码运行后的结果 **S=**（ **22**）

主程序：n=6;
S=f1(n)
子程序：function y=f1(n)
if(n==0)
y=1;
else
y=n+f1(n-1);
end
6+5+4+3+2+1+1=22

4.下列 **matlab** 代码用于近似计算 **pi**，请补充完整。

```
clear;clc;close all;
n=10000;
xs=rand(2,n);xs=2*xs-1;
g=inline('sum(x.*x)-1');
k=sum(g(xs)<0)+sum(g(xs)==0);
mypi=(4*k/n);
```

5. 已知递归方程，$\begin{cases} f(n)=2f(n-1)+n \\ f(0)=3 \end{cases}$ 则 f(n) 的非递归表达式为

$f(n)=2^{n+1}+3\cdot 2^n-n-2$

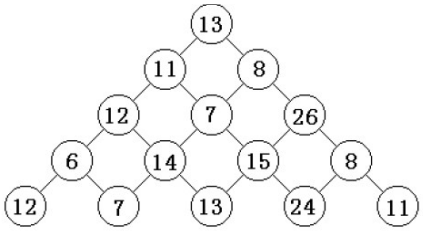
2. 已知非齐次递归方程：$\begin{cases} f(n)=bf(n-1)+g(n) \\ f(0)=c \end{cases}$，其中，b、c 是常数，

g(n)是 n 的某一个函数。则 f(n)的非递归表达式为：$f(n)=cb^n+\sum_{i=1}^{n} b^{n-i}g(i)$。

6.有 11 个待安排的活动，它们具有下表所示的开始时间与结束时间，如果以贪心算法求解这些活动的最优安排（即为活动安排问题：在所给的活动集合中选出最大的相容活动子集合），得到的最大相容活动子集合为活动（ {1，4，8，11} ）。

i	1	2	3	4	5	6	7	8	9	10	11
S[i]	1	3	0	5	3	5	6	8	8	2	12
f[i]	4	5	6	7	8	9	10	11	12	13	14

8. 数塔问题。有形如下图所示的数塔，从顶部出发，在每一结点可以选择向左走或是向右走，一起走到底层，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;_____1_____;c++)

if(t[r+1][c]>t[r+1][c+1]) ____2____;

else ____3____ ;

10. 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]=∞;____1____ }

d[s]=0 **Relax(u,v,w)** if d[v]>d[u]+w(u,v) then

{ d[v]=d[u]+w[u,v];

2 }

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 3

do 4

6. 所谓贪心选择性质是指（所求问题的整体最优解可以通过一系列局部最优解的选择，即贪心选择来达到 ）。

7.所谓最优子结构性质是指（问题的最优解包含了其子问题的最优解）。

8.回溯法是指（具有限界函数的深度优先生成法）。

9.用回溯法解题的一个显著特征是在搜索过程中动态产生问题的解空间。在任何时刻，算法只保存从根结点到当前扩展结点的路径。如果解空间树中从根结点到叶结点的最长路径的长度为 **h(n)**，则回溯法所需的计算空间通常为（**O(h(n))**

10. 用回溯法解 **0/1** 背包问题时，该问题的解空间结构为（子集树）结构。

11.用回溯法解批处理作业调度问题时，该问题的解空间结构为（排列树）结构。

20. 用回溯法解图的 m 着色问题时，使用下面的函数 OK 检查当前扩展结点的每一个儿子所相应的颜色的可用性，则需耗时（渐进时间上限）（ ）。

```
Bool Color::OK(int k)
{//
  for(int j=1;j<=n;j++)
    if((a[k][j]= =1)&&(x[j]= =x[k])) return false;
  return true;
}
```

21. 旅行售货员问题的解空间树是（ ）。

12. 算法就是一组有穷的（规则），它们规定了解决某一特定类型问题的（ 一系列运算）。

13. 算法的复杂性是（ 算法效率）的度量，是评价算法优劣的重要依据。计算机的资源最重要的是（时间和空间 ）资源。因而，算法的复杂性有（时间复杂度和空间复杂度 ）之分。

14.f(n)= 6×2^n +n^2 ，f(n)的渐进性态 f(n)= O(2^n)

15.贪心算法总是做出在当前看来（最好）的选择。也就是说贪心算法并不从整体最优考虑，它所做出的选择只是在某种意义上的（局部最优选择）。许多可以用贪心算法求解的问题一般具有 2 个重要的性质：（贪心选择和最优子结构 ）

三、简答

1.算法的重要特性是什么？算法分析目的是什么？

重要特性：输入、输出、确定性、有限性、可实现性。
算法分析的目的：分析算法占用计算机资源的情况，对算法做出比较和评价，设计出更好的算法。

2.分支限界法的基本思想和主要步骤是什么？

基本思想：分支定界法的基本思想是对有约束条件的最优化问题的所有可行解（数目有限）空间进行搜索。
用分支限界法设计算法的步骤是：
(1)针对所给问题，定义问题的解空间（对解进行编码）
(2)确定易于搜索的解空间结构（按树或图组织解）
(3)以广度优先或以最小耗费（最大收益）优先的方式搜索解空间，并在搜索过程中用剪枝函数避免无效搜索。

3.简述动态规划算法的基本思想和主要步骤？

基本思想：将待求解问题分解成若干个子问题；由于子问题有重叠，动态规划算法能够保存已解决的子问题的答案，而在需要时再找出已求得的答案，就可以避免大量重复计算
设计动态规划算法的主要步骤为：
1.找出最优解的性质，并刻划其结构特征。
2.递归地定义最优值。
3.以自底向上的方式计算出最优值。
4.根据计算最优值时得到的信息，构造最优解。

4. 备忘录方法和动态规划算法相比有何异同？

动态规划算法的基本要素：
1　最优子结构性质
当问题的最优解包含了其子问题的最优解时，称该问题具有最优子结构性质。
2　重叠子问题性质
动态规划算法对每个问题只解一次，将其解保存在一个表格中，当再次需要解此问题时，用常数时间查看一下结果。因此，用动态规划算法通常只需要多项式时间。

备忘录方法：
•用一个表格来保存已解决的子问题的答案，用的时候查表即可。
•采用的递归方式是自顶向下。
•控制结构与直接递归相同，区别在于备忘录方式为每个解过的子问题建立备忘录。

