第一章 算法求解基础

λ 算法的概念

λ 算法特征（输入、输出、确定性、可行性、有穷性）——掌握每种特征的含义、算法和程序的区别

λ 描述算法的方法（自然语言、流程图、伪代码、程序设计语言）

λ 欧几里德算法（辗转相除法）——递归/迭代程序实现及其变形

λ 常见算法种类——精确算法、启发式算法、近似算法、概率算法

λ 数学归纳法证明；

λ 例题：

λ 1、欧几里得算法主要用来求（ A ）。

λ A．最大公约数 B．最小公倍数

λ C．最小公约数 D．最优解

λ 2、算法的（ B）是指算法的每一条指令必须有确切的定义，没有二义性。

λ A．输入 B．确定性 C．能行性 D.有穷性

λ 3、算法的（ D ）是指算法的每一条指令必须足够基本，它们可以通过已经实现的基本运算执行有限次来实现。

λ A．输入 B．输出 C．确定性 D．能行性 E.有穷性

λ 4. 欧几里德算法是求两个数的（ C ）。

λ A．和 B． 积 C．最大公约数 D．最小公倍数

λ

第二章 算法分析基础

λ 算法复杂度——运行一个算法所需的时间和空间。

λ 好算法的四个特征（正确性、简明性、效率、最优性）

正确性 vs 健壮性 vs 可靠性

最优性——算法（最坏情况下）的执行时间已达到求解该类问题所需时间的下界。

λ 影响程序运行时间的因素（程序所依赖的算法、问题规模和输入数据、计算机系统性能）

λ 算法的渐近时间复杂度 ——数量级上估计（O、Ω、Θ)

λ 最好、最坏、平均时间复杂度——定义

——课后习题 2-8（通过考察关键操作的执行次数）

λ 时间复杂度证明

——课后习题 2-10，2-13，2-17

λ 算法按时间复杂度分类：多项式时间算法、指数时间算法

多项式时间算法：$O(1)<O(\log n)<O(n)<O(n\log n)<O(n^2)<O(n^3)$

指数时间算法：$O(2^n)<O(n!)<O(n^n)$

例题：

1、以下（ C ）不是多项式时间算法的渐进时间复杂度。

A. $O(\log n)$ B. $O(n\log n)$ C. $O(n!)$ D. $O(n^{100})$

2、程序的（ D ）是指算法的执行时间已达到求解该类问题所需时间的下界。

A．正确性　　B．健壮性　　C．可靠性　　D．最优性

3、下面程序段的渐近时间复杂度是 O（　n^2　）

```
for (int i=1;i<=n;i++)
    for (int j=1;j<=i/2;j++)
        x++;
```

4、T(n)表示当输入规模为 n 时的算法效率，以下算法效率最优的是（　　C　）

A．T(n) = T(n-1)+1，T(1) = 1　　B．T(n) = 2n2

C．T(n) = T(n/2)+1，T(1) = 1　　D．T(n) = 3nlog2n

5、多项式时间复杂度 O(logn)的大小介于 O(1) 和 $O(n^2)$两者之间。

6、下面矩阵乘法的时间复杂度为（ A ）

```
for (i=0; i<n; i++)
    for (j=0; j<n; j++)
    {   c[i][j]=0;
        for (k=0; k<n; k++)      c[i][j]+=a[i][k]*b[k][j];
    }
```

A．$O(n^3)$　　B．$O(n^2)$　　C．O(logn)　　D．O(n)

7、下面程序段的时间复杂度为（　B　）。

```
for (int i=1; i≤n;i++)
    for (int j=1; j≤i;j++)
        for (int k=1; k≤j;k++)
```

x++;

A．$O(n^{1/2})$　　B．O(n^3)　　C．O($n\log_2 n$)　　D．O($\log_2 n$)

8、写出下列复杂性函数的偏序关系（即按照渐进阶从低到高排序）：

2^n　3^n　$\log n$　$n!$　$n\log n$　n^2　n^n　10^3

答：O（10^3）< O(logn) < O(nlogn) < O(n^2) < O(2^n) < O(3^n) < O(n!) < O(n^n)

9、下面程序段的时间复杂度为（　D　）。

```
for (int i=1; i≤n;i++)
    for (int j=i; j≤n;j++)
        x++;
```

A．$O(n^{1/2})$　　B．O(n)

C．O(nlogn)　　D．O(n^2)

10、设 T(*n*)由递推式 $$T(n)=\begin{cases}T(2)=4\\T(n)=2T(\lfloor n/2\rfloor)+2n\log n\end{cases}$$ **定义，则 T(*n*)的时间复杂度为（　C　）。**

A．$O(2^n)$　　B．O(n)

C．$O(n\log^2 n)$　　D．O(n logn)

第五章 分治法

λ　分治法——求解的基本要素：将一个难以直接求解的复杂问题分解成若干个规模较小、相互独立但类型相同的子问题，然后求解这些子问题；如果这些子问题还比较复杂而不能直接求解，还可以继

续细分，直到子问题足够小，能够直接求解为止；最后将子问题的解组合成原始问题的解。这种问题求解策略称为分治法。

λ 分治法很自然的导致一个递归算法。

λ 平衡子问题思想

λ 递归算法的时间复杂度分析:

递推式 $T(n)=aT(n/b)+cn^k$，$T(1)=c$

——递推式中每部分的含义

——求解得到算法的渐近时间复杂度（分三种情况）

——改进思路

λ 求最大最小元

λ 二分搜索算法框架

λ 对半搜索

——程序实现

——对半搜索二叉判定树（树的构成）

——对半搜索二叉判定树性质（左右子树结点数、树高等）

——对半搜索的时间复杂度分析（搜索成功/失败、最好/最坏/平均）。

◆ 二叉判定树的性质→对半搜索的时间复杂度

成功搜索：平均、最坏 $O(\log n)$

失败搜索：平均、最好、最坏都是 $\Theta(\log n)$

◆ （通过关键字值间的比较，搜索指定关键字值元素）这类搜索算法最坏情况下的时间下界为 $O(\log n)$，因此对半搜

索是最优算法。

——课后习题 5-8

λ 最优算法

λ 两路合并排序

——分治法排序过程

——程序实现

——时间复杂度分析（最好、最坏、平均）

——空间复杂度分析

——是否最优算法

λ 快速排序

——排序过程（每一趟分划后的排列和子问题划分）

——时间复杂度分析（最好、最坏递推式、平均）

——课后习题 5-11

λ 斯特拉森矩阵乘法——求解时间的递推关系式，时间复杂度，相应的证明。

λ 棋盘覆盖问题

——算法设计思想

——时间复杂度分析

——最优算法

λ 实验补充——（线性时间选择算法）寻找第 k 个最小元

λ 例题：

1、分治法的设计思想是将一个难以直接解决的大问题分割成规模较

小的子问题，分别解决子问题，最后将子问题的解组合起来形成原问题的解。这要求原问题和子问题（ C ）。

A. 问题规模相同，问题性质相同

B. 问题规模相同，问题性质不同

C. 问题规模不同，问题性质相同

D. 问题规模不同，问题性质不同

2、有7名同学参加游泳比赛，学号分别是5，8，11，24，33，39，45，用对半搜索查找学号33的同学的过程中，依次访问到的学号是（ D ）。

A．11,24, 33　　B．24, 33　　C．11, 45, 33

D．24, 39, 33

3、快速排序的最坏情况时间复杂度为是（ B ）。

A．$O(n^3)$　　B．$O(n^2)$　　C．O(nlogn)　　D．O(n)

4、两路合并排序算法的空间复杂度为 O(n) 。

5、斯特拉森矩阵乘法通过减少 子矩阵相乘次数 来降低时间复杂度。

6、下列排序算法中是最优算法的是（ A ）。

A．两路合并排序　　B．冒泡排序　　C．选择排序　　D．快速排序

7、有6名同学参加乒乓球比赛，学号分别是2，5，9，10，15，23，用对半搜索查找学号为10的同学的过程中，依次访问到的学号是（ D ）。

A．5,10　　B．15,9,10　　C．9,10

D．9,15,10

8、分治法通过将一个复杂的问题分解成若干个规模较小、相互独立、类型相同的子问题求解。然后求解子问题的解，并组合成原问题的完整答案。

（1）请叙述分治法中向下分解过程的“平衡子问题思想”。

（2）请用规模为 n 的实例分治框架举例说明原因。

答：（1）平衡子问题思想是指：用分治法设计算法时，使子问题的规模大致相同，即:将一个问题分成大小相等的 k 个子问题，这种处理方法的性能几乎总是比子问题规模不等的做法要好。

（2）举例说明：

规模为 n 的实例，若每次分解为规模大致相当的 2 个子问题，则递归深度为 logn 次，总的求解时间复杂度为 O(nlogn)。

规模为 n 的实例，若每次分解为规模为 1 和 n-1 的子问题，则总共 n-1 轮递归，总的求解时间为(n-1)+(n-2)+……+1=O(n^2)。

可见，子问题规模大致相当的情况，求解的时间复杂度较小。

9、关于对半搜索算法，下列说法（ A ）是错误的。

A． 对半搜索二叉判定树中右子树的结点数和左子树的结点数相等。

B．对半搜索二叉判定树的树高是$\lceil \log(n+1) \rceil$**。**

C．对半搜索二叉判定树是搜索算法的二叉判定树中最矮的。

D．对半搜索是最优算法。

10、目标函数取最大（或最小）值的可行解称为（ A ）。

A．最优解　　B．近似解　　C．最大解　　D．最小解

11、快速排序在最坏情况下，时间递推式为（ B ）。

A. $T(n)=2T(n/2)+\Theta(n)$

B. $T(n) \le T(n-1)+n+1$

C. $T(n)=n+1+\frac{1}{n}\sum_{k=0}^{n-1}(T(k)+T(n-1-k))$

D. $T(n)=n+1+\frac{1}{n}\sum_{k=0}^{n-1}T(k)$

12、分治法是将一个难以直接求解的复杂问题分解成若干个 规模较小 、 相互独立 、且 类型相同 的子问题，然后求解这些子问题，直到子问题足够小能够直接求解，再将子问题的解组合成原始问题的完整答案。

13、斯特拉森矩阵乘法的处理方法是：将待乘的两个 n×n 的矩阵 A 和 B，分别分解成 4 个(n/2)×(n/2)的矩阵相乘。

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

先执行 7 次子矩阵乘法和 10 次加（减）法得到 7 个中间子矩阵：

$P=(A_{11}+A_{22})(B_{11}+B_{22})$　　$Q=(A_{21}+A_{22})B_{11}$

$R=A_{11}(B_{12}-B_{22})$　　$S=A_{22}(B_{21}-B_{11})$

$T=(A_{11}+A_{12})B_{22}$　　$U=(A_{21}-A_{11})(B_{11}+B_{12})$

$V=(A_{12}-A_{22})(B_{21}+B_{22})$

然后再使用 8 次子矩阵加（减）法得到：

$C_{11}=P+S-T+V$　　$C_{12}=R+T$

$C_{21}=Q+S$　　$C_{22}=P+R-Q+U$

λ 请写出斯特拉森矩阵分治求解的时间递推关系式，并给出求得的时间复杂度。

λ 通过手算证明子矩阵 $C_{12}=R+T$ 的正确性(不能省略中间步骤)。

答：（1）$T(n)=\begin{cases} b & n \le 2 \\ 7T(n/2)+O(n^2) & n>2 \end{cases}$

求得时间复杂度为： $O(n^{\log_2 7})$ 或$O(n^{2.81})$

（2）证明：

$$C_{12} = R + T = A_{11}(B_{12} - B_{22}) + (A_{11} + A_{12})B_{22} = A_{11}B_{12} - A_{11}B_{22} + A_{11}B_{22} + A_{12}B_{22}$$

λ $$= A_{11}B_{12} + A_{12}B_{22} = C_{12}$$

λ **14、分治法的步骤中不包括下列哪一个（ C ）**

λ **A．Divide B．Combine C． Sort D．Conquer**

λ **15、快速排序中，每趟分划操作结束后，原序列中的元素以主元为轴心重新排列，主元左侧子序列中所有元素都（ B ）主元。**

λ **A．小于 B．不大于 C．大于 D． 不小于**

λ **16、编程题（10 分）**

λ **请用递归法输出斐波那契数列的前 20 个数。**

λ **long Fib(int n)**

λ **{**

λ **if (（1）n<=1)**

λ **{**

λ **Return （2）1 ;**

λ **}**

λ **else**

λ **{**

λ **return （3）Fib（n-1） + （4）Fib(n-2) ;**

λ **}**

λ **}**

λ int main(int argc, char* argv[])

λ {

λ // 递归法

λ for (int i=1; i<=20; i++)

λ {

λ printf("%d ", （5）Fib(i));

λ }

λ return 0;

