Big-O – upper bound $f(n) \le C\,g(n)$

Show that $n+1$ is $O(n^2)$

Proof $n \le n^2 \quad \forall n \ge 2$

$1 \le n^2 \quad \forall n \ge 2$

$\therefore n+1 \le 2n^2$

$\therefore n+1$ is $O(n^2)$

Show that $n^2 + 4n + 10$ is $O(n^2)$

$n^2 \le n^2 \quad \forall n$

$4n \le n^2 \quad \forall n \ge 4$

$10 \le n^2 \quad \forall n \ge 4$

$n^2 + 4n + 10 \le 3n^2 \quad \forall n \ge 4$

$\therefore n^2 + 4n + 10$ is $O(n^2)$

Big Omega – lower bound $f(n) = \Omega(g(n))$, $f(n) \ge C\,g(n)$

show that $2n^2 + n + 10$ is $\Omega(n)$

Proof $2n^2 \ge n \quad \forall n \ge 1$

$n \ge n \quad \forall n$

$10 \ge n \cdot 0 \quad \forall n$

$2n^2 + n + 10 \ge 2n \quad \forall n \ge 1$

show that $5n^2 + 8n + 20$ is $\Omega(n^2)$

Proof $5n^2 \ge n^2 \quad \forall n$

$8n \ge n^2 \cdot 0 \quad \forall n$

$20 \ge n^2 \cdot 0 \quad \forall n$

$5n^2 + 8n + 20 \ge n^2 \quad \forall n \ge 0$

Big-theta $C_1 g(n) \le f(n) \le C_2 g(n)$

Show that $5n^2 + 5n + 5 = \theta(n^2)$

Proof $\forall n \quad n^2 \le 5n^2$ | $5n^2 \le n^2 \quad \forall n$

$\forall n \quad 0 \cdot n^2 \le 5n$ | $5n \le n^2 \quad \forall n \ge 1$

$\forall n \quad 0 \cdot n^2 \le 5$ | $5 \le n^2 \quad \forall n \ge 3$

$\therefore 2n^2 \le f(n) \le 3n^2 \quad \forall n \ge 3$

from code

loop = n   loop ซ้อน loop = n^2

code	จำนวนครั้ง # คำสั่ง	เวลา
max = $-\infty$	1	O(1)
for (i=0 ; i < n ; i++)	1 + (n+1) + (n) = 2n+2	O(n)
if A[i] > max	n	O(n)
max = A[i]	≤ n	O(n)

total time = $O(1) + O(n) + O(n) + O(n)$

$f = O(1) \le C_1 \cdot 1 \le C_2 \cdot n \le C_3 \cdot n \le C_4 \cdot n$

$f \le C \cdot 1 \le C_1 + (C_2 + C_3 + C_4) \cdot n$

$\le (C_1 + C_2 + C_3 + C_4) \cdot n$

$\le C_5 \cdot n = O(n) \quad \forall n \ge \max(n_0, n_2, n_3, n_4)$

Sorting

Array : A collection of data of the same type.

	0	1	2	3	4	...	10
A	10	50	40	~~8~~ 10	15	...	99

A[0] A[3] A[4]

A[0] = 10   A[3] = 10   A[4] = 15

Selection sort → $O(n^2)$

Find min

8 5 3 1 2 7 4 9 10

1 5 3 8 2 7 4 9 10

2 3 8 5 7 4 9 10

3 8 5 7 4 9 10

next min

- Pick min (A[m] is minimum)
- Swap A[m] ↔ A[i]

```
O(n)          --- for (i=0 ; i<n ; i++)
O(1)·O(n)     ---   min=∞
O(n)·O(n)     ---   for (j=i ; j<n ; j++)
O(1)·O(n^2)   ---     if min > A[j]
O(1)·O(n^2)   ---       min = A[j]
                        m = j            find min
tmp = A[i]
A[i] = A[m]      swap A[m], A[i]
A[m] = tmp
```

