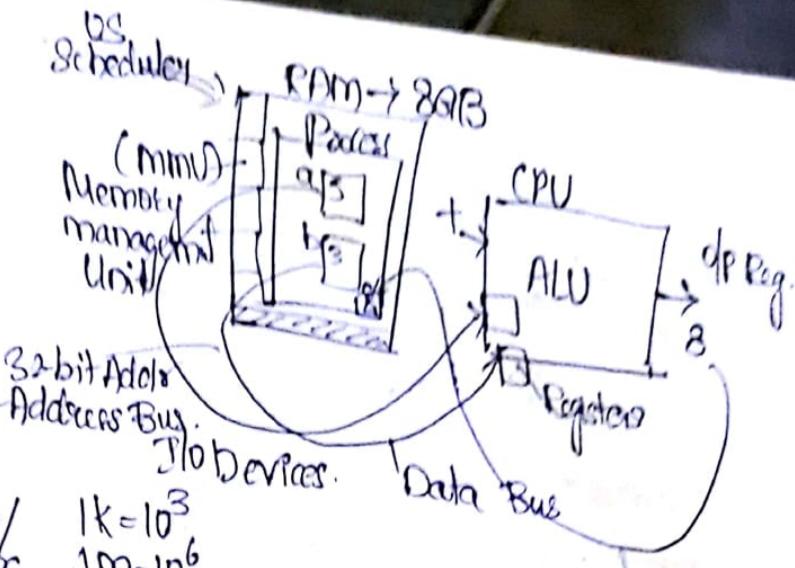


A
`int a=5, b=3;
`int c=a+b;
`point c;

A B
1GHz 2GHz
 $10^9 \text{ clockcycles/sec}$ $2 \times 10^9 \text{ inst/sec}$
 10^9 inst/sec



$$\begin{aligned} 1k &= 10^3 \\ 1M &= 10^6 \\ 1G &= 10^9 \end{aligned}$$

$$1 \text{ ClockCycle} = C \geq 1 \text{ Inst.}$$

→ just by looking at freq value of processor, cannot decide fastness → depends on no. of Inst. can be done in 1cc

2. → Depends on RAM (Memory)

- If RAM has less memory.

- more no. of swaps - Thrashing inc, performance decreases.

3. → Depends on OS (New version - more optimization). faster

4. → No. of Cores

5. → Hard Disk - Solid State Disk (faster), Hard disk (Mechanical slow)

→ GPU. Digital.

→ I/O Interrupt.

*. Life Cycle of a Process.

A (32-bit)

4GB (RAM)

B (32-bit)

- 8GB (RAM).

1 Bytes = 8 bits

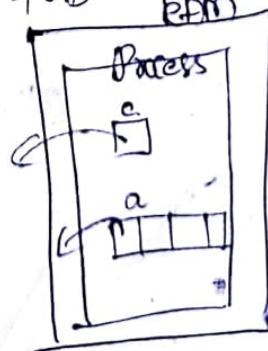
1 KB = 2^{10} B = 1024

1 MB = 1024 KB = 2^{20} B.

1 GB = 1024 MB = 2^{30} B.

char c;
int a;

32-bits Port H/W & S/W.
 $4 \text{ GB} = 4 \times 2^{30} = 2^{32}$ B → 32 bits.



Smallest fetchable unit from RAM is 1 byte

RAM-4GB.

$$1GIB = 2^{90} B$$

B > 8GB.

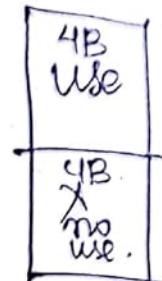
$$= 22^{\circ} 22' 30'' B$$

$$= 2^{3g} \rightarrow 3g \text{ bits.}$$

RAM = 8GiB \Rightarrow 33 bits Addressing are needed.

But Address Bits are only 32-bits:

so we cannot even address next 4GB.



Place the system with UGIB RAM

and SystemB with 8G13 with 32-bit A/D Bus have same Performance

3/w. 4/w.

32 on 32

64 bits in 32 bits. (Cannot carry 64 bits data as there is only 32 bits)

82 on 64 ✓

64 on 64 ✓

④ Primary Rep. for 10 :- $\frac{0}{2} \frac{0}{6} \frac{0}{5} \frac{0}{4} \frac{1}{3} \frac{0}{2} \frac{1}{1} \frac{0}{0}$

should use
g's comp.

unsigned.
10

$$2^{\text{d}} \text{ comp} = 1^{\text{e}} \text{ comp} + 1$$

-128-

64
32
16
~~8~~
~~16~~
118

~~0000 1000~~ 1800010:-11110101

~~not there~~
as it has
a sup. for
single value 0.

$$\begin{array}{r}
 & & & & & 1 & 1 \\
 & & & & & \hline
 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\
 -2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\
 \hline
 1 & 0 & 0 & 0 & 1 & 6 & 8 & 4 & 2 & 1
 \end{array}$$

~~-128
118
10~~

$$19^{\text{r}} \quad 00010011$$

1's comp : $1\ 1\ 1\ 0\ 1\ 1\ 0\ 0$

$$\begin{array}{r} + \\ \hline 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1 \end{array}$$

$\frac{128}{128} \ 64 \ 32 \ 16 \ 8 \ 4 \ 2 \ 1$

$$\begin{array}{r} 64 \\ 32 \\ 16 \\ \hline 109 \\ -19 \end{array}$$

$$-128 + 64 + 32 + 8 + 4 + 1 = -19.$$

$$\begin{array}{r} 105 \\ 64 \\ 32 \\ \hline 41 \\ -32 \\ \hline 9 \end{array}$$

$$105^{\text{r}} \quad 01101001$$

1's c : $1\ 0\ 0\ 1\ 0\ 1\ 1\ 0$

$$\begin{array}{r} + \\ \hline 1\ 0\ 0\ 1\ 0\ 1\ 1 \end{array}$$

-105^{r}

$$\begin{array}{r} 28 \\ 64 \\ \hline 14 \end{array}$$

$$78^{\text{r}} \quad 01001110$$

$1\ 0\ 1\ 1\ 0\ 0\ 0\ 1$

$$\begin{array}{r} + \\ \hline 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0 \end{array}$$

