《算法分析与设计》2012-2013-2 学期期中测试（信息安全专业 DQ 教学班）

姓名：_____学号：_____得分：_____

1. 证明$O\left(f\left(n\right)\right)+O\left(g\left(n\right)\right)=O\left(f\left(n\right)+g\left(n\right)\right)$。（10 分）

证明：对于任意 $f_1(n) \in O(f(n))$，存在正常数 c_1 和自然数 n_1，使得对所有 $n \geq n_1$，有 $f_1(n) \leq c_1 f(n)$成立。类似，对于任意 $g_1(n) \in O(g(n))$，存在正常数 c_2 和自然数 n_2，使得对所有 $n \geq n_2$，有 $g_1(n) \leq c_2 g(n)$成立。

令 $c_3 = \max\{c_1, c_2\}$，$n_3 = \max\{n_1, n_2\}$，则对所有的 $n \geq n_3$，有

$$\begin{aligned} f_1(n) + g_1(n) &\leq c_1 f(n) + c_2 g(n) \\ &\leq c_3 f(n) + c_3 g(n) \\ &= c_3(f(n) + g(n)) \end{aligned}$$

即$O\left(f\left(n\right)\right)+O\left(g\left(n\right)\right)=O\left(f\left(n\right)+g\left(n\right)\right)$成立。

2. 将下列 5 个函数按渐近增长率由低至高进行排序，要求写出比较过程。（15 分）

$f_1(n)=n(\log n)^n$，$f_2(n)=\log n^{100\log n}$，$f_3(n)=n^2\log n$，$f_4(n)=2^{\log n+\log\log n}$，$f_5(n)=\sqrt[10]{n}$

解： $f_2(n)=\log n^{100\log n}=100\log^2 n$，$f_4(n)=2^{\log n+\log\log n}=n\log n$，

(1) $f_2(n)$是对数函数的幂，$f_5(n)$是幂函数，因此$f_2(n)=O\left(f_5(n)\right)$；

(2) $\lim\limits_{n\to\infty}\dfrac{f_4\left(n\right)}{f_5\left(n\right)}=\lim\limits_{n\to\infty}\dfrac{n\log n}{n^{1/10}}=\lim\limits_{n\to\infty}\left(n^{9/10}\log n\right)=\infty$，因此$f_5(n)=O\left(f_4(n)\right)$；

(3) $\lim\limits_{n\to\infty}\dfrac{f_4\left(n\right)}{f_3\left(n\right)}=\lim\limits_{n\to\infty}\dfrac{n\log n}{n^2\log n}=\lim\limits_{n\to\infty}\dfrac{1}{n}=0$，因此$f_4(n)=O\left(f_3(n)\right)$；

(4) 对 $f_1(n)$和 $f_3(n)$取对数，有

$\log f_1(n)=\log n+n\log\log n=\Theta\left(n\log\log n\right)=\Omega\left(n\right)$，$\log f_3(n)=2\log n+\log\log n=\Theta\left(\log n\right)$，

因为$\log n=O\left(n\right)$，所以$f_3(n)=O\left(f_1(n)\right)$；

因此，5 个函数按渐近增长率由低至高排序为 $f_2(n), f_5(n), f_4(n), f_3(n), f_1(n)$。

3. 给定按升序排列的 n 个不同整数存于数组 $a[1{:}n]$中，请设计$O\left(\log n\right)$的算法找到下标 i，$1 \leq i \leq n$，使得 $a[i] = i$，如不存在这样的下标，则返回 0。（15 分）

解：

令 head = 1, rear = n.