Insertion sort → $O(n^2)$

5, 3, 6, 2, 9, 4, 1, 8, 7   insert element

3, 5, X

X, 3, 5, 6

2, 3, 5, 6, X

2, 3, X, 5, 6, 9

X, 2, 3, 4, 5, 6, 9

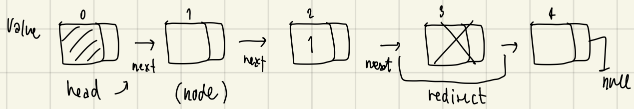
1, 2, 3, 4, 5, 6, X, 9

1, 2, 3, 4, 5, 6, X, 8, 9

1, 2, 3, 4, 5, 6, 7, 8, 9

find place to insert		insert		total
O(n)	+	O(n)	=	O(n)
if use binary search O(log n)				for 1 element
O(n)	+	O(1)	=	O(n)

Linked list : list of data, chain of number



node : int value → node value

comparison	Array	linked list
Access	O(1)	O(n)
modify	O(1)	access + O(1)
delete	O(n)	access + O(1)
Insert	O(n) sorted	access + O(1) sorted
find max	O(n) | O(1)	O(n) | O(1)

Bubble sort

k = 1

5 4 2 7 8 6

4 5 2 7 8 6

4 2 5 7 8 6

4 2 5 7 8 6

4 2 5 7 8 6

4 2 5 7 6 8

k = 2

4 2 5 7 6 8

2 4 5 7 6 8

2 4 5 7 6 8

2 4 5 7 6 8

2 4 5 6 7 8

2 4 5 6 7 8

* if we can flip two consecutive objects, we can sort ...

```
for k = 1 to n
  for i = 1 to n
    if A[i] > A[i+1]
      swap (A[i], A[i+1])
```

* another code that works too but is not bubble sort

```
for i = 1 to n
  for j = i+1 to n
    if A[i] > A[j]
      Swap (A[i], A[j])
```

Analysis

$O(n^2)$

```
for i=1 to n
  for j=1 to n
    if i < j
      sum += i+j
```

$O(n)$

```
i = 0
while i < n
{ j = 1
  while (j < n/10)     -> 450   j = n,  j < n/10
  } sum += j
    j++
  }
  i = i+j
}
```

$O(n \log n)$

```
i = 0 ;
while i < n
{ j = 1
  while j < n
  } sum += j
    j++
  }
  i++
}
```

$n = 2^k \quad O(\log n) \le H_n \le O(\log n)$

$H_n = \theta(\log n)$

$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4}+\frac{1}{5}+\frac{1}{6}+\frac{1}{7} + \frac{1}{8} + \frac{1}{9}$

$\le 1 + \frac{1}{2} + \frac{1}{2} + \frac{1}{4}+\frac{1}{4}+\frac{1}{4}+\frac{1}{4} + \frac{1}{8} + \frac{1}{8}$

$H_n \le k + 0$

$H_n \le \log(n) + \frac{1}{n}$

$H_n \approx \log(n)$

* upper bound

Time complexity

```
int f(int A[1..n])
{ max = -∞                    O(1)
  for i = 1 to n              O(n)
    if max < A[i]             O(1) x O(n) = O(n)
      max = A[i]              O(1) x O(1) x O(n) = O(n)
  return max }                O(1)
```

$T(n) = O(1) + O(n) + O(n) + O(n) + O(1) = O(n)$

$\forall n \geq n_1$

$\leq c_1 n + c_2 n + c_3 n + c_4 n + c_5 n \leq c_6 n$

$c_6 = \max(c_1, c_2, c_3, c_4, c_5)$ $n_6 = \max(n_1, n_2, n_3, n_4, n_5)$

$f(n) = O(g(n))$ $f = O(n)$ | $f = O(n)$

$f(n) \leq c\, g(n)$ $f \leq c_1 n$ | $f \leq c_2 n$

$\forall n \geq n_0, c > 0$ $\forall n \geq n_1$ | $\forall n \geq n_2$

ex. $T(n) = 2T(n-1) + 1$ (find max)

```
T(n)    int max (A[1..n])
O(1)    if n = 1, return A[1]
T(n-1)  x = max (A[1...n-1])
T(n-1)  y = max (A[2...n])
O(1)    if x > y
O(1)        return x
O(1)    else
O(n)        return y
```