-78^{r}

$$\begin{array}{r} 200 \\ 128 \\ \hline 72 \end{array}$$

~~$200^{\text{r}} \quad 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0$~~

$$\begin{array}{r} 2^0 \ 2^1 \ 2^2 \ 2^3 \ 2^4 \ 2^5 \ 2^6 \ 2^7 \ 2^8 \ 2^9 \ 2^{10} \\ 1024 \ 512 \ 256 \ 128 \ 64 \ 32 \ 16 \ 8 \ 4 \ 2 \ 1 \end{array}$$

8 bits range.
-128 to +127

$$\begin{array}{r} 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0 \\ \underline{1} \end{array}$$

correct notation.

25. 00011001

Incr. -12
Notation

$$\begin{array}{r} 10001100 \\ + \\ \hline 10100100 \\ \underline{(101\ 001\ 00)} \\ 64\ 32\ 16\ 8\ 4\ 2 \end{array}$$

$-37 \cdot X$ as $25-12=13$
& not -37

$$\begin{array}{r} 127 \quad 00001100 \\ 11110011 \\ + \\ \hline 11110100 \\ -127 \quad 00011001 \\ 0000011001 \\ \underline{-128\ 64\ 32\ 16\ 8\ 4\ 2} \\ 8+4+1=13 \end{array}$$

- 1) Zero has 2 rep.
- 2) Add, sub. becomes easy.

<u>Data Types.</u>		<u>C/C++</u>	<u>depends.</u>
int - 4B	short long	00 8B.	32-bits \Rightarrow long - 4B, longlong - 8B
char - 1B	signed unsigned	(4B) (8B)	64-bits \Rightarrow long - 2B, longlong - 2B
float - 4B			Java - long - 8B
double - 8B			
(bool) Boolean -	Byte	1 bits But repn bytels.	0 - False 1 - True - 1 B. 4

For signed 4 bits :- $\begin{array}{cccc} -8 & 4 & 2 & 1 \end{array}$ $\begin{cases} \text{Min} = -8 = 1000 \\ \text{Max} = 4 = 0111 \end{cases}$

<u># bits</u>	<u>Signed</u>	<u>Unsigned</u>	
1	-1, 0	0, 1	$0 \rightarrow 0 \text{ Max}$
2	-2, 1	0, 3	$\frac{1}{-1} \rightarrow -1 \text{ Min.}$
3	-4, 3	0, 7	
4	-8, 7	0, 15	
8	-128, 127	0, 255	

$$n \rightarrow (-2^n, 2^{n-1})$$

$$(0, \overset{1}{\cancel{2}} - 1)$$

min max

~~9B~~ \Rightarrow 4B \Rightarrow No. of bits = $4 \times 8 = 32$ bits. $(-\frac{2^{31}}{2}, \frac{2^{31}-1}{2})$ (0, $\frac{2^{32}-1}{2}$)
Signed. Unsigned.

$2^{10} = 1024 \approx 10^3$.
 $2^{10} = 10^6$.
 $2^{20} = 10^9$.

$(-\frac{2 \times 10^9}{2}, \frac{2 \times 10^9-1}{2})$ (0, $\frac{4 \times 10^9-1}{2}$)
Signed. Unsigned.

$(-\frac{2 \times 10^9}{2}, \frac{2 \times 10^9-1}{2}), (0, \frac{4 \times 10^9-1}{2})$

Java long $= 8B = 8 \times 8 = 64$ bits. $\Rightarrow (-\frac{2^{63}}{2}, \frac{2^{63}-1}{2})$, (0, $\frac{2^{64}-1}{2}$)
 $2^{30} = 10^9$.
 $2^{40} = 10^{12}$.
 $2^{50} = 10^{15}$.
 $2^{60} = 10^{18}$.
 $2^{10} = 10^3$.
 $(2^{10})^6 = (10^3)^6$.
 $= \underline{\underline{10^18}}$.

char \Rightarrow 1B \Rightarrow $(-128, 127)$, (0, 255)
int \Rightarrow 4B \Rightarrow $(-\frac{2 \times 10^9}{2}, \frac{2 \times 10^9-1}{2})$, (0, 4×10^9)
long \Rightarrow 8B \Rightarrow $(-\frac{2 \times 10^{18}}{2}, \frac{2 \times 10^{18}-1}{2})$, (0, 16×10^{18})
float \Rightarrow 4B \Rightarrow (dp. Int. -? bit sign) 3 bits.
= 32 bits (dp. fraction) =? 23 bits.

* Double \Rightarrow 8B. \rightarrow if Integer \Rightarrow 11 bits
zoom high precision = 64 bits \rightarrow if fraction = 7(52) bits (52)

long double = 12B. \rightarrow x₁
= 96 bits \rightarrow y₁

0010000000000000

```

long
int solve (int arr[], int N)
{
    long sum = 0;
    for (int i = 0; i < N; i++)
        sum += arr[i];
    return sum;
}

```

$$1 \leq N \leq 10^5$$

$$-10^6 \leq arr[i] \leq 10^6$$

$$-10^{11} \leq sum \leq 10^{11}$$

~~max~~
~~no overflow~~

N can be ~~int~~ (✓)

int range:

~~(2x10⁹, 2x10⁹)~~, (2x10⁹, 2x10⁹)
~~signed~~

int - (-2x10⁹, 2x10⁹)
Java long (8B) → (-8x10¹⁸, 8x10¹⁸)
C/C++ long long (8B) → (-8x10¹⁸, 8x10¹⁸)

N - int ↵

array index → int ↵

sum → change data-type to long.
→ change return type also

$$1 \leq N \leq 10^4$$

$$1 \leq arr[i] \leq 10^8$$

$$1 \leq sum \leq 10^{12}$$

Constraints.

