1-1 Time Complexity


```
for (int i=0; i<N; i++)
  for (int j=0; j<i*i; j++)
    for (int k=1; k<j; k=k*2)
      print("Hello");
```

$f(N) = \underbrace{(1 \log 1 + 4 \log 4 + 9 \log 9 + \ldots. + N^2 \log N^2)}_{N項}$

$f(N) \leqq (N * N^2 \log N^2)$

$= \Theta (N^3 \log (N))$

$f(N) \geqq (\frac{N}{2} * \left(\frac{N}{2}\right)^2 \log \left(\frac{N}{2}\right)^2)$

$= C\ N^3 \log \left(\frac{N}{2}\right)$

$= C\ N^3 \log (N) - C\ N^3$

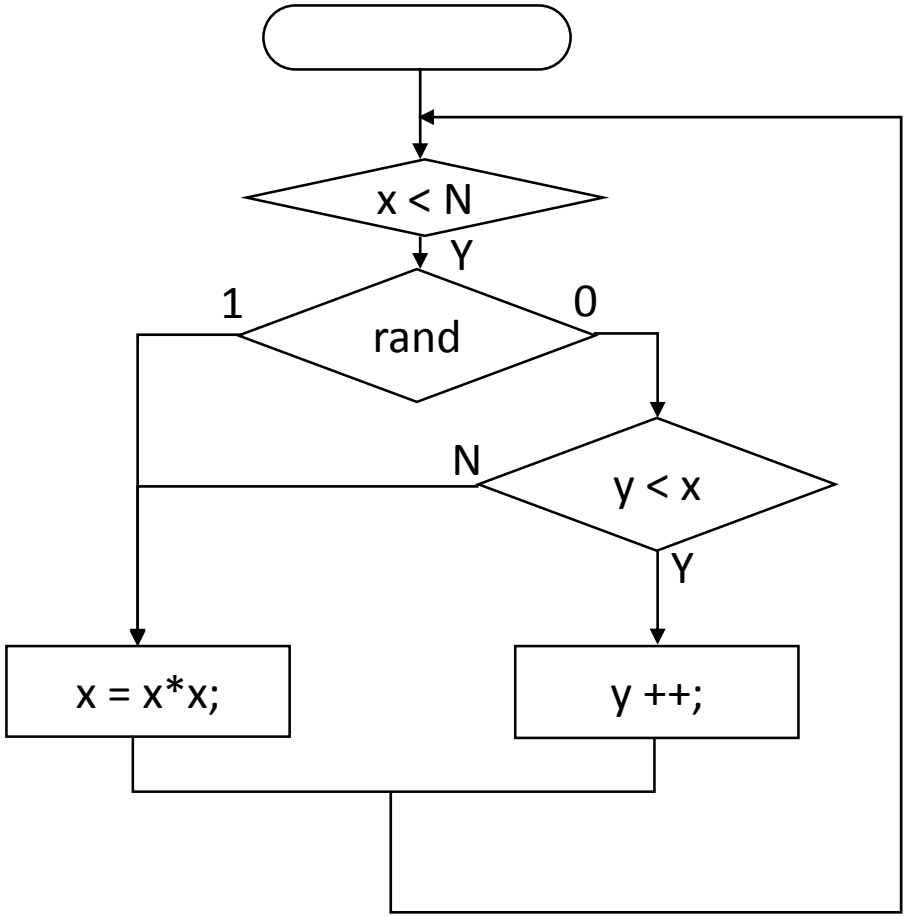
$= \Theta (N^3 \log (N))$

Ω($N^3 \log (N)$ **)**

O($N^3 \log (N)$ **)**

1-2 Time Complexity

```
x = 2; y = 2;
while (x<N) {
  j = random 0 or 1;
  if (j == 0 && y<x) {
    y=y+1;
  }else{
    x=x*x;
  }
}
```



