第一次作业：

2-4.

解：a. (1,5)，(2,5)，(3,5)，(4,5)，(3,4)

b. 满足A[1] > A[2] > ...... > A[n]的数组即<n，n-1，...，2，1>具有最多逆序对，有$\frac{n(n-1)}{2}$个

c. 逆序对越多，则插入排序要做的比较越多，运行时间越长。

d. 利用分治法，伪代码如下：　　证明？　-5

```
COUNT-INVERSIONS(A, p, r)
    inversions = 0
    if p<r
        q = ⌊(p+r)/2⌋
    inversions = inversions + COUNT-INVERSIONS(A, p, q).
    inversions = inversions + COUNT-INVERSIONS(A, q+1, r)
    inversions = inversions + MERGE-INVERSIONS(A, p, q, r)

MEGRE-INVERSIONS(A, p, q, r)
  n1 = q-p+1
  n2 = r-q
  let L[1,......, n1+1] and R[1,......, n2+1] be new arrays
  for i=1 to n1
     L[i] = A[p+i-1]
  for j=1 to n2
     R[j] = A[q+j]
  L[n1+1] = ∞
```

```
R[n2+1]=∞
  i=1
  j=1
 inversions=0
 counted=FALSE
对L[]和R[]均进行归并排序
for k=p to r
    if counted==FALSE and R[j]<L[i]
        inversions=inversions+n1-i+1
        counted=TRUE
    if L[i]≤R[j]
        A[k]=L[i]
         i=i+1
    else A[k]=R[j]
         j=j+1
        counted=FALSE
 return inversions
```

4.1-5.

解：从第一个正数起，记当前和、最大子数组和

①按数组顺序更新当前和为加上下一个元素，并以"打擂"方式更新最大子数组和

②若当前和大于0便产生第一个正数，重复执行①过程

③若当前和小于等于0，则当前和无用，开始寻找第一个正数并在找到后重新计算当前和，再执行②过程。

以此类推，直至遍历到最后一个元素，即可得出最大子数组。时间复杂度为O(n

4.3-2.

解：取 $c>0$，使 $T(n) \le c\lg n$

则 $T(\lceil \frac{n}{2} \rceil) \le c\lg\frac{n}{2}$

$\therefore T(n) \le c\lg\frac{n}{2} + 1$

$= c\lg n - c\lg 2 + 1$

$\le c\lg n$

其中，只要 $c \ge 1$，最后一步都会成立

$\therefore T(n) \le c\lg n$ 　　$T(n) \le c\lg(n-2)$

$T(1) = c\lg 1$

$\therefore T(n) = O(\lg n)$ 　　-5

4.3-9.

解：令 $m = \lg n \Rightarrow n = 2^m$

$T(2^m) = 3T(2^{\frac{m}{2}}) + m$

记 $T(2^m) = F(m)$

$\therefore$ 有 $F(m) = 3F(\frac{m}{2}) + m$

猜测 $F(m) = O(m\lg m)$

对 $c>0$ 有 $F(m) \le cm\lg m$ 　　$F(m) \le cm^{\lg 3} + dm$

$F(m) \le 3[c \cdot \frac{m}{2}\lg(\frac{m}{2})] + m$

$\le \frac{3}{2}c\lg\frac{m}{2} + m$

$= \frac{3}{2}cm\lg m - \frac{3}{2}cm\lg 2 + m$ 　　-5

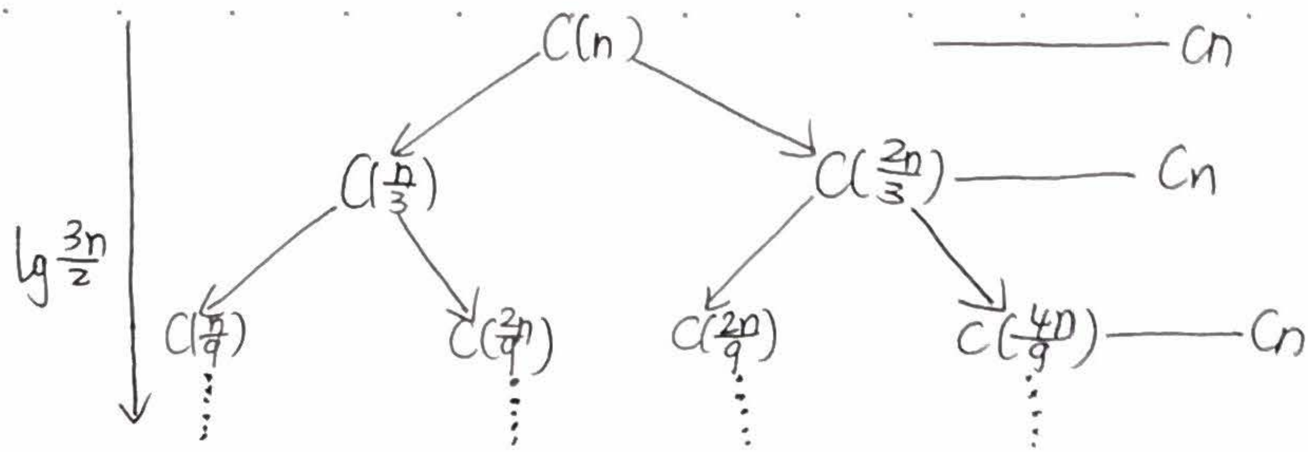
$= km\lg m \quad (k = \frac{3}{2}c)$

$\therefore$ 有 $T(n) = O(\lg n\lg n)$

若此式成立，c应该严格等于$\frac{3}{2}$，c是后边是分治法分解问题和合并问题的时间代价，严格"$c=\frac{3}{2}$"并不现实。

4.4-6.

