2.0 Mathematics Reviews

1) Exponents

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

$\dfrac{X^A}{X^B} = \quad X^{A-B}$

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

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

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

2) Logarithms

$X^A = B \quad \text{->} \log_x B = A$

$\log_A B = \quad \dfrac{\log_c A}{\log_c B}$

3) Series

อนุกรม

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

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

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

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

คำถาม

```
for ( int i=1 ; i<=n ; i++)
    sum = sum+i
```

$1 + 2 + 3 + \ldots + (N-1) = ?$

$$\frac{N(N+1)}{2}$$

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

for(int i

2.1 A Brief Introduction to Recursion :

Mathematical function

Circular logic?

ทำแบบไม่หยุด

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

2. $\boxed{y = \sin(x) * pi}$

Recursive โปรแกรมเรียกใช้ Function ตัวเอง

3. $\overset{y}{\boxed{f(x)}} = 2\boxed{f(x-1)}+x^2$

$f(0)=0$, x nonnegative integer

4. Factorial กำหนด x เป็นจำนวนเต็มที่ไม่เป็นลบ

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

Recursive ต้องมี Base Case

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;       Base case
    else
        return x* fact(x-1);
}
```

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

[ans =] มี return Value

Example 2 Factorial

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

return value
type int

```
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
}
```

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

return กลับ
ไปที่ สีม่วง

Example 3

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

F

∴ วนไปเรื่อยๆ ไม่ลง base case

1. ไม่มี base case
2. มี base case แต่ไม่ลง base case

ข้อสอบมีไล่ Recursive #

Example 4

123

```
void printout(int n)
{   if( n >=10 )
        printout(n/10);
    cout << n%10;
}
```

123	12	1
T	T	F
Printout (12)	Printout (1)	
ค้างไว้ 2	ค้าง 2	
พิมพ์ 3	พิมพ์ 2	พิมพ์ 1

Example 5

ออกสอบ

123

```
void printout(int n)
{  if( n >=10 )
   {  printout(n/10);
         cout << n%10;
   }
}
```

123	12	1
T	T	F
PO(12)	PO(1)	
ค้างไว้ 2	ค้างไว้ 2	
3	2	

45 23

การบ้าน 1 ***

จงเขียนโปรแกรมหาค่า Fibonacci

กำหนด function Fibonacci ดังภาพ

แสดงผลลัพธ์เป็นค่า fibonacci ตั้งแต่ 0-19

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

and

Function

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

for $n > 1$.

F(0) = 1

F(1) = 1

F(2) = 1

…

F(19) = 4181

การวิเคราะห์ขั้นตอน

2.2 Algorithm Analysis

เฉพาะ

Definition : An algorithm is a clearly specified set of

ชุดคำสั่งง่ายๆ

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

11

4 -3 5 -2 -1 2 6 -2 −2

4

4 + -3

4 + -3 + 5

4 + -3 + 5 + -2

…

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

-3

-3 + 5

-3 + 5 + -2

-3 + 5 + -2 + -1

…

-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;
}
```

ทำเหมือน instruction

$1 + \boxed{N + 1} + N$

j=0 j<N j++

Thissum = Thissum + A[j]

$2N$

return

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

$= 4N + 3$

$\cancel{2}N^2 + \cancel{N}$

2.2.1) Mathematical Background

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

- 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 $T(N) \le cf(N)$ when $N \ge n_0$.

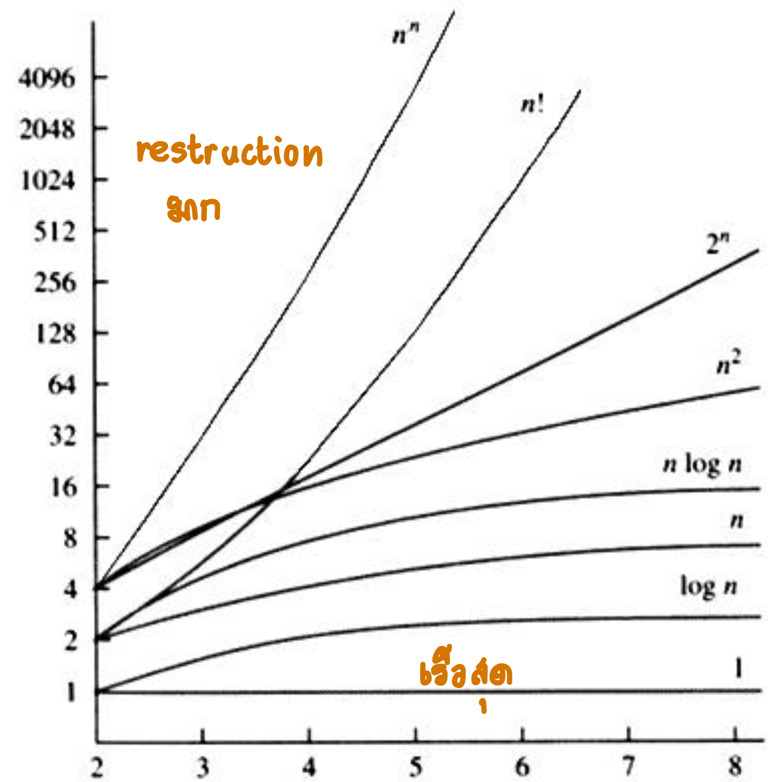
- $T(N) = 1000N$ — T
- $F(N) = N^2$ — F
- เมื่อ N=1, c=1 จะเห็นว่า $1000N > N^2$
- เมื่อ N มีค่ามากขึ้น จนถึง 1000 $1000N = N^2$
- เมื่อ N มากกว่า 1000 $1000N < N^2$

upper b ...

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

1000N เป็นฟังก์ชันที่โตไม่เร็วกว่า N^2

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



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++;
```

เชิงกรณีเกิด

$1+N+1+N$

$2N = \cancel{4}N + \cancel{2}$

อัตราการเติบโตของคำสั่ง

$O(N)$

k++ ;

k = k+1 ;

$2N$

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;
```

$O(N)$

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

$O(N^2)$

$\boxed{O(N)}$ $O(N^2)$ $O(N^2+\cancel{N})$

คำสั่งน้อยยิ่งเร็ว

คำสั่งมากยิ่งช้า

ตัดตัวหน้าออก

และตัวหลังออก

ถ้า 1,000,000 คำสั่งมันเยอะ

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

```
1 if (condition)          O(1)
1         S1
  else
          S2
```