•初始化为每个子问题的记录存入一个特殊的值，表示并未求解。在求解过程中，查看相应记录如果是特殊值，表示未求解，否则只要取出该子问题的解答即可。

备忘录方法与动态规划和递归的区别：

1、动态规划是自低向上 ，备忘录方法是自顶向下，递归是自顶向下

2、动态规划每个子问题都要解一次，但不会求解重复子问题；备忘录方法只解哪些确实需要解的子问题；递归方法每个子问题都要解一次，包括重复子问题• 。

4.简单描述分治法的基本思想。

分治法的基本思想是将一个规模为 n 的问题分解为 k 个规模较小的子问题，这些子问题互相独立且与原问题相同。递归地解这些子问题，然后将各个子问题的解合并得到原问题的解。

5.算法的复杂度分析涉及哪些方面？

算法复杂度主要是**时间复杂度和空间复杂度**

6.动态规划法的指导思想是什么？

其基本思想也是将待求解问题分解成若干个子问题，先求解子问题，然后从这些子问题的解得到原问题的解。

7.简单描述分治法的基本思想。

分治算法的基本思想是将一个规模为 N 的问题分解为 K 个规模较小的子问题，这些子问题相互独立且与原问题性质相同。求出子问题的解，就可得到原问题的解。即一种分目标完成程序算法，简单问题可用二分法完成。

6、**分治法的基本思想**是将一个规模为 n 的问题分解为 k 个规模较小的子问题，这些子问题互相独立且与原问题相同；对这 k 个子问题分别求解。如果子问题的规模仍然不够小，则再划分为 k 个子问题，如此递归的进行下去，直到问题规模足够小，很容易求出其解为止；将求出的小规模的问题的解合并为一个更大规模的问题的解，自底向上逐步求出原来问题的解。

7、**“最优化原理”**用数学化的语言来描述：假设为了解决某一优化问题，需要依次作出 n 个决策 D1，D2，…，Dn，如若这个决策序列是最优的，对于任何一个整数 k，1 ＜ k ＜ n，不论前面 k 个决策是怎样的，以后的最优决策只取决于由前面决策所确定的当前状态，即以后的决策 Dk+1，Dk+2，…，Dn 也是最优的。

8、某个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**。

9、**回溯法的基本思想**是在一棵含有问题全部可能解的状态空间树上进行深度优先搜索，解为叶子结点。搜索过程中，每到达一个结点时，则判断该结点为根的子树是否含有问题的解，如果可以确定该子树中不含有问题的解，则放弃对该子树的搜索，退回到上层父结点，继续下一步深度优先搜索过程。在回溯法中，并不是先构造出整棵状态空间树，再进行搜索，而是在搜索过程，逐步构造出状态空间树，即边搜索，边构造。

10、 P(Polynomial 问题)：也即是多项式复杂程度的问题。

NP 就是 Non-deterministic Polynomial 的问题，也即是多项式复杂程度的非确定性问题。

NPC(NP Complete)问题，这种问题只有把解域里面的所有可能都穷举了之后才能得出答案，这样的问题是 NP 里面最难的问题，这种问题就是 NPC 问题。

12. 算法分析的目的是什么？

分析算法占用计算机资源的情况，对算法做出比较和评价，设计出更好的算法。

13. 算法的时间复杂性与问题的什么因素相关？

算法的时间复杂性与问题的规模相关，是问题大小 n 的函数

6.算法的渐进时间复杂性的含义？

当问题规模很大时，精确的计算 T(n)是很难实现而且也是没有必要的。对于算法时间性能的分析无需非要得到时间复杂度 T(n)的精确值，它的变化趋势和规律也能清楚地反映算法的时间耗费。基于此，引入了渐进时间复杂度作为时间性能分析的依据，它的含义就是：在问题规模 n 趋于无穷大时算法时间复杂度 T(n)的渐进上界，即函数 T(n)的数量级(阶)。

15.最坏情况下的时间复杂性和平均时间复杂性有什么不同？

5. 最坏情况下的时间复杂性和平均时间复杂性考察的是 n 固定时，不同输入实例下的算法所耗时间。最坏情况下的时间复杂性取的输入实例中最大的时间复杂度：

$$W(n) = \max\{ T(I) \}, \quad I \in Dn$$

平均时间复杂性是所有输入实例的处理时间与各自概率的乘积和：

$$A(n) \quad \sum P(I)T(n, I) \quad I \in Dn$$

16.简述二分检索（折半查找）算法的基本过程。

6. 设输入是一个按非降次序排列的元素表 A[i：j] 和 x，选取 A[(i+j)/2]与 x 比较，如果 A[(i+j)/2]=x，则返回(i+j)/2，如果 A[(i+j)/2]<x，则 A[i：(i+j)/2-1]找 x，否则在 A[(i+j)/2+1 j] 找 x。上述过程被反复递归调用。

17.背包问题的目标函数和贪心算法最优化量度相同吗？

ANS：不相同。目标函数：获得最大利润。最优量度：最大利润/重量比。

18.采用回溯法求解的问题，其解如何表示？有什么规定？

8. 问题的解可以表示为 n 元组：（$x_1, x_2, \cdots\cdots x_n$），$x_i \in S_i$，$S_i$ 为有穷集合，$x_i \in S_i$，（$x_1, x_2, \cdots\cdots x_n$）具备完备性，即（$x_1, x_2, \cdots\cdots x_n$）是合理的，则（$x_1, x_2, \cdots\cdots x_i$）（i<n)一定合理。

19.回溯法的搜索特点是什么？

9. 在解空间树上跳跃式地深度优先搜索，即用判定函数考察 x[k]的取值，如果 x[k]是合理的就搜索 x[k]为根节点的子树，如果 x[k]取完了所有的值，便回溯到 x[k-1]。

21.为什么用分治法设计的算法一般有递归调用？

子问题的规模还很大时，必须继续使用分治法，反复分治，必然要用到递归

22.为什么要分析最坏情况下的算法时间复杂性？

最坏情况下的时间复杂性决定算法的优劣，并且最坏情况下的时间复杂性较平均时间复杂性游可操作性。

13. T(n)是某算法的时间复杂性函数，f(n)是一简单函数，存在正整数 No 和 C，n〉No，有 T(n)<f(n)，这种关系记作 T(n)=0(f(n))。
14. 二分检索算法的最多的比较次数为 log n
15. 最坏情况下快速排序退化成冒泡排序，需要比较 n^2 次。
16. 是一种依据最优化量度依次选择输入的分级处理方法。基本思路是：首先根据题意，选取一种**量度标准；然后**按这种量度标准对这 n 个输入排序，依次选择输入量加入部分解中。如果当前这个输入量的加入，不满足约束条件，则不把此输入加到这部分解中。
17. 回溯法的解（$x_1, x_2, \cdots\cdots x_n$）的隐约束一般指个元素之间应满足的某种关系。
18. 讲数组一分为二，分别对每个集合单独排序，然后将已排序的两个序列归并成一个含 n 个元素的分好类的序列。如果分割后子问题还很大，则继续分治，直到一个元素。
19. 快速排序的基本思想是在待排序的 N 个记录中任意取一个记录，把该记录放在最终位置后，数据序列被此记录分成两部分。所有关键字比该记录关键字小的放在前一部分，所有比它