λ }

λ 17、下列问题中，（ D ）不适合采用分治法进行求解。

λ A．矩阵相乘问题 B．求最大最小元

λ C．排序问题 D．最长公共子序列

λ 18、斯特拉森矩阵乘法通过（ C ）来降低时间复杂度。

λ A．仅对原矩阵分块 B．减少子矩阵的加法次数C．减少子矩阵的乘法次数 D．提高计算机的运行速度

λ 19、出于“平衡子问题”的思想，通常分治法在分解原问题时，形成若干子问题，这些子问题的规模都大致 相同 （“相同”或“不同”）。

λ 20、快速排序算法在最坏情况下的空间复杂度为 O（n^2） 。

λ 21、以下排序算法中，（ D ）算法在最坏情况下的时间复杂度

为 O(nlogn)。

λ A．直接插入排序　　B．冒泡排序

λ C. 快速排序　　D．两路合并排序

λ 22、在快速排序算法中引入随机过程的主要目的是（ C ）。

λ A. 改善确定性算法的平均时间

λ B. 保证算法总能在 $O(n\log n)$ 时间内结束

λ C．避免了算法最坏情况下的发生

λ D．确定性算法最坏情形下的平均运行时间

λ 23、　　分划操作　　是快速排序的核心操作。

λ 24、　　状态空间树　　是描述问题解空间的树形结构。

λ 25. 请比较分治策略中的合并排序和快速排序算法的异同。（8分）

λ 问题分解过程：

λ 合并排序——将序列一分为二即可。（十分简单）

λ 快速排序——需调用 Partition 函数将一个序列划分为子序列。（分解方法相对较困难）

λ 子问题解合并得到原问题解的过程：

λ 合并排序——需要调用 Merge 函数来实现。（Merge 函数时间复杂度为 O(n)）

λ 快速排序——一旦左、右两个子序列都已分别排序，整个序列便自然成为有序序列。（异常简单，几乎无须额外的工作，省去了从子问题解合并得到原问题解的过程）

λ 26、（1）若待排序元素存放于 l 数组中，且 Swap(i,j)函数的作

用为交换下标为 i 和 j 的两个元素 l[i]和 l[j]的值。 （8 分）

λ 请写出快速排序算法的 C++程序，包含以下三个函数：

λ int Partition(int left, int right)

λ void QuickSort()

λ void QuickSort(int left, int right)

λ (2) 对初始数据（5，8，4，6，3，7，1）执行快速排序，每次都选择左边第一个元素作为主元，请给出每一趟分划操作的结果。并用[]标识子序列的边界。 （4 分）

λ (3) 若待排序数据的规模为 n，快速排序算法最好、最坏情况什么时候发生？ （2 分）

λ 答：(1) ----8 分

λ int Partition(int left, int right)

λ { int i=left, j=right+1;

λ do

λ { do i++; while (l[i]<l[left]);

λ do j--; while(l[j]>l[left]);

λ if (i<j) Swap(i,j);

λ } while (i<j);

λ Swap(left,j);

λ return j;

λ }

λ **void QuickSort()**

λ **{ QuickSort(0,n-1); }**

λ **void QuickSort(int left, int right)**

λ **{ if (left<right)**

λ **{ int j=Partition(left,right);**

λ **QuickSort(left,j-1);**

λ **QuickSort(j+1,right);**

λ **}**

λ **}**

λ **(2) -----4 分**

初始状态	5	8	4	6	3	7	1	∞
第一趟	[3	1	4]	5	[6	7	8]	∞
第二趟	[1]	3	[4]	5	[6	7	8]	∞
第三趟	1	3	4	5	6	[7	8]	∞
第四趟	1	3	4	5	6	7	[8]	∞
排序结果	2	3	4		5		8	∞

λ **(3) 快速排序算法在最好、平均情况下的时间复杂度为**

O(nlogn)，在最坏情况下的时间复杂度是 $O(n^2)$。

λ 每次分划操作后，若左、右两个子序列的长度基本相等，则快速排序效率最高，为最好情况。原始序列正向有序或反向有序时，每次分划操作所产生的两个子序列，其中之一为空序列，则快速排序效率最低，为最坏情况。 ------2 分

λ

λ

第六章 贪心法

λ （求解最优化问题）贪心法的基本要素：

- 最优子结构性质
- 贪心选择性质

λ 一般背包问题

——课后习题 6-1

λ 最佳合并模式

——最小带权外路径长度

——课后习题 6-8，k 路最优合并, 虚结点

λ 最小代价生成树

——Prim 和 Kruskal 算法（构造过程、区别）

——共同的理论基础：MST 性质

——不同点和应用场合

——课后习题 6-9

——Prim 算法求解过程（邻接表、lowcost\nearest\mark 数组、输出的边集、构造的最小代价生成树）

λ 补充---聚类间隔的定义

λ 例题

λ 1、设有 5 个有序文件分别包含（30,10,20,15,20）条记录，现通过两两合并将其合并成一个有序文件。（若每一次合并读写两个文件的全部记录一次）则整个合并过程中至少需要读写总共（ B ）条记录。

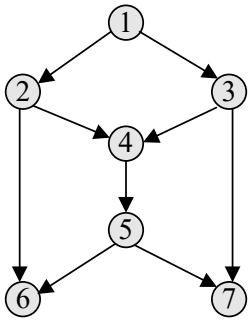
λ A．95 B．215 C．220 D．225

λ 2. n 个顶点的连通图至少有（ A ）条边。

λ A．n-1 B．n C．n+1 D．n+2

3、用贪心法求解最优化问题，每一步上用做决策依据的选择准则被称为 贪心准则 ，根据该标准选择的解分量是局部最优选择，若能够最终产生整体最优解，则称该问题具有 最优子结构 性质。

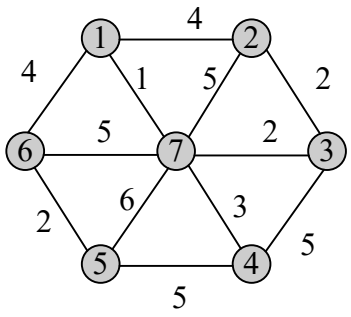
4、下面一个有向无环图，请将其转化为邻接矩阵表达。



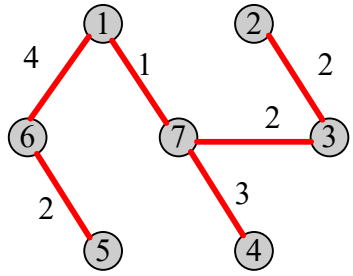
其邻接矩阵为：

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

λ 5、 请画出下图的 Prim 算法由顶点 1 出发获得的最小代价生成树。



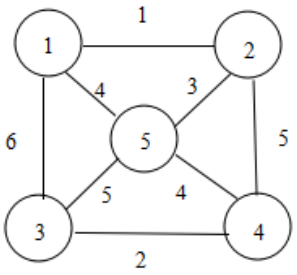
λ



解：

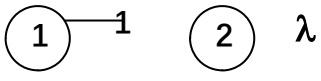
λ

6.下图给出了一个带权无向连通图，分别用 Kruscal 算法和 Prim 算法构造其最小代价生成树，请画出生成过程中每一步构造完毕的生成树。（从结点 1 出发）(8 分)

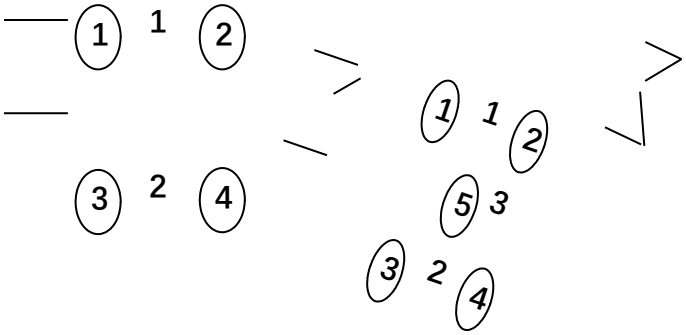


λ

λ 1）Kruskal 算法：



λ



λ

λ

λ

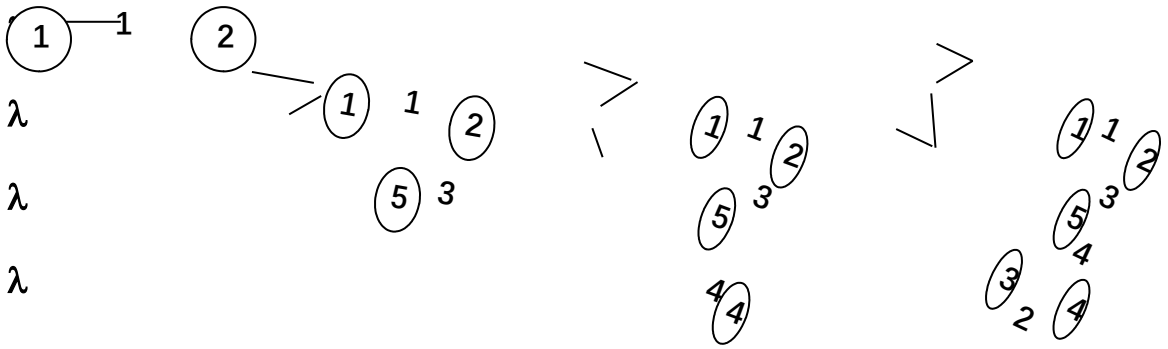
λ

λ （1 分） （1 分） （1 分） （1 分）

λ

λ

λ 2）Prim 算法：



λ （1 分） （1 分） （1 分） （1 分）

λ 7、在一般背包问题中，有载重为 15 的背包，4 件物品的重量分别为：(w0,w1,w2,w3)=(4,5,3,6)，物品装入背包的收益为：(p0,p1,p2,p3)=(20,18,15,22)，设问题的解为(x0,x1,x2,x3)，当解为（A ）时，物品装入背包可获得最大收益。

λ A．(1,0.4,1,1) B．(1,1,0,6/5) C．(0.5,0.8,1,1)

D．(1,1,1,0.5)

λ 8、图 G=(V, E)是一个无向带权连通图，其边数相对较少，则能较高效率得到最小代价生成树的算法是（ C ）。

λ A. 普里姆算法 B. 迪杰斯特拉算法

λ C.克鲁斯卡尔算法 D. 弗洛伊德算法

λ 9、设有 5 个有序子文件（F1，F2，F3，F4，F5），其长度分别为（24，35，30，12，7）。通过最佳合并模式，所需的读写记录数为 235 。

λ 10、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。对给定的 n 和 s，寻找一个方案，使得剩下的数字组成新的最小的正整数。

λ 如输入 n 为 178543，s 为 4，结果为 13

λ 试用贪心法解决此问题，请：

λ (1) 给出用贪心法求解该最优化问题的贪心选择策略。（4 分）

λ (2) 请补全下列贪心算法求出对正整数 n 去掉 s 个数字之后得到的新的最小的正整数。（4 分，每空 2 分）

λ (3) 此算法的渐进时间复杂度？（2 分）

λ

λ 注：正整数 n 存于字符串中，例：

λ string n = “178543”;

λ n.at(0) //返回字符串 n 的第 1 个字符

λ n.erase(2, 3)　　//删除 n 中索引为 2 开始的 3 个字符

λ （1）为了尽可能地逼近目标，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母。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，剩下的数字串便是最后的最优解。

λ (2)string MinNum(string n, int s)

λ {

λ　　while(s>0)

λ　　{

λ　　unsigned int i = 0;　//从串首开始找

λ　　while((i < n.length()) &&(n.at(i) < n.at(i+1))　　)

λ　　i++;

λ　　n.erase(i ,1);　　//删除字符串 n 中索引为 i 的字符

λ　　s--　;

λ　　}

λ　　while((n.length() > 1) &&(n.at(0) ==’0’))

λ　　n.erase(0, 1);

λ　　return n;

λ }

λ

λ （3）O（n^s）

λ 11、由若干权值分别为（1，2，3，3，5，6，7，9）的结点，按照贪心准则构造的3路最佳合并树的带权外路径长度为（A ）。

λ A．66　　B．75　　C．78　　D． 92

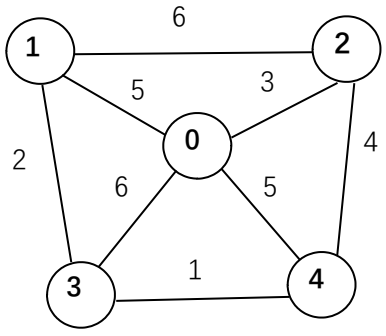
λ 12、Prim 算法和 Kruskal 算法的理论基础是 <u>　最小生成树性</u>质<u>　　</u>，两个算法的正确性都可以从该定理得到证明。

λ 13、在一般背包问题中，有载重为 20 的背包，3 件物品的重量为：（w_0，w_1，w_2）=(15，12，18)，物品装入背包的收益为：(p_0，p_1，p_2)=(10，9，15)，设问题的解为(x_0，x_1，x_2)，当解为（ A ）时，物品装入背包可获得最大收益。

λ A．(0, 1/6, 1)　　B．(1, 0, 6/5)　　C．(1/3 ,1/6 ,1)　　D．(0 ,1/3 ,1)

λ 14、设有 5 个有序子文件（F1，F2，F3，F4，F5），其长度分别为（20，30，30，10，5）。通过最佳两路合并模式，所需的读写记录数为<u>　205　</u>。