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

```
T(1)    int max (A[1..n])
O(1)    if n = 1, return A[1]
T(n-1)  m = max (A[1..n-1])
O(1)    if A[n] > m
O(1)        return A[n]
O(1)    else
T(n-1)      return max (A[1..n-1])
```

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

ex $T(n) = T(n-1) + 1$

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

$= 2(2T(n-2) + 1) + 1 = 4T(n-2) + 3$

$= 2(2(2T(n-3) + 1) + 1) + 1 = 8T(n-3) + 7$

$= 2(2(2(2T(n-4) + 1) + 1) + 1) + 1 = 16T(n-4) + 15$

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

when $k = n-1$

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

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

note $T(n) = T(n-1) + 1 \rightarrow O(n)$

$T(n) = 2T(n-1) + 1 \rightarrow O(2^n)$

Recursive function

```
int fac(n)      T(n)
{ if n < 1      O(1)
    return 1    O(1)
  else
    return n*fac(n-1)
           O(1) + T(n-1)
}
```

$\therefore T(n) = O(1) + O(1) \times T(n-1)$

$\leq c_1 \leq c_1 T(n-1)$

$T(n) \leq c_1 + c_2 T(n-1)$

$\forall n \geq \max(n_1, n_2)$

change $O(1) \rightarrow 1$, $O(n) \rightarrow 1$

$T(n) = 1 + T(n-1)$

$T(n) = T(n-1) + 1$

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

$= T(n-3) + 1 + 1 + 1$

$T(n-k) + 1 + 1 + 1 + \dots + 1$ (k)

when $T(1) = 1$

$\because T(n-1) = T(n-2) + 1$

$\because T(n-2) = T(n-3) + 1$

$\because T(n-3) = T(n-4) + 1$

$\therefore T(n) = T(1) + n - 1 = n$

$\therefore T(n) = O(n)$

ex $T(n) = T(n-1) + 1$

n
↓ +1
n-1
↓ +1
n-2
↓ +1
n-3
↓ +1
1

n-1 level

$= 1 \times T(1) + n - 1$

$= 1 + n - 1 = n$

$\therefore T(n) = O(n)$

ex. $T(n) = 2T(\frac{n}{2}) + 1$

$\exists k$ $n \leq 2^k$

$n \leq 2^k$

$K = \log n$

$T(n) = 2^k \times T(1) + 1 + 2 + 4 + \dots + 2^{k-1}$

$\simeq 1 + 2 + 4 + \dots + 2^{k-1} + 2^k = 2^k - 1$

$\therefore T(n) = n - 1$

$n = 2^k$

$2^{\log_2 n} = n$

$4^{\log_2 n} = n^2$

$8^{\log_2 n} = n^3$

$a^{\log b} = b^{\log a}$

$\log(2^{\log b}) = \log(b^{\log 2})$

$\log b \cdot \log 2 = \log a \times \log b$

Note

$T(n) = 2T(\frac{n}{2}) + 1 \rightarrow O(n)$

$T(n) = 2T(\frac{n}{2}) + n \rightarrow O(n \log n)$

$T(n) = 2T(\frac{n}{2}) + n^2 \rightarrow O(n^2)$

$T(n) = 2T(\frac{n}{2}) + n^3 \rightarrow O(n^3)$

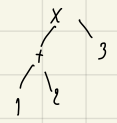
Binary Tree

Pre order : 1st time (Root) L → R

Inorder : 2nd time L → Root → R

Post order : 3rd time L → R → root

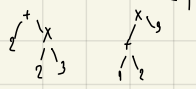
ex $12 + 3X$ * Post

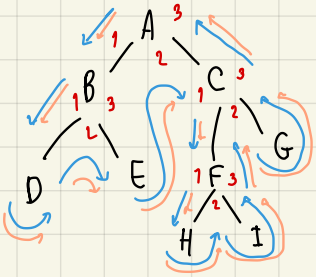


ex $23-$

ex $1 + 3 - 2 + 2 \times 3 + 5 - 4 = ((1+3)-2) + ((2\times3)+(5-4))$

* Inorder $1+(2\times3) \neq (1+2)\times3$





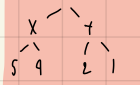
Pre-order ABDECFHIG

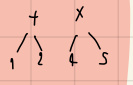
Inorder DBEAHFICG

Post order DEBHIFGCA

Preorder ≠ Post order

$-X\,5\,9 + 2\,1$ ≠ $12 + 45X -$





$(5\times9)-(2+1) \neq (1+2)-(4\times5)$

Yogouin1

(1) Show that $3n^2 - 6n + 8 = \Theta(n^2)$ (theta)

