算法设计与分析第一次作业

曲宇勋 201928014628016 人工智能学院

1. 试确定下述程序的执行步数，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  for(int i=0; i<m; i++)
4          for(int j=0; j<p; j++){
5                  T sum=0;
6                  for(int k=0; k<n; k++)
7                          Sum+=a[i][k]*b[k][j];
8                  C[i][j]=sum;
9          }
10 }
```

Solution:列出以上代码的执行步数统计表

语句行号	s/e	频率	总步数
1	0	0	0
2	0	0	0
3	1	m+1	m+1
4	1	m(p+1)	m(p+1)
5	1	mp	mp
6	1	mp(k+1)	mp(k+1)
7	1	mpk	mpk
8	1	mp	mp
9	0	0	0
10	0	0	0

表 1: 程序执行步数统计表

代码的执行总步数为

$$(m+1)+m(p+1)+mp+mp(k+1)+mpk+mp = 2mpk+4mp+2m+1$$

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？

```
template<class T>
bool MinMax(T a[], int n, int& Min, int& Max)
{
if(n<1) return false;
Min=Max=0;
for(int i=1; i<n; i++){
        if(a[Min]>a[i]) Min=i;
        if(a[Max]<a[i]) Max=i;
}
return true;
}
```

```
template<class T>
bool MinMax(T a[], int n, int& Min, int& Max)
{
if(n<1) return false;
Min=Max=0;
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:

对于**算法一**，算法所耗时间与数组a的元素无关，每次循环内都包含两次比较。

所以算法一的比较次数为$2(n-1)$

对于**算法二**，在最坏情况下，数组呈升序排列，每次循环$a[Min] > a[i]$条件均不满足，即每次循环都执行else语句

所以算法二的最差比较次数为$2(n-1)$

5.下面那些规则是正确的？为什么？

1).$\{f(n) = O(F(n)), g(n) = O(G(n))\} \rightarrow f(n)/g(n) = O(F(n)/G(n))$

2).$\{f(n) = O(F(n)), g(n) = O(G(n))\} \rightarrow f(n)/g(n) = \Omega(F(n)/G(n))$

3).$\{f(n) = O(F(n)), g(n) = O(G(n))\} \rightarrow f(n)/g(n) = \Theta(F(n)/G(n))$

4).$\{f(n) = \Omega(F(n)), g(n) = \Omega(G(n))\} \rightarrow f(n)/g(n) = \Omega(F(n)/G(n))$

5).$\{f(n) = \Omega(F(n)), g(n) = \Omega(G(n))\} \rightarrow f(n)/g(n) = O(F(n)/G(n))$

6).$\{f(n) = \Theta(F(n)), g(n) = \Theta(G(n))\} \rightarrow f(n)/g(n) = \Theta(F(n)/G(n))$

Solution:

(1)(2)(3)(4)(5)是错误的，(6)是正确的，以下举其对应的反例

1). $f(n) = n^4, g(n) = n^3, F(n) = n^5, G(n) = n^5$,

$f(n)/g(n) = n, F(n)/G(n) = 1, f(n)/g(n) = \Omega(F(n)/G(n))$

2). $f(n) = n^4, g(n) = n^3, F(n) = n^7, G(n) = n^5$,

$f(n)/g(n) = n, F(n)/G(n) = n^2, f(n)/g(n) = O(F(n)/G(n))$

3). 反例同(1)(2)

4). $f(n) = n^4, g(n) = n^3, F(n) = n^3, G(n) = n$,

$f(n)/g(n) = n, F(n)/G(n) = n^2, f(n)/g(n) = O(F(n)/G(n))$

5). $f(n) = n^4, g(n) = n^3, F(n) = n^2, G(n) = n^2$,

$f(n)/g(n) = n, F(n)/G(n) = 1, f(n)/g(n) = \Theta(F(n)/G(n))$

6).由于$f(n) = \Theta(F(n))$，所以

$$\lim_{n\to\infty}\left(\frac{f(n)}{F(n)}\right) = c_1, (c_1 \neq 0)$$

同理

$$\lim_{n\to\infty}\left(\frac{g(n)}{G(n)}\right) = c_2, (c_2 \neq 0)$$

所以

$$\lim_{n\to\infty}\left(\frac{\frac{f(n)}{g(n)}}{\frac{F(n)}{G(n)}}\right) = \lim_{n\to\infty}\left(\frac{\frac{f(n)}{F(n)}}{\frac{g(n)}{G(n)}}\right) = \frac{\lim\limits_{n\to\infty}\frac{f(n)}{F(n)}}{\lim\limits_{n\to\infty}\frac{g(n)}{G(n)}} = \frac{c_1}{c_2}, (\frac{c_1}{c_2} \neq 0)$$

同理

$$\lim_{n\to\infty}\left(\frac{\frac{F(n)}{G(n)}}{\frac{f(n)}{g(n)}}\right) = \frac{c_2}{c_1}, (\frac{c_2}{c_1} \neq 0)$$

所以$f(n)/g(n) = \Theta(F(n)/G(n))$

6. 按照渐近阶从低到高的顺序排列以下表达式:

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

Solution:

从低到高的顺序为

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

7. 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)假设第二台计算机上用t秒完成的问题规模为n'，完成的运算量为$T(n') = 3 * 2^{n'}$.当运算速度提升为64倍时，可完成的运算量满足

$$\begin{aligned} T(n') =& 64T(n) \\ 3 * 2^{n'} =& 64 * 3 * 2^n \\ n' =& n + 6 \end{aligned}$$

可完成规模为n+6的问题

(2)假设第二台计算机上用t秒完成的问题规模为n'，完成的运算量为$T(n') = n'^2$.当运算速度提升为64倍时，可完成的运算量满足

$$\begin{aligned} T(n') =& 64T(n) \\ n'^2 =& 64 * 3 * 2^n \\ n' =& 8\sqrt{3 * 2^n} \end{aligned}$$

可完成规模为$8\sqrt{3 * 2^n}$的问题

(3)当问题规模的时间复杂度为T(n)=8，是一个常数时，问题所需时间消耗和问题规模无关，所以可完成规模为无穷大的问题

8. 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:

当$n > 1$时，

$$\begin{aligned} F(n) =& F(n-1) + F(n-2) \\ F(n-1) =& F(n-1) \end{aligned}$$

用矩阵乘法可以表示为

$$\begin{bmatrix} F(n) \\ F(n-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F(n-1) \\ F(n-2) \end{bmatrix}$$

迭代$n-1$次可得

$$\begin{bmatrix} F(n) \\ F(n-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} \begin{bmatrix} F(1) \\ F(0) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

令$A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$,求解A的特征值与特征向量,解$(A - \lambda I)x = 0$可得

$$\lambda_1 = \frac{1+\sqrt{5}}{2}, x_1 = \frac{1}{\sqrt{\lambda_1^2 + 1}} \begin{bmatrix} \lambda_1 \\ 1 \end{bmatrix}$$

$$\lambda_2 = \frac{1-\sqrt{5}}{2}, x_2 = \frac{1}{\sqrt{\lambda_2^2 + 1}} \begin{bmatrix} \lambda_2 \\ 1 \end{bmatrix}$$

所以对A相似对角化可得$A = S\Lambda S^{-1}$,其中S为正交矩阵，即$SS^T = I$

$$S = [x_1, x_2], \Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

所以

$$
\begin{aligned}
\begin{bmatrix} F(n) \\ F(n-1) \end{bmatrix} &= A^{n-1} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
&= (S\Lambda S^{-1})^{n-1} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
&= S\Lambda^{n-1} S^{-1} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{\sqrt{\lambda_1^2+1}}\lambda_1 & \frac{1}{\sqrt{\lambda_2^2+1}}\lambda_2 \\ \frac{1}{\sqrt{\lambda_1^2+1}} & \frac{1}{\sqrt{\lambda_2^2+1}} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}^{n-1} \begin{bmatrix} \frac{1}{\sqrt{\lambda_1^2+1}}\lambda_1 & \frac{1}{\sqrt{\lambda_2^2+1}}\lambda_2 \\ \frac{1}{\sqrt{\lambda_1^2+1}} & \frac{1}{\sqrt{\lambda_2^2+1}} \end{bmatrix}^T \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{\sqrt{\lambda_1^2+1}}\lambda_1 & \frac{1}{\sqrt{\lambda_2^2+1}}\lambda_2 \\ \frac{1}{\sqrt{\lambda_1^2+1}} & \frac{1}{\sqrt{\lambda_2^2+1}} \end{bmatrix} \begin{bmatrix} \lambda_1^{n-1} & 0 \\ 0 & \lambda_2^{n-1} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{\lambda_1^2+1}}(\lambda_1+1) \\ \frac{1}{\sqrt{\lambda_2^2+1}}(\lambda_2+1) \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{\sqrt{\lambda_1^2+1}}\lambda_1 & \frac{1}{\sqrt{\lambda_2^2+1}}\lambda_2 \\ \frac{1}{\sqrt{\lambda_1^2+1}} & \frac{1}{\sqrt{\lambda_2^2+1}} \end{bmatrix} \begin{bmatrix} \frac{\lambda_1^{n-1}}{\sqrt{\lambda_1^2+1}}(\lambda_1+1) \\ \frac{\lambda_2^{n-1}}{\sqrt{\lambda_2^2+1}}(\lambda_2+1) \end{bmatrix} \\
&= \begin{bmatrix} \frac{\lambda_1^n}{\lambda_1^2+1}(\lambda_1+1) + \frac{\lambda_2^n}{\lambda_2^2+1}(\lambda_2+1) \\ \frac{\lambda_1^{n-1}}{\lambda_1^2+1}(\lambda_1+1) + \frac{\lambda_2^{n-1}}{\lambda_2^2+1}(\lambda_2+1) \end{bmatrix}
\end{aligned}
$$

所以

$$F(n) = \frac{\lambda_1^n}{\lambda_1^2+1}(\lambda_1+1) + \frac{\lambda_2^n}{\lambda_2^2+1}(\lambda_2+1)$$

$n = 0$与$n = 1$情况经过验证亦符合上述等式,其中$\lambda_1 = \dfrac{1+\sqrt{5}}{2}, \lambda_2 = \dfrac{1-\sqrt{5}}{2}$

经过化简之后可得

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