解：递归树如下所示：



从根结点到叶子结点的最长路径为 $n\to\frac{2}{3}n\to(\frac{2}{3})^2n\to\dots\to 1$

当 $k=\log_{\frac{3}{2}}n$，即树的最长路径长度时 $(\frac{2}{3})^k n=1$

当 $k=\log_3 n$，即树的最短路径长度时 $(\frac{1}{3})^k n=1$

∵每层开销均为 cn

∴解为 $cn\log_3 n=\Omega(n\log n)$

4.5-1.

解：a、该递归式与 $T(n)=aT(\frac{n}{b})+f(n)$ 相比（主方法）

有 $a=2$，$b=4$

∵ $\log_4 2=\frac{1}{2}$ ∴ $n^{\log_b a}=\sqrt{n}$

又∵ $f(n)=1=O(n^{\log_b a-k})$ 且 $k=\frac{1}{2}$

∴ $n^{\log_b a}>f(n)$

$T(n)=\Theta(\sqrt{n})$

b、该递归式与 $T(n)=aT(\frac{n}{b})+f(n)$ 相比

其中 $a=2$，$b=4$

∵ $\log_b a=\log_4 2=\frac{1}{2}$

∴ $n^{\log_b a}=\sqrt{n}$

又∵ $f(n)=\sqrt{n}=n^{\log_b a}$

∴ $T(n)=\Theta(\sqrt{n}\lg n)$

c、该递归式与 $T(n)=aT(\frac{n}{b})+f(n)$ 相比.

第二次作业：

15.1-3：

解：将问题更改为每做一次切割要付出固定成本C。那当钢条长度为0和不做切割时不用付出成本C。

故对于 r_n $(n \geq 1)$，用更短的钢条的最优切割收益描述它变为：

$$r_n = \max(p_n, r_1 + r_{n-1} - C, r_2 + r_{n-2} - C, \ldots, r_{n-1} + r_1 - C)$$

故用带备忘的自顶向下法实现为：

```
MEMOIZED-CUT-ROD(p, n)
    let r[0...n] be a new array
    for i=0 to n
     r[i]=-∞
    return MEMOIZED-CUT-ROD-AUX(p, n, r)
```

```
MEMOIZED-CUT-ROD-AUX(p, n, r)
    if r[n]≥0
        return r[n]
    if n==0
        q=0
    else  q=-∞
        for i=1 to n-1   //为n时无切割成本
            q=max(q, p[i]+MEMOIZED-CUT-ROD-AUX(p, n-i, r
    q=max(q-C, p[n])
    r[n]=q
    return q
```

-5

最后结果和无切割成本的p(n)作比较

15.2-1:

解：由题意得：

矩阵	A_1	A_2	A_3	A_4	A_5	A_6
规模	5×10	10×3	3×12	12×5	5×50	50×6

由MATRIX-CHAIN-ORDER得到m、s表如下：

m:

	1	2	3	4	5	6
6	m[1,2]+m[3,6]+5×3×6	m[2,2]+m[3,6]+10×3×6	m[3,4]+m[5,6]+3×5×6	m[5,6]+12×5×6	5×50×6	0
5	m[1,4]+m[5,5]+5×5×50	m[2,2]+m[3,5]+10×3×50	m[3,4]+3×5×50	12×5×50	0	
4	m[1,2]+m[3,4]+5×3×5	m[3,4]+10×3×5	3×12×5	0		
3	m[1,2]+5×3×12	10×3×12	0			
2	5×10×3	0				
1	0					

s:

	1	2	3	4	5
6	2	2	4	4	5
5	4	2	4	4	
4	2	2	3		
3	2	2			
2	1				

∴ 由上得 $s[1,6] \to ((A_1A_2)(A_3A_4A_5A_6))$

$s[1,2] \to (A_1A_2)$

$s[3,6] \to (A_3A_4)(A_5A_6)$

故最终其最优括号化方案为 $(A_1A_2)((A_3A_4)(A_5A_6))$

且 $m[1,6] = m[1,2] + m[3,6] + 5\times3\times6$

$= 150 + (m[3,4] + m[5,6] + 3\times5\times6) + 90$

$= 150 + (180 + 1500 + 90) + 90$

$= 2010$

15.2-5:

