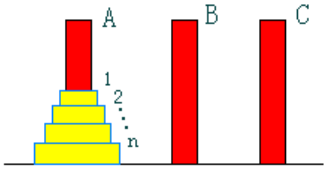
《算法分析与设计》期末复习题

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

2.Hanoi 塔问题如下图所示。现要求将塔座 A 上的的所有圆盘移到塔座 B 上，并仍按同样顺序叠置。移动圆盘时遵守 Hanoi 塔问题的移动规则。由此设计出解 Hanoi 塔问题的递归算法正确的为：（B）



Hanoi 塔

A.
```
void hanoi(int n, int A, int C, int B)
{
    if (n > 0)
    {
       hanoi(n-1,A,C, B);
       move(n,a,b);
       hanoi(n-1, C, B, A);
    }
```

B.
```
void hanoi(int n, int A, int B, int C)
{
    if (n > 0)
    {
       hanoi(n-1, A, C, B);
       move(n,a,b);
       hanoi(n-1, C, B, A);
    }
```

C.
```
void hanoi(int n, int C, int B, int A)
{
    if (n > 0)
    {
       hanoi(n-1, A, C, B);
       move(n,a,b);
       hanoi(n-1, C, B, A);
    }
```

```
D. void hanoi(int n, int C, int A, int B)
   {
       if (n > 0)
       {
          hanoi(n-1, A, C, B);
          move(n,a,b);
          hanoi(n-1, C, B, A);
       }
```

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（D）。

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

5. 以下关于渐进记号的性质是正确的有：（A）

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

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

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

9. 程序块（A）是回溯法中遍历排列树的算法框架程序。

A.

```
void backtrack (int t)
{
  if (t>n) output(x);
    else
      for (int i=t;i<=n;i++) {
        swap(x[t], x[i]);
        if (legal(t)) backtrack(t+1);
        swap(x[t], x[i]);
      }
}
```

B.

```
void backtrack (int t)
{
  if (t>n) output(x);
    else
      for (int i=0;i<=1;i++) {
        x[t]=i;
        if (legal(t)) backtrack(t+1);
      }
}
```

C.

```
void backtrack (int t)
{
  if (t>n) output(x);
    else
      for (int i=0;i<=1;i++) {
        x[t]=i;
        if (legal(t)) backtrack(t-1);
      }
}
```

D.

```
void backtrack (int t)
{
  if (t>n) output(x);
    else
      for (int i=t;i<=n;i++) {
        swap(x[t], x[i]);
        if (legal(t)) backtrack(t+1);
      }
}
```

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. k 带图灵机的空间复杂性 S(n)是指（B）

A. k 带图灵机处理所有长度为 n 的输入时，在某条带上所使用过的最大方格数。

B. k 带图灵机处理所有长度为 n 的输入时，在 k 条带上所使用过的方格数的总和。

C. k 带图灵机处理所有长度为 n 的输入时，在 k 条带上所使用过的平均方格数。

D. k 带图灵机处理所有长度为 n 的输入时，在某条带上所使用过的最小方格数。

13. NP 类语言在图灵机下的定义为（D）

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 NDTM 所接受的语言}；

14. 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 $n \geq n_0$ 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：$0 \leq cg(n) < f(n)$ }；

15. 记号Ω的定义正确的是（B）。

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 $n \geq n_0$ 有：$0 \leq cg(n) \leq f(n)$ }；

C. (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. (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：$0 \leq cg(n) < f(n)$ }；

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. 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

3. 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的选择，即贪心选择来达到）。
4. 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）。
5. 回溯法是指（具有限界函数的深度优先生成法）。
6. 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)) ）。
7. 回溯法的算法框架按照问题的解空间一般分为（子集树）算法框架与（排列树）算法框架。
8. 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。
9. 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。
10. 用回溯法解 0/1 背包问题时，计算结点的上界的函数如下所示，请在空格中填入合适的内容：

```
Typep Knap<Typew, Typep>::Bound(int i)
{// 计算上界
   Typew cleft = c - cw;  // 剩余容量
   Typep b = cp;          // 结点的上界
   // 以物品单位重量价值递减序装入物品
   while (i <= n && w[i] <= cleft) {
      cleft -= w[i];
      b += p[i];
      i++;
      }
   // 装满背包
   if (i <= n)  (b += p[i]/w[i] * cleft) ;
   return b;
}
```

