中国科学院大学

University of Chinese Academy of Sciences

计算机算法设计与分析

083500M01001H

Chap 1-2 课程作业

2022 年 9 月 11 号

Professor: 刘玉贵

学生: 周胤昌
学号: 202228018670052
学院: 网络空间安全学院
所属专业: 网络空间安全
方向: 安全协议理论与技术

Problem 1

试确定下述程序的关键操作数, 该函数实现一个 $m \times n$ 矩阵与一个 $n \times p$ 矩阵之间的乘法:

```
1  template <class T>
2  void Mult(T **a, T **b, int m, int n, int p) {
3      //m×n 矩阵 A 与 n×p 矩阵 B 相成得到 m×p 矩阵 C
4      for(int i = 0; i < m; i++) {
5          for(int j = 0; j < p ;j++) {
6              T sum = 0;
7              Tfor(int k = 0; k < n; k++)
8              Tsum += a[i][k]*b[k][j];
9              C[i][j] = sum;
10         }
11     }
12 }
```

Solution:

此算法的关键操作为**第 8 行**, 该语句的操作数为 2 (1 次加法和 1 次乘法). 则三层循环的总关键操作数为:

$$T = \sum_{i=0}^{m-1} \sum_{j=0}^{p-1} \sum_{k=0}^{n-1} 2 = 2mnp$$

故该算法的时间复杂度为

$$T(m,n,p) = O(2mnp) = O(mnp)$$

Problem 2

函数 MinMax 用来查找数组 $a[0:n-1]$ 中的最大元素和最小元素, 以下给出两个程序. 令 n 为实例特征. 试问: 在各个程序中, a 中元素之间的比较次数在最坏情况下各是多少?

```cpp
/* 找最大最小元素 (方法一)*/
template <class T>
bool MinMax(T a[], int n, int& Min, int& Max) {
    //寻找 a[0:n-1] 中的最小元素与最大元素
    //如果数组中的元素数目小于 1, 则返回 false
    if(n<1) return false;
    Min=Max=0; //初始化
    for(int i=1; i<n; i++) {
        if(a[Min]>a[i]) Min=i;
        if(a[Max]<a[i]) Max=i;
    }
    return true;
}
```

```cpp
/* 找最大最小元素 (方法二)*/
template <class T>
bool MinMax(T a[], int n, int& Min, int& Max) {
    //寻找 a[0:n-1] 中的最小元素与最大元素
    //如果数组中的元素数目小于 1, 则返回 false
    if(n<1) return false;
    Min=Max=0; //初始化
    for(int i=1; i<n; i++) {
        if(a[Min]>a[i]) Min=i;
        else if(a[Max]<a[i]) Max=i;
    }
    return true;
}
```

Solution:

不论数组 a 是单调递增还是单调递减, for 循环内部的两次判断都得执行, 所以方法 1 在任何情况下的元素比较次数都为 $2 \times (n-1)$; 而对于方法 2 来说, 当数组 a 单调递减 (最好情况) 时, for 循环内部的第一个判断条件一定满足, 那么 else if 这个判断自然就不用执行了, 即此情形下的元素比较次数为 $1 \times (n-1)$. 但当数组 a 单调递增 (最坏情况) 时, 第一个循环条件在各轮循环中都不能满足, 所以紧跟着需要执行后边的 else if 判断, 即此情形下的元素比较次数为 $2 \times (n-1)$.

Problem 3

证明以下关系式不成立: (1). $10n^2 + 9 = O(n)$; (2). $n^2 \log n = \Theta\left(n^2\right)$.

Solution:

证明. **(1).** 考虑极限 $\lim\limits_{n\to\infty} \dfrac{10n^2+9}{n} = +\infty > c \ \ (\forall c > 0)$, 故根据大 O 比率定理可知该式不成立;

(2). 考虑极限 $\lim\limits_{n\to\infty} \dfrac{n^2 \log n}{n^2} = +\infty > c \ \ (\forall c > 0)$, 故根据 Θ 比率定理可知该式不成立. □

Problem 4

按照渐近阶从低到高的顺序排列以下表达式:

$$4n^2, \log n, 3^n, 20n, n^{2/3}, n!$$

Solution:

根据 Stirling 公式:

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \ (\text{as } n \to +\infty)$$

可知有渐进阶的顺序为

$$\log n \ll n^{2/3} \ll 20n = \Theta(n) \ll 4n^2 = \Theta\left(n^2\right) \ll 3^n \ll n!$$

Problem 5

(1). 假设某算法在输入规模是 n 时为 $T(n) = 3 * 2^n$. 在某台计算机上实现并完成该算法的时间是 t 秒. 现有另一台计算机, 其运行速度为第一台的 64 倍. 那么, 在这台计算机上用同一算法在 t 秒内能解决规模为多大的问题?