— Correct Data Types.

OPERATORS.

Arithmetic Op. (+, -, *, /, %).

1) Logical & &, ||, !

2) Relational op. (>, <, >=, <=, ==, !=).

3) Bitwise &, |, ^, ~, <<, >>

4) Ternary op. ((?: :))

5) Assignment, sizeof, , , , ,

6) Assignment, sizeof, , , , ,

Truth Table.

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

int a=25, b=18;

int c=a&b

a&b

a&b

~a

int a=(), b(), c();

int d= a/a=a

a&a=a

a|a=0

a|0=a

a&0=0

a|1 = a+(1-a%2)

$$(+) \times (\times)$$

$$\begin{array}{r} 64 \\ 32 \\ 16 \\ 8 \\ 4 \\ 2 \\ 1 \end{array} \quad \begin{array}{r} 128 \\ 64 \\ 32 \\ 16 \\ 8 \\ 4 \\ 2 \\ 1 \end{array}$$

$$a. 25: \begin{array}{r} 100011001 \end{array}$$

$$b. 18: \begin{array}{r} 00010010 \end{array}$$

$$a&b: \begin{array}{r} 00011011 \end{array} = 27$$

$$a&b: \begin{array}{r} 00010000 \end{array} = 16$$

$$a|b: \begin{array}{r} 00001011 \end{array} = 11$$

$$\&~a: \begin{array}{r} 11100110 \end{array} = 26$$

$$\begin{array}{l} \text{32 bits} \\ \text{int } 801.111100110 \end{array}$$

$$\begin{array}{r} \text{odd}/1 = \text{odd} \\ \text{even}/1 = \text{even} \end{array}$$

25:

$$\begin{array}{r} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{array}$$

$$\begin{array}{r} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{array} \quad a/1 \xrightarrow{\text{odd}=0} \quad a/\text{even}=1$$

$$\begin{array}{r} \text{Q5: } 00011001 \\ \text{Q. 1: } 00000001 \\ \hline 00000001 = 1. \end{array}$$

$\text{Odd \& 1} \Rightarrow 1$

$a \& 1 = \text{fast}$

$a \% 2 = \text{slow.}$

$a \& 1 \begin{cases} \rightarrow 0 \text{ (even)} \\ \rightarrow 1 \text{ (odd)} \end{cases}$

$a \& 1 = a \% 2.$

& is faster than modulus as

(Alternative)

$$15 \% 4 = 3$$

$$15 - 4 = 11 - 4 = 7$$

series of 3's

$$a \& b \& c = a \& c \& b = b \& a \& c = b \& c \& a =$$

Bitwise operators are all
Associative & Commutative

Left Shift ($<<$)

$$\begin{array}{l} \text{a: } 00001001 \quad (9) \quad 9. \\ \text{a} << 1 : 00010010 \quad (18) \quad 9 * 2. \\ \text{a} << 2 : 00100100 \quad (36) \quad 9 * 2^2 \\ \text{a} << 3 : 01001000 \quad (72) \quad 9 * 2^3 \\ \vdots \\ \text{a} << i : \dots \quad \dots \quad 9 * 2^i. \end{array}$$

Right Shift ($>>$)

$$\begin{array}{l} \text{a: } 00001001 \quad (9) \\ \text{a} >> 1 : 00000100 \quad (4) \\ \text{a} >> 2 : 000000010 \quad (2) \end{array}$$

$9 >> 9 = a / 2^9. \text{ Right Shift.}$

left shift.
(multiply)
 $a << i = a * 2^i$

$$9 / 2^1 = 4$$

$$9 / 2^2 = 9 / 4 = 2.$$

left shift : $a \ll i = a \cdot 2^i$ (doubles by $\times 2$)
 right shift : $a \gg i = a / 2^i$ (halves)

16. $8 \cdot 4 \cdot 2 \cdot 1$
Halves.

Beyond certain limit, gets 0.
shifts all the bits, either to left (\ll) or to right (\gg).

$$\text{e.g.: } a \ll 100 = 0 \\ a \gg 100 = 0.$$

int pow2(int N).

```

    int ans=1;
    for(int i=0; i<N; i++)
        ans=ans*2;
  
```

return ans;

$$a \ll i = a \cdot 2^i$$

long long int pow(int N).

```

    return 1LL << N;
    // (long long int)1 << N.
  
```

g.

for $0 \leq N \leq 63$ unsigned datatype $\rightarrow 0$ to $2^{63}-1$.

$0 \leq N \leq 70$ - long double
(96 bits)

$a=3$

011	$011 = 3$
<u>001</u>	<u>$100 = 4$</u>
<u>001</u>	<u>$1000 = 8$</u>
<u>110</u>	<u>$1000100 = 9$</u>

$0111 = 7$
<u>$1000 = 8$</u>

Q) $0 \leq x, y \leq 25$. Set x^{th} and y^{th} bits.

int setBits (int x, int y)

return $(1 \ll x) + (1 \ll y)$.