11. 用回溯法解布线问题时，求最优解的主要程序段如下。如果布线区域划分为n×m的方格阵列，扩展每个结点需 O(1)的时间，L 为最短布线路径的长度，则算法共耗时 （ O(mn) ），构造相应的最短距离需要（O(L)）时间。

```
for (int i = 0; i < NumOfNbrs; i++) {
          nbr.row = here.row + offset[i].row;
          nbr.col = here.col + offset[i].col;
          if (grid[nbr.row][nbr.col] == 0) {
              // 该方格未标记
              grid[nbr.row][nbr.col]
                 = grid[here.row][here.col] + 1;
              if ((nbr.row == finish.row) &&
                 (nbr.col == finish.col)) break; // 完成布线
          Q.Add(nbr);}
          }
```

12. 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）（O（mn））。

```
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;
}
```

13. 旅行售货员问题的解空间树是（排列树）。

三、　证明题

1. 一个分治法将规模为 n 的问题分成 k 个规模为 n / m 的子问题去解。设分解阀值 n0=1，且 adhoc 解规模为 1 的问题耗费 1 个单位时间。再设将原问题分解为 k 个子问题以及用 merge 将 k 个子问题的解合并为原问题的解需用 f(n) 个单位时间。用 T(n) 表示该分治法解规模为 |P|=n 的问题所需的计算时间，

则有：$T(n)=\begin{cases} O(1) & n=1 \\ kT(n/m)+f(n) & n>1 \end{cases}$

通过迭代法求得 T（n）的显式表达式为：$T(n)=n^{\log_m k}+\sum_{j=0}^{\log_m n-1} k^j f(n/m^j)$

试证明 T（n）的显式表达式的正确性。

2. 举反例证明 0/1 背包问题若使用的算法是按照 p_i/w_i 的非递减次序考虑选择的物品，即只要正在被考虑的物品装得进就装入背包，则此方法不一定能得到最优解（此题说明 0/1 背包问题与背包问题的不同）。

证明：举例如：p={7,4,4},w={3,2,2},c=4 时，由于 7/3 最大，若按题目要求的方法，只能取第一个，收益是 7。而此实例的最大的收益应该是 8，取第 2，3 个。

3. 求证：O(f(n))+O(g(n)) = O(max{f(n),g(n)}) 。

证明：对于任意 f1(n)∈ O(f(n)) ，存在正常数 c1 和自然数 n1，使得对所有 n≥n1，有 f1(n)≤ c1f(n) 。

类似地，对于任意 g1(n) ∈ O(g(n)) ，存在正常数 c2 和自然数 n2，使得对所有 n≥n2，有 g1(n) ≤c2g(n) 。

令 c3=max{c1, c2}， n3 =max{n1, n2}，h(n)= max{f(n),g(n)} 。

则对所有的 n ≥ n3，有

f1(n) +g1(n) ≤ c1f(n) + c2g(n)

≤ c3f(n) + c3g(n)

= c3(f(n) + g(n))

≤ c32 max{f(n),g(n)}

= 2c3h(n) = O(max{f(n),g(n)}) .

4. 求证最优装载问题具有贪心选择性质。

（最优装载问题：有一批集装箱要装上一艘载重量为 c 的轮船。其中集装箱 i 的重量为 Wi。最优装载问题要求确定在装载体积不受限制的情况下，将尽可能多的集装箱装上轮船。 设集装箱已依其重量从小到大排序，$(x_1, x_2, \cdots, x_n)$是最优装载问题的一个最优解。又设 $k = \min_{1 \le i \le n}\{i \mid x_i = 1\}$ 。如果给定的最优装载问题有解，则有 $1 \le k \le n$ 。）

证明：

四、 解答题

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。

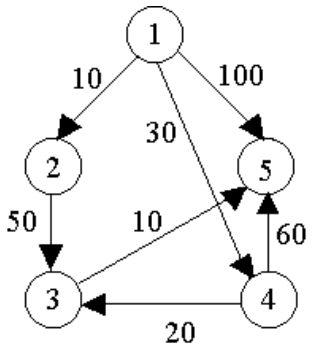
解：利用给出的关系式，此时有：b=2, c=1, g(n)=1, 从 n 递推到 1，有：

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

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						

4. 请写出用回溯法解装载问题的函数。

装载问题：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，其中集装箱 i 的重量为 wi，且 $\sum_{i=1}^{n} w_i \le c_1 + c_2$。装载问题要求确定是否有一个合理的装载方案可将这 n 个集装箱装上这 2 艘轮船。如果有，找出一种装载方案。

解：
```
void backtrack (int i)
{// 搜索第 i 层结点
   if (i > n)  // 到达叶结点
   更新最优解 bestx,bestw;return;
   r -= w[i];
   if (cw + w[i] <= c) {// 搜索左子树
      x[i] = 1;
      cw += w[i];
      backtrack(i + 1);
      cw -= w[i];      }
   if (cw + r > bestw)  {
      x[i] = 0;  // 搜索右子树
      backtrack(i + 1);      }
   r += w[i];
}
```

5. 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。

7. 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);
}
```

8.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);
            }
}
```