20. 在定义一个过程或者函数的时候又出现了调用本过程或者函数的成分，既调用它自己本身，这称为直接递归。如果过程或者函数 P 调用过程或者函数 Q，Q 又调用 P，这个称为间接递归。消除递归一般要用到栈这种数据结构。

21. 哈密顿环是指一条沿着图 G 的 N 条边环行的路径，它的访问每个节点一次并且返回它的开始位置。

22. 当前选择的节点 X[k]是从未到过的节点，即 X[k]≠X[i](i=1,2,„,k-1)，且 C(X[k-1], X[k])≠∞，如果 k=-1，则 C(X[k], X[1])≠∞。

23. 思路是：最初生成树 T 为空，依次向内加入与树有最小邻接边的 n-1 条边。处理过程：首先加入最小代价的一条边到 T，根据各节点到 T 的邻接边排序，选择最小边加入，新边加入后，修改由于新边所改变的邻接边排序，再选择下一条边加入，直至加入 n-1 条边。

7.贪心算法的基本思想？

基本思想：从问题的初始解出发逐步逼近给定的目标，每一步都做出当前看来是最优的选择（贪心选择），最终得到整个问题的最优解

8.回溯法的解（$x_1, x_2, \cdots\cdots, x_n$）的隐约束一般指什么？

回溯法的解（$x_1, x_2, \cdots\cdots, x_n$）的隐约束一般指个元素之间应满足的某种关系。

9.阐述归并排序的分治思路。

将数组一分为二，分别对每个集合单独排序，然后将已排序的两个序列归并成一个含 **n** 个元素的分好类的序列。如果分割后子问题还很大，则继续分治，直到一个元素。

10.快速排序的基本思想是什么？

快速排序的基本思想是在待排序的 **N** 个记录中任意取一个记录，把该记录放在最终位置后，数据序列被此记录分成两部分。所有关键字比该记录关键字小的放在前一部分，所有比它大的放置在后一部分，并把该记录排在这两部分的中间，这个过程称作一次快速排序。之后重复上述过程，直到每一部分内只有一个记录为止。

四、算法分析与设计

1.用展开法求解递推关系：

$$T(n)=\begin{cases}1 & n=1\\ 2T(n-1)+1 & n>1\end{cases}$$

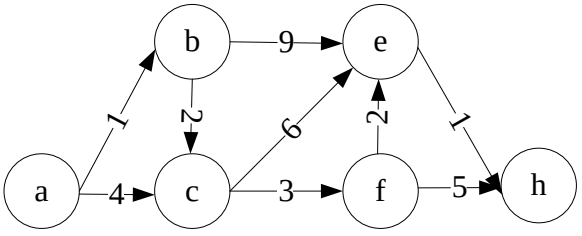
解答：已知非齐次递归方程：$f(n)=\begin{cases}c & n=0\\ bT(n-1)+g(n) & n>1\end{cases}$。其中 b，c 为常数，

则 f(n)的非递归表达式为： $f(n)=cb^n+\sum_{i=1}^{n}b^{n-i}g(i)$。

根据题意得：b=2，c=1，有

$$\begin{aligned}T(n)&=cb^{n-1}+\sum_{i=1}^{n-1}b^{n-1-i}g(i)\\ &=2^{n-1}+2^{n-2}+2^{n-3}+\cdots+2^2+2+1\\ &=2^n-1\end{aligned}$$

2.请写出 Dijkstra 算法的具体计算步骤，并根据代码求从定点 a 到其它所有顶点的最短距离。



伪代码：

1. 初始化数组 dist、path 和 s；
2. while (s 中的元素个数<n)

2.1 在 dist[n]中求最小值，其下标为 k（则 vk 为正在生成的终点）；
2.2 输出 dist[j]和 path[j]；
2.3 修改数组 dist 和 path；
2.4 将顶点 vk 添加到数组 s 中；

5.利用分支限界算法求解如下背包问题，要求给出解空间树的剪支搜索过程。

$$\max \sum_{i=1}^{n} b_i x_i \qquad W =<10,8,5>$$

$$\begin{cases} \sum_{i=1}^{n} w_i x_i \le \max W \\ x_i \in \{0,1\}, 1 \le i \le n \end{cases} \qquad \begin{array}{l} B =<5,4,1> \\ \max W = 16 \end{array}$$