解：观察MATRIX-CHAIN-ORDER的伪代码，可得知3层循环、最里层循环调用m[i,j]的次数固定为2次，故：

$$\therefore \sum_{i=1}^{n}\sum_{j=i}^{n} R(i,j) = \sum_{l=2}^{n}\sum_{i=1}^{n-l+1}\sum_{k=i}^{i+l-2} 2 \quad \bigstar$$

$$= \sum_{l=2}^{n}\sum_{i=1}^{n-l+1} 2(l-1)$$

$$= \sum_{l=2}^{n} 2(l-1)(n-l+1)$$

设 $t=l-1$，则原式 $= \sum_{t=1}^{n-1} 2t(n-t)$

$$= 2\sum_{t=1}^{n-1}(nt-t^2)$$

$$= 2\sum_{t=1}^{n-1} nt - 2\sum_{t=1}^{n-1} t^2$$

$$= \cancel{2}n\frac{(1+n-1)(n-1)}{\cancel{2}} - \cancel{2}\frac{(n-1)n(2n-1)}{\cancel{6}\,3}$$

$$= n^2(n-1) - \frac{(n-1)(2n^2-n)}{3}$$

$$= \frac{n^3-n}{3}$$

故得证。

(方法二)：由书P214页m表可知只有计算 $m[1\ldots i-1, j]$ 和 $m[i, j+1\ldots n]$ 时才会访问 $m[i,j]$，即共计 $i-1+n-j$ 次。

即：$R(i,j) = i-1+n-j$.

$$\therefore \sum_{i=1}^{n}\sum_{j=i}^{n} R(i,j) = \sum_{i=1}^{n}\sum_{j=i}^{n}(i-1+n-j)$$

$$= \sum_{i=1}^{n}(i-1+n$$

15.3-6：

解： ①当交易佣金为0，即不需要支付佣金时：

假设从货币1到货币k，要取得最优兑换序列，所做的最后一次兑换是从货币i兑换到了货币k。那么从货币1到货币i必然是其区间的最优兑换序列。如果不是，那么我们可以找到一个从货币1到货币i的兑换策略其收益大于原从货币1到货币k的兑换序列，我们用新找到的从1到i的兑换策略替换原来1到i的序列，从而使得原从1到k的问题得到更大的收益，这与最初的解是原问题最优解的前提相矛盾。因此，不可能存在更优的解。原问题的子问题的解应是其自身的最优解，故该问题具有最优子结构性。

②当交易需要支付佣金，k次交易需支付c_k佣金，c_k为任意值时：

$$D_m = \max\{(D_i - c_i) r_{i,j}\}$$
$$= \max\{D_i r_{i,j} - c_i r_{i,j}\}$$

两个子问题相关，具有相同变量$r_{i,j}$，同个原问题的一个子问题的解会影响另一个子问题的解，不具备最优子结构性。

如：用数据

$r_{i,j}$	1	2	3	4
1	1	2	$\frac{5}{2}$	6
2	$\frac{1}{2}$	1	$\frac{3}{2}$	3
3	$\frac{2}{5}$	$\frac{2}{3}$	1	3
4	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	1

Let $c_1=2$ and $c_2=c_3=3$

此时从货币1到货币4的最优兑换序列为：$1\to2\to3\to4$，

收益为：$2\times\frac{3}{2}\times3 - c_3 = 9-3=6$

而从货币1到货币3的最优兑换序列为：$1\to3$，

收益为：$\frac{5}{2} - c_1 = \frac{5}{2} - 2 = \frac{1}{2}$ 并不是 $1\to2\to3$：$2\times\frac{3}{2} - c_2 = 3-3=0$

即：从$1\to4$的最优并不含从$1\to3$的最优．

故该问题不具有最优子结构性质

15.4-1:

解:

i	j	0	1	2	3	4	5	6	7	8	9
	y_j		0	1	0	1	1	0	1	1	0
0	x_i	0	0	0	0	0	0	0	0	0	0
1	1	0	↑0	↖1	←1	↖1	↖1	←1	↖1	↖1	←1
2	0	0	↖1	↑1	↖2	←2	←2	↖2	←2	←2	↖2
3	0	0	↖1	↑1	↖2	↑2	↑2	↖3	←3	←3	↖3
4	1	0	↑1	↖2	↑2	↖3	↖3	↑3	↖4	↖4	←4
5	0	0	↖1	↑2	↖3	↑3	↑3	↖4	↑4	↖4	↖5
6	1	0	↑1	↖2	↑3	↖4	↖4	↑4	↖5	↖5	↑5
7	0	0	↖1	↑2	↖3	↑4	↑4	↖5	↑5	↑5	↖6
8	1	0	↑1	↖2	↑3	↖4	↖5	↑5	↖6	↖6	↑6

故按行主次序，得到LCS一个为：

$\langle 1,0,0,1,1,0\rangle$

15.5-2.

解：由算法OPTIMAL-BST(p, q, n)的伪代码得出了关键字的e、w、root表分别如下：

e:

j \ i	1	2	3	4	5	6	7	8
7	3.12	2.61	2.13	1.55	1.20	0.78	0.34	0.05
6	2.44	1.96	1.48	1.01	0.72	0.32	0.05	
5	1.83	1.41	1.04	0.57	0.30	0.05		
4	1.34	0.93	0.57	0.24	0.05			
3	1.02	0.68	0.32	0.06				
2	0.62	0.30	0.06					
1	0.28	0.06						
0	0.06							