(2). 若上述算法改进后的新算法的时间复杂度为 $T(n) = n^2$, 则在新机器上用 t 秒时间能解决输入规模为多大的问题?

(3). 若进一步改进算法, 最新的算法的时间复杂度为 $T(n) = 8$, 其余条件不变, 在新机器上运行, 在 t 秒内能够解决输入规模为多大的问题?

Solution:

(1). 设问题规模为 M, 则新机器上的求解时间为 $t = 3 \times 2^M/64$, 老机器的求解时间 $t = 3 \times 2^n$, 即解得 $M = n + 6$;

(2). 同理设问题规模为 M, 则新机器上的求解时间为 $t = M^2/64 = n^2$, 老机器的求解时间 $t = n^2$, 即解得 $M = 8n$;

(3). 因为该算法的时间复杂度是常数阶的, 也就意味着问题规模不影响求解时间 (当问题规模很大时), 所以在任何 (可以运行该算法的) 机器上, t 秒内可以解决任意规模的问题.

Problem 6

Fibonacci 数有递推关系:

$$F(n)=\begin{cases}1, & n=0\\ 1, & n=1\\ F(n-1)+F(n-2), & n>1\end{cases}$$

试求出 $F(n)$ 的表达式.

Solution:

解法 1 (数学解法): 根据递推关系可知其对应的特征方程为

$$r^2=r+1\Rightarrow r_1=\frac{1+\sqrt{5}}{2}, r_2=\frac{1-\sqrt{5}}{2}$$

根据二阶常系数齐次差分方程解的结构定理可知解形如 $F\left(n\right)=C_1\left(\frac{1+\sqrt{5}}{2}\right)^n+C_2\left(\frac{1-\sqrt{5}}{2}\right)^n$.

又由于 $F(0)=F(1)=1$, 所以 $C_1=\frac{1}{\sqrt{5}}, C_2=-\frac{1}{\sqrt{5}}$, 即方程解为

$$F\left(n\right)=\frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n-\left(\frac{1-\sqrt{5}}{2}\right)^n\right]$$

解法 2 (矩阵的快速幂解法): 考虑如下

$$\begin{pmatrix}F_{n+1}\\F_n\end{pmatrix}=\begin{pmatrix}F_n+F_{n-1}\\F_n\end{pmatrix}=\begin{pmatrix}1&1\\1&0\end{pmatrix}^1\begin{pmatrix}F_n\\F_{n-1}\end{pmatrix}=\cdots=\begin{pmatrix}1&1\\1&0\end{pmatrix}^n\begin{pmatrix}F_1\\F_0\end{pmatrix}$$

矩阵 $\begin{pmatrix}1&1\\1&0\end{pmatrix}$ 的特征值为 $\lambda_1=\frac{1-\sqrt{5}}{2}, \lambda_2=\frac{1+\sqrt{5}}{2}$. 对角矩阵 Λ 和相似变换矩阵 P 分别为

$$\Lambda=\begin{pmatrix}\frac{1-\sqrt{5}}{2}&0\\0&\frac{1+\sqrt{5}}{2}\end{pmatrix},\quad P=\begin{pmatrix}\frac{1-\sqrt{5}}{2}&\frac{1+\sqrt{5}}{2}\\1&1\end{pmatrix}$$

所以

$$\begin{pmatrix}F_{n+1}\\F_n\end{pmatrix}=\begin{pmatrix}1&1\\1&0\end{pmatrix}^n\begin{pmatrix}F_1\\F_0\end{pmatrix}=P\Lambda^nP^{-1}\begin{pmatrix}1\\1\end{pmatrix}=\begin{pmatrix}\frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{n+1}-\left(\frac{1-\sqrt{5}}{2}\right)^{n+1}\right]\\\frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{n}-\left(\frac{1-\sqrt{5}}{2}\right)^{n}\right]\end{pmatrix}$$

Problem 7

下面的无向图以邻接链表存储, 而且在关于每个顶点的链表中与该顶点相邻的顶点是按照字母顺序排列的. 试以此图为例描述讲义中算法 DFNL 的执行过程.