λ 15、下图给出了一个带权无向连通图，用克鲁斯卡尔算法构造其

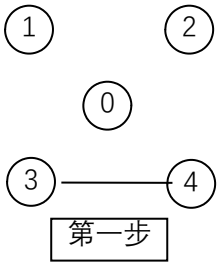


最小代价生成树，请画出构造过程，并给出最小代价。

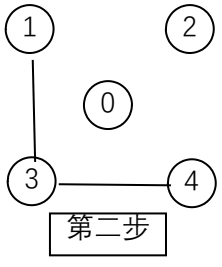
λ （a）

λ

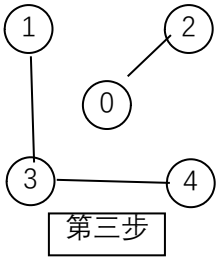
λ 答：



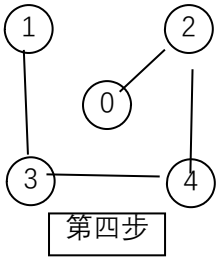
第一步



第二步



第三步



第四步

λ 最小代价为：_____10_____

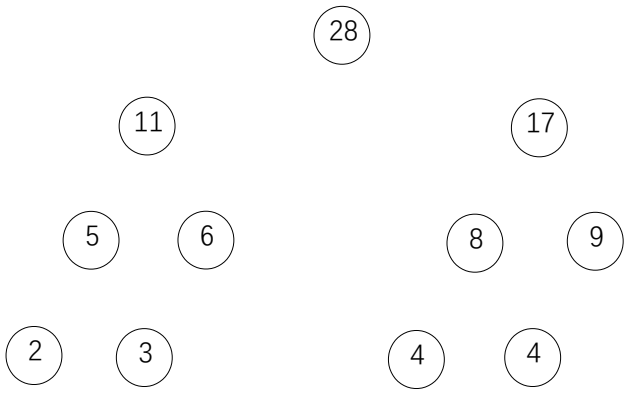
λ 16、有 5 个物品，重量分别为（w_0,w_1,w_2,w_3,w_4）=（2,1,3,5,6），价值分别为（p_0,p_1,p_2,p_3,p_4）=（2,3,1,4,5）。若物品可分割，背包载重为 6，请给出贪心法求解的最优量度标准，并求放入背包物品的最大价值和最优解元组。

λ 答：选择的最优量度标准为：单位重量的价值 w_i/p_i=（1, 3, 1/3, 4/5, 5/6）。

λ　　因此最优解元组为（1, 1, 0, 0, 1/2），最大价值为 7.5。

λ　17、设 W={2，3，4，4，6，9}，请按照贪心准则，构造一棵 2 路最佳合并树。

λ　解：生成的 2 路合并树为



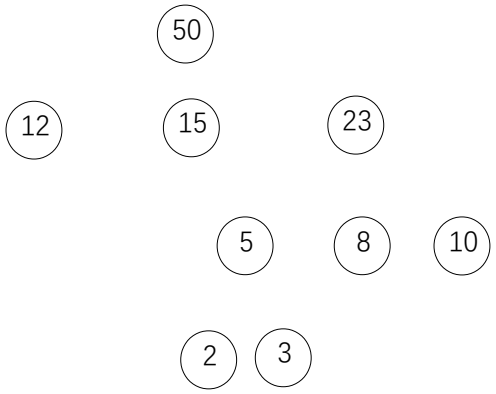
λ

λ　18、Prim 算法和 Kruskal 算法都是求图的(C)的算法。

λ　A．最短回路　　B．最长路径　　C．最小代价生成树 D ．最小带权外路径长度

λ　19、设 W={2,3, 8, 10,12,15}，请按照贪心准则，构造一棵最优 3 路合并树。

λ　答：n=6，k=3，(n-1)%(k-1)=1。所以需补 1 个虚节点。



λ

λ

λ　20、设有一般背包问题实例 n=6，　M=11，$(w_0, w_1, \cdots, w_7) = (3,5,3,4,1,6)$，物品装入背包的收益为：

$(p_0, p_1, \cdots, p_7) = (15,4,9,10,6,6)$，给出求解这一实例的最优解和最大收益。（请给出求解过程）

λ 答：$p/w = (5, 0.8, 3, 2.5, 6, 1)$，解为$(1,0,1,1,1,0)$，最大收益值为 40

λ 或：按p/w从大到小排列为$(6,5,3,2.5,1,0.8)$，解为$(1,1,1,1,0,0)$，最大收益 40

λ

λ

第七章 动态规划法

λ （求解最优化问题）动态规划法的基本要素：

- 最优子结构性质
- 重叠子问题性质

λ 动态规划法求解思路——自底向上、保存子问题的解

λ 动态规划法求解步骤：

- 1）刻画最优解的结构特性
- 2）递归定义最优解值
- 3）以自底向上方式计算最优解值
- 4）根据计算得到的信息构造一个最优解

λ 备忘录方法

1）动态规划法的变形。

2）采用分治法的思想。

3）自顶向下直接递归的方式求解。

——两者的异同和适用场合

λ 多段图问题

——从后向前/从前向后递推式

——结点的 cost 和 d 值求解过程

——最短路径长度，并根据 d 值构造最短路径

注意：d[j]的含义和最短路径的构造。

——课后习题 7-1，7-2

λ 关键路径问题

——earliest、latest 的递推式和求解过程

——寻找关键活动

——构造关键路径

最长路径长度——完成工程的最短时间

——程序实现

——补充题

注意：应由关键活动<i,j>（而不是事件 i）来确定关键路径。

λ 弗洛伊德算法

——d_k 数组和 $path_k$ 数组更新的递推式

——每次迭代后的 d 数组和 path 数组元素值

——程序实现和时间复杂度

λ 最长公共子序列问题

——c 和 s 数组元素的求解递推式

——c 和 s 数组元素求值，得最优解值

——回溯构造最优解

——程序实现

——课后习题 7-9

λ 0/1 背包问题动态规划法求解——自底向上，递推式

λ 0/1 背包问题（实数重量）阶跃点集合求解——非启发式方法、启发式方法

注意：被支配的阶跃点和所有 X>M 的阶跃点均应该去除。

——课后习题 7-15、补充题

λ 实验补充——装载问题。

λ 例题：

λ 1、动态规划算法通常按照（ A ）的顺序执行以下 4 个步骤。

λ ① 以自底向上方式计算最优解值

λ ② 刻画最优解的结构特征

λ ③ 递归定义最优解值

λ ④ 根据计算得到的信息构造一个最优解

λ A．②③①④ B．①②③④ C．③②①④

D．③②④①

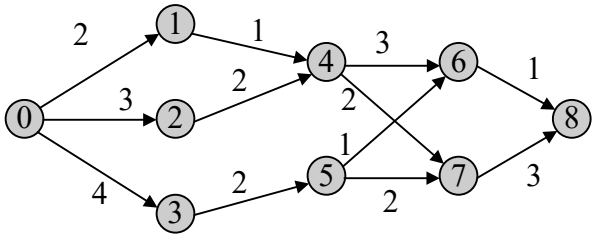
λ 2、字符串 adadecdfcbd 和 bdacdbdf 的公共最长子序列是（1）dacdbd 。

λ 3、关键路径问题中，完成工程所需的最短时间是 AOE 网络中从开始结点到完成结点的 最长 路径的长度（填“最长”或“最短”）。分析关键路径的目的在于找出图中的 关键活动 （即那

些如果不能如期完成就会影响整个工程最短工期的活动）。

λ 4、多段图问题适合用___动态规划__算法求解。

λ 5、下图的关键路径和长度分别是什么？



λ

λ

0 1 2 3 4 5 6 7 8

earliest 0 2 3 4 5 6 8 8 11

latest 0 5 4 4 6 6 10 8 11

——每空 1 分，共 6 分

关键路径是：___(0,3,5,7,8)___ ——1 分

路径长度是：____11____ ——1 分

λ 6.如果只计算最长公共子序列的长度，而无需构造最优解，则只需保存两行元素就可以计算最长公共子序列的长度。请改写下列的程序得到改进的算法，使算法的空间复杂度为 $\Theta(\min(m,n))$ 。

λ Void LCS::CLCS(int i, int j)

λ {

λ if(i==0||j==0) return;

λ if(s[i][j]==1){

λ CLCS(i-1,j-1);

λ cout<<a[i];

```
λ   }
λ       else if (s[i][j]==2) CLCS(i-1,j);
λ       else CLCS(i,j-1);
λ   }
λ   int LCS::LCSLength()
λ   {
λ       for(int i=1; i<=m; i++)  c[i][0]=0;
λ    for(int i=1; i<=n; i++)  c[0][i]=0;
λ       for(i=1; i<=m; i++)
λ       {
λ          for(int j=1; j<=n; j++)
λ               if(x[i]==y[j]) c[i][j]=c[i-1][j-1]+1;
λ               else if(c[i-1][j]>=c[i][j-1]) c[i][j]=c[i-1][j];
λ                       else c[i][j]=c[1][j-1];
λ       }
λ       return c[m][n];
λ   }
```

改写后：答：(1) ----8 分

```
int Partition(int left, int right)
{   int i=left, j=right+1;
    do
    {   do i++; while (l[i]<l[left]);
```

```
        do j--; while(l[j]>l[left]);
        if (i<j) Swap(i,j);
    } while (i<j);
    Swap(left,j);
    return j;
}
void QuickSort()
{   QuickSort(0,n-1);    }
void QuickSort(int left, int right)
{   if (left<right)
    {   int j=Partition(left,right);
        QuickSort(left,j-1);
        QuickSort(j+1,right);
    }
}
```

λ 六、设计题（10 分）

λ 一个人上台阶，台阶有 n 级，他可以一次上 1 级，也可以一次上 2 级。问上这个 n 级的台阶一共有多少种上法？请给出动态规划法求解思路和递推式。

λ 解：如果台阶只有 1 级，那么他一次就可以上去，很显然，上法只有 1 种；

λ 如果台阶有 2 级，那么他可以 1-1，也可以直接上到 2 级，这时一共有 2 种上法；

λ 当 n 大于等于 3 时，f(n)=f(n-1)+f(n-2)，即若要跳到 n 级台阶等于从 n-1 级台阶再跳 1 级，或从 n-2 级台阶再跳 2 级。

λ 7、动态规划算法的基本要素为（ C ）

λ 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

λ 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

λ 8、若序列 X={B, C, A, D, B, C, D}，Y={A, C, B, A, B, D, C, D}，请给出序列 X 和 Y 的最长公共子序列的长度 5 。

λ 9、动态规划算法有一个变形方法，是 备忘录方法 。这种方法是“自顶向下”的递归方向，在求解过程中，保存每个解过的子问题，从而避免相同子问题的重复求解。

λ 10、求解 0/1 背包问题：设有 4 件物品重量分别为 (w0,w1,w2,w3)=(10,15,6,9), 每个物品的收益为 (p0,p1,p2,p3)=(2,5,8,1)，背包的载重为 M=32。利用非启发式方法计算 $S^i(0\le i\le 3)$，并求出最大收益值。

答：$S^0=\{(0,0),(10,2)\}$，

$S_1^1=\{(15,5),(25,7)\}$，

$S^1=\{(0,0),(10,2),(15,5),(25,7)\}$，

$S_1^2=\{(6,8),(16,10),(21,13),(31,15)\}$，

$S^2=\{(0,0),(6,8),(16,10),(21,13),(31,15)\}$

$S_1^3=\{(9,1),(15,9),(25,11),(30,14),(40,16)\}$，

$S^3=\{(0,0),(6,8),(15,9),(16,10),(21,13),(30,14),(31,15)\}$

-----8 分，S^0~S^3 每个集合 2 分。

(2) 最优解值为 15 -----1 分

(3) 最优解为（1,1,1,0） -----1 分

λ 十一、编程题（20 分）

λ 1、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：

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

λ 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。

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

λ void LCSLength(int m，int n，char *x，char *y，int **c，int **b)

λ { int i，j;

λ for (i = 1; i <= m; i++) c[i][0] = 0;

λ for (i = 1; i <= n; i++) c[0][i] = 0;

λ for (i = 1; i <= m; i++)

λ for (j = 1; j <= n; j++)

λ {

λ if (x[i]==y[j])

λ { c[i][j]=c[i-1][j-1]+1;

λ b[i][j]=1; }

λ else if (c[i-1][j]>=c[i][j-1])

λ {

λ c[i][j]= c[i-1][j] ; b[i][j]=2;

λ }

λ else { c[i][j]= c[i][j-1] ; b[i][j]=3; }

λ 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）

λ void LCS(int i，int j，char *x，int **b)

λ { if (i ==0 || j==0) return;

λ if (b[i][j]== 1)

λ {LCS(i-1,j-1,x,b); cout<<x[i]; }

λ else if (b[i][j]== 2) LCS(i-1,j,x,b) ;

λ else LCS(i,j-1,x,b) ;