(1) 当 head <= rear 时，令 mid = ⌊(head + rear)/2⌋；

(2) 如果 a[mid] = mid，返回 mid 值，结束。

如果 a[mid] > mid，令 rear = mid – 1，返回(1)继续执行；

如果 a[mid] < mid，令 head = mid + 1，返回(1) 继续执行；

(3)返回 0 值，结束。

```
public static int Search(int [] a, int n)
{
    // 在 a[0] <= a[1] <= ... <= a[n-1] 中搜索 a[i] = i
    // 找到 a[i] = i 时返回 i，否则返回 0
    int left = 0; int right = n – 1;
    while (left <= right)
    {
        int mid = ⌊(left + right)/2⌋;
        if (a[mid] == mid) return mid;
        if (a[mid] > mid) right = mid – 1;
        else left = mid + 1;
    }
    return 0; // 未找到 a[i] = i
}
```

4. 利用主方法给出下列递归式的渐近界，并用数学归纳法证明式(2)的渐近界。（20 分）

(1) $T(n)=4T(n/2)+n$, (2) $T(n)=4T(n/2)+n^2$, (3) $T(n)=4T(n/2)+n^3$

解：(1) $a=4,b=2,\log_b a=\log_2 4=2,$

$f(n)=n=O(n^{2-\varepsilon}),$ if $\varepsilon=0.5,$

因此， $T(n)=\Theta(n^2)$.

(2) $a=4,b=2,\log_b a=\log_2 4=2,$

$f(n)=n^2=\Theta(n^2),$

因此， $T(n)=\Theta(n^2\log n)$.

(3) $a=4,b=2,\log_b a=\log_2 4=2,$

$f(n)=n^3=\Omega(n^{2+\varepsilon}),$ if $\varepsilon=0.5,$

而且 $4f(n/2)=4(n/2)^3=n^3/2\le 3n^3/4=cf(n),$

其中 $n=3/4$，

因此， $T(n)=\Theta(n^3)$.

证明：假设当 $k<n$ 时， $T(k)\le ck^2\log k$ ，其中 c 为常数。

$$\begin{aligned}T(n)&=4T(n/2)+n^2\\&\le 4c(n/2)^2\log(n/2)+n^2\\&=cn^2(\log n-1)+n^2\\&=cn^2\log n-(c-1)n^2\\&\le cn^2\log n \text{ if } c\ge 1\end{aligned}$$

因此，命题得证。

5. 利用直接展开法求解下列递归式的渐近界。（20 分）

(1) $T(n)=4T(n/2)+n^2$, (2) $T(n)=\sqrt{n}T(\sqrt{n})+n$

解：(1)

$$\begin{aligned}&T(n)=4T(n/2)+n^2\\&=4\left(4T(n/2^2)+n^2/2^2\right)+n^2\\&=4^2T(n/2^2)+2n^2\\&=4^2\left(4T(n/2^3)+n^2/2^4\right)+2n^2\\&=4^3T(n/2^3)+3n^2\\&=\cdots\\&=4^kT(n/2^k)+kn^2\\&=\cdots\\&=4^{\log n}T(n/2^{\log n})+n^2\log n\\&=n^2T(1)+n^2\log n\\&=O(n^2\log n)\end{aligned}$$

(2) $T(n)=\sqrt{n}T(\sqrt{n})+n$

$$\begin{aligned}&T(n)=\sqrt{n}T(\sqrt{n})+n\\&\frac{T(n)}{n}=\frac{T(n^{1/2})}{n^{1/2}}+1\\&=\frac{T(n^{1/4})}{n^{1/4}}+1+1\\&=\frac{T(n^{1/8})}{n^{1/8}}+1+1+1\\&=\cdots\\&=\frac{T(n^{1/2^k})}{n^{1/2^k}}+k\\&=\cdots\\&=\frac{T(2)}{2}+k',\end{aligned}$$

其中 $2=n^{1/2^{k'}}$ ，则

$$\begin{aligned}1/2^{k'}&=\log_n 2\\2^{k'}&=\log n\\k'&=\log\log n\end{aligned}$$

所以， $\frac{T(n)}{n}=\frac{T(2)}{2}+\log\log n=\Theta(1)+\log\log n$ ，

即 $T(n)=O(n\log\log n)$.