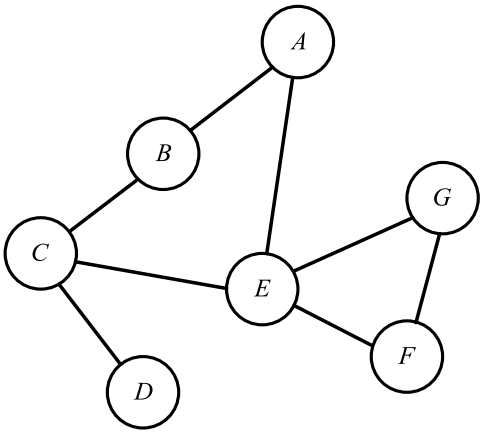

图 1: 一个无向图 G

Solution:

先利用深度优先搜索 (**DFS**) 获取深度优先搜索树 T(深索数 DFN 也顺便求出来了), 再利用后根遍历来搜索 T, 最后**从下到上、从右往左**依次计算出各顶点的最低深索数. 此过程可以如下模拟:

Step 1. 出入栈的模拟:

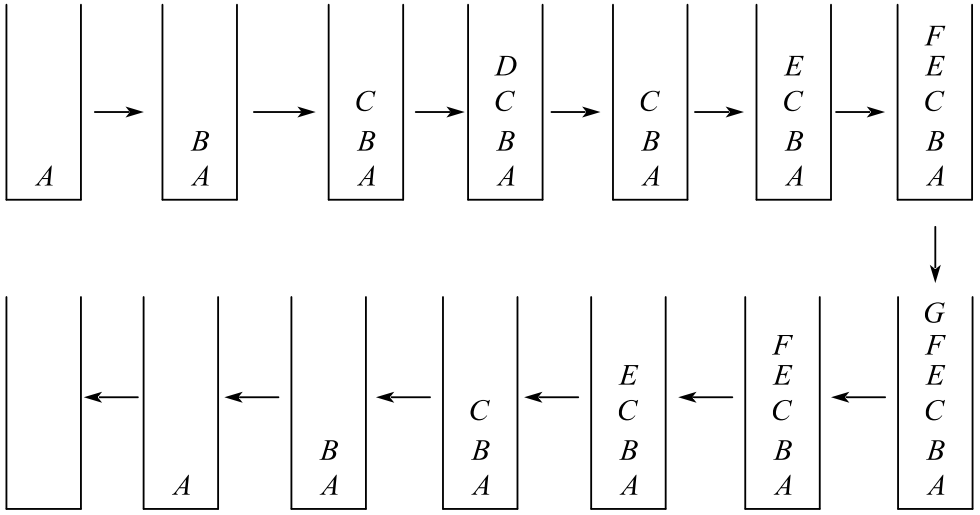

图 2: 出入栈模拟

Step 2. 求出深度搜索树 T 和余边, 并利用后根遍历来搜索 T, 最后从下到上、从右往左依次计算出各顶点的最低深索数: 该步骤的模拟过程见如下 (右图为深度优先搜索树 T, 左图的红边为余边 (对应到搜索树 T 的红色虚线), 红色圈数字为最低深索数).

