武汉大学计算机学院
2006 级《算法设计与分析》考试试卷

1. （10 分）对下表中的每一对 f (n)， g (n)，问 f (n) 是否 $O(g(n)), \Omega(g(n)), \Theta(g(n))$， $o(g(n))$，若是，则在相应空格打√。

【P19 复杂性类比较层次、P17-18 举例】

f (n)	g (n)	O	Ω	Θ	o
n^5	1.2^n	√			√
$(\log n)^3$	n^2	√			√
3^n	2^n		√		

2. （10 分）求解下列递归关系

$f(n)=1$ （n=1）

$f(n)=8f(n/2)+n^2$ $(n>=2)$

其中 $n=2^k, k$ 为一个非负整数。

解： f(n)=2(n^3)-n^2…… P57

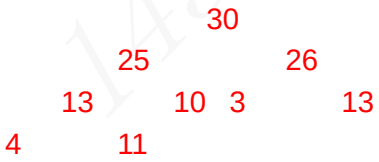
3. （15 分）回答下列问题。

（a）依次将 13，4，25，26，10，3，30，11，13 插入一个非空的极大堆，画出所得到的极大堆的表示；（5 分）

（b）高度为 h 的堆最少有几个元素？最多有几个元素？（二叉树的高度定义如下：根结点的层次为 0，若一个结点的层次为 l，则其儿子结点的层次为 $l+1$，树中结点的最大层次为树的高度）；（5 分）

（c）证明具有 n 个元素的堆的高度为 $\lfloor \log n \rfloor$。（5 分）

解：(a).



(b). 高度为 h 的堆最少有 2^h 个元素（不是 2^(h-1)，因为高度从 0 开始不是从 1 开始）；最多有 2^（h+1）个元素

(c). 证明：运用数学归纳法:

4.（10 分）给定数组 A=（10,12,17,26,33,75,81,96），使用二分搜索依次搜索 4,12,11,35,87，请统计一共进行了多少次比较？

解： 4→26/12/10 【3】
12→26/12 【2】
11→26/12/10 【3】

	1	2	3
1	0	4	6
2	6	0	2
3	3	7	0

35→26/75/33 【3】
87→26/75/81/96 【4】
一共比较次数： 3+2+3+3+4=15（次）

5.设含有 3 个结点的有向图 G 的成本矩阵如下图所示，试用 FLOYD 算法求解出该有向图的所有结点对间的最短距离。

	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

解：p136

	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

	1	2	3
1	0	4	11
2	6	0	2
3	3	7	0

	1	2	3
1	0	4	6
2	5	0	2
3	3	7	0

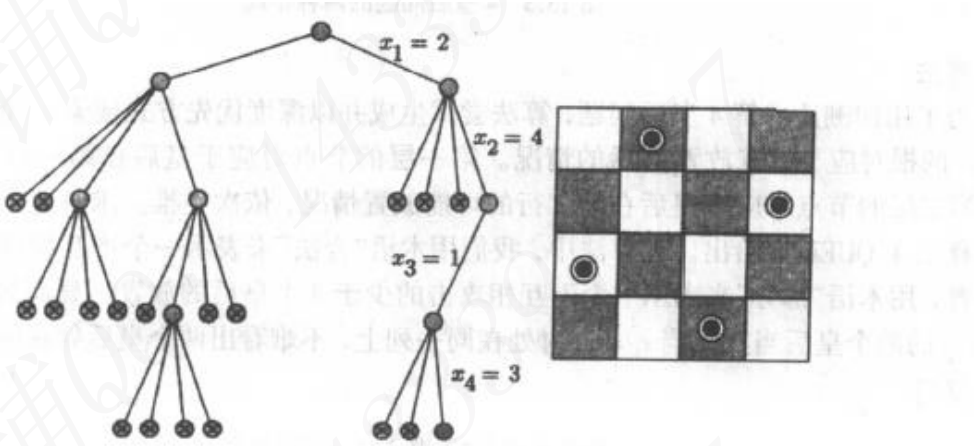
6.(10 分)KRUSKAL 是找无向图的最小生成树的算法，证明在含权无向图中，KRUSKAL 能正确找出最小生成树。

解：p153

证明：我们用对 T 的大小施行归纳法来证明，T 是最小生成树边集的子集。初始时，$T=\{\}$命题平凡地真。对于归纳步：假设在此算法第 7 步中加入边 $e=(x,y)$之前有 $T\subset T^*$，这里 T^* 是最小生成树 $G^*=(V,T^*)$边的集合。设 X 是包含 x 的子树的顶点集，设 $T'=T\cup\{e\}$，我们将证明 T'也是最小生成树中边集的子集。根据假设归纳，$T\subset T^*$，如果 T^* 包含 e，那么就不需要进一步证明，否则根据定理 3.1(c)，$T^*\cup\{e\}$恰好包含以 e 为一条边的一个回路，因为 $e=(x,y)$连接了 X 中的一个顶点和 $V-X$ 中的另一个顶点，T^* 必定也包含另一条边 $e'=(w,z)$，使得 $w\in X$，并且 $z\in V-X$。我们得知 $cost(e')\geqslant cost(e)$；否则 e'应该在前边添加，因为它不与 T^*（它含有 T 的边）的边构成回路。如果我们现在构造 $T^{**}=T^*-\{e'\}\cup\{e\}$，注意到 $T'\subseteq T^{**}$。并且，T^{**}是最小值生成树的边的集合，因为 e 是连结 $V-X$ 中顶点与 X 中顶点的边中值最小者。