6. 某超市中有 n 种商品搞活动，每种商品 i 的重量是 w_i，其价格为 v_i，现在给你发一个容量为 C 的购物车，你可以在这 n 种商品中选一些放入你的购物车中免费带走。但是要求所选的商品重量之和不能大于购物车容量 C，而且超市中每种商品每人最多选两件。请问在这种情况下你如何选择商品使得你能带走的免费商品的价格达到最大？（20 分）

(a) 为该问题设计一个动态规划算法，要求写出分析过程和递归式。

(b) 若该超市共有 3 种商品搞活动，商品的价值依次为 $v = (25, 30, 15)$，商品的重量依次为 $w = (2, 3, 1)$，购物车容量为 $C = 5$。运用自底而上的方法求解上述问题，要求画出表格，并给出最优解与最优值！

解：

方法 1

- 将 n 种商品全部复制一份得到 $2n$ 种商品，这样每种商品最多只能选择 1 件。
- 定义 $m(i, j)$为购物车容量为 j，由第 1, …, i 种商品装入购物车的最优值。
- **Case 1:** 不选择第 i 种商品
  - 则 $m(i, j)$为当重量限制为 j 时，$\{1, \ldots, i-1\}$种商品装入购物车所产生的最大价值
- **Case 2:** 选择第 i 种商品.
  - 新的重量限制为 $j - w_i$
  - $m(i, j - w_i)$ 为新重量限制下，$\{1, \ldots, i-1\}$种商品装入购物车所产生的最大价值

因此，递归式如下：

$$m(i,j)=\begin{cases}0 & \text{if } i=0 \text{ or } j=0\\ m(i-1,j) & \text{if } w_i > j\\ \max\{m(i-1,j), v_i + m(i-1, j-w_i)\} & \text{otherwise}\end{cases}$$

C = 5

Item	Value	Weight
1	25	2
1’	25	2
2	30	3
2’	30	3
3	15	1
3’	15	1

W + 1 →

n + 1 ↓

	0	1	2	3	4	5
ϕ	0	0	0	0	0	0
{ 1 }	0	0	25	25	25	25
{ 1, 1’ }	0	0	25	25	50	50
{ 1, 1’, 2 }	0	0	25	30	50	55
{ 1, 1’, 2, 2’ }	0	0	25	30	50	55
{ 1, 1’, 2, 2’, 3 }	0	15	25	40	50	65
{ 1,1’,2,2’,3,3’}	0	15	30	40	55	65

最优解：选择商品 1,1’,3, 即选择两个商品 1，一个商品 3

最优值 = 25+25+15=65

方法 2

- 定义 $m(i, j)$为购物车容量为 j，由第 1, …, i 种商品装入购物车的最优值。
- Case 1: 不选择第 i 种商品
  - 则 $m(i, j)$为当重量限制为 j 时，{1, …,$i-1$}种商品装入购物车所产生的最大价值
- Case 2: 仅选择 1 格第 i 种商品.
  - 新的重量限制为 $j - w_i$
  - $m(i, j - w_i)$ 为新重量限制下，{1, …, $i-1$}种商品装入购物车所产生的最大价值
- Case 3: 选择两个第 i 种商品
  - 新的重量限制为 $j - 2w_i$
  - $m(i, j - 2w_i)$ 为新重量限制下，{1, …,$i-1$}种商品装入购物车所产生的最大价值

因此，递归式如下：

$$m(i,j)=\begin{cases}0 & \text{if } i=0 \text{ or } j=0\\ m(i-1,j) & \text{if } j<w_i\\ \max\{m(i-1,j),v_i+m(i-1,j-w_i)\} & \text{if } w_i\le j<2w_i\\ \max\begin{Bmatrix}m(i-1,j),v_i+m(i-1,j-w_i),\\ 2v_i+m(i-1,j-2w_i)\end{Bmatrix} & \text{if } j\ge 2w_i\end{cases}$$

C = 5

Item	Value	Weight
1	25	2
2	30	3
3	15	1

W + 1 →

n + 1 ↓

	0	1	2	3	4	5
ϕ	0	0	0	0	0	0
{1}	0	0	25	25	50	50
{1, 2}	0	0	25	30	50	55
{1, 2, 3}	0	15	30	40	55	65