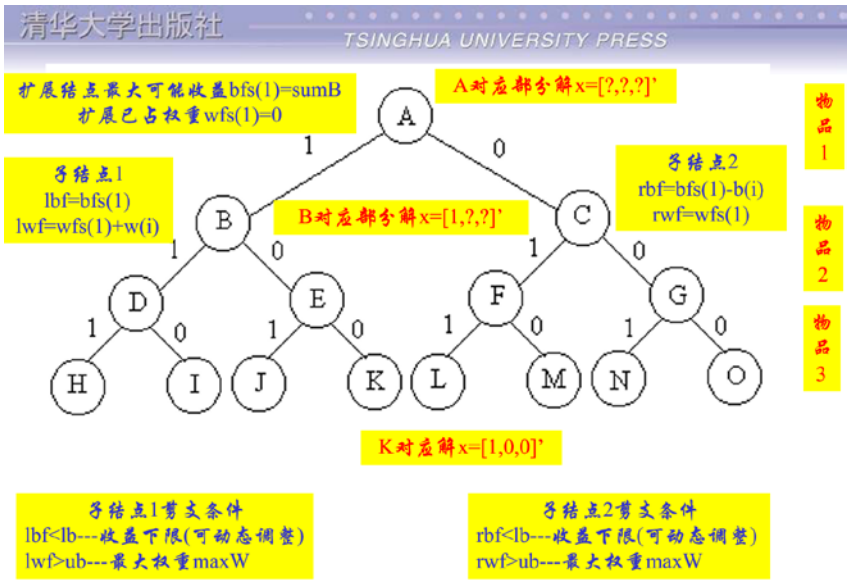
基本步骤：

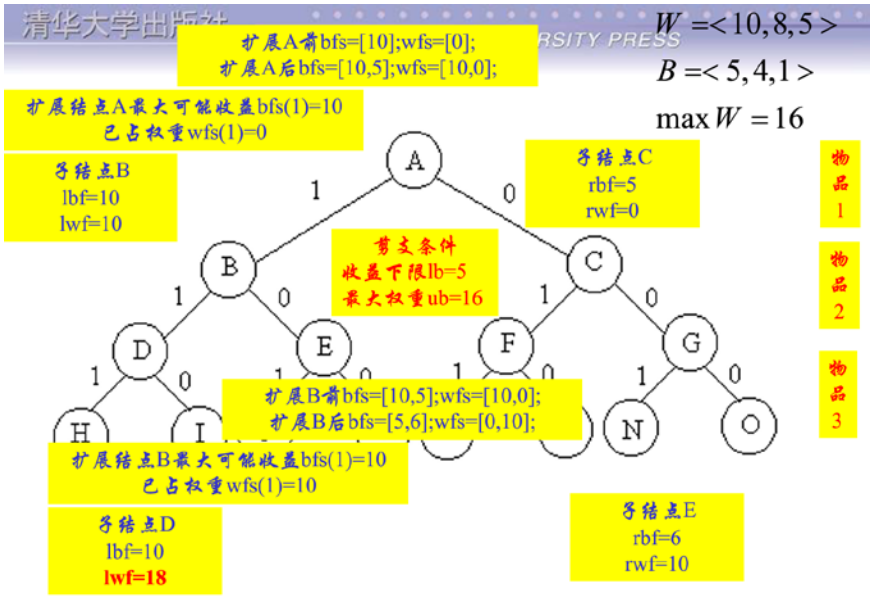
step1 限界函数确定目标函数的界[lb, ub] 。

step2 按照广度优先策略遍历问题的解空间树：依次搜索当前结点的所有子结点，分别估算这些子结点的目标函数可能取值，若超出界限，将其丢弃；否则，将其加入活结点队列。

step3 按一定原则从活结点队列中取出一个作为当前扩展结点，重复上述过程，直到找到最优解。

注意：可以根据需要动态调整界限；限界方式不固定





6.用动态规划法求 $A_{10*30}B_{30*20}C_{20*10}D_{10*200}$ 运算量最小的乘积顺序。要求写出求解过程，并将结果填入数组 m[4][4]中。

解答：递归定义 $m[i][j]=\begin{cases}0 & i=j\\ \min\limits_{i\le k<j}\{m[i]m[k]+m[k+1][j]+p_{i-1}p_kp_j\} & i<j\end{cases}$

$p_0=10, p_1=30, p_2=20, p_3=10, p_4=200$

$m[1][1]=0$

$m[1][2]=m[1][1]+m[2][2]+p_0p_1p_2=0+0+10*30*20=6000$

$m[2][3]=m[2][2]+m[3][3]+p_1p_2p_3=0+0+30*20*10=6000$

$m[3][4]=m[3][3]+m[4][4]+p_2p_3p_4=20*10*200=40000$

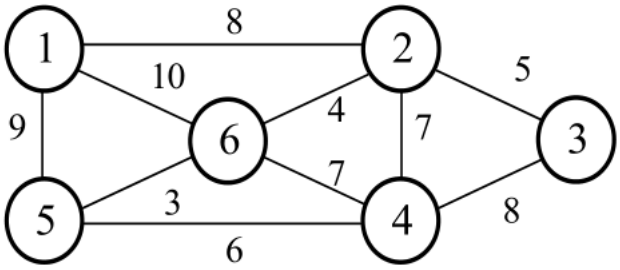
$$m[1][3]=\min\begin{cases}m[1][1]+m[2][3]+p_0p_1p_3=0+6000+10*30*10=9000\\ m[1][2]+m[3][3]+p_0p_2p_3=6000+0+10*20*10=8000\end{cases}$$

$$m[2][4]=\min\begin{cases}m[2][2]+m[3][4]+p_1p_2p_4=0+40000+30*20*200=160000\\ m[2][3]+m[4][4]+p_1p_3p_4=6000+0+30*10*200=66000\end{cases}$$

$$m[1][4]=\min\begin{cases}m[1][1]+m[2][4]+p_0p_1p_4=0+66000+10*30*200=126000\\ m[1][2]+m[3][4]+p_0p_2p_4=6000+40000+10*20*200=86000\\ m[1][3]+m[4][4]+p_0p_3p_4=8000+0+10*10*200=28000\end{cases}$$

$$m[i][j]=\begin{array}{c}\\1\\2\\3\\4\end{array}\begin{array}{c}\begin{array}{cccc}1&2&3&4\end{array}\\\begin{bmatrix}0&6000&8000&28000\\&0&6000&66000\\&&0&40000\\&&&0\end{bmatrix}\end{array}$$

7.用贪心法求下图的最小生成树




```
function [Q,P]=prim(W)
n=size(W,1);Q=zeros(n,n);V=ones(1,n);
P=-ones(1,n);P(1)=1;
M=max(W(W<inf))+1;
while norm(P-V)>0
      S=ones (n,n);S(P>0,P<0)=0;
      wmin=min(W(:)+M*S(:));
      [us,vs]=find(W+M*S==wmin);
      u=us(1);v=vs(1);
      Q(u,v)=W(u,v);
      P(v)=1;
end
```

8.请设计概率算法计算 $\int_a^b f(x)dx$，**给出伪代码或 matlab 代码。**

例如求 $\int_0^1 x^2 dx$，求 $\int_0^1 e^x dx$

```
clear all;clc;
n=1000000;N=0;
for i=1:n
      x=rand;
      y=rand;
      if y<x^2
            N=N+1;
      end
end
S=N/n
%%
结果输出为：0.3334
```

```
clear all;clc;
n=1000000;N=0;
for i=1:n
      x=rand;
      y=rand*exp(1);
      if y<exp(x)
            N=N+1;
      end
end
S=N/n*exp(1)
%%
结果输出为：1.7143
```