$(1 \ll 2) + (1 \ll 2)$

does not work if
arithmetic both x and
addition y bits are same.

$$0100 = 4$$

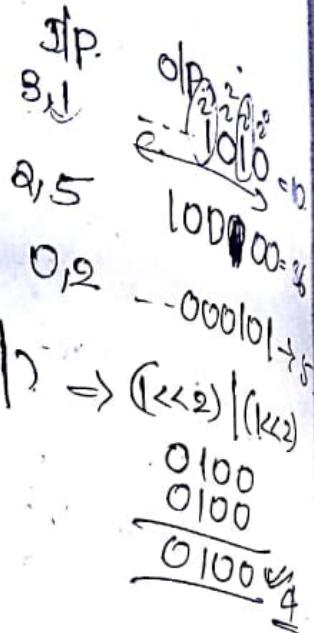
$$\begin{array}{r} 10100 \\ - 100 \\ \hline 1000 \end{array}$$

$$1000 = 8$$

replace it with logical OR ($|$)

setBits(int x, int y)

return $(1 \ll x) | (1 \ll y)$



Q) Check if i^{th} bit is set or not:

bool checkBit (int N, int i)

{

$$\text{int } x = 1 \ll i;$$

$$\text{int ans} = N \& x;$$

If (ans).

return False

else

return True

Bool checkBit (int N, int i)

a) return $(1 \ll i) \& N$.

b) return $(N \gg i) \& 1$

c) return $(N \gg i) \% 2$

①

$$011001 = N \gg 3$$

$$\begin{array}{r} 001000 \\ - 001000 \\ \hline 000000 \end{array}$$

$$\begin{array}{r} 001000 \\ - 001000 \\ \hline 000000 \end{array}$$

25, 3

True.

25, 1

Op

false.

12, 2

True

1100

21st

Op

True.

$$a \& 1 = a \% 2$$

LSB position.

$$(N \gg i) \& 1$$

take i^{th} position
bit last.

$$0011001 \gg 3$$

$$\begin{array}{r} 000001 \\ - 000001 \\ \hline 000001 \end{array}$$

$$\begin{array}{r} 000001 \\ - 000001 \\ \hline 000001 \end{array}$$

set.

(*) x no. of ones, y no. of zeroes. \Rightarrow return int value: $x+y = 2^k$

int create(int x, int y)

{ int sum;

for(i=x+y; i>x; i--)

sum = sum | (1<<(i-1)),

from 3
to 2 iterate 3, 2

IP O/P.

11100 = 28

from 5
to 2 times
iterate

2, 5 1100000

4, 1 11110.

3.

= 15 = 16 - 1

$2^{x+y} - 1$

20. 11

11111 = 31 = 32 - 1

11 ... $x = 2^7 - 1$

11

10

4 3 2 1 0
↓ ↓ ↓ 0 0

1
2⁰

11
2¹
2²

11111

$2^{x+y} = 2^5 = 32$

or no. of ones, y no. of zeroes, int create(int x, int y)

1) $(1 << x) - 1 \leq y$

2) $(1 << y) + ((1 << x) - 1)$

3) $(1 << (x+y)) - (1 << y)$

Count the no. of set bits:-

(*) int BITS(int N) { } $0 \leq N \leq 7$

ans=0

for(int i=0; i<x+y; i++)

ans=ans+(1<<i).

① {

int count=0, i=0;
while(~~i<31~~)
 if((1<<i)&N)

 count++;

 i++;

IP O/P.

10(1010) $\Rightarrow 2$.

25(11001) $\Rightarrow 3$

16(10000) $\Rightarrow 1$.

② {

int c=0;
while(N!=0) {

 if((N&1)==1)

checks if LSB
position is
set

c++;

N=N>>1; get the next
bit in LSB position.

$1 << i$:

will not work for $i=31$.
so go only till 30.

check only

25 \Rightarrow 11001
10 \Rightarrow 00010

4 5
6 7
{ till 2nd bit

8 9
10 11
{ till 3rd bit

16 to 31 \Rightarrow till 4th bit
32 to 63 \Rightarrow till 5th bit

③ int countNoOfSetBits(int N) click only till true bits are set,
 {
 int c=0;
 for(int i=0; i<= $\log_2 N$; i++) no need of clearing beyond
 {
 if (checkBit(N, i)==1) // optimized code.
 c++;
 }
 return c;

⇒ for checking power of 2, if $c=1$, then the given no. is power.

	<u>N</u>	<u>N-1</u>	<u>N & (N-1)</u>	<u>N</u>	<u>N-1</u>	<u>N & N-1</u>
(11001)	25	24	11000	(11001)	25	24
11000	24		(&4)	(1100)	12	11
11000				(10010)	18	17
				(100101)	37	36
				(101010)	42	41
						41(10100)

2) $N=12$. (1100)
 $N-1=11$. (1011)
 $N \& (N-1)=8$. $\underline{\underline{1000}} = 8$.

3) $N=18$: 10010
 $N-1=17$ $\underline{\underline{10001}}$
 $(N \& N-1)$ $\underline{\underline{10000}} = 16$.

4) 37 : 100101
 36 : $\underline{\underline{100100}}$
 100100

5) 42 : 101010
 41 : $\underline{\underline{101001}}$
 101000

Bit. Resets the least significant Bit: $N \& N-1$

int c=0;
 while(N!=0)
 {
 if(N & (N-1))
 c++;
 }

① Compute power.(a, N).

int pow(int a, int N)

{
 int res = 1;

 for (int i=1; i<=N; i++)
 res = res * a;

 return res;

}

Python has no data types &
can do it by dividing the
parts of nw. into string.
but C, C++, Java cannot do this.
long numbers multiplication

()%m → e [0, m-1]

m = 10 + 7 → Prime Number.

