Intro To Time Complexity
Friday, 16 August 2024 5:00 PM

TLE

Q ⇒ How to decide whether solⁿ is optimal or not.

Check for a prime — I
Find square root — I

	PodcastI	P II
	Algo 1	Algo 2
P1	5.1 secs ⇓ C/C++	8 secs ⇓ Python
P2	5.1 secs ⇓ mobile phone ⇓ ~~2.1 sec~~	4.9 secs ⇓ Macbook ⇓ 4.9 secs
	4.1 sec	3.5 sec

Sort 500

I/P ⇒

	Algo 1	Algo 2
Prime no	N-2	$\sqrt{N}-1$
Find square root	$\sqrt{N}$	$\log_2(N)+2$

```
for (int i=0; i<n; i++) {     1+n+1+n+2n
    2 print (a[i]);            = (4n+2) ⇒ f(n)
}
    ⇒ 2N            0 ⇒ n
```

Arithmetic o/p ⇒ +, -, ×, ÷, %
Comparison o/p ⇒ ==, <=, >=, !=
Assignment ⇒ a = 5, b = 10, c = 100000
Bitwise ⇒ &, |, ^, >>, <<
Logical o/p ⇒ &&, ||
function calls ⇒ add (a, b), sqrt(a)
⇒ return a;
array op ⇒ arr[i], a[i][j]
increment / decrement o/p ⇒ a = a+1 (a+1), b++; ++a;

```
for (int i=0; i<n; i++) {  ⇒ n
    for (int j=0; j<n; j++) {
        print a[i][j];          ⇒ (4n+2) × n
}
```

$(4n+2)n + \underset{i=0}{1} + \underset{i<n}{n+1} + \underset{i++}{n}$

$4n^2 + 2n + 1 + n + 1 + n$

$g(n) = \boxed{4n^2 + 4n + 2}$

$f(n) = 4n + 2$ — 1D
$g(n) = 4n^2 + 4n + 2$ — 2D — big value ↑n
$h(n) = 4n^3 + 3n^2 + 2n + 1$

Eg ⇒ I ⇒ $4n^2 + \sqrt{n} + 3\log(n) +$
II ⇒ $4n + 8n^2 + 9\log(\frac{n}{2})$ > No

n↑ ⇒ no. of o/p ↑

I		II
Hospital	Market	America (US)
1 km	100 m	Bhat don
2 km	1 km	12000 ₹
500 m	4 km	18 hrs flight
Walking	Walking	16 hrs
5 mins	100 yrs	24000 km

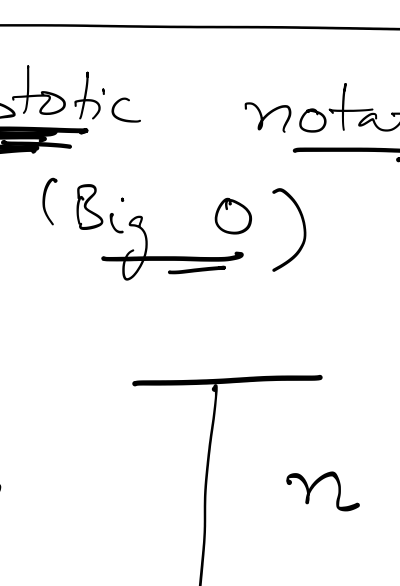
x — y
4n² — 4n³, 5n³ — z

Asymptotic notations

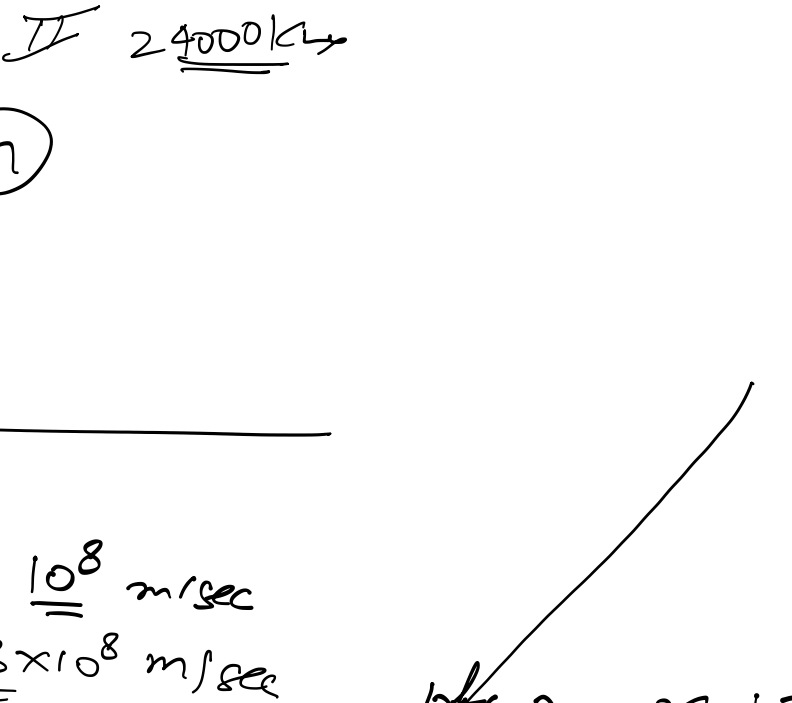
(Big O)

n ↑

n ⇒ ∞ ≈ 10^8 m/sec, 3×10^8 m/sec



4n³ — n, n², n³



⇒ Order of rate of increase of time w.r.t. input size.