9.写一贪心算法，求解 0-1 背包问题。

0-1 背包问题描述：给定 n 种物品和一个背包。物品 i 的重量是 Wi，其价值为 Vi，背包的容量为 C。问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?若进一步讨论：就 0-1 背包问题的应用作简单的描述。

10a.对如下算法进行计算复杂度分析

```
MERGESORT(low，high)
      if low<high；
      then mid←(low，high)/2；
      MERGESORT(low，mid)；
      MERGESORT(mid+1，high)；
      MERGE(low，mid，high)；
endif
end MERGESORT
```

解答：递归方程为：$T(n)=\begin{cases} a & n=1 \\ 2T(n/2)+cn & n>1 \end{cases}$

令 $2^n=k$，解此递归方程，

$$\begin{aligned} T(n) &= 2T(n/2)+cn \\ &= 2\left(2T(n/4)+n/2\right)+cn \\ &= 4T(n/4)+2cn \\ &= 4\left(2T(n/8)+cn/4\right)+2cn \\ &= 8T(n/8)+3cn \\ &= 2^kT(1)+kcn \\ &= an+cn\log n \end{aligned}$$

所以时间复杂度为 $T(n)=O(an)+O(cn\log n)$

10b.对如下算法进行计算复杂度分析

```
procedure S1(P，W，M，X，n)
i←1; a←0
while i≤ n do
      if W(i)>M then return endif
      a←a+i
      i←i+1 ;
repeat
end
```

解答：i←1; a←0 时间为 O(1)，
while i≤ n do 循环 n 次，循环体内所用时间为 O(1)

所以总体时间为：$T(n)=O(1)+nO(1)=O(1)$

二、复杂性分析

1、 递归方程

$$T(n)=\begin{cases} a & n=1 \\ 2T(n/2)+cn & n>1 \end{cases}$$

$$\begin{aligned} T(n) &= 2(2T(n/4)+cn/2)+cn \\ &= 4T(n/4)+2cn \\ &\cdots\cdots \\ &= 2^kT(1)+kcn \\ &= an+cn\log n \end{aligned}$$

设 $n=2^k$

解递归方程：

2、

i←1 ;s←0　　时间为：O(1)

while　i≤ n do　　循环 n 次

　　循环体内所用时间为 O(1)

所以 总时间为:

T(n)=O(1)+ nO(1)= O(n)

3、最多的查找次数是 p-m+1 次

4、F2（2，n,1,1）的时间复杂度为：

T(n)=O(n-2);　因为 i≤n 时要递归调用 F2，一共是 n-2 次

当 n＝1 时 F1(n)的时间为 O(1)

当 n>1 时 F1(n)的时间复杂度与 F2(2,n,1,1)的时间复杂度相同即为为 O(n)

5、

　　xmax←A(1);j←1　　时间为：O(1)

for i←2 to n do　　循环最多 n-1 次

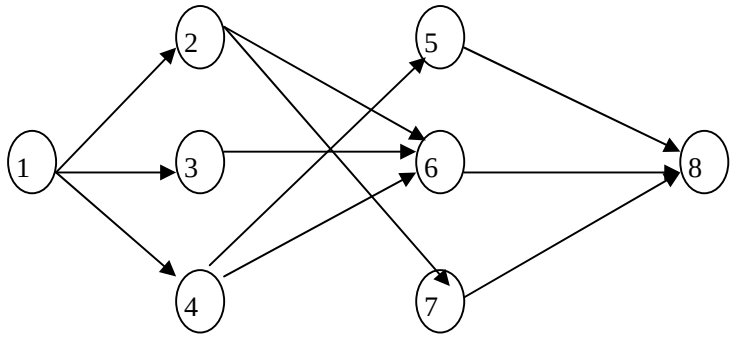
　所以 总时间为:

T(n)=O(1)+ (n-1)O(1)= O(n)

6、$\log_2 n+1$

三、算法理解

1、 写出多段图最短路经动态规划算法求解下列实例的过程，并求出最优值。



各边的代价如下：

C(1,2)=3， C(1,3)=5 ， C(1,4)=2

C(2,6)=8 ， C(2,7)=4 ， C(3,5)=5 ， C(3,6)=4， C(4,5)=2，C(4,6)=1

C(5,8)=4， C(6,8)=5 ， C(7,8)=6

1、

Cost(4,8)=0

Cost(3,7)= C(7,8)+0=6 ， D[5]=8

Cost(3,6)= C(6，8)+0=5, D[6]=8

Cost(3,5)= C(5，8)+0=4 D[7]=8

Cost(2,4)= min{C(4，6)+ Cost(3,6), C(4，5)+ Cost(3,5)}

= min{1+ 5, 2+4}=6 D[4]=6

Cost(2,3)= min{C(3，6)+ Cost(3,6) }

= min{4+5}=9 D[3]=5

Cost(2,2)= min{C(2，6)+ Cost(3,6), C(2，7)+ Cost(3,7)}

= min{8+5, 4+6}=10 D[2]=7

Cost(1,1)= min{C(1,2)+ Cost(2,2), C(1,3)+ Cost(2,3), C(1,4)+ Cost(2,4)}

= min{3+10, 5+9,2+6}= 8

D[1]=4

1→4→6→8

2、 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)

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

3、 快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)

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	i	p
65	70	75	80	85	55	50	2	2	8
65	2	75	80	85	55	50	70	3	7
65	2	50	80	85	55	75	70	4	6
65	2	50	55	85	80	75	70	4	6
55	70	75	80	85	65	50	2		

4、 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)

48, 12, 61, 3　　5, 19, 32, 7

48, 12　　61, 3　　5, 19　　32, 7

12, 48　　3, 61　　5, 19　　7, 32

3, 12, 48, 61　　5, 7, 19, 32

3, 5, 7, 12, 19, 32, 48, 61

8、写出归并排序算法对下列实例排序的过程。

(6,2,9,3,5,1,8,7)

调用第一层次　6, 2, 9, 3　　5, 1, 8, 7　　分成两个子问题

调用第二层次　6, 2　9, 3　　5, 1　8, 7　分成四个子问题

调用第三层次　6　2　9　3　　5　1　8　7　分成八个子问题

调用第四层次　只有一个元素返回上一层

第三层归并　2 , 6　3, 9　1, 5　7, 8　返回上一层

第二层归并　2 , 3, 6, 9　1, 5, 7, 8　返回上一层

第一层归并　1, 2 , 3, 5 , 6, 7, 8, 9　排序结束，返回主函数

9、写出用背包问题贪心算法解决下列实例的过程。

P=(18,12,4,1)