λ 12、关键路径问题中，完成工程所需的最短时间是 AOE 网络中从开始结点到完成结点的（ B ）路径的长度。

λ A．最短　　B．最长　　C．最优　　D．最差

λ 13、一个最优化多步决策问题是否适合用动态规划法求解有两个要素：<u>最优子结构</u>性质和<u>重叠子问题</u>性质。

λ 14、备忘录方法是<u>动态规划法</u>的变形。

λ 15、给定字符串 $X=(x_1,x_2,...,x_6)=$ abacbd 和 $Y=(y_1,y_2,...,y_5)=$ baacd。

λ （1）请给出 LCS 算法求解最长公共子序列长度过程中数组 c 和数组 s 最后三行的值，

λ c[i][j]保存子序列 $Xi=(x_1,...,x_i)$和 $Yj=(y_1,...,y_j)$的最长公共子序列的长度。

λ s[i][j]记录 c[i][j]的值是由三个子问题 c[i-1][j-1]+1，c[i][j-1]和 c[i-1][j] 中的哪一个计算得到的。若 c[i][j]=c[i-1][j-1]+1 时 s[i][j]=1； c[i][j]=c[i-1][j]时 s[i][j]=2； c[i][j]=c[i][j-1]时 s[i][j]=3。

λ （2）求最长公共子序列的长度。

λ （3）请用箭头画出在 s 上执行 CLCS(int i, int j)算法构造最长公共子序列的追溯路径，并构造最长公共子序列。

λ 答：（1）

		0	b	a	a	c	d
		0	1	2	3	4	5
0	0	0	0	0	0	0	0
a	1	0	0	1	1	1	1
b	2	0	1	1	1	1	1
a	3	0	1	2	2	2	2
c	4	0	1	2	2	3	3
b	5	0	1	2	2	3	3
d	6	0	1	2	2	3	4

c[i][j]数组

		0	b	a	a	c	d
		0	1	2	3	4	5
0	0	0	0	0	0	0	0
a	1	0	2	1	1	3	3
b	2	0	1	2	2	2	2
a	3	0	2	1	1	3	3
c	4	0	2	2	2	1	3
b	5	0	1	2	2	2	2
d	6	0	2	2	2	2	1

s[i][j]数组

λ （2）最长公共子序列长度为：4

λ （3）回溯路径如图所示。S 数组的回溯路径

λ 最长公共子序列为：aacd

λ 17、有 0/1 背包问题实例（w_0,w_1,w_2,w_3）=（3,4,4,2），（p_0,p_1,p_2,p_3）=（4,8,6,2），和 M=10。

λ （1）若 f(j,X)表示当背包载重为 X，可供选择的物品为 0,1，……j 时的最优解值（即放入背包的物品所获收益最大）。请写出动态规划法求解该最优解值的递推关系式。

λ （2）若用 S^j 表示对应于 f(j,X)曲线的阶跃点集合，试计算上面实例的 S^j，0≤j<4。

λ （3）给出上面实例的最大收益和最优解元组。

λ 解：（1）递推式为：

λ $f(j,X)=\max\{f(j-1,X), f(j-1,X-w_j)+p_j\}$ $0\le j<n$

λ （2）

λ $S^0=\{(0,0),(3,4)\}$ $x_0=0$

λ $S^1=\{(0,0),(3,4),(4,8),(7,12)\}$ $x_1=1$

λ $S^2=\{(0,0),(3,4),(4,8),(7,12),(8,14)\}$ $x_2=1$

λ $S^3=\{(0,0),(2,2),(3,4),(4,8),(6,10),(7,12),(8,14),(10,16)\}$ $x_3=1$

λ （3）最大收益为 16。最优解元组为（0,1,1,1）。

λ 18、若 n 个矩阵$\{A_1,A_2,\cdots\cdots A_n\}$中两个相邻矩阵 A_i 和 A_{i+1} 均是可乘的，A_i 的维数为 $p_i\times p_{i+1}$，A_{i+1} 的维数为 $p_{i+1}\times p_{i+2}$。求 n 个矩阵 $A_1A_2\cdots\cdots A_n$ 连乘时的最优计算次序，以及对应的最少数乘次数。

λ 二维数组 m[i][j]定义为：计算 A[i：j]所需的最少数乘次数；

λ 二维数组 s[i][j]定义为：计算 A[i：j]的最优计算次序中的断开位置（例如：若计算 A[i：j]的最优次序在 A_k 和 A_{k+1} 之间断开，$i\le k<j$，则 s[i][j]=k）。

λ 填空完成下面用动态规划法求解矩阵相乘问题的程序，求得最优的计算次序以使得矩阵连乘总的数乘次数最少，并输出加括号的最优乘法算式。

```
λ void MatrixChain(int n,int *p,int **m,int **s)
λ {
λ     for (int i=1;i<=n;i++)  m[i][i]=0;       //为 m[i][i]赋初值
λ     for (int r=2;r<=n;r++)       //r 是连乘矩阵个数
λ         for(int i=1;i<=n-r+1;i++)   //i 是矩阵连乘的起点
λ         {
λ             int j=   (1)   ;          //j 是矩阵连乘的终点
λ             m[i][j]= m[i+1][j]+p[i]*p[i+1]*p[j+1];  //取断开位置
i，对 m[i][j]赋初值
λ             s[i][j]= i;  //取断开位置 i，对 s[i][j]赋初值
```

```
λ                for(int k=i+1;k<j;k++)
λ                {
λ                    int t= m[i][k]+m[k+1][j]+__(2)__;
λ                    if (__(3)__)
λ                    {
λ                        m[i][j]=t  ;
λ                        __(4)__;
λ                    }
λ                }
λ            }
λ  }
λ  void Traceback(int i,int j,int **s)
λ  {
λ       if (i==j) {cout<<'A'<<i<<' ';return;}
λ       if (i<s[i][j]) cout<<'(';
λ       Traceback(i,__(5)__,s);
λ       if (i<s[i][j]) cout<<')';
λ
λ       if (s[i][j]+1<j) cout<<'(';
λ       Traceback( s[i][j]+1,j,s);
λ       if (s[i][j]+1<j) cout<<')';
λ  }
```

λ 答：·· （1） i+r-1

λ （2） p[i]*p[k+1]*p[j+1]

λ （3） t<m[i][j]

λ （4） s[i][j]=k

λ （5） s[i][j]

λ 19、备忘录方法和动态规划法的主要区别是（ A ）。

λ A．备忘录方法是自顶向下求解，动态规划法是自底向上求解。

λ B．备忘录方法是自顶向下求解，动态规划法是自底向上求解。

λ C．备忘录方法需要保存子问题的解，而动态规划法不需要保存子问题的解。

λ D．备忘录方法不需要保存子问题的解，而动态规划法需要保存子问题的解。

λ 20、序列 zxyyz 和 xzxyzz 的最长公共子序列长度为（ C ）。

λ A．2 B．3 C．4 D．5

λ 21、0/1 背包问题用 动态规划 法可求得最优解。若物品 j 的重量为 w_j，价值为 p_j， f(j,X)表示物品 0~j 可选、剩余背包载重为 X 时的背包最大收益，则求解最优解值的递推式为：f(j,X) = max{ f(j-1,X) , f(j-1,X-w_j)+p_j }。

λ 22、（1）请写出运用 Floyd 算法求任意点对间的最短路径长度 d[i][j]的递推式，以及当 d[i][j]发生更新时最短路径上 j 的前一个顶点信息 path[i][j]更新的递推式。

λ （2）若在一个有 4 个结点的带权有向图中求解完毕得到的 d_{n-1} 和 $path_{n-1}$ 数组如下图所示，请给出从结点 2 到结点 3 的最短路径长度和最短路径。

λ

λ 答：(1) d3 数组 path3 数组

λ $d_k[i][j]=$ min{ $d_{k-1}[i][j]$, $d_{k-1}[i][k]+d_{k-1}[k][j]$ }

λ $path_k[i][j]=$ $path_{k-1}[k][j]$

λ （2）从结点 2 到结点 3 的最短路径长度为 6 ，最短路径为（2,0,1,3）

λ

第八章 回溯法

λ 状态空间树——描述问题解空间的树形结构

- 问题状态（树中每个结点）
- 解状态（候选解元组）
- 答案状态（可行解元组）
- 最优答案结点（目标函数取最优值的答案结点）

λ 回溯法和分支限界法都是通过搜索问题的状态空间树求解，

λ 比较回溯法与分支限界法的异同。

λ 剪枝函数（约束函数、限界函数）——可以剪去不必要搜索的子树，压缩问题求解所需要实际生成的状态空间树的结点。

- 约束函数——剪去不含答案状态（可行解）的子树
- 限界函数——剪去不含最优答案结点的子树

λ 约束函数（显式约束、隐式约束）

- 显示约束——规定了所有可能的元组，组成问题的候选解集（解空间）
  - 排列树——用于确定 n 个元素的排列满足某些性质的状态空间树，一般有 n!个叶结点（解状态）。

- 子集树——从 n 个元素的集合中找出满足某些性质的子集的状态空间树，一般有 2^n 个解状态。

■ 隐式约束——给出了判定一个候选解是否为可行解的条件。

λ 回溯法深度优先搜索问题的状态空间树，用剪枝函数（往往是约束函数）进行剪枝，通常求问题的一个或全部可行解

λ 蒙特卡罗(Monte Carlo)算法——估计回溯法处理一个实例时，状态空间树上实际生成的结点数的方法：$m=1+m_0+m_0m_1+m_0m_1m_2+...$

λ n-皇后问题

——算法思想和程序实现

——显式约束、隐式约束条件，显式约束对应的状态空间树

——Monte Carlo 算法估计实际生成的状态空间树结点数

——补充题（蒙特卡罗方法）

λ 子集和数问题（画出状态空间树）

——（可变长度解、固定长度解）对应的状态空间树不同（P183-184）

——约束函数定义

——课后习题 8-2（固定长度解元组）

λ 图的 m-着色问题——四色定理（四种颜色可对地图着色），如何将地图转化为平面图？

λ 哈密顿环——课后习题 8-10

λ 例题

λ 1、状态空间树中，如果从根到某个解状态的路径代表一个作为

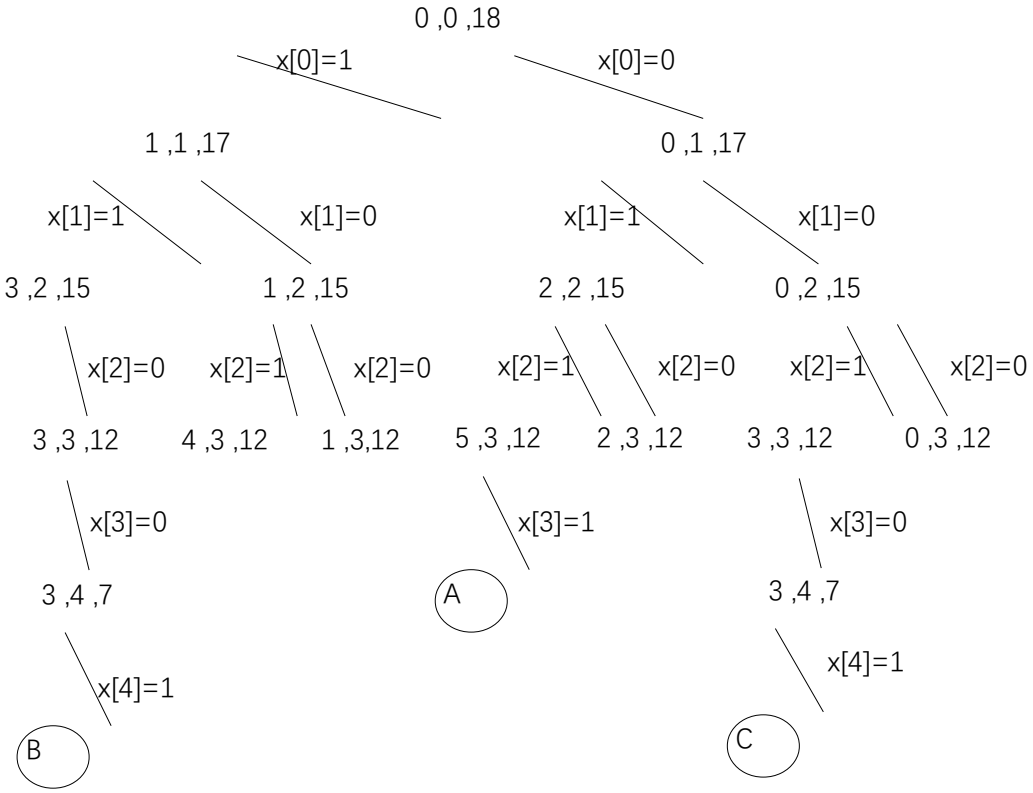
可行解的元组，则称该解状态为（　C　）。

λ　A．解状态　　B．问题状态

λ　C．答案状态　　D．最优答案结点

λ　2　、　设　有　n=5　个　正　数　的　集　合　W={w0,w1,...,w4}=（1，2，3，5，7）和整数 M=10。求 W 中元素之和等于 M 的所有子集。画出对于这一实例由 SumOfSub 算法实际生成的那部分状态空间树，并给出所有可行解。

