Lecture: 01 : Prerequisite

Advanced Module Description

Arrays — 3-4
Bit manipulation — 2
Recursion — 2
Math — 1
Hashing — 2
Sorting — 2

Contest 1 — Arrays. Bit. Recursion, Math. Hashing

Searching — 3
Class objects & LL basics
Stack — 2
Queue — 1
Trees — 3
Revision — DSA 1 & 2

Contest: sorting, searching, LL, stack, Queue, Trees

Math — 2
Two pointers — 1
Backtracking — 2
LL problems — 2
Trees — 2
Hashing internal implementation

Contest: Math, 2 pointers, Backtracking, LL, trees

Heap — 2
Greedy — 1
Interview problems
DP — 4.
Graph — 4
Revision — DSA 3 & 4

Contest: Heap, Greedy, DP & Graph

Contest: full syllabus { Mandatory to be eligible for placements }

Q1 $1 + 2 + 3 + 4 + 5 \cdots\cdots\cdots n.$

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

1 —— $s = 1 + 2 + 3 + 4 + 5 \cdots\cdots (n-1) + n$

2 —— $s = n + (n-1) + (n-2) + (n-3) \cdots\cdots 2 + 1$

$2s = (n+1) + (n+1) + (n+1) + (n+1) \cdots\cdots (n+1) + (n+1)$

n

$2s = (n+1) * n$

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

Qu2 Nos in range $[3, 8]$ → inclusive

→ inclusive

$3, 4, 5, 6, 7, 8 \longrightarrow$ 6 Ans

$8 - 3 + 1 =$ 6 Ans

$[a, b] \longrightarrow b - a + 1.$

Ex: $[1, 8] \longrightarrow 8 - 1 + 1 = 8.$

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

Qu Given a no. n, find the count of factors of n.

n = 12 ⟶ 1, 2, 3, 4, 6, 12 Ans = 6

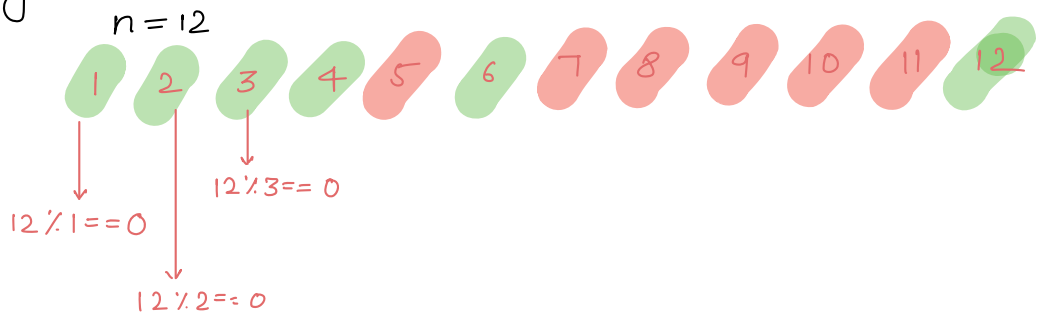
factors: n % i == 0

[1 - n]

n = 24 ⟶ 1, 2, 3, 4, 6, 8, 12, 24 Ans = 8

n = 10 ⟶ 1, 2, 5, 10 Ans = 4.

Logical




```
int countfactors(n) {
    count = 0;
    for(i=1; i<=n; i++) {
        if( n%i==0) {
            count++;
    }   }
    return count;
}
```

ite = n.

Execution Time: Depends on external factors

fun game: 10^8 itr in 1 sec ——— Assumption

itr = n.

```
int countfactors(n) {
    count = 0;
    for(i=1; i<=n; i++) {
        if(n%i==0) {
            count++;
        }
    }
    return count;
}
```

$n = 10^9$ ⟶ 10^9 itr

10^8 itr = 1 sec

1 itr = $\frac{1}{10^8}$ sec

10^9 itr = $\frac{10^9}{10^8}$ sec = 10 sec

$n = 10^{18}$ ——— 10^{18} itr

1 itr = $\frac{1}{10^8}$ sec

10^{18} itr = $\frac{10^{18}}{10^8}$ sec = 10^{10} sec ≃ 317 years

You ⟶ children ⟶ grandchildren

⟶ 4th gen ⟶ 5th gen.