$g(n) = 4n^2 + 4n + 2$ ⇒ n^2

$n^2 + \underset{n^2}{\underbrace{n}} + \underset{n^2}{1}$

$n^2 + n^2 + n^2$

$= 3n^2$

T ⇒ $O(n^2)$

$(n-3)\,n^2$

$n^3 - 3n^2$

3 ⇒ n = 10, n² = 100

Q ⇒ Print all the even no's from 1 to n

⇒ O(n) Google

```
for (i=0; i<n; i+=2) { ⇒ n/2
    print(i);          (i=i+2)
}
0 1 2 | 4 | 6 | 8 | ... | n
```

$O(\frac{n}{2})$ × — $\frac{n}{2}$ ⇒ constant

⇒ O(n)

n — n² — n³

Q ⇒
```
int i = N, a = 0
while (i >= 0) {
    1 a = a+v
    1 i = i/2
}
```

add (a, b) {
1 return (a+b) }
O(1)

O(1)

i = N/2⁰ ← 1st iter
i = N/2 ← 2nd
i = N/4 = N/2² ← 3rd
i = N/2³ — 4
i = 1 = N/2^{k-1} ← kth iter

i ⇒ N

int = (1/2) = 0

```
for (i=0; i<N; i++) {
    print (a[i]);
}
```
⇒ 2

```
i=5
print (a[i]);
```
⇒ 3 ⇒ constant O(1)

$\frac{N}{2^{k-1}} = 1$

15, 7, 3, 1, 0

$N = 2^{k-1}$

$\log_2 N = \log_2 2^{k-1}$

$\log_2 N = (k-1)\log_2 2$

$k = \log_2 N + 1$

$k = \log_2 N$

T.C. = $O(\log_2 N)$

$N = 2^{k-1}$

$\log_3 N = \log_3 2^{k-1}$

$\log_3 N = (\log_3 2)(k-1)$

$k-1 = \frac{\log_3 N}{\log_3 2}$

log ⇒ Same bucket

$\log_{10} N$ ↓ — $\log_2 N$

O(log N)

```
for (i=1; i<n; i=i*2) {      i=N  i>0  i=i/2
    print (i)
}
```

1 ⇒ 1
2 ⇒ 2
4 ⇒ 3
8
2^N ⇒ k

$K = 2^N$

Quest
```
int K = 0                     n = 
for (i=1; K<n; i++) {         O(N) ×
    K = K + i;                1 i++
}
```

K = 0 — i = 1 ⇒ initially
K = 1 — i = 2 ⇒ 1st iteration
K = 1+2 — i = 3 ⇒ 2nd
K = 1+2+3 — i = 4 ⇒ 3rd
K = 1+2+3+4 — i = 5 ⇒ 4

⋮ x — i = x+1

K = 1+2+3+4 ⋯ x

1+2+3+4 ⋯ x = N

⇒ $\frac{(x)(x+1)}{2} = N$

$x^2 + x = N$ ⇒ $\sqrt{N}$

$\log_2 x^2 = \log_2 N$
$(\log_2 x)2 = \log_2 N$

$x^2 + x - N = 0$

$x = \frac{-1 \pm \sqrt{1^2 - 4(-1)(N)}}{2(1)}$

$x = \frac{-1 \pm \sqrt{1+4N}}{2}$

$O(n) \simeq O(\sqrt{N})$

```
for (i=0; i<N; i++) {
    for (j=i; j<N; j++) {
        print i, j;
    }
}
```
⇒ $O(N^2)$

0 1

```
int j=0;  1, 2, 3
for (i=0; i<N; i++) {          ⇒ O(N)
    while (j<N && arr[j] < arr[i]) {
        j++;                    ⇒ O(n)
    }
}
```

O(1)

i=0, a[0]

1	2	3	4	5	6	7	8	9	10	11	12

N

O(n) + O(n) = O(2n) ⇒ O(n)

0 = n-1 ⇒ N
1 ⇒ 1 ⇒ n-1 ⇒ N-1
2 ⇒ 2 ⇒ n-2 ⇒ N-2
⋮
1

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

```
for (i=0; i<n; i++) {
  ⇒ while (j<n && arr[j] < arr[i]) {
        j++;
    }
}
```

once (n)

0	1	2	3	4	5	6	7
2	3						

i=1, j=1

once (n)

(n+n) = 2n
= n

( ) ⇒ x ⇒ O(n)
+
( ) ⇒ y ⇒ O(y)

Sorting ⇒ quick, merge > n log n

n/2 ⇒ O(n²)

O(1)

Binary, Trinary > more optimal / n ⇒ n/2

W/ Φ — W/ Φ theta

O

x² + n = N

x = O( )

while (n) {
n = n & (n-1)
count++;
}

Bit Manip

Lectures + Unlocking
Some library — Java

```
for (i=n; i>1; i=i/2) {
    for (i=0; i<n; i=2*i) {
```
5

n = 2^k = ⇒ log(n)