λ



λ　（2）可行解有：（1，1，0，0，1）、（0，1，1，1，0）和（0，0，1，0，1）

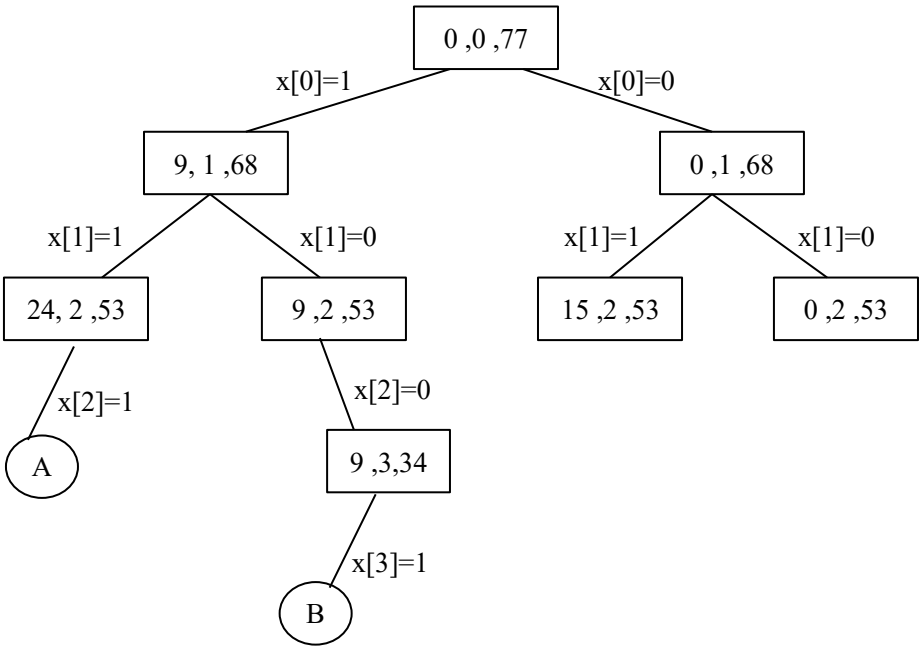
λ　——3 分

λ

λ　3　、　设　有　n=4　个　正　数　的　集　合

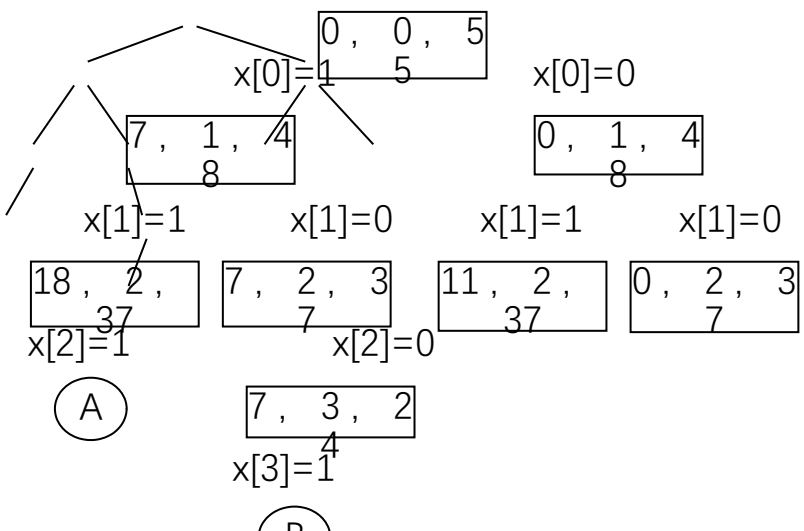
W={w0,w1,...,w4}=（9，15，19，34）和整数 M=43。求 W 中元素之和等于 M 的所有子集。画出对于这一实例由 SumOfSub 算法实际生成的那部分状态空间树，并给出所有可行解。

λ



λ 4、子集和数的问题：设有 n=4 个正数的集合 W={w_0,w_1,w_2,w_3}={7,11,13,24}和正数 M=31，求 W 的所有满足条件的子集，使子集中的正数之和等于 M。请画出由 SumOfSub 算法实际生成的那部分状态空间树，并给出所有可行解。（采用固定长度 4-元组表示解）（10 分）

答：（（w0,w1,w2,w3）=（7,11,13,24），M=31



λ

λ

λ 5、对于含有 n 个元素的子集树问题，当解结构是固定长度元组时，解空间的叶结点数目为（ B ）。

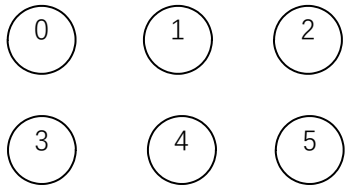
λ A. n! B. 2^n C. 2^{n+1}-1 D. ∑n!/i!

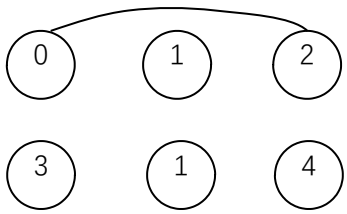
λ 6、一般称从 n 个元素的集合中找出满足某些性质的子集的状态空间树为 子集树（7） 。

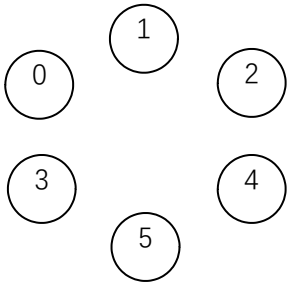
7、对于含有 n 个元素的子集树问题，当解结构是固定长度元组时，解空间的叶结点数目为（ B ）。

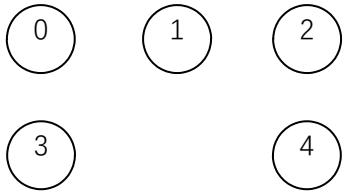
A. n! B. 2^n C. 2^{n+1}-1 D. ∑n!/i!

8、下面图中（ D ）不包含哈密顿环。

A． 

B． 

C． 

D． 

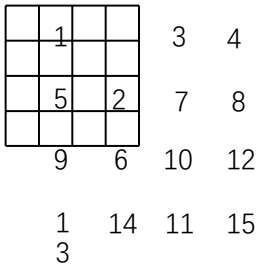
9、所有可在多项式时间内用确定算法求解的判定问题是(A)。

A．P 类问题　B.NP 类问题　C.NP 完全问题　D.P 类语言

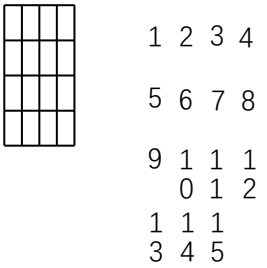
10、使用剪枝函数的深度优先策略生成状态空间树中结点的算法称为 回溯法 。采用广度优先策略生成结点，并使用剪枝函数的算法称为 分支限界法 。

11、用蒙特卡罗方法计算回溯法的时间效率，m_0 为第 2 层不受限制的孩子数目，m_1 为第 3 层不受限制的孩子数目，以此类推，则整个状态空间树上将实际生成的结点数估计为 $1+m_0+m_0m_1+m_0m_1m_2+\ldots$ 。

12、15 谜问题：在一个 4×4 的方形棋盘上放置了 15 块编了号的牌，还剩下一个空格。号牌的一次合法移动是指将位于空格四周（上、下、左、右）的一块号牌移入空格位置的四种移动方式。请给出公式，判断由如图(b)所示的初始状态出发，经过一系列合法的号牌移动能否到达图(c)所示的目标状态？为什么？



(b)



(c)

答：

Less(1)=	**0**	**Less(2)=**	**0**	**Less(3)=**	**1**	**Less(4)=**	**1**
Less(5)=	**1**	**Less(6)=**	**0**	**Less(7)=**	**1**	**Less(8)=**	**1**
Less(9)=	**1**	**Less(10)=**	**0**	**Less(11)=**	**0**	**Less(12)=**	**1**
Less(13)=	**1**	**Less(14)=**	**1**	**Less(15)=**	**0**	**Less(16)=**	**14**

由于：

$$\sum_{k=1}^{16} less(k) + i + j$$

$$= (0+14+1+1+1+0+1+1+1+0+0+1+1+1+0+0) + (0+1)$$

或……$+(1+2)$

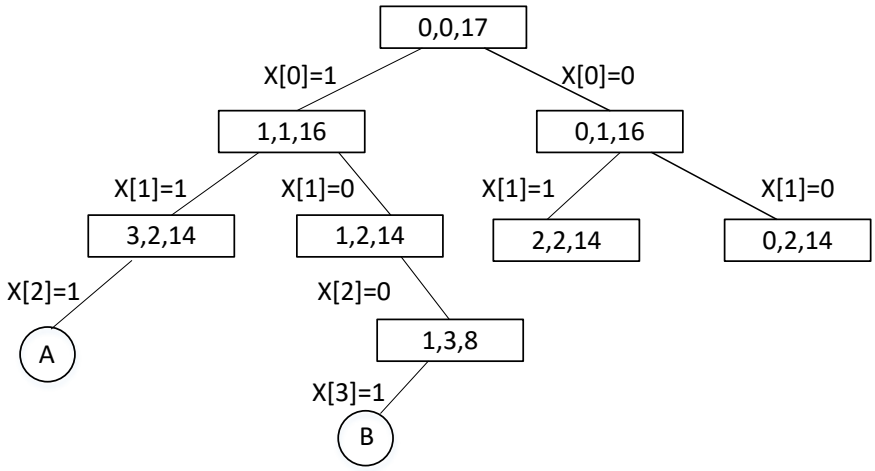
$= 24$或26

为偶数。

所以：可由如图(a)所示的初始状态出发到达图(b)所示的目标状态。

13、求解子集和数的问题：设有 n=4 个正数的集合 S={1,2,6,8}和整数 M=9，请画出回溯法求解过程中产生的状态空间树，并给出所有子集和为 M 的可行解（x_0,x_1,x_2,x_3）。

答：



可行解：{1，1，1，0} 和 {1，0，0，1}

14、一幅没有飞地的地图至少需要（ B ）种颜色进行着色，就可以使图中任意两个相邻国家均着不同颜色。

A.3　　B. 4　　C. 5　　D. 6

15、8 皇后问题运用蒙特卡罗方法估计状态空间树上实际生成的结

点数，如果随机选取的路径为（4,1,3,0,2,7,5），求实际生成的结点数估计值。

解：

m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
8	5	3	3	3	1	1	0

结点数估计值

=1+8+8*5+8*5*3+8*5*3*3+8*5*3*3*3+8*5*3*3*3*1+8*5*3*3*3*1*1

=3769

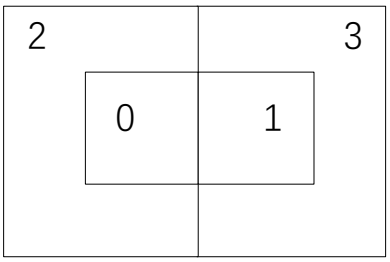
16、8 皇后问题运用蒙特卡罗方法估计状态空间树上实际生成的结点数，如果随机选取的路径为（2,5,3,0,4），求实际生成的结点数估计值。

答：

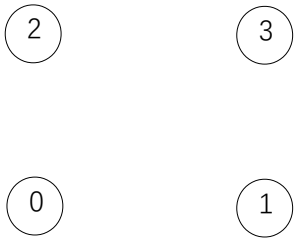
m_0	m_1	m_2	m_3	m_4
8	5	3	3	2

结点数估计值=1+8+8*5+8*5*3+8*5*3*3+8*5*3*3*2=1249

17、地图着色问题中的地图可以转化为对应的平面图表示。请给出下面地图对应的平面图。



答：对应的平面图为



18、请简述回溯法和分枝限界法的相同点和不同点。

答：相同点：（1）都通过搜索问题的状态空间树求解。（2）都通过剪枝函数进行剪枝。

不同点：（1）回溯法采用深度优先遍历，分枝限界法采用广度优先遍历。

（2）回溯法求解所有可行解，分枝限界法求解最优解。

（3）回溯法主要采用约束函数，剪去不含可行解的分支。分枝限界法主要采用限界函数，剪去不含最优解的分支。

（4）对当前扩展结点的扩展方式不同。回溯法中的每个活结点可能多次成为当前扩展结点，纵深方向扩展其一个儿子，然后再回溯后扩展其他儿子；而分枝限界法中每一个活结点只有一次机会成为扩展结点，一次产生所有孩子结点，自身成为死结点，之后无需再返回该结点处。

λ **16、N 皇后回溯法，请填写程序中的空格，其中：**

λ **限界函数 bool PLACE(int k, int i, int *x)，用以确定在 k 列上能否放置皇后；**

λ **函数 void NQUEENS(int k, int n ,int *x)，用以摆放 N 个皇后；**

```
λ  bool PLACE(int k, int i, int *x){          //检查 x[k]位置是否合法
λ    for (int j=0;j<k;j++)
λ      if ((___1____) || (____2____)) return false;
λ    return true;
λ  }
```

λ void NQUEENS(int k, int n ,int *x){

λ /*此过程使用回溯算法求出在 n*n 棋盘上放置 n 个皇后，使其既不同行，也不同列，也不在同一斜角线上*/

λ for (int i=0;i<n;i++) { //显示约束的第一种观点，x[k]=0,1,…,n-1

λ if(___3___){ //约束函数

λ x[t]=i;

λ if (k==n-1) {

λ for (i=0;i<n;i++)count<< x[i] <<" "; //输出一个可行解；

λ count<<endl;

λ }

λ else ___4___ ; //深度优先进入下一层；

λ }

λ }

λ }

λ 答：（1）___abs(k-j)==abs(x[j]-i)___;

λ （2）___x[j]= =i___;

λ （3）___PLACE(k,i,x)___;

λ （4）___NQueen(k+1,n,x)___;

