2.0 Mathematics Reviews

1) Exponents

$X^A X^B = X^{A+B}$

$\frac{X^A}{X^B} = X^{A-B}$

$(X^A)^B = X^{AB}$

$X^N + X^N = 2X^N \neq X^{2N}$

$X^N + X^N = 2X^{N+1}$

2) Logarithms

$X^A = B$ -> $\log_x B = A$

$\log_A B = \frac{\log_c A}{\log_c B}$

3) Series

1 + 2 + 3 + … + N

$$\sum_{i=1}^{N} i = \frac{N(N+1)}{2}$$

$1^2 + 2^2 + 3^2 + \ldots + N^2$

$$\sum_{i=1}^{N} i^2 = \frac{N(N+1)(2N+1)}{6}$$

```
for (i= 1; i <= N, i++)
    sum = sum + i;
```

คำถาม

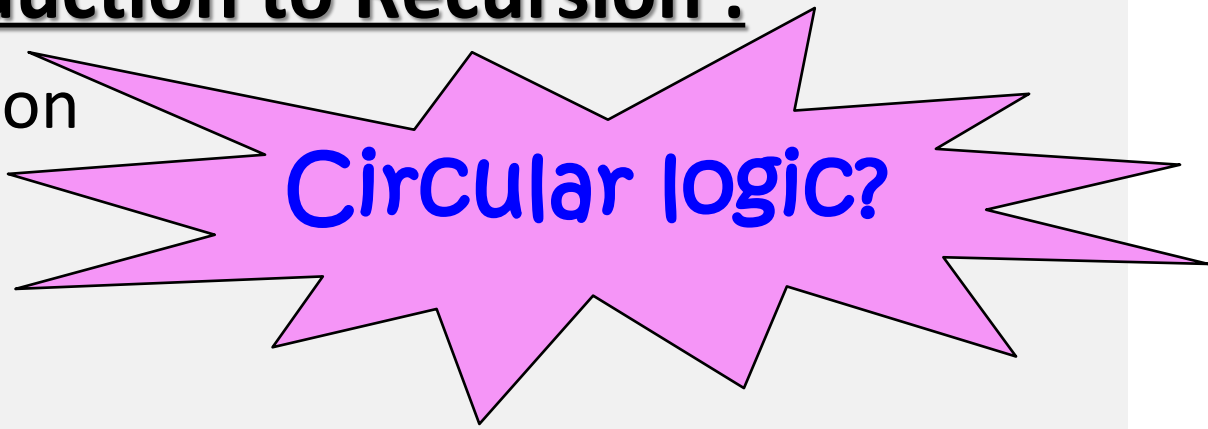
1 + 2 + 3 + … + (N-1) = ?

$$\frac{N(N+1)}{2} = \frac{(N-1)(N-1+1)}{2} = \frac{N(N-1)}{2}$$ #

2.1 A Brief Introduction to Recursion :

Mathematical function

1. C=2(F-32)/9



2. y = sin(x) * pi

3. $f(x) = 2f(x-1)+x^2$

   f(0)=0 , x nonnegative integer

4. Factorial กำหนด x เป็นจำนวนเต็มที่ไม่เป็นลบ

$f(x) = x * f(x-1)$, $f(1) = 1$ และ $f(0)=1$

5. Fibonacci number

$f(n) = f(n-1) + f(n-2)$

$f(0)=0$, $f(1) = 1$

Example 1

```
#include <stdio.h>
int fact(int x)
{   if(x <= 1)
        return 1;
    else
        return x* fact(x-1);
}
```

```
int main()
{    int ans;
     ans = fact(3);
     cout << ans;
}
```

Example 2 Factorial

```
int main()
{   int ans;
    ans = 6      fact(3);
    cout << ans;
}
```

```
int fact(int x)   //2
{  if(x <= 1)
       return 1;
   else
       return x* 1
}
```

```
int fact(int x)   //3
{  if(x <=1)
       return 1;
   else
       return x* 2
}
```

fact(x-1);

```
int fact(int x)   //1
{  if(x <= 1)
       return 1;
   else
       return x* fact(x-1);
}
```