计算时 倒数第一行：$e[1,0]=q_0$

$e[2,1]=q_1$

$e[3,2]=q_2$

……

倒数第二行：$e[1,1]=e[1,0]+e[2,1]+w[1,$

$e[2,2]=e[2,1]+e[3,2]+w[2,$

……

倒数第三行：$e[1,2]=\min\{e[1,0]+e[2,2]+w[1,$ / $e[1,1]+e[3,2]+w[1,$

……

倒数第四行：$e[1,3]=\min\{e[1,0]+e[2,3]+w[1,3$ / $e[1,1]+e[3,3]+w[1,3$ / $e[1,2]+e[$

基于回理

w: i

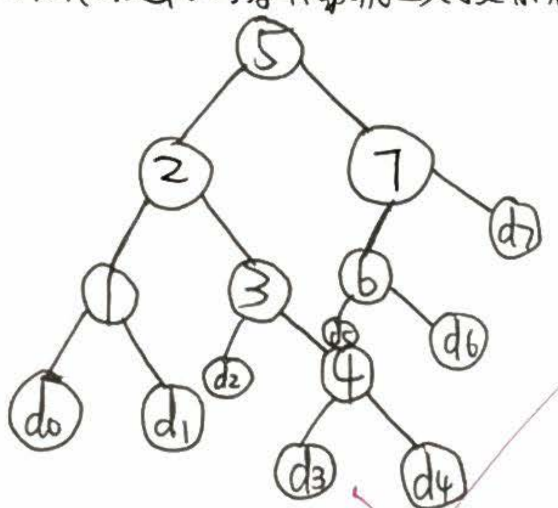
j \ i	1	2	3	4	5	6	7	8
7	1.00	0.90	0.78	0.64	0.56	0.41	0.24	0.05
6	0.81	0.71	0.59	0.45	0.37	0.22	0.05	
5	0.64	0.54	0.42	0.28	0.20	0.05		
4	0.49	0.39	0.27	0.13	0.05			
3	0.42	0.32	0.20	0.06				
2	0.28	0.18	0.06					
1	0.16	0.06						
0	0.06							

计算时 $w[i, j] = w[i, j-1] + p_j + q_j$

root: i

j \ i	1	2	3	4	5	6	7
7	5	5	5	6	6	7	7
6	3	5	5	5	6	6	
5	3	3	4	5	5		
4	2	3	3	4			
3	2	3	3				
2	2	2					
1	1						

故综上计算和画表可得其最优二叉搜索树为:



搜索代价 $e[1, 7] = 3.12$

15-9.

解：设m[i, j]为每次从i到j拆分花费的代价。用s[i, j]保存i→j中最优拆分点，L[n]表示有n个字符的字符串，以L作为算法输入参数：

则有：

$$m[i,j]=\begin{cases}0 & i=j\\ \min\limits_{i\le k\le j}\{m[i,k]+m[k+1,j]+\underbrace{\min\{L[j]-L[k]+1,\ L[k]-L[i]+1\}}_{\text{为此次拆分花费的代价}} & i<j\end{cases}$$

每次划分可以选择拷贝右边的还是左边的。

算法伪代码如下：

```
MIN-Time(L)
    t=L.length-1
    let m[1...t, 1...t] and s[1...n-1, 2...n] be new tables
    for i=1 to t
        m[i,j]=0
    for l=2 to t
        for i=1 to t-l+1
            j=i+l-1
            m[i,j]=∞
            for k=i to j-1
                q=m[i,k]+m[k+1,j]+min{L[j]-L[k], L[k]-L[i]}+1
                if q<m[i,j]
                    m[i,j]=q
                    s[i,j]=k
    return m and s
```

m[1...t, 1...t]：记录拆分代价；s[1...n-1, 2...n]：记录最优拆分点。

构造生成最优解，上述S为全局变量：

```
PRINT(S, 1, m)
    if i==j return;
    else print("%d", s[i,j]  PRINT(S, i, s[i,j]); PRINT(S, s[i,j]+1, j)
```

15-11.

解：依然运用自底向上算法，设r[n]保存每个子问题的最小化成本，即每个子问题的最优解。设P为需要付出的额外代价，即：如果有设备未售出而剩余，则P为库存成本，否则，P为额外劳动的雇佣成本.

```
MIN-COST(D, p)
  Let r[0...n] be a new array
    r[0]=0
    for j=1 to n
      q=+∞
      for i=1 to j
        q=min{q, p[i]+r[j-i]}
      r[j]=q
  return r[n]
```

时间复杂度说明?

16.1-4

解：修正算法：用这种算法的时间是将活动按时间来进行排序的时间
将所有的活动按开始时间和结束时间排序，按照活动开始时间进行遍历，把每个活动分配到那个时刻空闲的教室。同时维护两个列表：在时刻t空闲的教室列表free和已经安排了活动i的教室列表busy（活动：开始$s_i \le t$，结束$f_i > t$）。当t是某个活动的开始时刻时，把该活动分配到一个空闲教室，然后把该教室从free列表中移到busy列表。

当t是某个活动的结束时刻时，把该活动分配的教室，从busy列表中移到free列表中。

为了避免使用不必要的教室，在分配教室时，总是分配已经办过活动的教室，如果没有，再启用新教室

这就保证了使用尽可能少的教室.