λ

第九章 分枝限界法

λ 分枝限界法广度优先搜索问题的状态空间树，用剪枝函数（往往是限界函数）进行剪枝，通常求问题的最优解

λ 根据活结点表采用的数据结构不同，分枝限界法分类：

- FIFO 分枝限界法
- LIFO 分枝限界法
- LC 分枝限界法

λ 十五谜问题（定理 9-1：判定初始状态是否可以到达目标状态）

——FIFO、LIFO、LC 分枝限界法

——LC 分枝限界法中搜索代价 $\hat{c}(x)$ =f(x)+ $\hat{g}(x)$

——补充题（十五谜问题的 LC 分支限界法求解，生成的状态空间树）

λ 上、下界函数——与最优化问题的目标函数有关

- $\hat{c}(x)$ 是代价函数 c(X)的下界函数
- $u(x)$ 是代价函数 c(X)的上界函数

λ 如何用上、下界函数进行剪枝：

①（若目标函数取最小值时为最优解）

- 则用上界变量 U 记录迄今为止已知的最小代价上界（即迄今为止已知的可行解中目标函数最小值），可以确定最小代价

答案结点（最优解）的代价值不会超过 U。

- 对任意结点，若 $\hat{c}(X)$ ≥U，则 X 子树可以剪枝。
- 为了不至误剪去包含最小代价答案结点的子树，若 X 代表部分向量，则 U=min {u(X)+ε,U}；若 X 是答案结点，则 U=min {cost(X),U}。

②（若目标函数取最大值时为最优解）

- 则用下界变量 L 记录迄今为止已知的最大代价下界（即迄今为止已知的可行解中目标函数最大值），最大代价答案结点（最优解）的代价值不会小于 L。
- 对任意结点，若 $u(X)$ ≤L，则 X 子树可以剪枝。
- 为了不至误剪去包含最大代价答案结点的子树，若 X 代表部分向量，则 L=max {$\hat{c}(x)-\varepsilon$,L}；若 X 是答案结点，则 L=max {cost(X),L}。

λ 带时限的作业排序

——前提：作业按时限排序，以便判断是否可行。

——画出 JSFIFOBB 算法（FIFO 分枝限界法）实际生成的状态空间树（目标函数为损失最小）

——从活结点表中选取扩展结点时，应保证扩展结点满足 $\hat{c} < U$ ，否则剪枝。

——扩展结点生成孩子时，应剪去不可行的孩子结点（即：子集内的作业不能在时限内完成）

——对于可行的孩子结点，进一步计算其损失下界 $\hat{c}$ 和损失

上界 u。当 $\hat{c} < U$ 时生成该结点，否则剪枝。

——每生成一个孩子，需同时检查是否要用 u 更新上界变量值 U。

——求最优解值（最大作业收益=所有作业收益之和-最优解对应的最小损失）和最优解（入选的作业编号，可变长度解）

——课后习题 9-2

例题

1、分枝限界法根据活节点表的不同分为 （2）FIFO 分枝限界法、LIFO 分枝限界法和 （3）LC 分枝限界法。

2、请简述回溯法与分枝限界法的异同。

答：相同点：　　——（共 4 分。答对 1 条给 2 分，答对 2 条给 4 分。）

都是在问题的状态空间树上搜索问题解的算法,都通过活结点表实现。都用到了剪枝函数——约束函数剪去不含答案结点的分枝,用限界函数剪去不含最优解的分枝。

不同点：　　——（共 6 分。答对 1 条给 3 分，答对 2 条给 6 分。）

（1）求解目标不同：回溯法的求解目标是找出解空间树中满足约束条件的所有可行解；而分枝限界法的求解目标则是找出满足约束条件的一个可行解，或某种意义下的最优解。

（2）搜索方式不同：回溯法以深度优先的方式搜索解空间树，而

分枝限界法则以广度优先的方式搜索解空间树。

（3）对当前扩展结点的扩展方式不同：回溯法中的每个活结点可能多次成为当前扩展结点，纵深方向扩展其一个儿子，然后再回溯后扩展其他儿子；而分枝限界法中每一个活结点只有一次机会成为扩展结点，一次产生所有孩子结点，自身成为死结点，之后无需再返回该结点处。

3、根据（　D　）可将分支限界法分为 FIFO、LIFO、LC 分支限界法三类。

A．状态空间树中解空间大小不同

B．求解可行解或最优解的目标不同

C．构造的剪枝函数不同

D．从活结点表中选择下一个扩展节点的次序不同

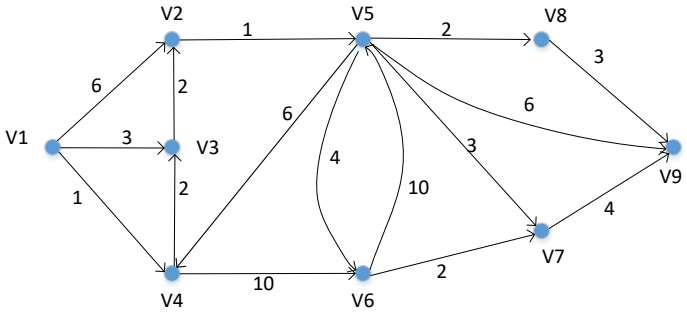
4、使用剪枝函数的广度优先策略生成状态空间树中结点的算法，称为 分支限界法 。

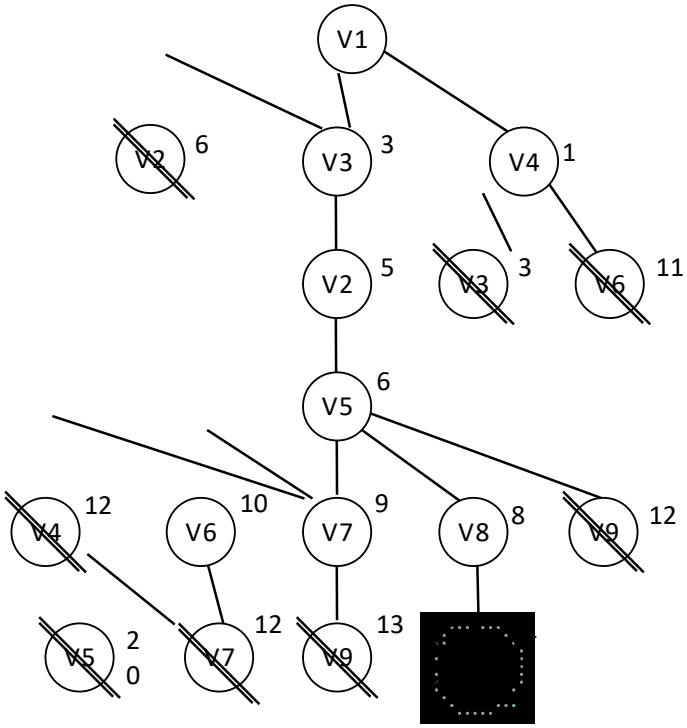
5、以代价估计函数 $\hat{c}(\cdot)$作为选择下一个扩展结点的评价函数，即总是选取 $\hat{c}(\cdot)$值 最小 的活节点为下一个 E-结点。（“最大”或“最小”）

6、约束函数和 限界 函数的目的相同，都是为了剪去不必要搜索的子树，减少问题求解所需实际生成的状态节点数，它们统称为剪枝函数。

7、根据优先队列式分支限界法，求下图中从 v1 到 v9 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〇框起，最优解用双圆圈◎框起。

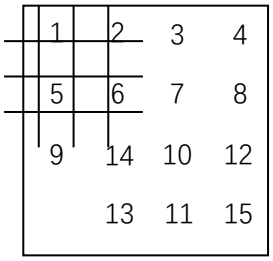
剪枝函数：在算法扩展结点的过程中，一旦发现一个结点的下界不小于当前找到的最短路长，则算法剪去以该结点为根的子树

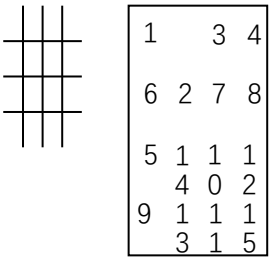


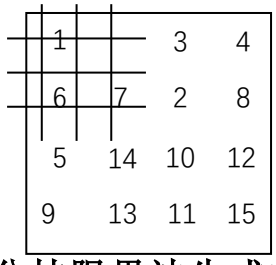


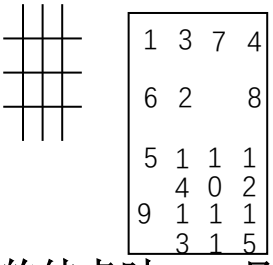
8、下列 4×4 方形棋盘上给定的 15 个号牌和一个空格的初始排列中，

（ C ）经过一系列合法的号牌移动，始终无法到达目标状态。

A. 

B. 

C. 

D. 

9.分枝限界法生成状态空间树中的结点时，一旦一个活结点成为（4） 扩展节点 后，算法将依次生成它的孩子结点并加入到（5）优先权队列 中。

10、根据不同的(D)形成 FIFO、LIFO、LC 三种不同的分枝限界法。
A．状态空间树　　B．目标函数　　C．约束函数　　D．活节点表

11、采用广度优先产生状态空间树的结点，并使用剪枝函数的方法称为（ D ）

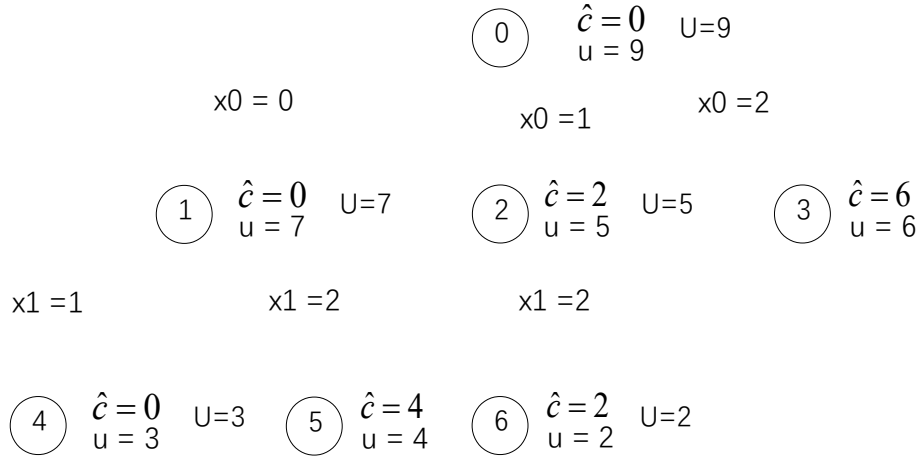
A. 贪心法　　B. 动态规划法　C. 回溯法　　D. 分枝限界法

12、带时限的作业排序问题实例：n=3，（t_0,d_0,p_0）=(2,3,2)，（t_1,d_1,p_1）=(2,4,4)，（t_2,d_2,p_2）=(3,5,3)，求使得总收益最大的作业子集，解向量形式为可变长度元组解（其中的解分量 x_i 为入选作业的下标）。

（1）请标明下图中 FIFO 分枝限界法生成的状态空间树（未入选的作业损失为目标函数），求解过程中每个结点的：损失下界 $\hat{c}$、损失上界 u，以及损失上界阈值变量 U 的更新值。

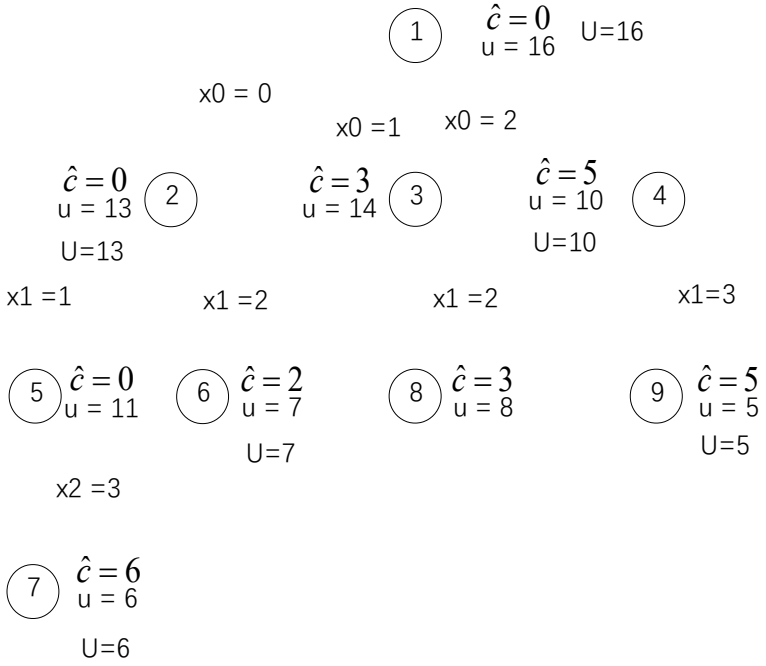
（2）给出求得的最大作业收益和最优解元组。

答：



最大作业收益为：4+3=7
最优解元组为（1，2）
13、带时限的作业排序问题实例：n=4，（t_0,d_0,p_0）=(1,1,3)，（t_1,d_1,p_1）=(1,2,2)，（t_2,d_2,p_2）=(2,3,6)，（t_3,d_3,p_3）=(3,5,5)解向量形式为可变长度元组解（x_i为入选作业的下标），选取目标函数为未入选的作业造成的损失。
（1）当采用LC分支限界法时，若从活结点表（优先权队列）中取出的新的扩展结点因限界函数$\hat{c} \geq U$被剪枝，则剩余活结点表中的结点均会被丢弃。请据此设计：活结点表中满足什么条件的结点能优先成为新的扩展结点?
（2）请画出该实例用LC分枝限界法求解生成的状态空间树，标出树中每个结点的目标函数损失下界$\hat{c}$、损失上界u，以及损失上界阈值U的每次更新值（U值请加方框）。
——注意：状态空间树中的结点编号请按生成顺序编号。
（3）请给出求得的最大作业收益和对应的作业子集解元组。