Example 3

```
0 int bad(int n)
1 {   if (n==0)
2         return 0;
3    else
4         return bad(n/3 + 1+ n – 1);
5 }
```

1. ไม่มี base case
2. มี base case แต่ไม่ลง

Example 4

```
0 void printout(int n)
1 {   if( n >=10 )
2         printout(n/10);
3      cout << n%10;
4 }
```

123 → T → printout(12) → ค้าง 2

12 → T → printout(1) → ค้าง 2

1 → F → พิมพ์ 1 เสร็จ

return ไป บรรทัด 2.

Example 5

n 123

```
0  void printout(int n)
1  {   if( n >=10 )
2      {  printout(n/10);
3           cout << n%10;
4      }
5  }
```

n หาร เลข

2/10

456 | 45 | 4

T | T | F

PO(45) | PO(4) | พิมพ์ 4 return กลับไป 45

n/10 | n/10

ค้าง 2 | ค้าง 2 | เอา 4 % 10 = 0 ไม่พิมพ์

เอา 456 % 10 | เอา 45 % 10

= 6 | = 5

123 | 12 | 1

T | T | F

PO(12) | PO(1) | เสร็จ พิมพ์ 1 กลับไป 2 พิมพ์ทันที

ค้าง 2 | ค้าง 2

พิมพ์ 3 | พิมพ์ 2

การบ้าน 1

จงเขียนโปรแกรมหาค่า Fibonacci

กำหนด function Fibonacci ดังภาพ

แสดงผลลัพธ์เป็นค่า fibonacci ตั้งแต่ 0-19

F(0) = 1

F(1) = 1

F(2) = 1

...

F(19) = 4181

$$F_0 = 0, \quad F_1 = 1,$$

and

$$F_n = F_{n-1} + F_{n-2}$$

for $n > 1$.

2.2 Algorithm Analysis

Definition : An algorithm is a clearly specified set of simple instructions to be followed to solve a problem.

- correct
- time or space

Example 5 Running time Calculations

```
int sum(int n)
{       int partialSum;
        partialSum=0;
        for(int i=1; i<=n; i++)
                partialSum += i*i*i;
        return partialSum;
}
```

Example 6 Maximum subsequence sum

6 4 3 5 11 9

4 -3 5 -2 -1 2 6 -2

4	-3
4 + -3	-3 + 5
4 + -3 + 5	-3 + 5 + -2
4 + -3 + 5 + -2	-3 + 5 + -2 + -1
…	…
4 + -3 + 5 + -2 + -1 + 2 + 6 + -2	-3 + 5 + -2 + -1 + 2 + 6 + -2

-4 -5 -6 2 5 -1 3 -18

Example 6 Maximum subsequence sum

4 -3 5 -2 -1 2 6 -2

```
int MasSubsequenceSum(int a[], int N)
{       int ThisSum, MaxSum, j;
        for(j=0; j<N; j++)
        {       ThisSum += A[j];
                if( ThisSum > MaxSum)
                        MaxSum = ThisSum;
                else if( ThisSum < 0 )
                        ThisSum=0;
        }
        return MaxSum;
}
```

Ex

x = x + y; = 1 + N + 1 + N + 2N

= 4N + 3

1 + N + 1 + N

ThisSum = ThisSum + A[j]

= 2N

1 + N + 1 + N + 2N + 1

= 4N + 3

2.2.1) Mathematical Background

Definition : T(N)=O(f(N)) if there are positive constants c and n_0 such that $T(N) \leq cf(N)$ when $N \geq n_0$.

- Give two functions
- we compare their relative rates of growth.

Although 1000N is larger than N^2 for small value of N, N^2 grow at a faster rate, and thus N^2 will eventually be the larger function.