16.2-6

解：分数背包问题 $O(n)$

改进的算法中不需对重量进行排序，因为如果n种物品重量和不大于W，无需知道它们之间的顺序。

运行时间：$T(n)=T(\frac{n}{2})+cn$ 同时 $T(1)=d$

时间复杂度为 $O(n)$

$O(n)$算法： $avg_i=\frac{v_i}{w_i}$

选择一个物品作为主元，对所有物品进行Paitition操作。

将avg分为三个集合 $G=\{a_i : avg_i > m\}$ $Q=\{a_i : avg_i = m\}$

$P=\{a_i : avg_i < m\}$ 分别对G、Q、P中元素重量求和，得 S_G, S_Q, S_P

①若 $W<S_G$，则令物品集合为 $O=G$，对集合O递归进行上述操作

②若 $S_G \le W \le S_G+S_Q$，则G中集合元素全部放入包中，然后对Q重复①②

③若 $W>S_G+S_Q$，则将G、Q中元素全部放入包中，令物品集合 $O=P$，总重 $W=W-S_G-S_Q$，递归进行上述过程.

16.2-7

解：由题意可知，特定排序不会影响结果，假设A中元素按照升序排列。

接下来证明当B也按照升序排列时，回报最大化.

用反证法，假设不成立，存在 $i<j$ 使 $a_i<a_j$ 且 $b_i>b_j$.

考虑指数 b_i，b_j 对结果的影响，对于 $a_i^{b_i}a_j^{b_j}$，若交换 b_i 和 b_j 的顺序，则得到 $a_i^{b_j}a_j^{b_i}$

∵ $\dfrac{a_i^{b_j}a_j^{b_i}}{a_i^{b_i}a_j^{b_j}} = \left(\dfrac{a_j}{a_i}\right)^{b_i-b_j} > 1$

∴ $a_i^{b_j}a_j^{b_i} > a_i^{b_i}a_j^{b_j}$

∴ 在B的这种排序下，回报不能最大化

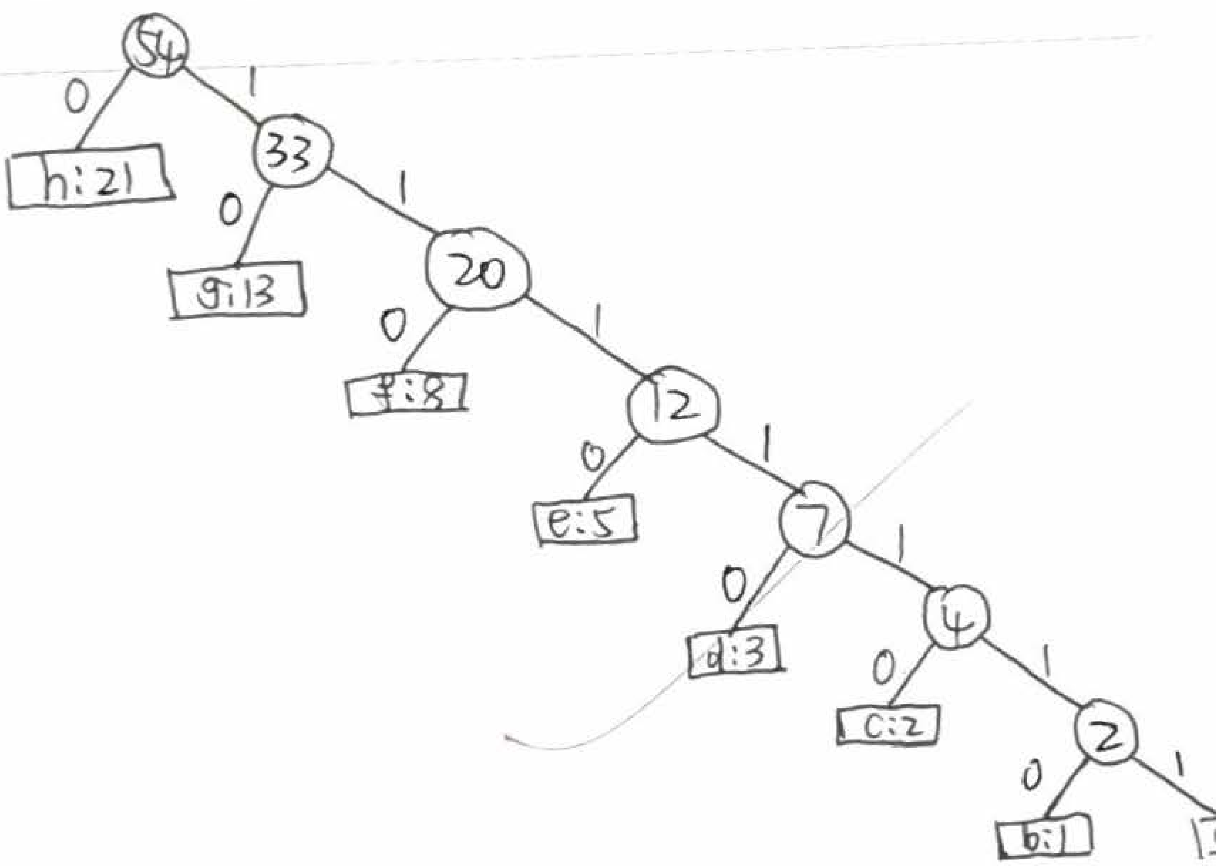
算法：将A升序排序且B升序排序时回报最大化

排序的时间复杂度为 $O(n\lg n)$

16.3-3

解：

a	b	c	d	e	f	g	h
1	1	2	3	5	8	13	21



赫夫曼编码为：

h:0
g:10
f:110
e:1110
d:11110
c:111110
b:1111110
a:1111111

推广：假设有n个字符，则第i个字符编码为：

$$\begin{cases} \text{若} i=1，\text{有} n-1 \text{位的"1"} \\ \text{若} i>1，\text{前}(n-i)\text{位是"1"，最后一位为"0"} \end{cases}$$

16-1.