W=(12,10,8,3)

M=25

10、实例符合 P(i)/W(i)≥P(i+1)/W(i+1)的顺序。

CU←25, X←0

W[1]< CU: x[1]←1; CU←CU-W[1]=13;

W[2]< CU: x[2]←1; CU←CU-W[2]=3;

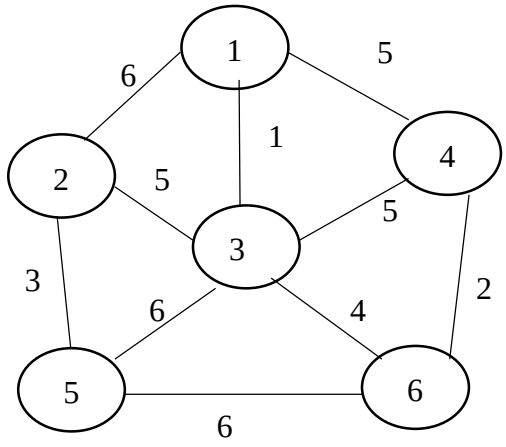
W[3]>CU: x[3]←CU/ W[3]=3/8;

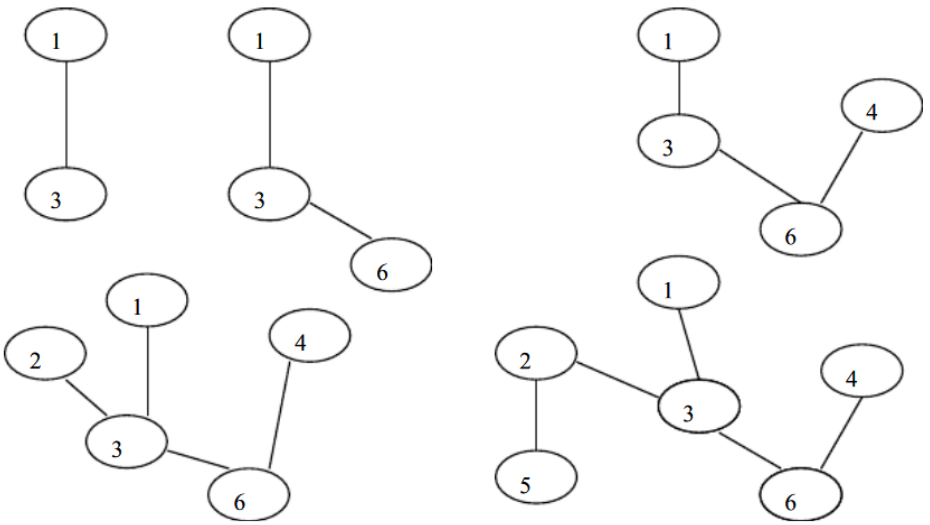
实例的解为：（1，1，3/8，0）

11、 有一个有序表为{1，3，9，12，32，41，45，62，75，77，82，95，100}，当使用二分查找值为 82 的结点时，经过多少次比较后查找成功并给出过程。

一共要要执行四次才能找到值为 82 的数

11.使用 prim 算法构造出如下图 G 的一棵最小生成树。





12.有如下函数说明

int f(int x,int y)
{
f=x Mod y +1;
}

已知 a=10,b=4,c=5 则执行 k=f(f(a+c,b),f(b,c))后，k 的值是多少并写出详细过程。

解答：本题即求 $k = f\left(f\left(15,4\right), f\left(4,5\right)\right)$

1. $f\left(15,4\right) = 15 \bmod 4 + 1 = 4$
2. $f\left(4,5\right) = 4 \bmod 5 + 1 = 5$
3. $f\left(4,5\right) = 4 \bmod 5 + 1 = 5$

$\therefore k = 5$

13.McCathy 函数定义如下：

当 x>100 时 m(x)=x-10;
当 x<=100 时 m(x)=m(m(x+11));
编写一个递归函数计算给定 x 的 m(x)值。

```
function f1=f(x)
if x>100
    f1=x-10;
elseif x<=100
    f1=f(f(x+11));
end
```

14.设计一个算法在一个向量 A 中找出最大数和最小数的元素。

```
%% 主函数
A=[5 2 1 8 6 9 10];
[Amax,Amin]=f(A)
结果输出为Amax=10，Amin=1
%% 子程序f.m
function [Amax,Amin]=f(A)
Amax=A(1);Amin=A(1);
n=length(A);
for i = 2:n
    if A(i)>Amax
        Amax=A(i);
    elseif A(i)<Amin
        Amin=A(i);
    end
end
```

1. 机器调度问题。

问题描述：现在有 n 件任务和无限多台的机器，任务可以在机器上得到处理。每件任务的开始时间为 s_i，完成时间为 f_i，$s_i<f_i$ 。$[s_i, f_i]$为处理任务 i 的时间范围。两个任务 i，j 重叠指两个任务的时间范围区间有重叠，而并非指 i，j 的起点或终点重合。例如：区间[1，4]与区间[2，4]重叠，而与[4，7]不重叠。一个可行的任务分配是指在分配中没有两件重叠的任务分配给同一台机器。因此，在可行的分配中每台机器在任何时刻最多只处理一个任务。最优分配是指使用的机器最少的可行分配方案。

问题实例：若任务占用的时间范围是{[1，4]，[2，5]，[4，5]，[2，6]，[4，7]}，则按时完成所有任务最少需要几台机器？（提示：使用贪心算法）

画出工作在对应的机器上的分配情况。

2. 已知非齐次递归方程：$\begin{cases} f(n) = bf(n-1) + g(n) \\ f(0) = c \end{cases}$，其中，b、c 是常数，g(n)是 n 的某一个函数。则 f(n)的非递归表达式为：$f(n) = cb^n + \sum_{i=1}^{n} b^{n-i} g(i)$。

现有 Hanoi 塔问题的递归方程为：$\begin{cases} h(n) = 2h(n-1) + 1 \\ h(1) = 1 \end{cases}$，求 h(n)的非递归表达式。