最优解：选择两个商品 1，一个商品 3

最优值 = 25+25+15=65

第 2 章作业：算法分析基础

1. 算法与程序的区别

(1).算法特性之一是有穷性，程序不一定满足有穷性。

(2). 计算机程序是用来给计算机读的，而算法是给人来读的，直接将算法输入计算机是不能运行的。

(3).算法是解决问题的一种方法或一个过程，而程序则是算法用某种程序设计语言的具体实现。

2. 将下列函数按渐进增长率由低到高排列出来。

$f_1(n)=2^{\sqrt{\log n}}, f_2(n)=n^{n^2}, f_3(n)=\left(\sqrt{\log n}\right)^{4/3}, f_4(n)=n(\log n)^{\log n}, f_5(n)=2^{2^n}, f_6(n)=n^{\log n},$

令 $m=\log n$ ，则有

$g_1(n)=\log f_1(n)=m^{1/2},$

$g_3(n)=\log f_3(n)=\frac{2}{3}\log m,$

$g_4(n)=\log f_4(n)=m+m\log m=\Theta(m\log n),$

$g_6(n)=\log f_6(n)=m^2,$

显然上述 4 个函数的渐近增长率排序为 $g_3(n)\le g_1(n)\le g_4(n)\le g_6(n)$；

$g_2(n)=\log f_2(n)=n^2\log n,$

$g_5(n)=\log f_5(n)=2^n,$

$g_6(n)=\log f_6(n)=\log^2 n,$

显然上述 3 个函数的渐近增长率排序为 $g_6(n)\le g_2(n)\le g_5(n)$；

因此，原函数的渐近增长率排序为 $f_3(n)\le f_1(n)\le f_4(n)\le f_6(n)\le f_2(n)\le f_5(n)$ 。

3. 已知 $g(n)=O(f(n))$，证明 $f(n)+g(n)=O(f(n))$。

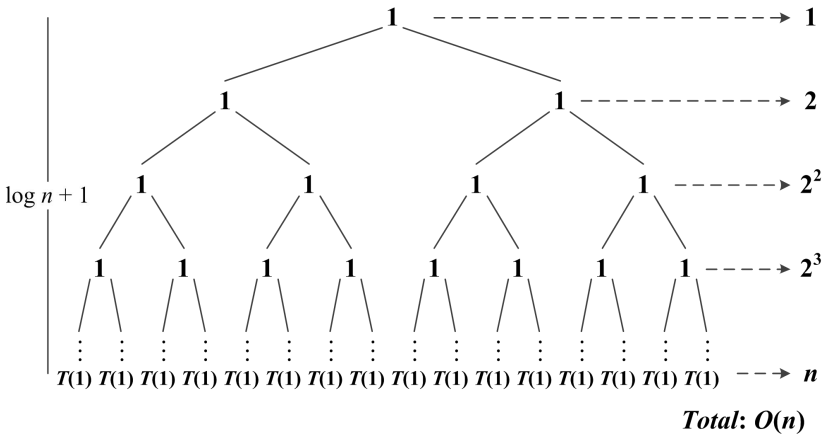
证明：因为 $g(n)=O(f(n))$，存在正常数 c_0 和自然数 n_0，使得对所有 $n\ge n_0$，有 $g(n)\le c_0 f(n)$ 成立。
令 $c_1=c_0+1$，则对所有的 $n\ge n_0$，有

$$\begin{aligned} f(n)+g(n) &\le f(n)+c_0 f(n) \\ &\le (1+c_0)f(n) \\ &= c_1 f(n) \end{aligned}$$

即 $f(n)+g(n)=O(f(n))$ 成立。

第 3 章作业：分治递归

1.画出 T(n)=2T(n/2)+1 的递归树，并给出其解的渐进界。然后用数学归纳法证明给出的界。



证明：假设当$k<n$时，$T(k)\le c_1k-c_2$，其中 c_1 和 c_2 为常数。

$$\begin{aligned}T(n)&=2T(n/2)+1\\&\le 2\left[c_1(n/2)-c_2\right]+1\\&=c_1n-(2c_2-1)\\&\le c_1n-c_2 \ \text{ if } \ c_2\ge 1\end{aligned}$$