解：a) 总是给能给的最高面额的硬币

然后重复这个过程直到剩余找零的总量降到0

b) 最理想方案 $(x_0, x_1, \ldots x_k)$，x_i表示硬币的面值 c^i 的数量

首先规定对于每个 $i<k$ 有 $x_i<c$

假设我们有 $x_i \geq c$，就能让 x_i 减 c 同时 x_{i+1} 加1

这样做使硬币总数减1同时总金额没有变化

因此最初的解决办法不是最理想的

这种硬币的分配方案和使用贪心算法每次尽可能取最高面值

的硬币相同。这是因为总值 V，$x_k = \lfloor Vc^{-k} \rfloor$ 并且对于 $i<k$，有

$x_i \lfloor (V \bmod c^{i+1}) c^{-i} \rfloor$。

这是唯一一种满足性质（除了最大面值的硬币没有其他的）

因为硬币总量是 $V \bmod c^k$ 的 c 基础代表

c) 硬币面额为 $\{1, 3, 4\}$，找零6

贪心算法算出方案 $\{1, 1, 4\}$，而最佳方案为 $\{3, 3\}$

d) 算法 MAKE-CHANGE(S, V) 是一个动态规划解法

第一个for循环运行n次，内层for循环运行k次，之后当循环

最多运行n次时，总运行时间为 $O(nk)$

MAKE-CHANGE(S, V)

let numcoins and coin be empty arrays of length V and

any any attempt to access them at indices in the range -max(S),

-1 should return ∞

```
for i from 1 to v do
    bestcoin = nil
    bestnum = ∞
    for c in S do
        if numcoins[i-c] + 1 < bestnum then
            bestnum = numcoins[i-c]
            bestcoin = c
        end if
    end for
    numcoins[i] = bestnum
    coin[i] = bestcoin
endfor
let change be an empty set
iter = V
while iter > 0 do
    add coin[iter] to change
    iter = iter - coin[iter]
endwhile
return change
```

25

设计证明题：

22.2-7

解：给每个结点增加一个属性，定为好人或坏人

链表中的结点的属性必与该链表头结点属性相反，按照这个规则去遍历链表，第一个链表表头结点属性设为好人。若在遍历过程中，发现新定义的属性和先前定义的属性有冲突，则不可以划分，否则可以划分，并且属性为好人和坏人的结点就是相应的划分。

由于使用BFS算法，所以复杂度为O(n+r)

24.1-3

解：若经过一轮松弛操作后，各点记录均无变化，则表明到达最后状态，可在松弛分支语句中加入对标志位的修改，若未修改，则表明无变化。

代码如下：

```
BELLMAN-FORD-UP(G, w, s)
INITIALIEE-SINGLE-SOURCE(G, s)
   flag=1
while(flag){    flag=0
   for each edge(u, v)∈G, E
      RELAX(u, v, w)
   for each edge(u, v)∈G, E
      if(v.d>u.d+w(u, v)){
            flag=1
            return FALSE
   return TRUE
```

24-3..

解：由 $R[i_1, i_2]\cdot R[i_2, i_3]\cdot \ldots \cdot R[i_{k-1}, i_k]$

$= \lg R[i_1, i_2] + \lg R[i_2, i_3] + \ldots + \lg R[i_{k-1}, i_k]$

则不等式可转化为：

$(-\lg R[i_1, i_2]) + \ldots + (-\lg R[i_k, i_1]) < 0$

由可令 n 种货币为 n 个结点，$-\lg R[i_1, i_2]$ 为货币 C_1 到 C_2 的边权重。

则问题转化为寻找图中负环路问题。

a. 可对 Floyd-Warshall 算法改进实现。

当运行 Floyd-Warshall 算法时，检测到有：$D^{(k)}[i, i] < 0$ 则退出算法.

Floyd-Warshall 运行时间为 $O(n^3)$，则运行时间为 $O(n^3)$

b. 由 Floyd-Warshall 的前驱矩阵可找出路径，运行时间为 $O(n^2)$

25-1.

解：a. 若插入边 (x, y) 则有如下更新

```
Update(t, u, v)
  for i = 1 to n
    for j = 1 to n
      t_ij^(k+1) = t_ij^(k) ∨ (t_ia^(k) ∧ t_bj^(k))
  return T^(n)
```

运行时间为 $O(n^2)$

b. 可给出一个链式：

$1 \to 2 \to 3 \to \ldots \to n$

则在布尔矩阵中有 $O(n^2)$ 个 0，此时给出边 $(n, 1)$，则需把 $O(n^2)$ 个 0 更新为 1.

c. 对于每个顶点我们都有一棵树，树中的顶点能够有路径通往树

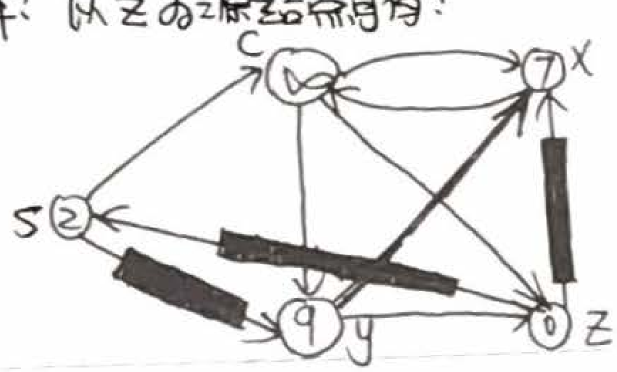
中的每个顶点，以及第一棵树包含能够从该顶点出发到达的顶点，第二棵树即可每一步的传递闭包，这样每插入一条边(u,v)，首先关注u的后继，把v加入后继中，如果在这一过程中不插入边，就可以在这棵树中停止搜索，对v的祖先进行同样操作.

