Notes By: Md Sadman Hafiz
SWE, SUST

Sub :

Day

Time : Date : / /

Basics

Big oh, Omega, theta Notation

→ If the order (power) of any function is n.

then, upper bound = anything greater or equal to n.

lower bound = anything $< n$.

big-oh (O) → lowest of the upper bound.

Omega (Ω) → highest of the lower "

theta (Θ) → n.

complexity:

$1 < \log n < \sqrt{n} < n < n \log n < n^2 < n^3 \ldots < 2^n < 3^n <$

Sub :

Day

Time : Date : / /

complexity of program

c-1

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

i	j
0	0
1	1
2	2
⋮	

$\frac{n(n+1)}{2} \simeq n^2$

$\boxed{O(n^2)}$

c-2

```
p = 0
for (i=1; p<=n; i++)
    p = p+i;
```

i	p
1	0+1
2	1+2
3	1+2+3
4	1+2+3+4
⋮	
k	1+2+...k

$\therefore O(\sqrt{n})$.

when $p > n$, it will stop : $\frac{k(k+1)}{2} > n$

$\therefore k^2 > n$

$\therefore k > \sqrt{n}$.

Sub :

Day

Time : Date : / /

C-3 :

for (i = 1; i < n; i = i * 2)

Stoping point.

$i >= n$.

$2^k >= n$.

$k = \log_2 n$.

$O(\log_2 n)$

i
1×2
$2^1 \times 2$
$2^2 \times 2$
$2^3 \times 2 =$
$\therefore 2^k$

C - 4 for (i = n; i >= 1; i = i / 2)

$\frac{n}{2^k}$.

$\therefore O(\log_2 n)$.

Day
Sub :
Time : Date : / /

A for (i=0; i*i < n; i++)

$O(\sqrt{n})$

* p = 0
for (i=1; i < n; i = i*2)
{ p++; } → $O(\log n)$

for (j=i; j<p; j = j*2)
{ } → $\log p$

$\therefore O(\log \log n)$

Sub :

Day

Time : Date : / /

* for (i = 0; i < n; i++) ——— n

for (j = 1; j < n; j = j * 2)

{ n log n.

}

---

n + n log n.

O (n log n).

Complexity Analysis

Merge Sort

$$T(n) = T(n/2) + T(n/2) + n$$

$$T(n) = 2T(n/2) + n \quad \text{---} (1)$$

substitute $n/2$ in place in eq (1)

$$T(n/2) = 2T(n/4) + n/2 \quad \text{---} (2)$$

Substitute eq (2) in eq (1)

$$T(n) = 2\{2T(n/4) + n/2\} + n$$

$$T(n) = 2^2 T(n/2^2) + 2n \quad \text{---} (3)$$

substitute $n/4$ in ... (3)

$$T(n/4) = 2T(n/8) + n/4 \quad \text{---} (4)$$

$$T(n) = 2^2\{2T(n/8) + n/4\} + 2n$$

$$= 2^3 T(n/2^3) + 3n \quad \text{---} (5)$$

$$\therefore T(n) = 2^4 T(n/2^4) + 4n \text{----}$$

$$T(n) = 2^i T\left(\frac{n}{2^i}\right) + in$$

let, $\frac{n}{2^i} = 1$

$\Rightarrow 2^i = n$

$\Rightarrow i = \log_2 n$

$\therefore T(n) = n T(1) + n \log_2 n$

$= n + n \log_2 n$

$= O(n \log_2 n)$

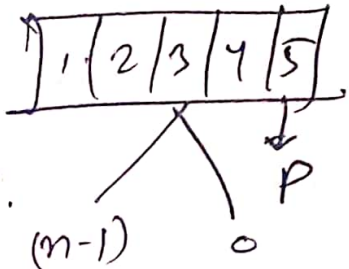
Best, worst, Avg : $O(n \log_2 n)$

Space $= O(n)$

quicksort

partition

Worst case: Already sorted.