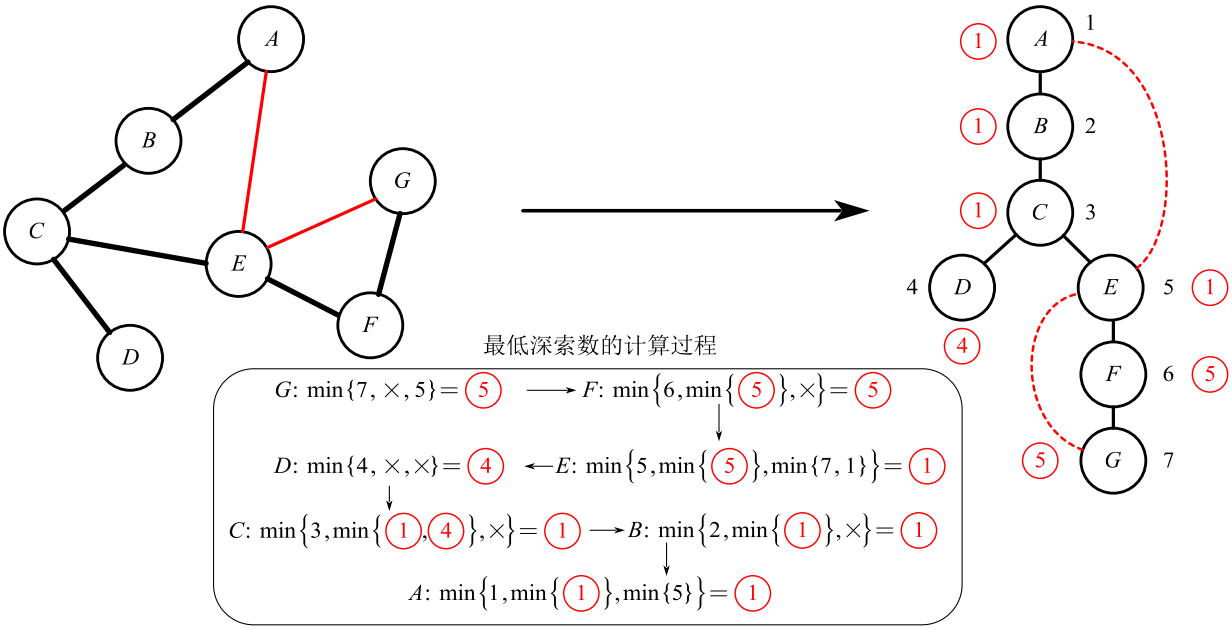

图 3: 第二步骤的模拟

Step 3. 求出割点以及 2-连通分支: 右图搜索树 T 的根节点 A 由于没有至少两个子节点, 故不可能是割点; 而 D, G 作为叶子节点更不可能是割点; 通过非根节点的割点判别充要条件[1]可知: E, C 均为割点, 于是可以简单得出如下 2-连通分支 (下图中红色的即为 2-连通分支, 图中分叉只是为了体现出拆解过程):

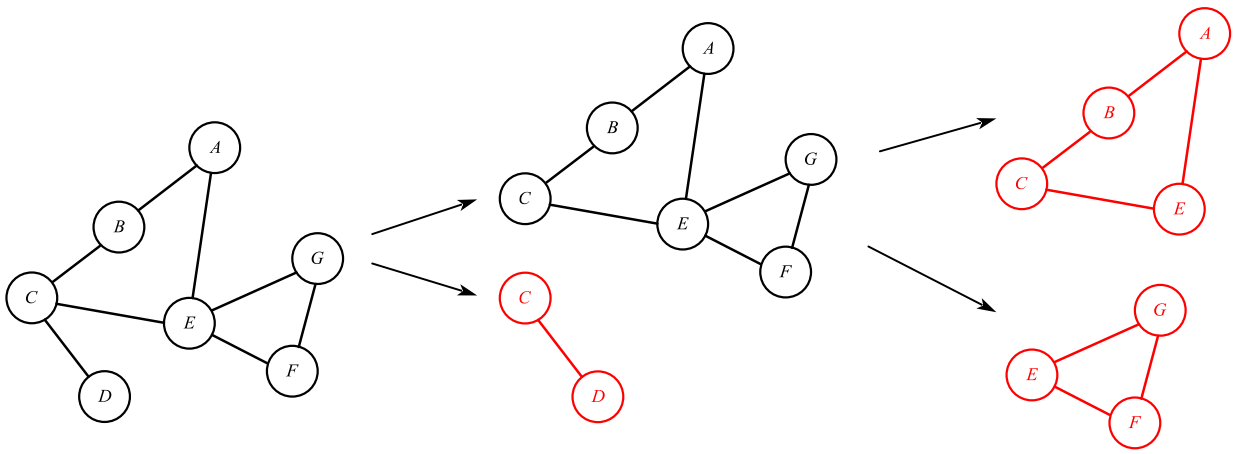

图 4: 第三步骤的模拟

至此, 此例的 DFNL 算法模拟执行完毕.

[1]对于非根节点 u 是割点 $\Leftrightarrow$ 节点 u 存在儿子 w, 使得 $L(w) \geq \mathrm{DFN}(u)$.

Problem 8

考虑下述选择排序算法1所示:

Algorithm 1 选择排序

Input: n 个不等整数的数组 $A[1..n]$
Output: 按递增次序排序的 A

```
1: for i := 1 to n do
2:     for j := i + 1 to n do
3:         if A[j] < A[i] then
4:             A[i] ↔ A[j]
5:         end if
6:     end for
7: end for
8: 输出排序后的数组 A
```

问: (1) 最坏情况下做多少次比较运算?
(2) 最坏情况下做多少次交换运算? 在什么输入时发生?