最坏时间复杂度证明

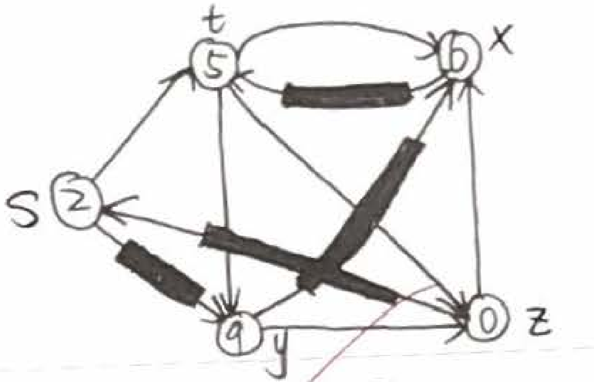
计算题：

24.1-1

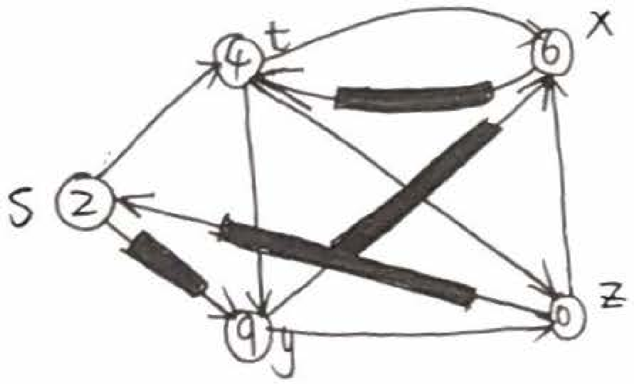
解：以z为源结点可有：



①



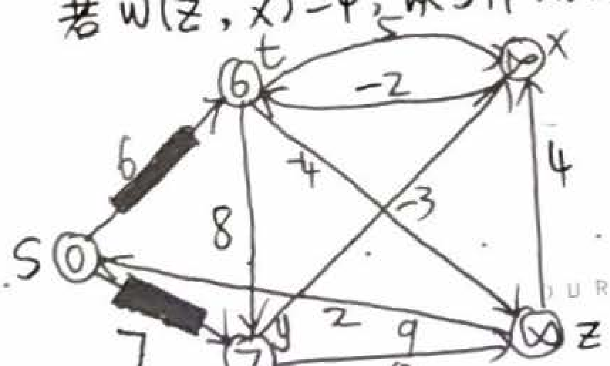
②

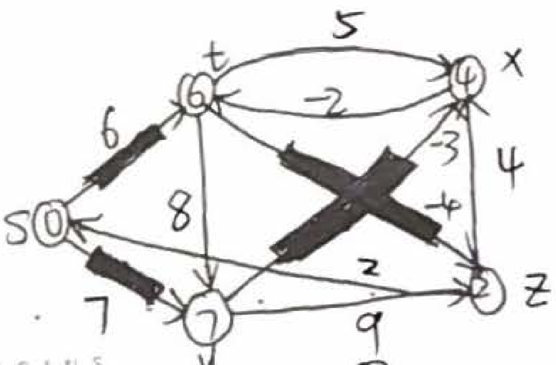


③

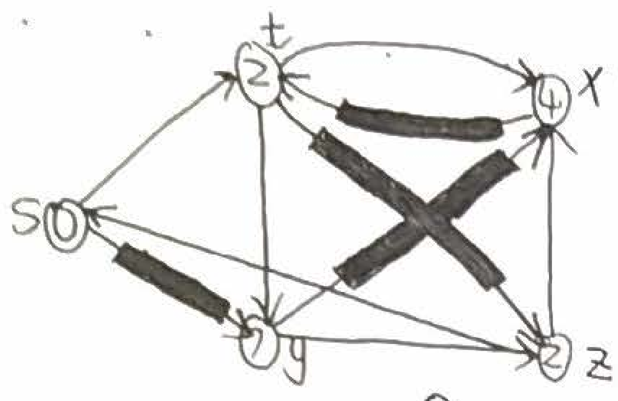
第④⑤次均与第③次结果相同.