因此，命题得证。

2.直接展开法求解递归式 T(n)=T(n/2)+1

$$\begin{aligned}&T(n)=T(n/2)+1\\&=T(n/2^2)+2\\&=T(n/2^3)+3\\&=\cdots\\&=T(n/2^k)+k\\&=\cdots\\&=T(n/2^{\log n})+\log n\\&=T(1)+\log n=O(\log n)\end{aligned}$$

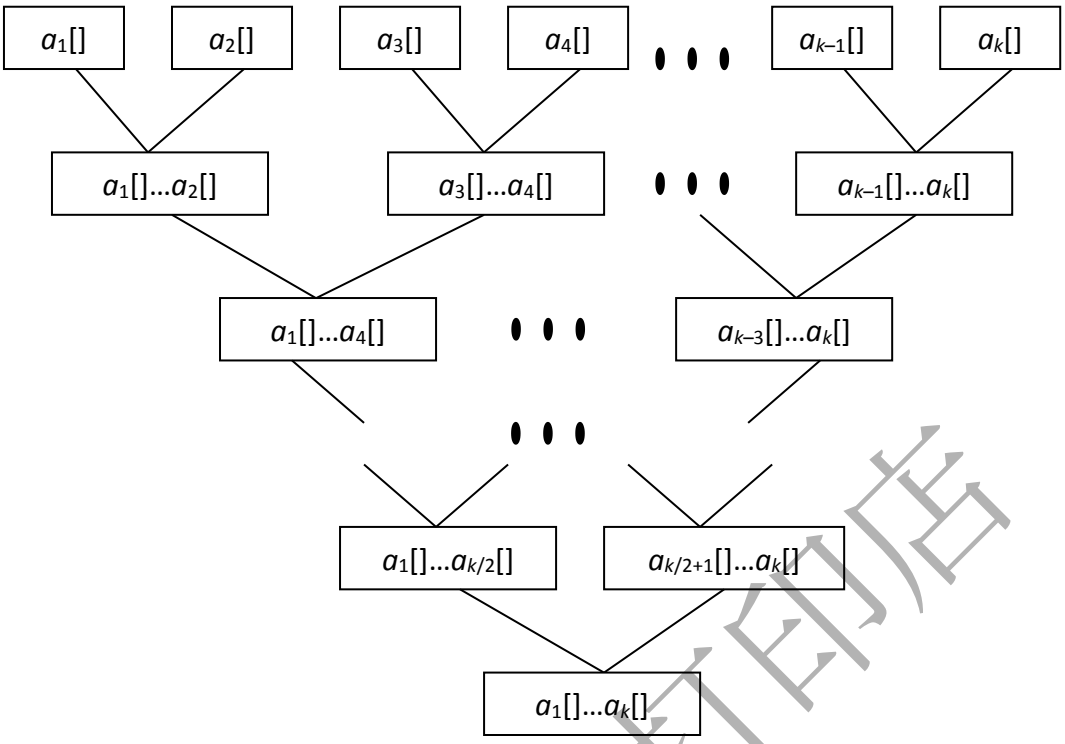
3.用主方法求解递归式

1.T(n)=4T(n/2)+n, 2.T(n)=4T(n/2)+n^2, 3.T(n)=4T(n/2)+n^3, 4.T(n)=3T(n/2)+ n^2

请参考期中测试答案。

第 4 章作业：动态规划

1、设计一个 O(n*log k) 时间的分治递归算法，将 k 个排好序的序列合并成一个排好序的序列，其中 k 个序列共计包含 n 个元素。



将 k 个有序数列两两合并排序，得到$\lceil k/2 \rceil$个有序数列，再次两两合并排序，得到$\lceil k/4 \rceil$个有序数列，重复操作直至得到一个长度为 n 的有序数列。共计 $\log k$ 次合并排序，每次合并排序数据的总体规模为 n，则时间复杂度为 O(n log k)

2、给定数组 a[1…n]，n 为偶数。设计一个算法，在最坏情况下用 3n/2 – 1 次比较找出 a[1…n]中元素的最大值和最小值。

将 a[1]…a[N]两两分成一组，即为 a[1]和 a[2]， a[3]和 a[4]， a[5]和 a[6]， … 共计 N/2 组，每组中的两个元素进行比较，得到最大值和最小值，共计 N/2 次比较；