7.（15 分）利用回溯法求 4 皇后问题，设解的形式为（x_1,x_2,x_3,x_4），其中 x_i 表示第 i 个皇后放在第 i 行，第 x_i 列，画出求解过程中生成的部分状态树（搜索树），

（10 分）给出最终解的形式（2 分），并统计算法总共生成了多少个结点。（3 分）
(p222)解：



最终解：（2，4，1，3）或者（3，1，4，2）
总共产生结点 27 个

8.（10 分）已知可满足性问题是 NP 完全问题，证明团集问题也是 NP 完全问题。
解： 定义 10.4+推论 10.1+ p180（可满足性“多项式时间规约到”团集）

9.（10 分）编辑距离问题：设 A 和 B 是 2 个字符串，要用最少的字符操作将字符串 A 换为字符串 B，这里所说的字符操作包括（1）删除一个字符；（2）插入一个字符；（3）将一个字符该为另一个字符。将字符串 A 变换为字符串 B 所用的最少字符操作数称为字符串 A 到 B 的编辑距离，记为$d(A,B)$。设计一个动态规划算法，给定任意两个字符串 A 和 B，算法能输出该实例的编辑距离$d(A,B)$。（要求写出算法的思想和算法的代码或者伪码）。

解：

```
//完整程序代码//
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int _Min(int a,int b,int c)
{
int min=a;
if (b <min)
min=b;
if(c <min)
min=c;
return min;
}

int ComputeDistance(char s[],char t[])
```

```
{
int n = strlen(s);

int m = strlen(t);

//int d[][] = new int[n + 1, m + 1]; // matrix
int **d = (int **)malloc((n+1) * sizeof(int *));
for(int i=0; i<=n; ++i)
{
d[i] = (int *)malloc((m+1) * sizeof(int));
}
// Step 1
if (n == 0)
{
return m;
}

if (m == 0)
{
return n;
}

// Step 2
for (int i = 0; i <= n; i++)
{
d[i][0] =i;
}

for (int j = 0; j <= m; d[0][j] = j++)
{
d[0][j] =j;
}

// Step 3
for (int i = 1; i <= n; i++)
{
//Step 4
for (int j = 1; j <= m; j++)
{
// Step 5
int cost = (t[j-1] == s[i-1]) ? 0 : 1;
```

```
// Step 6
d[i][j] = _Min(d[i-1][j]+1, d[i][j-1]+1,d[i-1][j-1]+cost);
}
}
// Step 7
return d[n][m];
}

int main(int argc, char *argv[])
{
char a[9999];
char b[9999];
printf("请输入字符串 1\n");
scanf("%s",&a);
printf("请输入字符串 2\n");
scanf("%s",&b);
int result= ComputeDistance(a,b);
printf("%d\n",result);
system("PAUSE");
return 0;
}
```

编辑距离的性质

计算两个字符串 s1+ch1, s2+ch2 的编辑距离有这样的性质:

1. d(s1,””) = d(“”,s1) = |s1| d(“ch1”,”ch2”) = ch1 == ch2 ? 0 : 1;
2. d(s1+ch1,s2+ch2) = min(d(s1,s2)+ ch1==ch2 ? 0 : 1 ,
d(s1+ch1,s2),
d(s1,s2+ch2));

第一个性质是显然的。

第二个性质: 由于我们定义的三个操作来作为编辑距离的一种衡量方法。

于是对 ch1,ch2 可能的操作只有

1. 把 ch1 变成 ch2
2. s1+ch1 后删除 ch1 d = (1+d(s1,s2+ch2))
3. s1+ch1 后插入 ch2 d = (1 + d(s1+ch1,s2))

对于 2 和 3 的操作可以等价于:

_2. s2+ch2 后添加 ch1
d=(1+d(s1,s2+ch2))

_3. s2+ch2 后删除 ch2
d=(1+d(s1+ch1,s2))

因此可以得到计算编辑距离的性质 2。

复杂度分析

从上面性质 2 可以看出计算过程呈现这样的一种结构(假设各个层用当前计算的串长度标记，并假设两个串长度都为 n)

可以看到，该问题的复杂度为指数级别 3 的 n 次方，对于较长的串，时间上是无法让人忍受的。

分析:　　在上面的结构中，我们发现多次出现了 (n-1,n-1), (n-1,n-2)……。换句话说该结构具有重叠子问题。再加上前面性质 2 所具有的最优子结构。符合动态规划算法基本要素。因此可以使用动态规划算法把复杂度降低到多项式级别。

动态规划求解

首先为了避免重复计算子问题，添加两个辅助数组。

一.　保存子问题结果。

M[|s1| ,|s2|] , 其中 M[i , j] 表示子串 s1(0->i) 与 s2(0->j) 的编辑距离

二.　保存字符之间的编辑距离.

E[|s1|, |s2|] , 其中 E[i, j] = s[i] = s[j] ? 0 : 1

三.　新的计算表达式

根据性质 1 得到

M[0,0] = 0;

M[s1i, 0] = |s1i|;

M[0, s2j] = |s2j|;

根据性质 2 得到

```
M[ i, j ]    = min(      m[i-1,j-1] + E[ i, j ] ,
                         m[i, j-1] ,
                         m[i-1, j]   );
```

复杂度

从新的计算式看出，计算过程为

```
i=1 -> |s1|
    j=1 -> |s2|
        M[i][j] = ……
```

因此复杂度为 O(|s1| * |s2|) ，如果假设他们的长度都为 n，则复杂度为 O(n^2)