To Do: 125 * 125

② Modulo Arithmetic:

$$(a+b)\%m = ((a\%m) + (b\%m)) \%m$$

$$(a+b)\%m = ((a\%m) + (b\%m)) \%m$$

$$(a-b)\%m = ((a\%m) - (b\%m) + m) \%m$$

$$(a/b)\%m = \text{Not distributive}$$

$$(a/b)\%m = (a+b^{-1}) \%m$$

$$= ((a\%m) + (b^{-1}\%m)) \%m$$

Inverse Modulo. Algorithm

2, 30 ✓ int

2, 50 ✓ long.

a^N

3, 2 → 9

4, 3 → 64

5, 2 → 25

10, 3 → 1000.

2, 30 ✓ int a+1

2, 50 ✓ long (Java)

2, 100 X long long (C++).

7, 253 X

125, 372 X BigInteger Java

M should be in the

int range: -2×10^9 to 2×10^9

If M is taken like $10^7 + 7$ or so

(2 * ()) → (10^{34}) doesn't fit into long

$$(15+4)\%7 = 4$$

$$(15\%7) + (4\%7) = 1+4 = 4$$

$$(23+3)\%3 = 0$$

$$(23\%3) + (3\%3) = 2+0=0.$$

$$(11+5)\%7 = 6$$

$$(11\%7) + (5\%7) = 4+5 = 20$$

$$(20\%7)$$

$$\underline{\underline{0}}$$

$$*(a+b)\%m = ? \quad ((a\%m) + (b\%m)) \% m$$

$$\Rightarrow (4+5)\%3 = 7\%3 = \underline{1}$$

$$(2\%3) + (5\%3) = 2+2=4 \rightarrow 4\%3 = \underline{1}$$

$$3) (3+6)\%7 = 9\%7 = \underline{2}$$

$$(3\%7) + (6\%7) = 3+6 = \underline{9} \rightarrow 9\%7 = \underline{2}$$

$$** (a-b)\%m = ((a\%m) - (b\%m) + m) \% m$$

$$(10-3)\%5 = 7\%5 = \underline{2} \checkmark$$

$$(10\%5) - (3\%5) 0-3 = \underline{7} \checkmark \quad 2\%5 = \underline{2}$$

$$(20-13)\%7 = 7$$

$$20\%7 = \underline{3} \quad 13\%7 = \underline{13}$$

$$3-13 = -10 + 17 = \underline{7}$$

$$(31-25)\%7 = 6$$

$$31\%7 = 3 \quad 25\%7 = 4 \quad 3-4 = -1 + 7 = \underline{6} \checkmark$$

If +ve ✓

If -ve \Rightarrow then +m. ✓

$(a-b)\%m$

$$= ((a\%m) - (b\%m) + m)$$

$$*(a/b)\%m$$

$$(20/5)\%7 = 4\%7 = 4$$

$$20\%7 = 6 \quad 5\%7 = 5 \quad \frac{6}{5} = 1.2 \times$$

% cannot be distributed over. ✓

$$*(a/b)\%m = ((a\%m) / (b\%m)) \% m$$

$$(a * a + a * a * a) \% m$$

$$= (((a \% m) + (a \% m)) \% m) * ((a \% m) \% m) * ((a \% m) \% m)$$

$m = 10^9 + 7$ fits in int range.

$$\text{but } \text{ans} = \text{ans} * a \% 10^9 + 7 \\ 10^9 * 10^5 = 10^{14}$$

$$1 \leq a \leq 10^{15}$$

long long int pow(int a, int N)

$$\text{long long int ans} = 1, m = 10^9 + 7, a = a \% m,$$

for (int i = 1; i <= N; i++)

$$\text{ans} = (\text{ans} * a) \% m.$$

return ans;

e.g. int $a = 10^6, b = 10^5$

long long int $c = a * b$.

(10^{11}) ~~10¹¹~~ \rightarrow 10¹¹

$m = 10^9 + 7$ can be rep. as:

1) 1000000007

2) pow(10, 9) + 7

3) 1e9 + 7

int a = new int[10];

for 10⁵ length array.

\rightarrow int a = new int[(int)1e5]

\uparrow
double
typecast.

e.g. $x = 0.000057$

$1e-5 = 10^{-5}$

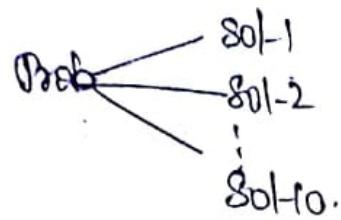
$\rightarrow 57e-6$

\rightarrow by default is double
typecast it to int.
or declare it as int.

int [1e5] X
 \uparrow doubt

int [(int)1e5]

Complexity Analysis of Algorithms



• Time & Space Comp.

ALGORITHMS

A. DFA will depend on the system's ~~int~~ set, there can be different so we count no. of transitions which are always same in all the systems irrespective of no. of induction.

①. solve (part N) {

```
int x=N, y=NH;  
for( int i=0; i<N; i++ )  
{
```

$$x = x + y$$

$$y = y + x;$$

$$x = x + x - y + y + Q + x + y$$

2

INT N=1000;
FOR (?)

108 (INT r=0;

{ =

8.

→ O(1)

3

for (int i=0; i< N; i++)

~~for (j=0; j<N; j++)~~

$$Q_{M8} + = \Psi^g_j + \Psi^{-j}_I$$

$\Rightarrow O(N^2)$

$$\textcircled{1} \text{ and } \textcircled{2} \Rightarrow O(N^2 + N) = O(N^2)$$

$$N + N = \underline{\underline{O(N)}}$$

N is insignificant w.r.t. \underline{N} .

$$\frac{N^2}{10^6} \quad \frac{N}{10^3}$$

$$\frac{10^3}{10^6} \times 100\% = 0.1\%$$

N is 0.1% of N₀.