Ω($\log\log(N)$ **)**

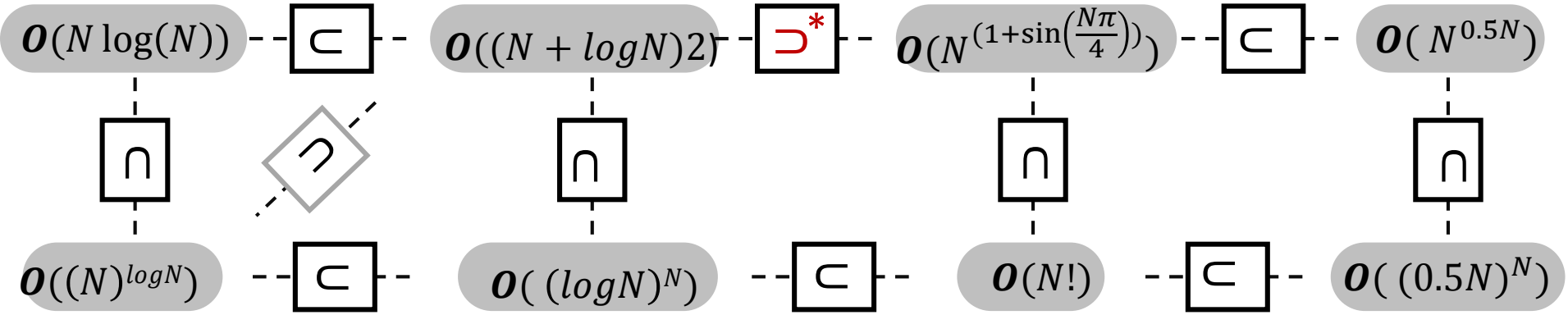
O(N **)**

2 Complexity Hierarchy

N = 2y

$(N)^{0.5N} = 2^y y^y$

N^2



N = 2^x

$(2^x)^x = 2^{(x^2)}$

N = 2^x

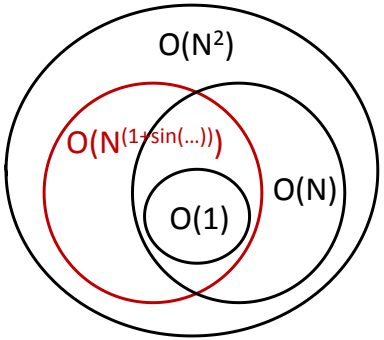
$x^{(2^x)}$

$O((N/3)^N)$

N = 2y

$(0.5N)^N = y^{2y} = y^y\ y^y$

* 謝謝19級林倢愷指出$O(N^{(1+\sin(\frac{N\pi}{4}))})$與$O(1), O(N), O(N^2)$的關聯如右圖。



3 Recursive MAX

```
int RMAX(int array[], int size)
{
    int m;

    if (size == 1) return array[0];

    int m1 = RMAX (array, size/2);
    int m2 = RMAX (array+size/2, size-size/2);

    if (m1>m2)  m = m1;
    else  m = m2;

    return m;
}
```

your code goes here

Time Complexity

- T(size) = **2 T(size/2) + 1**

  = 4 T(size/4) + 2 + 1
  = 8 T(size/8) + 4 + 2 + 1
  = …. +4 +2 +1

  log (size)

  = Θ ($2^{\log(size)}$) = **Θ (size)**

3 Recursive MAX


```
int RMAX(int array[], int size)
{
    int m;
    if (size == 1) return array[0];
    int m1 = array [0];
    for(int i=0; i<size/2; i++)
       if (m1 > array[i]) m1 = array[i];

    int m2 = RMAX (array+size/2, size-size/2);

    m = (m1 > m2)? m1 : m2;
    return m;
}
```

your code goes here

3 Recursive MAX


```
int RMAX(int array[], int size)
{
    int m;
    if (size == 1) return array[0];

    for(int i=0; i<size/2; i++) {
        if (array[i] < array[size-i-1])
            swap(array[i], array[size-i-1]);
    }

    m = RMAX (array, size/2);
    return m;
}
```