答：（1）$\hat{c}$值最小的活结点。
（2）



（3）

最大作业收益为：16-5=11

最优解元组为（2，3）

第十章 NP 完全问题

λ 不确定算法及其时间复杂度

λ 最优化问题与判定问题间的转换关系

λ 理解 P 类问题、NP 类问题、NP 难度问题、NP 完全问题的概念

λ 什么是多项式约化？

——课后习题 10-6、10-7

λ 了解 Cook 定理的内容（Steven Cook，1971 年，证明了可满足性问题是 NP 完全的）

——什么是可满足性问题？

λ NP 难度（NP 完全）问题的证明步骤（例：最大集团问题）

——课后习题 10-8

例题：

1、NP 问题是所有可在多项式时间内用（ B ）求解的判定问题的集合。

A. 确定算法　　B. 不确定算法　C. 蒙特卡罗算法　　D. 启发式算法

2、（ B ）是所有可在多项式时间内用不确定算法求解的判定问题的集合。

A. P 类问题　　B. NP 类问题 C. NP-complete 题　　D. NP-hard 问题

3、下列 CNF 公式中（ D ）不存在真值指派。

A. $(x_1 \vee x_2 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee x_3) \wedge (x_1 \vee \overline{x}_2 \vee \overline{x}_3)$

B. $(\overline{x}_1 \vee x_2) \wedge (\overline{x}_2 \vee x_3) \wedge (x_1 \vee \overline{x}_3)$

C. $(\overline{x}_1 \vee x_2) \wedge (\overline{x}_1 \vee \overline{x}_2) \wedge (x_1 \vee x_2)$

D. $(x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge \overline{x}_1 \wedge (\overline{x}_2 \vee \overline{x}_3)$

4、NP 问题是所有可在多项式时间内用 不确定 算法求解的判定问题的集合。对任意问题 $Q_1 \in NP$ 都有 $Q_1 \propto Q$，则称问题 Q 是 NP 难度（NP hard） 的。

5、NP 类问题是所有无法在多项式时间内用 （10）确定算法 求解的判定问题的集合。

6、所有可在多项式时间内用确定算法求解的判定问题是(A)。

A．P 类问题　　B.NP 类问题　　C.NP 完全问题　　D.P 类语言

7、第一个被证明是 NP 完全问题的是（ B ）。

A．最大集团问题　　B．CNF 可满足性问题

C．顶点覆盖问题　　D．图着色数判定问题

第十三章 密码算法（5%）

λ 信息安全的目标

- 机密性——加密
- 完整性——消息摘要
- 抗否认性——数字签名
- 可用性

λ 现代密码学的两个分支（密码编码学、密码分析学）

λ 两种密码体制（对称密码体制、非对称密码体制）的加/解密原理及优缺点

λ **RSA 算法的理论基础、加/解密原理、用途和安全性**

λ **例题：**

λ 1、在传统密码体制中，加密和解密所用的密钥是相同的，所以称为 对称 密码体制。

λ 2、信息安全的（ B ）是指维护信息的一致性，使信息在生成、传输、存储和使用过程中不发生非授权的篡改。

λ A. 机密性　　B. 完整性　　C. 抗否认性　　D. 可 用性

λ 3、下面哪一个关于 RSA 算法的说法是错误的？（ D ）

λ A. RSA 采用的非对称密码体制。

λ B. RSA 算法为每个用户生成一个公钥/私钥对。

λ C. RSA 算法可以用于加密。

λ D. RSA 算法不能用于数字签名。

λ 4、下面哪一个关于 RSA 算法的说法是错误的？（ D ）

λ A. RSA 采用的非对称密码体制。

λ B. RSA 算法为每个用户生成一个公钥/私钥对。

λ C. RSA 算法可以用于加密。

λ D. RSA 算法不能用于数字签名。

λ 5、信息安全的目标是保护信息的机密性、 完整性 ，并具有抗否认性 和可用性。

λ

λ 6. 下面（　　D ）不是针对 RSA 算法的攻击方式

λ A．大整数分解　　B．时间攻击

λ C．中间人攻击　　D．数字认证

λ 7、在传统密码体制中，加密和解密所用的密钥是相同的，这称为 对称 密码体制。（“对称”或“非对称”）

λ 8、信息安全的目标是保护信息的（8）机密性、（9）完整性，并具有（10）抗否认性和可用性。

复习参考题：

一、1、请写出分治法的算法思想。

2、分治法求解的问题通常得到一个递归算法，若算法的执行时间可以表示为 $T(n)=aT(n/b)+cn^k$，$T(1)=c$，请给出该算法渐近时间复杂度的解。

3、渐近时间复杂度的改进思路。

解：（1）分治法的算法思想是 1）将一个难以直接求解的复杂问题分解成若干个规模较小、相互独立，但类型相同的子问题。2）如果子问题还比较复杂而不能直接求解，还可以继续细分，直到子问题足够小，能够直接求解为止。3）最后，再通过组合子问题的解，来得到原始问题的完整解。

（2）
$$T(n)=\begin{cases}\Theta(n^{\log_b a}) & \text{若} a>b^k\\ \Theta(n^k\log n) & \text{若} a=b^k\\ \Theta(n^k) & \text{若} a<b^k\end{cases}$$

二、（1）若待排序元素存放于 l 数组中，且 Swap(i,j)函数的作用为交换下标为 i 和 j 的两个元素 l[i]和 l[j]的值。快速排序算法的 C++程序包含以下三个函数：

① int Partition(int left, int right)

② void QuickSort()

③ void QuickSort(int left, int right)

前两个函数的实现见右图，请补充完成函数：

```
int Partition(int left, int right)
{
int i=left, j=right+1;
do
{
do i++; while (l[i]<l[left]);
do j--; while(l[j]>l[left]);
if (i<j) Swap(i,j);
} while (i<j);
Swap(left,j);
return j;
}
```

```
void QuickSort()
{
QuickSort(0,n-1);
}
```

③ void QuickSort(int left, int right)

(2) 对初始数据（5，8，4，6，3，7，1）执行快速排序，每次都选择左边第一个元素作为主元，请给出每一趟分划操作的结果。并用[]标识子序列的边界。

(3) 若待排序数据的规模为 n，快速排序算法在最好、最坏、平均情况下的时间复杂度是多少？并请说明最好、最坏情况什么时候发生？

(4) 快速排序算法是否最优算法？

答：(1)

```
void QuickSort(int left, int right)
{
    if (left<right)
    {
        int j=Partition(left,right);
        QuickSort(left,j-1);
        QuickSort(j+1,right);
    }
}
```

(2)

初始状态	5	8	4	6	3	7	1	∞
第一趟	[3	1	4]	5	[6	7	8]	∞
第二趟	[1]	3	[4]	5	[6	7	8]	∞
第三趟	1	3	4	5	6	[7	8]	∞
第四趟	1	3	4	5	6	7	[8]	∞
排序结果	2	3	4		5		8	∞

(3)快速排序算法在最好、平均情况下的时间复杂度为 O(nlogn)，在最坏情况下的时间复杂度是 $O(n^2)$。

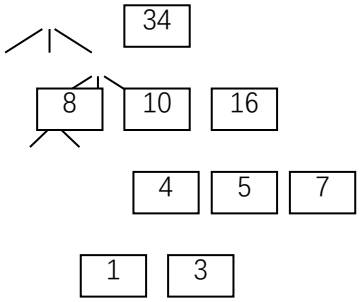
每次分划操作后，若左、右两个子序列的长度基本相等，则快速排序效率最高，为最好情况。

原始序列正向有序或反向有序时，每次分划操作所产生的两个子序列，其中之一为空序列则快速排序效率最低，为最坏情况。

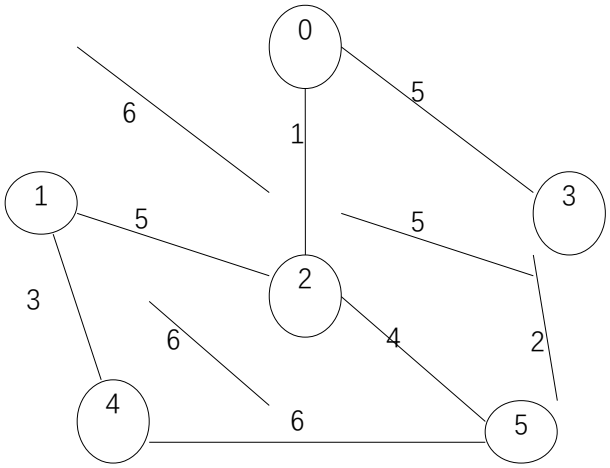
(4)快速排序算法不是最优算法。

三、请画出 W={1，3，5，7，8，10}的三路最佳合并树。

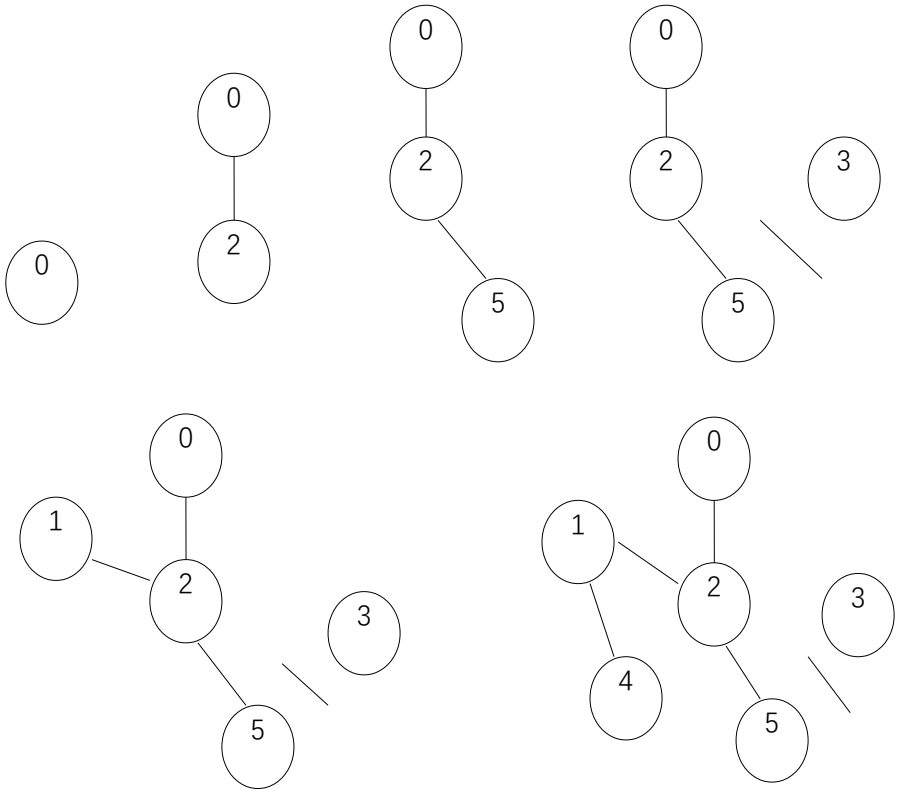
解：因为 n=6，k=3，(n-1)%(k-1)=1，需补齐一个虚结点。



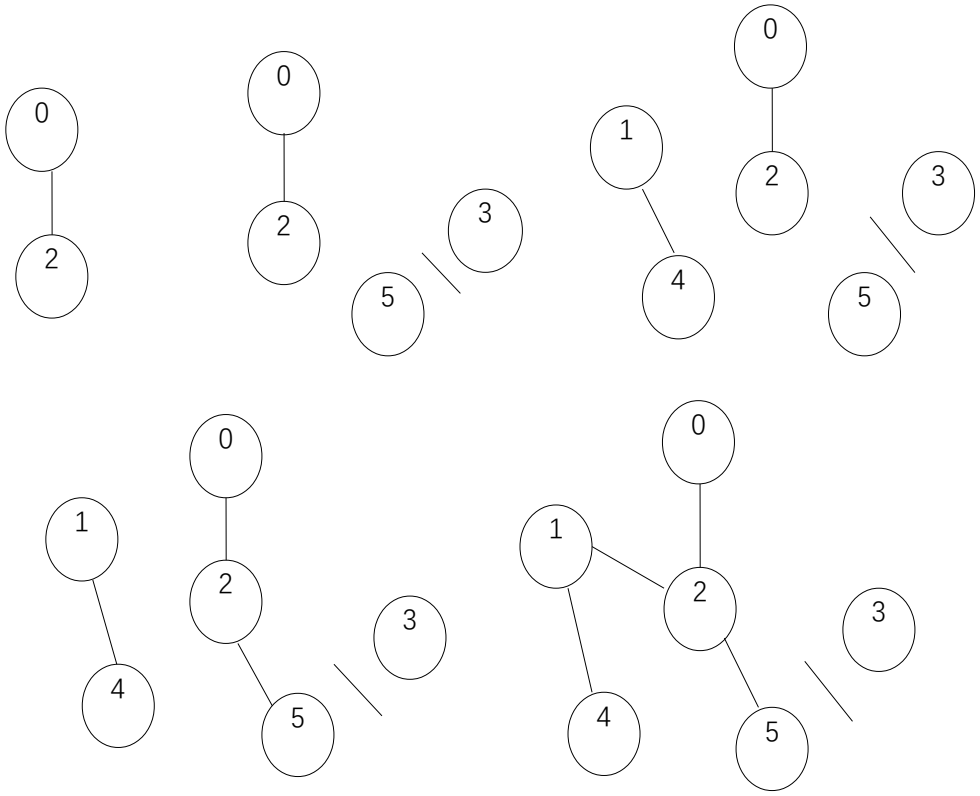
四、给出下图的带权无向连通图，使用 Prim 算法和 Kruskal 算法构造其最小代价生成树的过程（源结点为 0 号结点）。请给出生成过程中的每个子图。