④ $\text{for } (\text{int } i=0; i < N; i++)$
 $\quad \text{for } (\text{int } j=0; j < i; j++)$
 $\quad \quad \text{ans} = i * j + i - j;$

1 0 1 2 3 ... N
 1 0 1 2 3 ... N

i	j	#it.
0	0-0	0
1	0-1	1
2	0-2	2
3	0-3	3
...
N-1	0 to N-1	N-1

$$\left\{ \begin{array}{l} O(1) \\ (N-1)N \\ \frac{N^2}{2} \\ -O(N^2) \end{array} \right\}$$

⑤ $\text{for } (\text{int } i=0; i < N; i = i + 2)$.

{ $O(1)$. // No matter what is the value of N
 3. $i \neq 0 \Rightarrow i = i + 2 = 0$. all times.
 hence infinite loop.

⑥ $\text{for } (\text{int } i=1; i < N; i = i * 2)$

$N=1 \Rightarrow 1$ time
 $N=2 \Rightarrow 2$ times
 $N=3 \dots$

i	#it.
$2^0 \rightarrow 1$	1
$2^1 \rightarrow 2$	1
$2^2 \rightarrow 4$	1
$2^3 \rightarrow 8$	1
...	...
$2^K = N$	1

$\Rightarrow 2^K$ iterations.

K+1 iteration

$$2^K = N$$

$$K = \log_2 N$$

$$2^K = N-1$$

$$K = \log_2 (N-1)$$

$2^K \Rightarrow K+1$ iterations. $\Rightarrow (\log_2 N + 1)$ iterations
 ignore N.

$$1 \leq 2^K \leq N$$

$$2^K \leq N$$

$$\log_2 2^K \leq \log_2 N$$

$$K \leq \log_2 N$$

$$\text{stop at } K = \log_2 N + 1.$$

$$\underline{\underline{O(\log_2 N)}}$$

1)
Sol 1: $3N^3 + 4N^2 + 100 \rightarrow O(N^3)$. ^{if fast}
Sol 2: $100N^3 + N^2 + N \rightarrow O(N^3)$.
 Don't completely rely on O ,
 solve for exact no. of iterations to find which is better.

2) Sol 1: $10N^3 + 100 \rightarrow O(N^3)$.

Sol 2: $N^2 + 10000 \rightarrow O(N^2)$. ^{Better but not always.}
 Better ~~on~~ after some range of values

If $N=5$,

Sol 1: $1250 + 100 = 1350$ ^{not better}

Sol 2: $100 & 5 \times$

Linear Search:

```
④ bool f(int arr[], int N, int k) {
    for (int i=0; i<N; i++)
        if (arr[i] == k)
            return T;
    return F;
```

If $N=100$,

Sol 1: $10^7 + ()$

Sol 2:

Best Case: $O(1)$

Worst Case: $O(N)$.

Big-O gives the ~~worst case~~ ^{upper bound}

upper bound

* Big-Oh, $O(1)$ puts an upper bound on the time and space complexity of an algorithm based on the input size after a certain threshold.

	<u>1</u>	<u>2</u>	<u>3</u>	<u>total.</u>
1	1K	1to1K	1K	
2	2K	1to2K	2K	
3	3K	1to3K	3K	
				1
N	NK	1toNK	NK	

$$\text{Total} \Rightarrow 1^k + 2^k + 3^k + \dots + N^k$$

$$k=1 \Rightarrow 1+2+3+\dots+N = \frac{N(N+1)}{2} = O\left(\frac{N^2}{2}\right)$$

$$k=2 \Rightarrow 1^2 + 2^2 + 3^2 + \dots + N^2 = \frac{N(N+1)(2N+1)}{6} = O\left(\frac{N^3}{3}\right)$$

$$k=3 \Rightarrow 1^3 + 2^3 + \dots + N^3 = \frac{N^2(N+1)^2}{4} = O\left(\frac{N^6}{4}\right)$$

$$\therefore \Rightarrow O\left(\frac{N^{k+1}}{k+1}\right).$$

00001
00011

00011

0	1	2		
5	15	3.	3	5
2nd	1st	0th	15	15
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	1
1	1	1	1	1

$$\frac{a_8(9)}{x} = \frac{36}{99} \quad \textcircled{A}$$

111
3
3,5
3,5,15
5,15

Space Complexity:-

Solve ($\text{int } N$).

$$\left\{ \begin{array}{l} \text{int } x, y, z; \quad \text{⑫} \\ \text{int. arr[100];} \quad \text{⑯} \\ \text{int. d}_1, d_2, d[10]. \quad \text{⑭} \end{array} \right. \quad \left. \begin{array}{l} \text{int ba}[n] \\ \text{int a, ba}[n] \\ \underline{\text{O}(n)}. \end{array} \right. \quad \left. \begin{array}{l} \text{int a}[n][n] \\ \underline{\text{O}(n^2)} \end{array} \right.$$

$$\text{Space } (2+100+48-1)$$

constant.

$$\text{O}(1).$$

④ No. of Divisors:

$$12: 1, 2, 3, 4, 6, 12.$$

$$36: 1, 2, 3, 4, 6, 9, 12, 18, 36.$$

② Find divisors ($\text{int } N$)

$$\text{int count} = 0;$$

$$\text{for } (i=1; i \leq \sqrt{N}; i++)$$

$$\text{if } (m \% i == 0) \& \& (m = n/i))$$

$$\text{count++;}$$

$$\text{else if } (n \% i == 0)$$

$$\text{count} += 2;$$

else,

$$(GHz \Rightarrow 10^9 \text{ inst/sec.})$$

$$(sec \Rightarrow 1/10^9 \text{ sec})$$

$$N = 10^9$$

$$f_{ter} = 10^9$$

$$\text{inst} = 10^9$$

$$\frac{1}{T \text{ sec.}}$$

$$f_{ter} = f_{inst}$$

optimized

$$N = 10^{18}$$

$$f_{ter} = 10^{18}$$

$$\text{inst} = 10^{18}$$

$$1 \text{ sec.}$$

$$10^{18} = 10^9 \text{ secs}$$

$$= \frac{10^9}{60 \times 60 \times 24 \times 365 \times 365}$$

$$= 3.17 \times 10^{-16}$$

1.6×10^9 instructions/sec.