Solution:

(1). 任意情况下要比较 $(n-1)+(n-2)+\cdots+1=\frac{1}{2}n(n-1)$ 次. 再者, 此处不存在所谓的最好或最坏情况, 不论数组 A 是逆序还是升序排列, 计算机不可能因为提前得知 A 的所有情况而不执行判断语句, 即每次循环都需要进行比较运算;

(1). 最坏情况: 输入数组内元素为降序排列. 此时做 $(n-1)+(n-2)+\cdots+1=\frac{1}{2}n(n-1)$ 次交换运算. 当数组 A 内元素为升序排列时, 则交换次数为 0 (即为最好情况).

Problem 9

考虑下面的每对函数 $f(n)$ 和 $g(n)$, 比较他们的阶.
(1). $f(n)=\frac{1}{2}\left(n^2-n\right),\ g(n)=6n$; (2). $f(n)=n+2\sqrt{n},\ g(n)=n^2$;
(3). $f(n)=n+n\log n,\ g(n)=n\sqrt{n}$; (4). $f(n)=\log(n!),\ g(n)=n^{1.05}$;

Solution:

(1). $f(n)=\Theta\left(n^2\right)\gg\Theta(n)=g(n)$; **(2).** $f(n)=\Theta(n)\ll\Theta\left(n^2\right)=g(n)$;
(3). $f(n)=\Theta(n\log n)\ll\Theta\left(n^{1.5}\right)=g(n)$; **(4).** 根据**斯特林公式**可知:

$$\log(n!)\sim\log\left(\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\right)=\frac{1}{2}\log(2\pi n)+n(\log n-\log e)\sim n\log n\ (\text{as}\, n\to\infty)$$

所以有 $f(n)=\Theta(n\log n)\ll\Theta\left(n^{1.05}\right)=g(n)$.

Problem 10

在表1中填入 true 或 false.

	$f(n)$	$g(n)$	$f(n) = O(g(n))$	$f(n) = \Omega(g(n))$	$f(n) = \Theta(g(n))$
1	$2n^3 + 3n$	$100n^2 + 2n + 100$			
2	$50n + \log n$	$10n + \log\log n$			
3	$50n \log n$	$10n \log\log n$			
4	$\log n$	$\log^2 n$			
5	$n!$	5^n			

表 1: 原始表格

Solution:

解答如下表2所示:

	$f(n)$	$g(n)$	$f(n) = O(g(n))$	$f(n) = \Omega(g(n))$	$f(n) = \Theta(g(n))$
1	$2n^3 + 3n$	$100n^2 + 2n + 100$	False	True	False
2	$50n + \log n$	$10n + \log\log n$	True	True	True
3	$50n \log n$	$10n \log\log n$	False	True	False
4	$\log n$	$\log^2 n$	True	False	False
5	$n!$	5^n	False	True	False

表 2: 解答

Problem 11

用迭代法求解下列递推方程:

(1). $\begin{cases} T(n) = T(n-1) + n - 1 \\ T(1) = 0 \end{cases}$; (2). $\begin{cases} T(n) = 2T\left(\frac{n}{2}\right) + n - 1 \\ T(1) = 0 \end{cases}, n = 2^k$

Solution:

(1). 易知

$$\begin{cases} T(2) - T(1) = 1 \\ T(3) - T(2) = 2 \\ \vdots \\ T(n) - T(n-1) = n - 1 \end{cases} \xrightarrow{\text{各式相加}} T(n) = 1 + 2 + \cdots + n - 1 = \frac{1}{2} n(n-1) = \Theta\left(n^2\right)$$

(2). 令 $n = 2^k$, 则易知有如下:

$$\begin{aligned} T\left(2^k\right) &= 2T\left(2^{k-1}\right) + 1 \cdot 2^k - 1 \\ &= 2^2 T\left(2^{k-2}\right) + 2 \cdot 2^k - (1+2) \\ &= 2^3 T\left(2^{k-3}\right) + 3 \cdot 2^k - \left(1 + 2 + 2^2\right) \\ &\vdots \\ &= 2^k T(1) + k \cdot 2^k - \left(1 + 2 + \cdots + 2^{k-1}\right) \\ &= k \cdot 2^k + \left(1 - 2^k\right) = (k-1) \cdot 2^k + 1 \\ &= n \log n - n + 1 = T(n) = \Theta(n \log n) \end{aligned}$$

至此, Chap 1-2 的作业解答完毕.