Prim 算法：



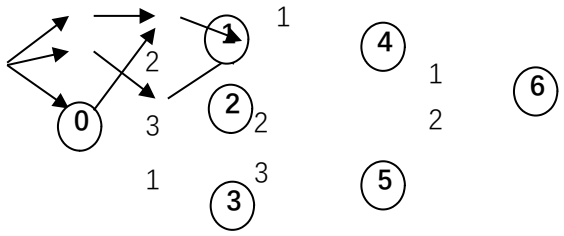
Kruskal 算法：



五、使用从前向后递推方法，求如图所示的多段图从源点 0 到汇点 6 的最短路径长度及最

短路径（请写出最短路径的构造过程）。

用 cost(j)表示源点 0 到顶点 j 的最短路径长度，用 d(j)记录从源点 0 到顶点 j 的最短路径上该结点 j 的前一个结点编号。



解：

cost(0)=	0		
cost(1)=	2	d(1)=	0
cost(2)=	3	d(2)=	0
cost(3)=	1	d(3)=	0
cost(4)=	3	d(4)=	1
cost(5)=	5	d(5)=	2
cost(6)=	4	d(6)=	4

最短路径长度为：4

最短路径为：(d(1)=0,d(4)=1,d(6)=4，6)=(0,1,4,6)

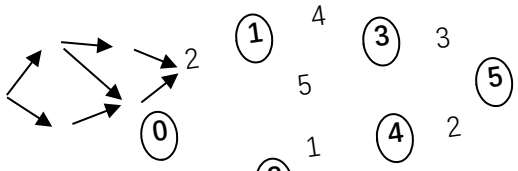
六、下图所示的 AOE 网中

（1）求每个事件（结点）i 可能的最早发生时间 earliest(i)和允许的最迟发生时间 latest(i)。

（2）求从源点 0 到汇点 5 的关键路径长度。

（3）找出图中所有的关键活动。

（4）写出图中所有的关键路径。



解：（1）

	0	1	2	3	4	5
earliest(i)	0	2	3	6	7	9
latest(i)	0	2	6	6	7	9

（2）关键路径长度：9

（3）关键活动：<0,1><1,3><1,4><3,5><4,5>

（4）关键路径：(0,1,3,5)和(0,1,4,5)

七、给定字符串 $X=(x_1,x_2,...,x_6)$=babcd 和 $Y=(y_1,y_2,...,y_5)$=abac。

（1）请给出 LCS 算法求解最长公共子序列长度过程中数组 c 的值，并求最长公共子序列的长度。

（2）请给出数组 s 的值，并用箭头指示构造最长公共子序列时在 s 中的追溯路径，给出最

长公共子序列。

（c[i][j]=c[i-1][j-1]+1 时 s[i][j]=1；c[i][j]=c[i-1][j]时 s[i][j]=2；c[i][j]=c[i][j-1]时 s[i][j]=3。）

解：（1）

		0	*a*	*b*	*a*	*c*
	0	0	0	0	0	0
b	1	0	0	1	1	1
a	2	0	1	1	2	2
b	3	0	1	2	2	2
c	4	0	1	2	2	3
d	5	0	1	2	2	3

c[i][j] 数组

最长公共子序列长度为 3。

（2）

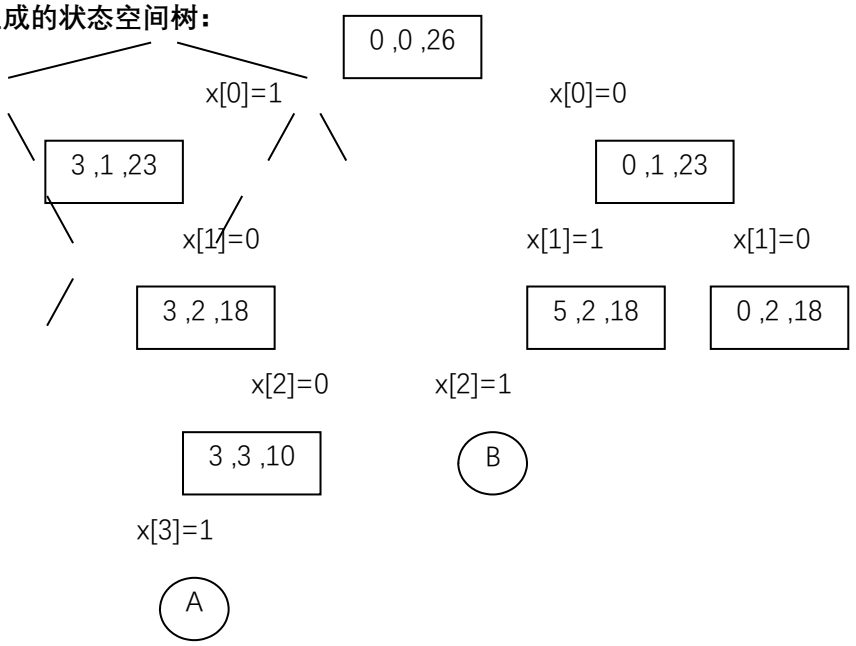
		0	*a*	*b*	*a*	*c*
	0	0	0	0	0	0
b	1	0	0	2	1	3
a	2	0	1	2	1	3
b	3	0	2	1	2	2
c	4	0	2	2	2	1
d	5	0	2	2	2	2

s[i][j]数组

最长公共子序列为：bac。

八、设有 n=4 个正数的集合 $W=\{w_0,w_1,w_2,w_3\}$=（3，5，8，10）和整数 M=13,求 W 中元素之和等于 M 的所有子集。画出对于这一实例由 SumOfSub 算法实际生成的那部分状态空间树，并给出所有可行解。

解：（1）实际生成的状态空间树：

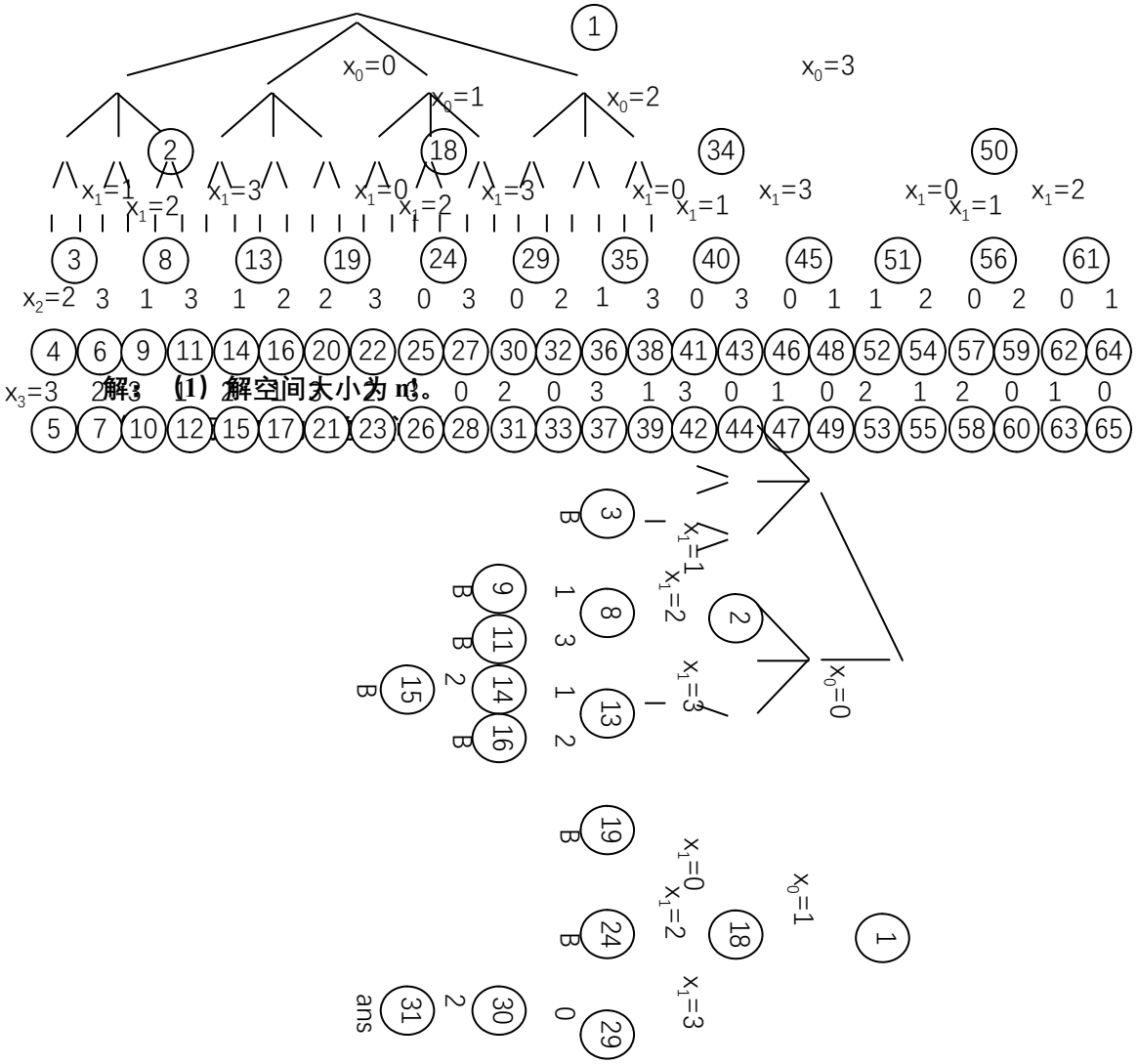


（2）可行解有：（1，0，0，1）和（0，1，1）

九、若给出求解 n-皇后问题的显式约束条件：$S_i=\{0,1,2,.....,n-1\}$，$0\le i<n$，且 $x_i\neq x_j(0\le i,j<n,\ i\neq j)$和隐式约束条件:对任意 $0\le i,j<n$,当 $i\neq j$ 时,$|i-j|\neq|x_i-x_j|$。

（1）则此时解空间大小为多少？

（2）当 n=4 时，该显式约束条件下的状态空间树如下图。请画出回溯法求解得到第一个可行解时，实际生成的状态空间树部分。



解：（1）解空间大小为 n!。

十、用 NQueens(0,4,x)调用下面程序，可以回溯法求解 4-皇后问题的所有可行解。若随机选择的路径为(0,2)和（0,3,1），请用蒙特卡罗方法估计实际生成的状态空间树结点数。

```
bool Place(int k,int i,int *x)
{     for (int j=0;j<k;j++)
      if ((x[j]=i)||(abs(x[j]-i)==abs(j-k))) return false;
return true;
}
void NQueens(int k,int n,int *x)
{  for (int i=0;i<n;i++)
{     if (Place(k,i,x))
      {  x[k]=i;
         if (k==n-1)
            {     for (i=0;i<n;i++) cout<<x[i]<<“ ”;    //输出一个可行解
```

```
                cout<<endl;
            }
        else NQueens(k+1,n,x); //深度优先进入下一层
        }
    }
    }
```

解：选择的路径为(0,2)时，状态空间树上实际生成的结点数为：1+4+4*2=13

选择的路径为(0,3,1)时，状态空间树上实际生成的结点数为：1+4+4*2+4*2*1=21

因此实际生成的结点数平均值为 17。

十一、请简述动态规划法与备忘录方法的异同。如何选择使用动态规划法或备忘录法?

答：1、动态规划法与备忘录方法的共同点：用一个表格来保存已求解的子问题的答案，使下次需要解此子问题时，只简单地查看答案，不重新计算。

不同点：备忘录的递归方式是自顶向下，而动态规划法的递归方式是自底向上。

2、动态规划法与备忘录方法的选择：

当一个问题的所有子问题都至少要解一次时，选用动态规划法较好，此时没有任何多余的计算，还可利用规则的表格存取方式，减少时间和空间需求。

当一个问题只有部分子问题需要求解时，选用备忘录法较好，它只解那些确实需要求解的子问题。