解：利用给出的关系式，此时有：b=2, c=1, g(n)=1, 从 n 递推到 1，有：

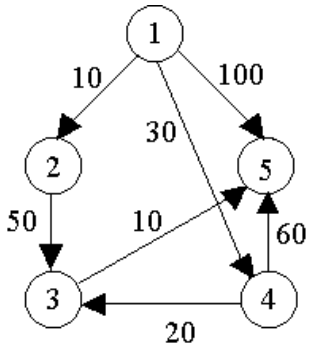
$$\begin{aligned} h(n) &= cb^{n-1} + \sum_{i=1}^{n-1} b^{n-1-i} g(i) \\ &= 2^{n-1} + 2^{n-2} + \ldots + 2^2 + 2 + 1 \\ &= 2^n - 1 \end{aligned}$$

3. 单源最短路径的求解。

问题的描述：给定带权有向图（如下图所示）G =(V,E)，其中每条边的权是非

负实数。另外，还给定 V 中的一个顶点，称为源。现在要计算从源到所有其它各顶点的最短路长度。这里路的长度是指路上各边权之和。这个问题通常称为单源最短路径问题。

解法：现采用 Dijkstra 算法计算从源顶点 1 到其它顶点间最短路径。请将此过程填入下表中。



迭代	S	u	dist[2	dist[3	dist[4	dist[5
初始	{1}	-	10	maxint	30	100
1						
2						
3						
4						

用分支限界法解装载问题时，对算法进行了一些改进，下面的程序段给出了改进部分；试说明斜线部分完成什么功能，以及这样做的原因，即采用这样的方式，算法在执行上有什么不同。

```
// 检查左儿子结点
  Type wt = Ew + w[i];   // 左儿子结点的重量
      if (wt <= c) {     // 可行结点
         if (wt > bestw) bestw = wt;
         // 加入活结点队列
         if (i < n) Q.Add(wt);
      }
  // 检查右儿子结点
      if (Ew + r > bestw && i < n)
          Q.Add(Ew);     // 可能含最优解
         Q.Delete(Ew);     // 取下一扩展结点
```

解答：斜线标识的部分完成的功能为：提前更新 bestw 值；

这样做可以尽早的进行对右子树的剪枝。具体为：算法 Maxloading 初始时将 bestw 设置为 0，直到搜索到第一个叶结点时才更新 bestw。因此在算法搜索到第一个叶子结点之前，总有 bestw=0，r>0 故 Ew+r>bestw 总是成立。也就是说，此时右子树测试不起作用。

为了使上述右子树测试尽早生效，应提早更新 bestw。又知算法最终找到的最优值是所求问题的子集树中所有可行结点相应重量的最大值。而结点所相应得重量仅在搜索进入左子树是增加，因此，可以在算法每一次进入左子树时更新 bestw 的值。

最长公共子序列问题：给定 2 个序列 X={x1,x2,…,xm}和 Y={y1,y2,…,yn}，找出 X 和 Y 的最长公共子序列。

由最长公共子序列问题的最优子结构性质建立子问题最优值的递归关系。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。其它情况下，由最优子结构性质可建立递归关系如下：

$$c[i][j]=\begin{cases}0 & i=0,j=0\\ c[i-1][j-1]+1 & i,j>0;x_i=y_j\\ \max\{c[i][j-1],c[i-1][j]\} & i,j>0;x_i\neq y_j\end{cases}$$

在程序中，b[i][j]记录 C[i][j]的值是由哪一个子问题的解得到的。

（1） 请填写程序中的空格，以使函数 LCSLength 完成计算最优值的功能。

```
void LCSLength(int m, int n, char *x, char *y, int **c, int **b)
{
        int i, j;
        for (i = 1; i <= m; i++) c[i][0] = 0;
        for (i = 1; i <= n; i++) c[0][i] = 0;
        for (i = 1; i <= m; i++)
           for (j = 1; j <= n; j++) {
              if (x[i]==y[j]) {
                   c[i][j]=c[i-1][j-1]+1; b[i][j]=1;}
             else if (c[i-1][j]>=c[i][j-1]) {
                   c[i][j]=c[i-1][j]; b[i][j]=2;}
              else { c[i][j]=c[i][j-1]; b[i][j]=3; }
              }
}
```

（2） 函数 LCS 实现根据 b 的内容打印出 Xi 和 Yj 的最长公共子序列。请填写程序中的空格，以使函数 LCS 完成构造最长公共子序列的功能（请将 b[i][j]的取值与（1）中您填写的取值对应，否则视为错误）。

```
void LCS(int i, int j, char *x, int **b)
{
      if (i ==0 || j==0) return;
      if (b[i][j]== 1) {
            LCS(i-1, j-1, x, b);
            cout<<x[i];
            }
      else if (b[i][j]== 2) LCS(i-1, j, x, b);
      else LCS(i, j-1, x, b);
}
```

对下面的递归算法，写出调用 f(4)的执行结果。

```
    void f(int k)
    { if( k>0 )
            { printf("%d\n  ",k);
              f(k-1);
              f(k-1);
            }
}
```

2. 设有 n 种面值为:

$d_1 \geq d_2 \geq \ldots\ldots \geq d_n$ 的钱币，需要找零钱 M，如何选择钱币 d_k，的数目 X_k，满足 $d_1 \times X_i + \ldots\ldots d_n \times X_n = M$ ，使得 $X_i + \ldots\ldots X_n$ 最小

请选择贪心策略，并设计贪心算法。

解：贪心原则：每次选择最大面值硬币。

CU←M;i←1;X←0　　// X 为解向量

While CU≠0 do

  X[i]←CU div d[i]　　// X[i]为第 i 中硬币数

  CU←CU-d[i]*X[i]

  i←i+1;

 repeat

3. 有n个物品，已知n=7, 利润为P=(10,5,15,7,6,18,3)，重量W=(2,3,5,7,1,4,1)，背包容积 M=15,物品只能选择全部装入背包或不装入背包，设计贪心算法，并讨论是否可获最优解。

解：定义结构体数组 G，将物品编号、利润、重量作为一个结构体：例如G[k]={1,10,2}

求最优解，按利润/重量的递减序，有

{5,6,1,6} {1,10,2,5}{6,18,4,9/2} {3,15,5,3} {7,3,1,3}{2,5,3,5/3} {4,7,7,1}

算法

```
procedure  KNAPSACK(P，W，M，X，n)
 //P(1：n)和 W(1；n)分别含有按
    P(i)/W(i)≥P(i+1)/W(i+1)排序的 n 件物品的效益值
    和重量。M 是背包的容量大小，而 x(1：n)是解向量//
 real P(1：n)，W(1：n)，X(1：n)，M，cu；
 integer i，n；
 X←0  //将解向量初始化为零//
 cu←M  //cu 是背包剩余容量//
 for i←1 to n do
    if  W(i)>cu  then  exit  endif
    X(i) ←1
     cu←cu-W(i)
 repeat
 end GREEDY-KNAPSACK
```