1 iteration = 1 instruction

$$\log_2 10^3 = \log_2 2^{10} = 10, \quad N = 10^6$$

$$(10^3) = 2^{10}$$

$$1 \text{ instruction} = \frac{1}{10^9} \text{ sec} \\ = 1 \text{ ns}$$

	N^3	$N^2 \log_2 N$	N^2	$N \log_2 N$	N	\sqrt{N}	$\log_2 N$	1	$(N=30)^N$	$(N=60)^N$
Iteration	10^{18}	20×10^{12}	10^{12}	$20 + 10^6$	10^6	10^3	20	1	$2^{30} = 10^9$	$2^{60} = 10^{18}$
Time:	31.7 yr.	5.5 hr.	16.6 min.	0.02 sec. - 20 ms	1 ms	1 μs	20 ns	1 ns	1 sec	31.7 yr

$$10 \cdot \log_2 10^6 = 10^{12} \log_2 (2^{10})^2 \\ = 20(10^{12})$$

$$N^2 \Rightarrow (10^6)^2 = 10^{12}$$

$$\text{Time} = \frac{10^{12}}{10^9} = 1000$$

It's = 1

$$\text{Time} = \frac{\text{It's.}}{10^9}$$

$$\frac{20(10^{12})}{10^9} \\ 20,000 \text{ sec}$$

$$N \log_2 N = 10^6 \log_2 10^6 \\ = 10^6 \log_2 2^{10} \\ = 20(10^6)$$

$$\text{Time} = \frac{20(10^6)}{10^9} =$$

0 1 2 3 4

④ arr: 1 5 -2 10 -6 subset array whose sum is K.

$$K = 16, -7, 12, 18$$

T T F

(HSTO) (1-3-6)

1, 5, -2, 10, -6

bool Subset (int arr[], int N, int K).

{

for (int i=0;

Problems on Complexity Analysis

1. Do

Find the Time Complexities of the following snippets of code:

$O(N+M)$

```
/* Assume that rand() takes constant amount of time */
int a = 0, b = 0;
for (int i = 0; i < N; i++) {
    a = a + rand();
}
for (int j = 0; j < M; ++j) {
    b = b + rand();
}
```

$O(N)$

$O(M)$

$\left. \begin{array}{l} O(N+M) \\ \end{array} \right\}$

if $N > M$
 $\Rightarrow O(N)$
 if $(M > N)$
 $\Rightarrow O(M)$

$O(N^2)$

(B)

```
int a = 0, b = 0;
for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
        a = a + j;
    }
}
for (int k = 0; k < N; k++) {
    b = b + k;
}
```

$\left. \begin{array}{l} O(N^2) \\ O(N) \end{array} \right\}$

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

$O(N^2)$

(C)

```
int a = 0;
for (int i = 0; i < N; i++) {
    for (int j = N; j > i; --j) {
        a = a + i + j;
    }
}
```

i	j	#Iter.
0	N	
1	N-1	
2	N-2	
3	N-3	
...	...	
N-1	1	

$$\begin{aligned} \text{total. No. of Iter.} &= N + N - 1 + \dots + 1 \\ &= \frac{N(N+1)}{2} \\ &= O(N^2) \end{aligned}$$

$O(\log_2 N)$

(D)

```
int a = 0, i = N;
while (i > 0) {
    a += i;
    i /= 2;
}
```

i	#Iter.
N	1
$N/2$	1
$N/2^2$	1
\vdots	
$N/2^K$	1

$$1 \leq 2^K \leq N$$

$$K \leq \log_2 N$$

$2^K \Rightarrow K+1$ iterations.

(K+1) iterations

$(\log_2 N + 1) \propto (\log N)$

$$O\left(\frac{N^{k+1}}{k+1}\right)$$

(13)

```

void fun(int N, int K) {
    for(int i = 1; i <= N; i++) {
        /* Assume that pow() takes constant time */
        int P = pow(i, K);
        for (int j = 1; j <= P; ++j) {
            /* Some constant amount of computation */
        }
    }
}

```

1-times

$\Theta(N)$. (F)

```
int count = 0;  
for(int i = N; i > 0; i /= 2) {  
    for(int j = 0; j < i; j++) {  
        count += 1;  
    }  
}
```

$$\left\{ N + \frac{N}{2} + \frac{N}{2^2} + \dots + \frac{N}{2^k} \right\}$$

$$N + \frac{N}{2} + \frac{N}{2^2} + \dots + \frac{N}{2^k}$$

```
int k = 0;  
for(int i = N/2; i <= N; ++i) {  
    for (int j = 2; j <= N; j =
```

```

    } } (int j = 2, j <= N, j =
        k = k + N/2;
}
}
```

