1. 习题一.

1. 确定程序的关键操作数.

矩阵乘法运算

```
template<class T>
void Mult(T **a, T **b, int m, int n, int p)
{//m×n 矩阵 a 与 n×p 矩阵 b 相成得到 m×p 矩阵 c
  for(int i=0; i<m; i++)
    for(int j=0; j<p; j++){
      T sum=0;
      for(int k=0; k<n; k++)
        Sum+=a[i][k]*b[k][j];
      C[i][j]=sum;
    }
}
```

外层循环 m 次
中层循环 p 次
内层循环 n 次
关键操作：1次乘，1次加.
故关键操作 $2\times n\times p\times m$ 次.

2.

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？

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，则还回 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;
}
```

方法一：n-1 次循环，每次循环比较 2 次，为 2n-2 次

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，则还回 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;
}
```

方法二：n-1 次循环，循环内至少做 1 次比较。如果整个数组是不降的，那么还需比较一次.
最坏 2n-2 次.

3.

3. 证明以下关系式不成立：

1). $10n^2+9=O(n)$;

2). $n^2\log n=\Theta(n^2)$;

1) 对于 $\forall c$，只要 $n>c$，都有 $10n^2+9>10c\cdot n>c\cdot n$
故不存在 c，使 $10n^2+9\le cn$.

2) 对于 $\forall c$，只要 $\log n>c$，即 $n>2^c$，都有 $n^2\log n>c\cdot n^2$
故不存在 c，使 $n^2\cdot\log n\le c\cdot n^2$

6. 6. 按照渐近阶从低到高的顺序排列以下表达式：

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

$4n^2 = O(n^2)$ $3^n = O(2^n)$ $20n = O(n)$

故顺序为 $\log n$ , $n^{\frac{2}{3}}$ , $20n$ , $40n^2$ , $3^n$ , $n!$

7. 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？

1)第一台机 $3\times 2^n = c\cdot t$

第二台机 $3\times 2^{n'} = c\cdot t \times 64$. 得 $n' = n+6$

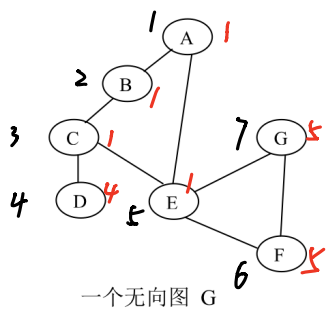
2) $T(n'') = (n'')^2 = T(n') = 3\times 2^n \times 2^6$

得 $n'' = 8\cdot\sqrt{3\times 2^n}$

3) 常数时间复杂度，t秒内能解决任意规模的问题.

2. 习题二.

5. 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。




一个无向图 G

图中黑色为DFN

红色为L.

① 访问A. DFN(A)=1, L(A)=1

② 访问B, DFN(B)=2, L(B)=2

③ 访问C, DFN(C)=3, L(C)=3

④ 访问D, DFN(D)=4, L(D)=4

⑤ 访问E, DFN(E)=5, L(E)=1

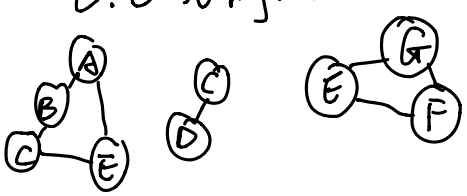
⑥ 访问F, DFN(F)=6, L(F)=6

⑦ 访问G, DFN(G)=7, L(F)=5

⑧ L(G)=5, L(F)=5

⑨ L(C)=1, L(B)=1.

C.E为割点.



3.

3.考虑下述选择排序算法：

输入：n 个不等的整数的数组 A[1..n]

输出：按递增次序排序的 A

```
For i:=1 to n-1
  For j:=i+1 to n
    If A[j]<A[i] then A[i]↔A[j]
```

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?

(1)冒泡排序，做 $n-1+n-2+\cdots+1=\frac{n(n-1)}{2}$

(2)最坏情况下，数组是递减的

需要做 $\frac{n(n-1)}{2}$ 次

4.

4.考虑下面的没对函数 f(n)和 g(n) ,比较他们的阶。

(1) $f(n)=(n^2-n)/2$, $g(n)=6n$ (2)$f(n)=n+2\sqrt{n}$, $g(n)=n^2$

(3)$f(n)=n+n\log n$, $g(n)=n\sqrt{n}$ (4)$f(n)=\log(n!)$, $g(n)=n^{1.05}$

(1) $\frac{f(n)}{g(n)}=\frac{n-1}{2\times 6}$ 故 $\frac{f(n)}{g(n)}=O(n)$ $g(n)=o(f(n))$

(2) $f(n)=O(n)$ $g(n)=O(n^2)$ $f(n)=o(g(n))$

(3) $f(n)=O(n\log n)$ $g(n)=O(n^{1.5})$ $f(n)=o(g(n))$

(4) $\lim_{n\to\infty}\frac{\log n!}{\log n^n}=\lim_{n\to\infty}\frac{\log(\sqrt{2\pi n}(\frac{n}{e})^n)}{\log n^n}=\lim_{n\to\infty}\frac{\frac{1}{2}\log(2\pi)+\frac{1}{2}\log n+n\log n-n}{n\log n}=1$

故 $\log(n!)$ 与 $n\log n$ 同阶.

而 $n\log n=o(n^{1.05})$ 故 $f(n)=o(g(n))$

5.

5.在表中填入 true 或 false .

| | f(n) | g(n) | f(n)=O(g(n)) | f(n)=Ω(g(n)) | f(n)=Θ(g(n)) |
|---|---|---|---|---|---|
| 1 | $2n^3+3n$ | $100n^2+2n+100$ | false | True | false |
| 2 | 50n+logn | 10n+loglogn | True | True | True |
| 3 | 50nlogn | 10nloglogn | false | True | false |
| 4 | logn | $Log^2n$ | True | false | false |
| 5 | n! | $5^n$ | false | True | false |

6.

6.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) $\begin{cases} T(n) = 2T(n/2) + n - 1 \\ T(1) = 0 \end{cases}$ ,$n=2^k$

(1) $T(n) = T(n-1) + n - 1 = T(n-2) + (n-1) + (n-2) = \cdots = T(1) + \frac{n(n-1)}{2}$

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

(2) 令 $2^k = n$.

$T(n) = 2 \cdot T(2^{k-1}) + 2^k - 1 + 2^2 \cdot T(2^{k-2}) + 2^k - 2 + 2^k - 1 = \cdots$

$= 2^k \cdot T(1) + k \cdot 2^k - (1 + 2 + 2^2 \cdots + 2^k)$

$= kn - 2^k + 1 = n\log n - n + 1 = O(n\log n)$

当 $2^{k-1} < n < 2^k$ 时，取 $2^{k-1} = c$，有 $T(c) < T(n) < T(2c)$，

即 $a_1 \cdot c\log c < T(n) < a_2 \cdot (c+1)\cdot \log(c+1)$，于 $T(n) = O(c\log c)$.

于是 $T(n) = O((n-b)\cdot \log(n-b)) = O(n\log n)$