根据算法得出的解：

X=（1,1,1,1,1,0,0）获利润 52， 而解

（1,1,1,1, 0, 1,0）可获利润 54

因此贪心法不一定获得最优解。

简答题：

1、对于下列各组函数 f(n)和 g(n)，确定 f(n)=O(g(n))或 $f(n)=\Omega(g(n))$ 或 $f(n)=\theta(g(n))$，并简述理由。（12 分）

(1) $f(n)=\log n^2; g(n)=\log n+5;$

(2) $f(n)=2^n; g(n)=100n^2;$

(3) $f(n)=2^n; g(n)=3^n;$

解：简答如下：

（1）$\log n^2=\theta(\log n+5)$，（2）$2^n=\Omega(100n^2)$，（3）$2^n=o(3^n)$

2、试用分治法实现有重复元素的排列问题：设 $R=\{r_1,r_2,...,r_n\}$ 是要进行排列的 n 个元素，其中元素 $r_1,r_2,...,r_n$ 可能相同，试计算 R 的所有不同排列。（13 分）

解：解答如下：

```
Template<class Type>
void Perm(Type list[],int k,int m)
```

```
{
  if(k= =m){
    for(int i=0;i<=m;i++) cout<<list[i]; ……..（4 分）
    cout<<endl;
  }
  else for(int i=k;i<=m;i++)
      if(ok(list,k,i)){
        swap(list[k],list[i]);
        Perm(list,k+1,m);
        swap(list[k],list[i]); ……..（8 分）
        };
 }
```

其中 ok 用于判别重复元素。

```
Template<class>
int ok(Type list[],int k,int i)
{
  if(i>k)
     for(int t=k;t<I;t++)
       if(list[t]= =list[i]) return 0;
  return 1;
}……..（13 分）
```

3、试用分治法对一个有序表实现二分搜索算法。（12 分）

解：解答如下：

```
  Template<class>
int BinarySearch(Type a[],const Type& x,int n)
 {//假定数组 a[]已按非递减有序排列，本算法找到 x 后返回其在数组 a[]中
  的位置，//否则返回-1
  int left=0,right=n-1;
  while(left<=right){
    int middle=(left+right)/2; ……..（4 分）
    if(x= =a[middle]) return middle+1;
    if(x>a[middle]) left=middle+1; ……..（8 分）
    else right=middle-1;
   }
   return -1;
 }……..（12 分）
```

4、试用动态规划算法实现 0-1 闭包问题。（15 分）

解：解答如下：

```
Template<class>
void Knapsack(Type v,int w,int c,int n,Type **m)
{
  Int jMax=min(w[n]-1,c);
```

```
        for(int j=0;j<=jMax;j++) m[n][j]=0;
        for(int j=w[n];j<=c;j++) m[n][j]=v[n]; ……..（5 分）
        for(int i=n-1;i>1;i--){
          jMax=min(w[i]-1,c);
          for(int j=0;j<=jMax;j++) m[i][j]=m[i+1][j];
          for(int j=w[i];j<=c;j++) m[i][j]=max(m[i+1][j],m[i+1][j-
w[i]]+v[i]); ……..（8 分）
        };
        m[1][c]=m[2][c];
        if(c>=w[1]) m[1][c]=max(m[1][c],m[2][c-w[1]]+v[1]); ……..（10 分）
      }
      Template<class>
      Void Traceback(Type **m,int w,int c,int n,int x)
      {
        for(int i=1;i<n;i++)
          if(m[i][c]= =m[i+1][c]) x[i]=0; ……..（12 分）
          else { x[i]=1,c-=w[i];};
        x[n]=(m[n][c])?1:0;
      }……..（15 分）
```

5、试用贪心算法求解下列问题：将正整数 n 分解为若干个互不相同的自然数之和，使这些自然数的乘积最大。（15 分）

解：解答如下：

```
      void dicomp(int n,int a[])
      {
        k=1;
        if(n<3){ a[1]=0;return;};
        if(n<5){ a[k]=1;a[++k]=n-1;return;}; ……..（5 分）
        a[1]=2;n-=2;
    while(n>a[k]){
         k++;
         a[k]=a[k-1]+1;
         n-=a[k];
        };……..（10 分）
        if(n= =a[k]){ a[k]++;n--;};
        for(int i=0;i<n;i++) a[k-i]++;
      }……..（15 分）
```

6、试用动态规划算法实现最大子矩阵和问题：求 $m \times n$ 矩阵 A 的一个子矩阵，使其各元素之各为最大。（15 分）

解：解答如下：

```
      int MaxSum2(int m,int n,int **a)
      {
        int sum=0;
```

```
    int *b=new int[n+1];
    for(int i=1;i<=m;i++){
       for(int k=1;k<=n;k++) b[k]=0; ……..（5 分）
       for(int j=i;j<=m;j++){
          for(int k=1;k<=n;k++) b[k]+=a[j][k];
          int max=MaxSum(n,b);
          if(max>sum) sum=max;
       }
    }
    return sum; ……..（10 分）
  }

  int MaxSum(int n,int *a)
  {
      int sum=0,b=0;
      for(int i=1;i<=n;i++){
         if(b>0) b+=a[i];
         else b=a[i];
         if(b>sum) sum=b;
      }
      Return sum; ……..（15 分）
  }
```

7、试用回溯法解决下列整数变换问题：关于整数 i 的变换 f 和 g 定义如下：$f(i)=3i; g(i)=\lfloor i/2 \rfloor$。对于给定的两个整数 n 和 m，要求用最少的变换 f 和 g 变换次数将 n 变为 m。（18 分）

解：解答如下：

```
    void compute()
    {
       k=1;
       while(!search(1,n)){
          k++;
          if(k>maxdep) break;
          init();
       };……..（6 分）
       if(found) output();
    else cout<<”No Solution!”<<endl;
  }……..（9 分）

  bool search(int dep,int n)
 {
    If(dep>k) return false;
    for(int i=0;i<2;i++){
       int n1=f(n,i);t[dep]=i; ……..（12 分）
```

```
        if(n1= =m||search(dep+1,n1)){
            Found=true;
            Out();
            return true;
        }
    return false; ……..（18 分）
}
```