T(N)=O(f(N)) if there are positive constants c and n_0 such that
T(N) ≤ cf(N) when N ≥ n_0.

- T(N) = 1000N
- F(N) = N^2
- เมื่อ N=1, c=1 จะเห็นว่า 1000N > N^2
- เมื่อ N มีค่ามากขึ้น จนถึง 1000 1000N = N^2
- เมื่อ N มากกว่า 1000 1000N < N^2

We can say that 1000N = O(N^2)

1000N เป็นฟังก์ชันที่โตไม่เร็วกว่า N^2

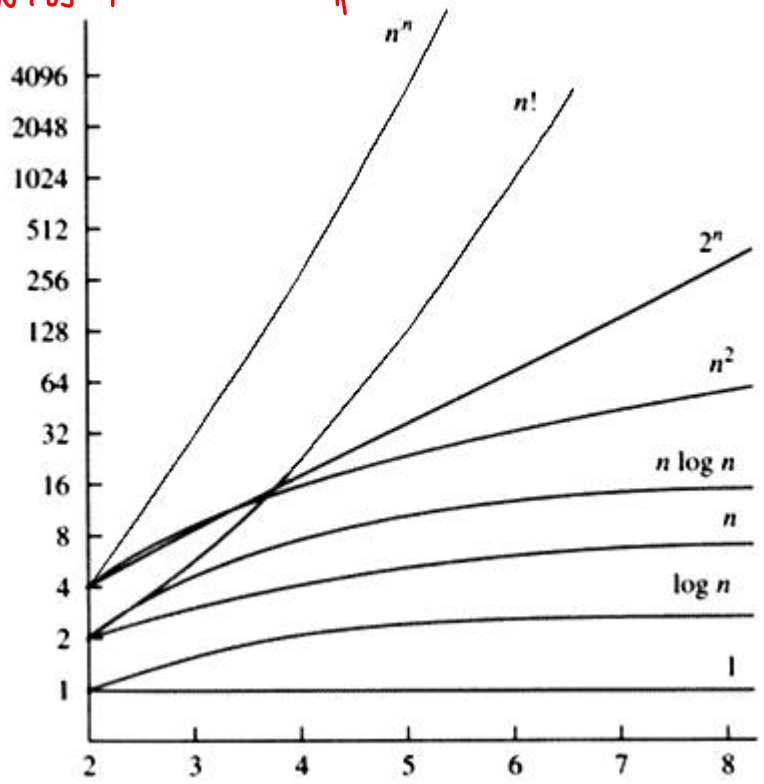
N เยอะยิ่งช้า

Function	Name
C	Constant
logN	Logarithmic
$\log^2 N$	Log-squared
N	Linear
NlogN	
N^2	Quadratic
N^3	Cubic
2^N	Exponential

n ยิ่งมากค่าลังเลอะยิ่งช้า

ช้าสุด

เร็วสุด



เร็วสุด

2.2.2) General Rules

Rule 1– For loop

The running time of a for loop is at most the running time of the statements inside the for loop(including tests) times the number of iterations.

```
for(i=0; i<n; i++)
        k++;
```

1 + N + 1 + N

2N

= ~~1~~N + ~~0~~

= N

Rule 2– Nested loops

Analyze these inside out. The total running time of a statement inside a group of nested loop is the running time of the statement multiplied by the product of the sizes of all the loops.

```
for(i=0; i<n; i++)
        for(j=0; j<n; j++)
                k++;
```

$O(N^2)$

$O(N)$

Rule 3– Consecutive statement For loop

Add.

```
for(i=0; i<n; i++)
        a[i]=0;
for(i=0; i<n; i++)
        for(j=0; j<n; j++)
                a[i]+=a[j]+i+j;
```

O(N)

$O(N^2)$

$O(N^2 + N)$

$= O(N^2)$

Rule 4– If/Else

The running time of an if/else statement is never more than the running time of the test plus the running times of S1 and S2

```
if (condition)
        S1
else
        S2
```

จริง 2

เท็จ 2