上述每组中的最大值组成数组 b[1…N/2]，冒泡排序(或其他排序方法)求得 b[1…N/2]中的最大值，即为原数组 a[1…N]的最大值，需 N/2 － 1 次比较；

同理，上述每组中的最小值组成数组 c[1…N/2]，冒泡排序(或其他排序方法)求得 c[1…N/2]中的最小值，即为原数组 a[1…N]的最小值，需 N/2 － 1 次比较；

因此，共需 3N/2 – 1 次比较.

3、设计 $O(n^2)$时间的动态规划算法，找出由 n 个数组成的序列的最长单调递增子序列，要求给出分析过程和递归表达式。

原序列记为 X, 对 n 个数递增排序, 构造一个新序列 Y, 对 X, Y 求其最长公共子序列即可.

设序列 $X=\{x_1,x_2,\ldots,x_m\}$和 $Y=\{y_1,y_2,\ldots,y_n\}$的最长公共子序列为 $Z=\{z_1,z_2,\ldots,z_k\}$ ，则

(1)若 $x_m=y_n$，则 $z_k=x_m=y_n$，且 z_{k-1} 是 x_{m-1} 和 y_{n-1} 的最长公共子序列。

(2)若 $x_m\neq y_n$ 且 $z_k\neq x_m$，则 Z 是 x_{m-1} 和 Y 的最长公共子序列。

(3)若 $x_m\neq y_n$ 且 $z_k\neq y_n$，则 Z 是 X 和 y_{n-1} 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]记录序列和的最长公共子序列的长度。其中， $X_i=\{x_1,x_2,\ldots,x_i\}$； $Y_j=\{y_1,y_2,\ldots,y_j\}$。当 i=0 或 j=0 时，空序列是 X_i 和 Y_j 的最长公共子序列。故此时 c[i][j]=0。其他情况下，由最优子结构性质可建立递归关系如下：

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

4、两个序列进行对齐时，匹配得 1 分，错配罚 1 分，空位罚 2 分。给定两个字符序列 X=CTACG 和 Y=TACAG，如何对齐才能使得得分最大？根据递归公式自底而上求解上述问题，要求画出求解过程表格，并给出最大得分和相应的对齐方式。

OPT(i, j) = 对齐 $x_1\ x_2\ \ldots\ x_i$ 和 $y_1\ y_2\ \ldots\ y_j$.最大的得分

Case 1: OPT 匹配 x_i-y_j.

x_i-y_j 配对的得分 + 对齐 $x_1\ x_2\ \ldots\ x_{i-1}$ 和 $y_1\ y_2\ \ldots\ y_{j-1}$ 最大的得分

Case 2a: OPT 留下 x_i 不匹配

$-2+$ 对齐 $x_1\ x_2\ \ldots\ x_{i-1}$ 和 $y_1\ y_2\ \ldots\ y_j$ 最大的得分

Case 2b: OPT 留下 y_j 不匹配

$-2+$ 对齐 $x_1\ x_2\ \ldots\ x_i$ 和 $y_1\ y_2\ \ldots\ y_{j-1}$ 最大的得分

$$OPT(i,j)=\begin{cases}-2j & \text{if } i=0\\ \max\begin{cases}\alpha_{x_iy_j}+OPT(i-1,j-1)\\ -2+OPT(i-1,j)\\ -2+OPT(i,j-1)\end{cases} & \text{otherwise}\\ -2i & \text{if } j=0\end{cases}$$

OPT(*i*,*j*) *j* / *i*	0	1	2	3	4	5
0	0	– 2	– 4	– 6	– 8	– 10
1	– 2	– 1	– 3	– 3	– 5	– 7
2	– 4	– 1	– 2	– 4	– 5	– 7
3	– 6	– 3	0	– 2	– 3	– 5
4	– 8	– 5	– 2	1	– 1	– 3
5	– 10	– 7	– 4	– 1	0	0