your code goes here

Time Complexity

- T(size) = **T(size/2) + size/2**

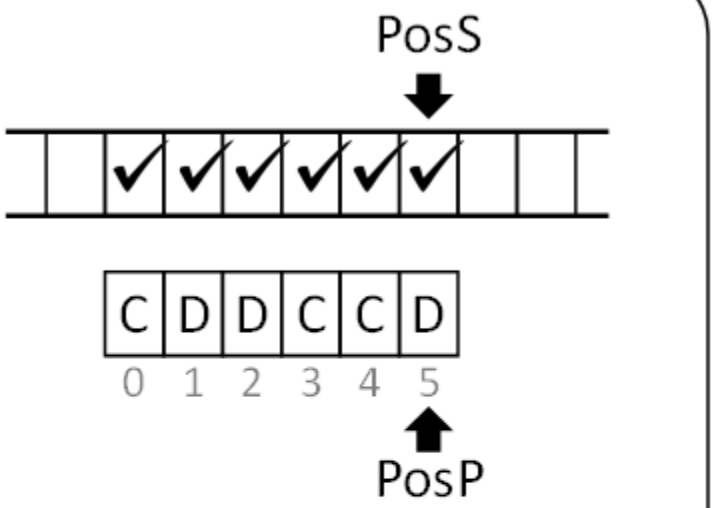
  = T(size/4) + size/4 + size/2
  = T(size/8) + size/8 + size/4 + size/2
  = 1 + … + size/8 + size/4 + size/2

  = **Θ (size)**

4-1 KMP

S	D	T	D	S	S	D	T	D	S	S	x
0	0	0	0	1	1	2	3	4	5	6	__7__ if x == 'D' __1__ if x == 'S' __0__ if x == 'T' __0__ otherwise

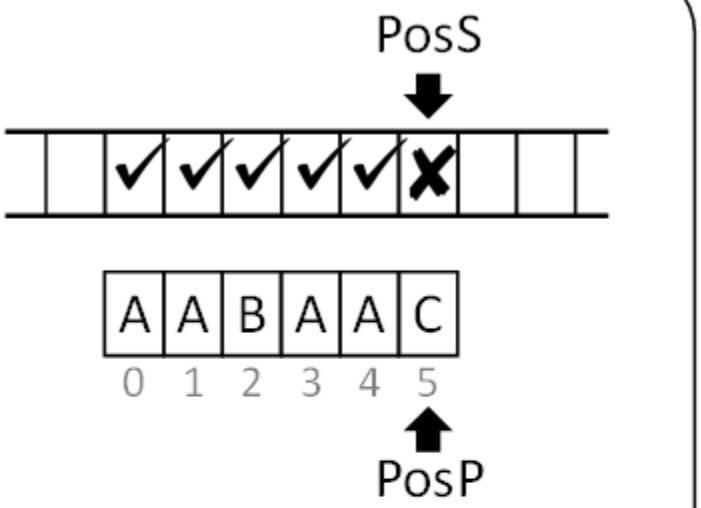
4-2 KMP



Next action:

PosS ++;

PosP = 2;



Next action:

PosS = PosS;

PosP = 2;

5 Infix to Postfix

bottom

Token	Stack						Output So Far															
A							A															
||	||						A															
(	||	(					A															
A	||	(					A	A														
+	||	(	+				A	A														
B	||	(	+				A	A	B													
*****	||	(	+	*			A	A	B													
C	||	(	+	*			A	A	B	C												
%	||	(	+	%			A	A	B	C	*											
9	||	(	+	%			A	A	B	C	*	9										
)	||						A	A	B	C	*	9	%	+								
<	||	<					A	A	B	C	*	9	%	+								
D	||	<					A	A	B	C	*	9	%	+	D							
==	||	==					A	A	B	C	*	9	%	+	D	<						
T	||	==					A	A	B	C	*	9	%	+	D	<	T					
&&	||	&&					A	A	B	C	*	9	%	+	D	<	T	==				
B	||	&&					A	A	B	C	*	9	%	+	D	<	T	==	B			
						Final output	A	A	B	C	*	9	%	+	D	<	T	==	B	&&	||	

Priority	Operator
High	In-coming (
	*, /, %
	+, -
	<, <=, >=, >
	==, !=
	&&
	||
Low	In-stack (

6 Asymptotic Notations

$F(N) + G(N) = O(N!)$

→ exist natural numbers c and N_0 such that
$F(N) + G(N) \leqq c\,N!$ for $N \geqq N_0$

$F(N)*G(N) \leqq (F(N) + G(N))^2/4$

$\leqq c^2/4\,(N!)^2$ for $N \geqq N_0$

$(N!)^2 < (2N)!$

Let $c' = c^2/4$ $N_0' = N_0$

$F(N)*G(N) \leqq c'\,(2N)! = O((2N)!)$ for $N \geqq N_0'$ QED

7-1 Three Basic Structures

- Sequential
- Selection (if-else)
- Iteration (loop)