$T(n) = T(n-1) + n$ → partition

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

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

$= T(n-K-1) + (n-K) \dots (n-2) + (n-1) + n.$

n → 1,7

n-1 → 2,7

n-2 → 3,7

n-3 → 4,7

n-k → 5,7

for terminating, $n-K-1 = 1.$

$\Rightarrow K = n-2$

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

$[n-(n-2)] + T(n-2)$

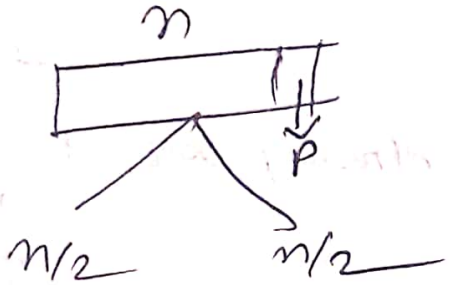
$T(n-(n-2)-1)$

$= n + (n-1) + (n-2) \dots 2+1$

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

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

Best Case: (Array is divided in two equal part)



$$T(n) = 2T(n/2) + n.$$

↗ partition

$O(n \log n)$ like as merge sort

Avg case:

1 element pivot 100 elements
100 elements pivot 1 element

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

$$= c_n + \frac{1}{n}\sum_{i=0}^{n-1} (T(i) + T(n-i-1))$$

$$= c_n + \frac{1}{n}\sum_{i=0}^{n-1} 2\,T(i)$$

$$T(n) = c_n + \frac{2}{n}\left(T(0) + T(1) + \cdots T(n-1)\right)$$

$$n\,T(n) = n^2 c + 2\left(T(0) + T(1) + \cdots T(n-1)\right) \quad \text{---} \ (1)$$

put $n = n-1$

$$(n-1)\,T(n-1) = (n-1)^2 = (n-1)^2 c + 2(T(0) + T(1) + \cdots T(n-2)) \quad \text{---} \ (2)$$

(1) − (2)

$$n\,T(n) - (n-1)\,T(n-1) = cn^2 - c(n-1)^2 + 2T(n-1)$$

$$= cn^2 - c(n^2 - 2n + 1) + 2T(n-1)$$

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

$$\therefore n\,T(n) = (n-1)c + T(n-1) + 2T(n-1) + 2cn - c$$

$$= T(n-1)\,(n+1) + 2cn - c$$

dividing both side by $n(n+1)$

$$\frac{T(n)}{n+1} = \frac{T(n-1)}{n} + \frac{2c}{n+1}$$ [remove c as it is constant]

$$\Rightarrow \frac{T(n)}{n+1} - \frac{T(n-1)}{n} = \frac{2c}{n+1} \quad \text{---(1)}$$

put $n = n-1$ →

$$\frac{T(n-1)}{n} - \frac{T(n-2)}{n-1} = \frac{2c}{n} \quad \text{---(2)}$$

put $n = n-2$ →

$$\frac{T(n-2)}{n} - \frac{T(n-1)}{n-2} = \frac{2c}{n-1} \quad \text{---(3)}$$

$$\therefore \frac{T(n)}{n+1} - \frac{T(0)}{1} = 2c\left(\frac{1}{n+1} + \frac{1}{n} + \cdots \frac{1}{3} + \frac{1}{2}\right)$$

$$\frac{T(n)}{n+1} = 2c \log(n+1)$$

$$T(n) = (n+1)\, 2c \log(n+1)$$

$$\approx n \log(n+1)$$

$$\approx n \log n$$

Sub :

Day

Time : Date : / /

Time and Space complexity

Alg	Best case	Avg	Worst	Space (worst)
Bubble	$\Omega(N)$	$\theta(N^2)$	$O(N^2)$	$O(1)$
Selection	$\Omega(N^2)$	$\theta(N^2)$	$O(N^2)$	$O(1)$
Insertion	$\Omega(N)$	$\theta(N^2)$	$O(N^2)$	$O(1)$
Merge	~~$\Omega(N)$~~ $\Omega(N \log N)$	$\theta(N \log N)$	$O(N \log N)$	$O(N)$
Quick	$\Omega(N \log N)$	$\theta(N \log N)$	$O(N^2)$	$O(N \log N)$
Heap	$\Omega(N \log N)$	$\theta(N \log N)$	$O(N \log N)$	$O(1)$

faster