最优对齐方式为

C	T	A	C		G
	T	A	C	A	G

第 5 章作业：贪心算法

1、请叙述分治递归、动态规划、备忘录方法和贪心算法各自的特点和基本要素，并比较它们之间的相同之处及其差异。

分治递归法所能解决的问题一般具有以下几个特征:

(1) 该问题规模缩小到一定的程度就可以容易地解决；

(2) 该问题可以分解为若干个规模较小的相同问题，即该问题具有递归子结构性质

(3) 利用该问题分解出的子问题的解可以合并为该问题的解；

(4) 该问题所分解出的各个子问题是相互独立的，即子问题之间不包含公共的子问题。

动态规划与分治的比较

(1) 共同点：递归子结构

将待求解问题分解成若干个规模较小的相同类型的子问题，先求解子问题，然后从这些子问题的解得到原问题的解。

(2) 不同点：重叠子问题

适合于用动态规划求解的问题，经分解得到子问题往往不是互相独立的。若用分治法来解这类问题，则分解得到的子问题数目太多，有些子问题被重复计算了很多次。

(3) 不同点：求解问题的顺序

分治递归、备忘录方法：自顶向下，动态规划：自底向上，

贪心算法的基本要素

(1) 贪心选择性质

(2) 最优子结构性质

贪心算法的特点

(1) 总是作出局部最优选择

(2) 不能保证得到全局最优解

贪心法与动态规划的比较

相同点：递推算法、最优子结构、局部最优解→全局最优解

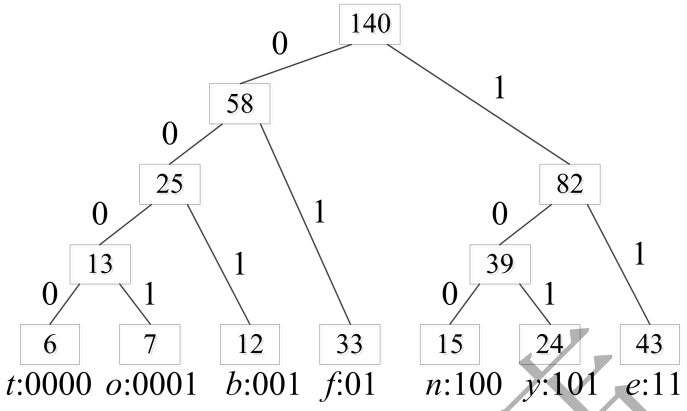
不同点：

动态规划：自底向上，贪心算法：自顶向下

贪心算法：当前局部最优解←上一步局部最优解，动态规划：当前局部最优解←之前所有局部最优解

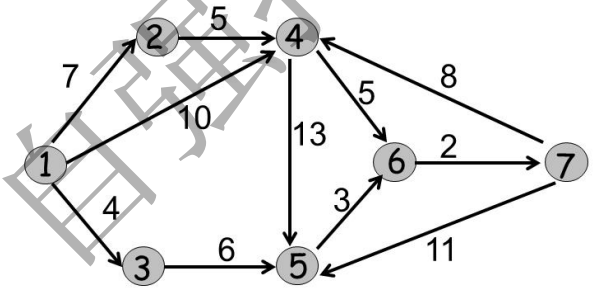
2、给定一个包含{b,e,f,n,o,t,y}字符集的文本文件，每个字符在文件中出现频率如下表所示，文件共有140个字符，试用huffman编码对该文本文件进行编码，要求画出编码的huffman树，给出每个字符的huffman编码，并求解编码后文件的长度。

字符	b	e	f	n	o	t	y
次数	12	43	33	15	7	6	24



编码后文件长度 $= (6 + 7) \times 4 + (12 + 15 + 24) \times 3 + (33 + 43) \times 2 = 357$ bit

3、用 Dijkstra 算法求下图中顶点1到其他所有顶点的最短路径，要求给出计算过程和最优解、最优值。



解1：

迭代	S	u	The distance from node 1 to node #					
			2	3	4	5	6	7
初始	{1}	-	7	4	10	∞	∞	∞
1	{1,3}		7	4	10	10	∞	∞
2	{1,3,2}		7	4	10	10	∞	∞
3	{1,3,2,4}		7	4	10	10	15	∞
4	{1,3,2,4,5}		7	4	10	10	13	∞
5	{1,3,2,4,5,6}		7	4	10	10	13	15
	{1,3,2,4,5,6,7}		7	4	10	10	13	15