7-2 Structured vs non-Structured

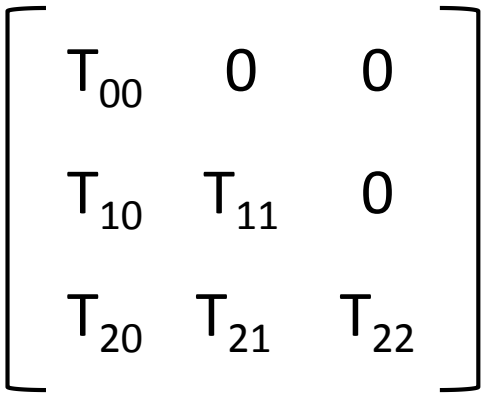
- Structured program cannot always achieve better speed than a non-structured one

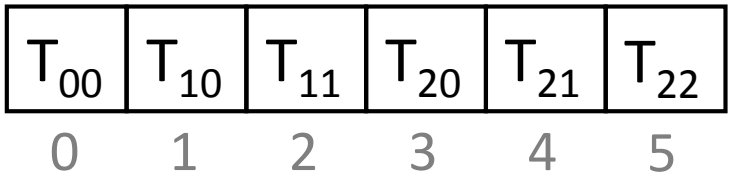
- Structured programs are always compiled into machine language programs, which are non-structured programs

7-3 Access levels of objects

- Changing the access levels of objects cannot affect the function of a program
  - Access levels are NOT designed for realizing functions
  - Access levels are NOT designed for protecting 智慧財產權
- Access levels are for maintaining a clean object interface
  - Preventing object users from accidently messing up the internal values of the object
  - Preventing object users from relying on the internal values of the object

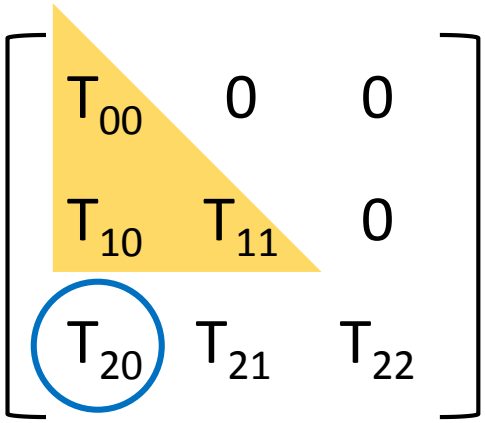
8 Lower-Triangular Matrix

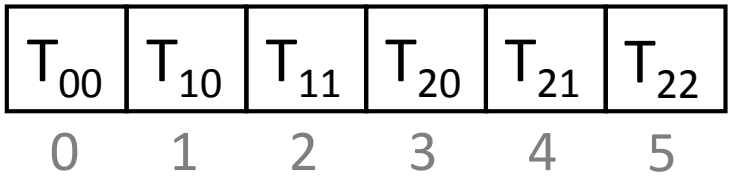




- Number of integers for storing directly using a raw array
- 1 + 2 + 3 +… + N = (1+N)N/2

8 Lower-Triangular Matrix





- Offset of T_{ij}

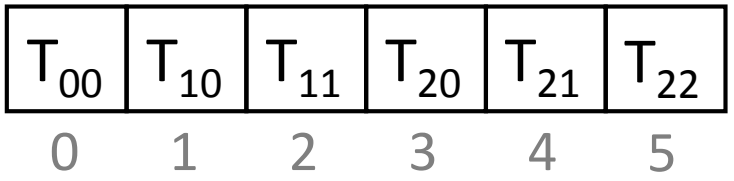
= (1+i) i /2 + j

- T_{20}

= (1+2)*2/2 + 0 = 3

Lower-Triangular Matrix

$$\begin{bmatrix} T_{00} & 0 & 0 \\ T_{10} & T_{11} & 0 \\ T_{20} & T_{21} & T_{22} \end{bmatrix}$$



# of integers = **(1-R)** * (1+N)N/2 *** 3** + 2

If the matrix is stored in a sparse matrix