Optimization.

$i * j = n$. { i & j are factors of n }

$j = \frac{n}{i}$ { i & $\frac{n}{i}$ " " " " }

$n = 24$

i	$j = n/i$
1	$24/1 = 24$
2	$24/2 = 12$
3	$24/3 = 8$
4	$24/4 = 6$
6	$24/6 = 4$
8	$24/8 = 3$
12	$24/12 = 2$
24	$24/24 = 1$

part1

$n = 100$

i	$j = n/i$
1	$100/1 = 100$
2	$100/2 = 50$
4	$100/4 = 25$
5	$100/5 = 20$
10	$100/10 = 10$
20	$100/20 = 5$
25	4
50	2
100	1.

Part1: $i <= \frac{n}{i}$

$i^2 <= n$.

$i <= \sqrt{n}$.

```
int countfactors(n) {
    count = 0;
    for(i=1; i*i<=n; i++) {        // go forward
        i <= sqrt(n)
        if(n % i == 0) {
            if(i == n/i) {
                cnt += 1;
            } else {
                cnt += 2;
            }
        }
    }
    return cnt;
}
```

$i = \sqrt{n}$.

$n = 10^{18}$.

its $= 10^{18} \longrightarrow \sqrt{10^{18}} = 10^9$ its

1 its $= \frac{1}{10^8}$ sec

10^9 its $= \frac{10^9}{10^8} = 10$ sec

317 years $\xrightarrow{\text{optimisation}}$ 10 sec

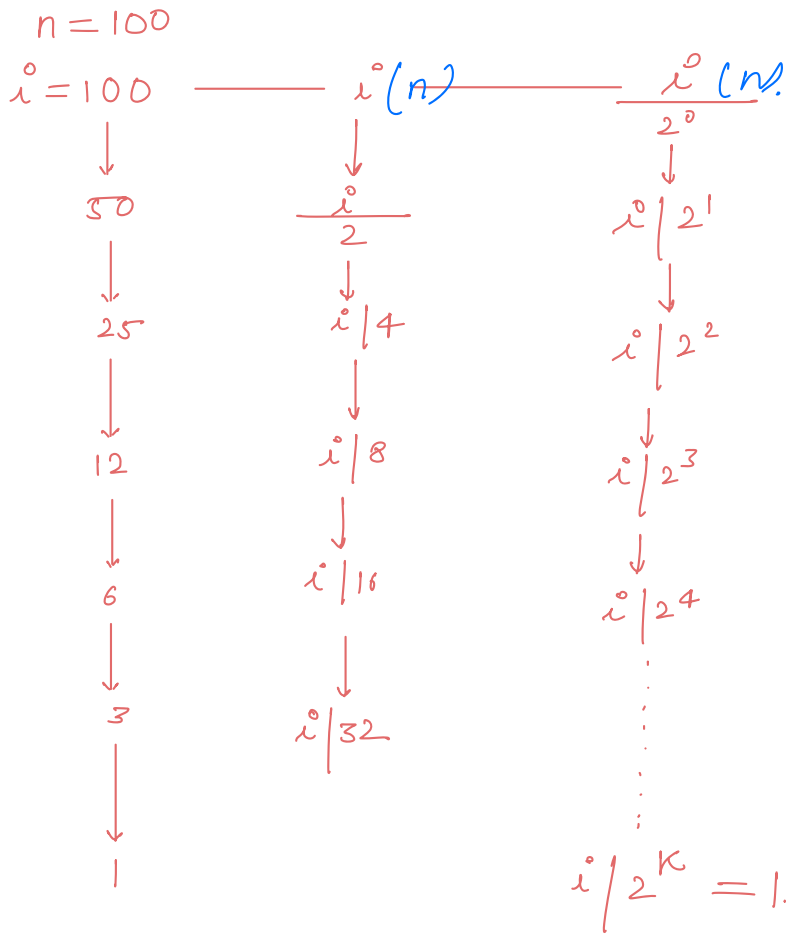
Qu $n \longrightarrow$ check it is prime or not?