若w(z,x)=4，从s作源结点，有：

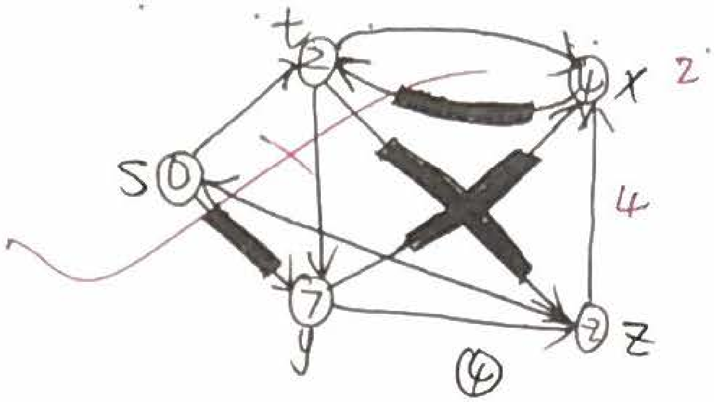




②



③



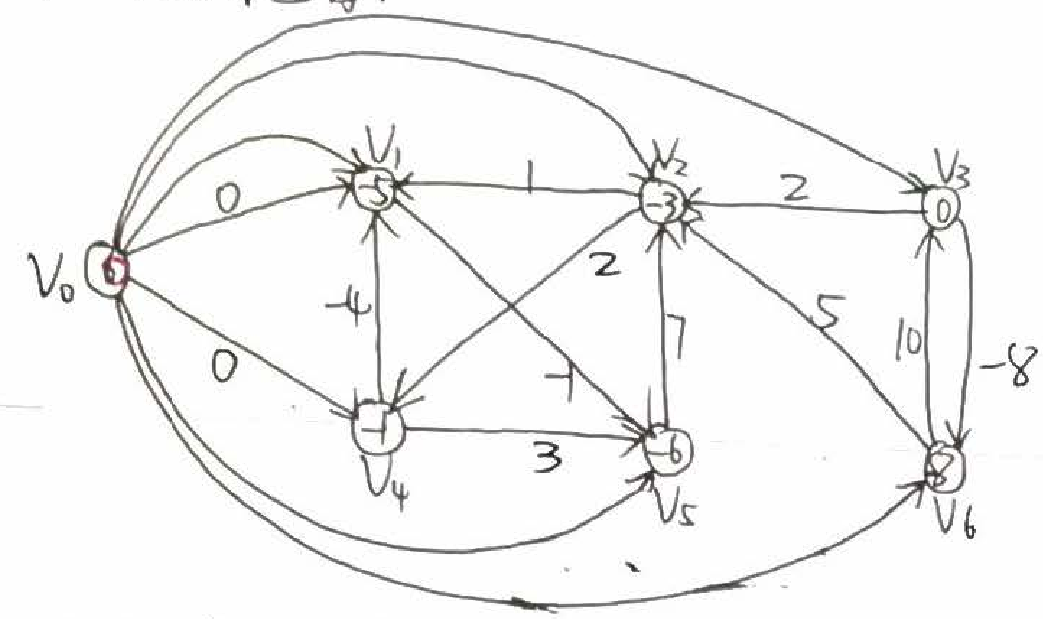
④

第⑤次与第④次相同.

24.4-1

没有输出的值　存在权重为负值的环路.

解：作约束图有：



运行Bellman Ford算法 可有：$X_1=-5$ $X_2=-3$ $X_3=0$ $X_4=-1$ $X_5=-6$ $X_6=-8$

25.2-1

解：

$$D^{(0)}=\begin{vmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & \infty & \infty \\ \infty & 2 & 0 & \infty & \infty & -8 \\ -4 & \infty & \infty & 0 & 3 & \infty \\ \infty & 7 & \infty & \infty & 0 & \infty \\ \infty & 5 & 10 & \infty & \infty & 0 \end{vmatrix} \quad D^{(1)}=\begin{vmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & 0 & \infty \\ \infty & 2 & 0 & \infty & \infty & -8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ \infty & 7 & \infty & \infty & 0 & \infty \\ \infty & 5 & 10 & \infty & \infty & 0 \end{vmatrix}$$

$$D^{(2)}=\begin{vmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & 0 & \infty \\ 3 & 2 & 0 & 4 & 2 & -8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ 8 & 7 & \infty & 9 & 0 & \infty \\ 6 & 5 & 10 & 7 & 5 & 0 \end{vmatrix} \quad D^{(3)}=\begin{vmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & 0 & \infty \\ 3 & 2 & 0 & 4 & 2 & -8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ 8 & 7 & \infty & 9 & 0 & \infty \\ 6 & 5 & 10 & 7 & 5 & 0 \end{vmatrix}$$

$$D^{(4)}=\begin{vmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ -2 & 0 & \infty & 2 & 0 & \infty \\ 0 & 2 & 0 & 4 & 2 & 8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ 5 & 7 & \infty & 9 & 0 & \infty \\ 3 & 5 & 10 & 7 & 5 & 0 \end{vmatrix}$$

$$D^{(5)}=\begin{vmatrix} 0 & 6 & \infty & 8 & -1 & \infty \\ -2 & 0 & \infty & 2 & -3 & \infty \\ 0 & 2 & 0 & 4 & -1 & \text{[illegible]} \\ -4 & 2 & \infty & 0 & -5 & \infty \\ 5 & 7 & \infty & 9 & 0 & \infty \\ 3 & 5 & 10 & 7 & 2 & 0 \end{vmatrix}$$

思考题：

25.2-6

解：若存在负权值的环路，则在最后通过Floyd-Warshall算法迭代生成的矩阵中，对角线上的值存在负值.

9.5