解 2：

迭代	S	u	The distance from node 1 to node #					
			2	3	4	5	6	7
初始	{1}	-	7	4	10	∞	∞	∞
1	{1,3}		7	4	10	10	∞	∞
2	{1,3,2}		7	4	10	10	∞	∞
3	{1,3,2,5}		7	4	10	10	13	∞
4	{1,3,2,5,4}		7	4	10	10	13	∞
5	{1,3,2,4,5,6}		7	4	10	10	13	15
	{1,3,2,4,5,6,7}		7	4	10	10	13	15

最短路径：

1→2：1,2

1→3：1,3

1→4：1,4

1→5：1,3,5

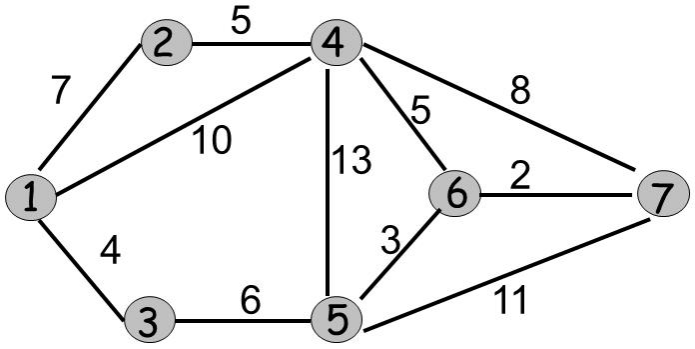
1→6：1,3,5,6

1→7：1,3,5,6,7

4、分别叙述用 Prim 算法和 Kruskal 算法求最小生成树的过程，并分别用上述两种算法求下图的最小生成树，要求给出最小生成树中边的生成顺序，画出最小生成树，并最小权重之和。（其中 Prim 算法从节点 1 开始执行）

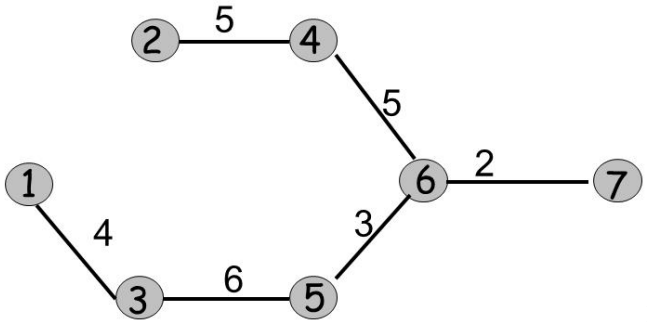
Prim‘s 算法. 首先置 S={1}，然后，只要 S 是 V 的真子集，就作如下的贪心选择：选取满足条件 i∈S，j∈V-S，且 c[i][j]最小的边，将顶点 j 添加到 S 中。这个过程一直进行到 S=V 时为止

Kruskal‘s 算法. 首先将 G 的 n 个顶点看成 n 个孤立的连通分支。将所有的边按权从小到大排序。然后从第一条边开始，依边权递增的顺序查看每一条边，并按下述方法连接 2 个不同的连通分支：当查看到第 k 条边(v,w)时，如果端点 v 和 w 分别是当前 2 个不同的连通分支 T1 和 T2 中的顶点时，就用边(v,w)将 T1 和 T2 连接成一个连通分支，然后继续查看第 k+1 条边；如果端点 v 和 w 在当前的同一个连通分支中，就直接再查看第 k+1 条边。这个过程一直进行到只剩下一个连通分支时为止。



Prim‘s 算法.(1,3), (3,5), (5,6), (6,7), (6,4), (4,2)

Kruskal‘s 算法. (6,7), (5,6), (1,3), (6,4), (4,2), (3,5)



权重之和为 15。