factors = 2. { 1 and no itself }

```
Boolean isPrime(n) {
    fac = countfactors(n);
    if (fac==2) {
        return true;
    }
    return false;
}
```

its $= \sqrt{n}$.

```
i = n.
while (i > 1) {
    i = i / 2;
}
```



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

$$2^k = n.$$

$$\log_2 2^k = \log_2 n.$$

$$\log_a a^b = b$$

$$\boxed{k = \log_2 n}$$

```
for(i=1 ——— n) {
    for(j=1 ——— n) {
        print(i+j)
    }
}
```

i	j [1-n]	# its
1	[1-n]	n.
2.	[1-n]	n.
3	[1-n]	n.
4	[1-n]	n
5	[1-n]	n.
⋮	⋮	⋮
n.	[1-n]	n.

n.

$n * n = n^2$ Ans

```
for(i=0 ——— n-1) {
    for(j=0 ——— i) {
    }
}
```

i	j [0-i]	# its
0	[0,0]	1
1	[0,1]	2.
2	[0,2]	3
3	[0,3]	4
4	[0,4)	5
⋮	⋮	⋮
n-1	[0, n-1)	n

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

Big O complexity

Steps

1. # itr
2. Ignore lower term
3. Ignore co-efficients

```
for( i=0 ——— n-1) {
    for(j=0 ——— i) {
    }
}
```

itr $= \frac{n^2}{2} + \frac{n}{2}$

$= O(n^2)$ Ans

itr $= n^2 + 10n$.

higher term $= n^2$

Lower term $= 10n$.

Input size	# itr	% of lower term contribution
$n = 10$	$10^2 + 10 * 10 = 200$	$\frac{10*100}{200} * 100\% = 50\%$
$n = 100$	$100^2 + 10*100$ $= 10000 + 1000$ $= 11000$	$\frac{1000}{11000} * 100\% = 9.\_\_\_\%$
$n = 1000$ (10^3)	$10^6 + 10*10^3$ $10^6 + 10^4 \longrightarrow$	$\frac{10^4}{10^6 + 10^4} \simeq$ very less
$n = 10^4$		very very less $< 1\%$

Best case & Worst case

```
boolean search(A[], k) {
    for(i=0 ——— n) {
        if(A[i]==k) {
            return true;
        }
    }
    return false;
}
```

2 8 1 6 5 14 13

Best case: k = 2.
TC: O(1)
~~# its~~: 1

Worst case: k = 13
value is not present in array.
~~# its~~ = O(n)
TC:

Space complexity

Algo is taking how much space

```
main() {
    int a=5;                    ——— O(1)
    int[] A = new int[10]       ——— 10 * 4 = 40 bits
                                    O(1)
    A = new int[n]              ——— n.
}
```

4 + 40 + 4 * n. = O(n)
int a, A[10], A[n]

Break: 10:23 — 10:31.

TLE

Time limit exceeded

1 sec $\simeq 10^8$ ops

Problem constraints

$n <= 10^5$

- $O(n^2)$ — $(10^5)^2 = 10^{10} > 10^8$ — Yes (tle)
- $O(n)$ $[10^5]$ — work
- $n \log_2 n$ — $10^5 * \log_2 10^5$, $10^5 * 23$, $23 * 10^5$, $2.3 * 10^6$ → work

$n = 10$

- $O(2^n)$ — $2^{10} \simeq 1024$ — work

Approach

Manas → Amazon

Q1 → Coded. → TLE

Q2

Q1

↓

Constraints

↓

Logic based on constraints

↓

Code.

↓

submit

Bitwise operator

(&)

a	b	a & b
0	0	0
0	1	0
1	0	0
1	1	1

→ 0 dominating

(OR)

a	b	a | b
0	0	0
0	1	1
1	0	1
1	1	1

→ 1 dominating

(^) XOR

a	b	a ^ b
0	0	0
0	1	1
1	0	1
1	1	0

{ same same puffy shame.

Not (~)

0 → 1

1 → 0