$\forall n \quad 3n^2 \le 3n^2$

$\forall n \quad 0 \cdot n^2 \le -6n$

$\forall n \quad 0 \cdot n^2 \le 8$

$\therefore 3n^2 \le f(n) \le 5n^2 \quad \forall n \ge 3$

(2) $n \log n = O(n^2)$

$n \le n$

$\log n \le n$

$n \log n \le n^2 \quad n \log n = O(n^2)$

(5)

```
int f2 (int n)
{   int sum = 0 , i , j;
    for i = 1 to n-1          ---- O(n)
        j = i ;
        while (j < n)         ---- O(n)
            sum = sum + 1;
            j = j + 1;
        end
    end
    return sum;
}
```

i	j
1	1, 2, 3, 4, .. n-1 → n-1
2	2, 3, 4, 5, .. n-1 → n-2
⋮	⋮
n-1	n-1

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

$= O(n^2)$

(7) Suppose $f(n) = \Omega(g(n))$ ($f(n) \ge g(n)$) and $\forall n \; f(n) > 0$

Show that whether or not $\log(f(n)) = O(\log(g(n)))$

Let $g(n) = 1$, $f(n) \ge g(n)$

$\log(g(n)) = 0 < \log(f(n))$

$\therefore \log(f(n)) \neq O(\log(g(n)))$

(8) $T(n) = 4T(n-2) + 2^n$ & $T(1) = 1$ find Big(o) of $T(n)$ n = odd integer

$T(n) = 4T(n-2) + 2^n$ $k=1$

$= 4T(n-4) + 2^{n-2} = 4^2 T(n-4) + 2^n \left(1 + \frac{1}{2^2}\right)$ $k=2$

$4^2\left(4T(n-6) + 2^{n-4}\right) + 2^n\left(1 + \frac{1}{2^2}\right) = 4^3 T(n-6) + 2^n\left(1 + \frac{1}{2^2} + \frac{1}{2^4}\right)$ $k=3$

$= 4^k T(n-2k) + 2^n\left(1 + \frac{1}{2^{2(1)}} + \frac{1}{2^{2(2)}} + \dots + \frac{1}{2^{2(k)}}\right)$

$k = \frac{n+1}{2}$

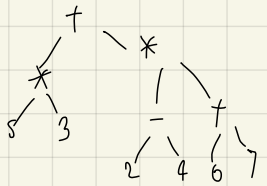
$T(n) = 4^k T(n-2k) + 2^n\left(\frac{1 - \left(\frac{1}{4}\right)^k}{1 - \frac{1}{4}}\right)$

$= 4^{\frac{n+1}{2}} T(1) + 2^n\left(\frac{1 - \left(\frac{1}{4}\right)^{\frac{n+1}{2}}}{\frac{3}{4}}\right)$

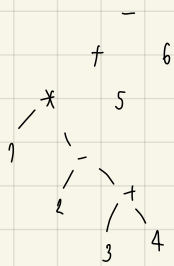
$= 2^{n+1} + 2^n\left(\frac{1 - \frac{1}{2^{n+1}}}{\frac{3}{4}}\right)$

$= O(2^n) + O(2^n) = O(2^n)$

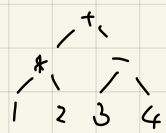
(10) (a) pre-order + * 5 3 * - 2 4 + 6 7



(b) post 1 2 3 4 + - * 5 + 6 -



(c) in-order 1 * 2 + 3 - 4



(11) pre-order B C A F G D E & post A G D F C E B