$\frac{9}{2}$	$\frac{9}{2}$	#178.
N	otan	N
$\frac{N}{2}$	otan $\frac{N}{2}$	$\frac{N}{2}$
$\frac{N}{2}$	otan $\frac{N}{2}$	$\frac{N}{2}$
(m)		
$\frac{N}{2}^F$	otan $\frac{N}{2}^F$	$\frac{N}{2}^F$

time * /

$$\begin{aligned}
 & K=1 \Rightarrow \frac{N(N+1)}{2} = O\left(\frac{(N^2)}{2}\right) = O(N^2) \\
 & K=2 \Rightarrow \frac{N(N+1)(2N+1)}{6} = O\left(\frac{N^2(N+1)^2}{4}\right) = O\left(\frac{N^4(N+1)^2}{16}\right) = O(N^4) \\
 & K=3 \Rightarrow \frac{N(N+1)(2N+1)(3N+1)}{4!} = O\left(\frac{N^4(N+1)^3(3N+1)}{24}\right) = O\left(\frac{N^7(N+1)^3}{24}\right) = O(N^7)
 \end{aligned}$$

$$O(N \log N)^{(G)}$$

```
int k = 0;  
for(int i = N/2; i <= N; ++i) {  
    for (int j = 2; j <= N; j =
```

```

    } } (int j = 2, j <= N, j =
        k = k + N/2;
}
}
```

```
int j = 0;  
for(int i = 0; i < N; ++i) {  
    while(j < N && arr[i] <= arr[j])  
        j++;
```

2 { while(j < N && arr[i] <= arr[j])
3 { j++;
4 }
5 }
6 }
7 }

per
doent

the following Time Complexities:

Linear D $\Delta(A) N^{K+G}$

Linear	D	4 (A) N^{x+6}
Logarithmic	F	3 (B) 5^{N+2}
Exponential	B	5 (C) $\Sigma \log$

Exponential B $S(C) \propto \log C$
Polynomial A $I(D) \propto D^2 N$

Polynomial A (D) 3^{2N}
 Linear Logarithmic C (E) $10N$
 Quadratic F (E) $10^3 N$

Quadratic $\mathcal{O}(F)$ 10^3 loc

* 2) { - log N. times

$$\rightarrow O\left(\frac{N}{2} \cdot \log N\right)$$

$$= \underline{\mathcal{O}(N \log N)}$$

Match the following Time Complexities:

- | | | |
|--------------------------|--|-----|
| (1) Linear D | 4 (A) N^{K+G} | (4) |
| (2) Logarithmic F | 3 (B) $5^{N \times 2}$ | (3) |
| (3) Exponential B | 5 (C) $\frac{1}{4} \log_2(\frac{N}{4000})$ | (5) |
| (4) Polynomial A | 1 (D) $3^{28}N + 10^5$ | (1) |
| (5) Linear Logarithmic C | 6 (E) $10N + 9 \frac{N}{100} + 340N^2$ | (6) |
| (6) Quadratic E | 8 (F) $10^3 \log_2(N+3N)$ | (2) |

④ $\text{arr}: \begin{matrix} & \text{index } 0 & 1 & 2 \\ & 5 & 10 & -6 \\ \text{and} & & 1^{\text{st}} & 2^{\text{nd}} \text{ bit} \\ \emptyset \Leftarrow 0 & 0 & 0 \end{matrix}$

No. of subsets = 2^N

$\dots \quad \begin{matrix} & 0 & 0 & 1 \\ & 0 & 1 & 0 \\ 5, 10 \Leftarrow 0 & 1 & 1 \\ 10 \Leftarrow 1 & 0 & 0 \\ 5-6 \Leftarrow 1 & 0 & 1 \\ 10, -6 \Leftarrow 1 & 1 & 0 \\ 5, 10, -6 \Leftarrow 1 & 1 & 1 \end{matrix}$

- ① Data Types
- ② Time Out

bool subset(int arr[], int n, int k)

```

    {
        for (int i=0; i<pow(2, n); i++)
        {
            if (checkBit(arr, i))
                sum += arr[i];
            if (sum == k)
                return T;
        }
    }

```

checkBit: $(i \ll j) \& N$

⑤ Time C. Space C.

```

    bool subset(int arr[], int n, int k)
    {
        for (int i=0; i<(1<<n); i++)
        {
            int sum=0;
            for (int j=0; j<n; j++)
            {
                if (checkBit(i, j))
                    sum = sum + arr[j];
            }
            if (sum == k)
                return T;
        }
    }

```

$O(2^N * N), O(1)$

```

int pow(int a, int N) {
    int m = 1e9 + 7;
    ans = (ans * a) % M;
}

```

$$1 \leq N \leq 10^9$$

$$1 < T \leq 10^3$$

HR, CC, CF, IB.....

Space

int arr[10⁷];

$$\text{Space: } 10^7 \times 4 \times 10 \times 2^{\text{space}} = 40 \text{ MB} \quad \text{OK}$$

long arr[10⁴];

$$8 \times 10^4 = 80 \text{ MB} \quad \text{OK}$$

bool arr[10⁸];

$$1 \times 10^8 = 100 \cdot 10^6 = 100 \cdot 2^{\text{space}} = 100 \text{ MB}$$

but if we have a $N \times N$ matrix
with $1 \leq N \leq 10^5$ & $1 \leq T \leq 100$

Read T
loop T times.

int[N][N].

T.C. = $O(TN^2)$.

S.C. = $O(N^2)$.

$$= 10^5 \cdot 10^5$$

$$= 10^{10}$$

$$\rightarrow 4 \cdot 10 \cdot 2^{\text{space}}$$

$$\rightarrow 40 \text{ GB}$$

Types of Errors:

- Compilation Error

- Runtime Error (Memory Limit) ~~not OK~~

- Time Limit Exceed \rightarrow Inifite Loop.

- Wrong Answer. \rightarrow Slow Input/Output

$O(T \times M) \leq 10^8$ see the
code to work
correctly.

$$\text{If } M=N \Rightarrow O(T \cdot N) = O(10^3 \cdot 10^3)$$

$$= O(10^{12})$$

$$\rightarrow \frac{10^{12}}{10^9} = 1000 \text{ sec}$$

$$= 16.6 \text{ min}$$

~~X~~ will give TLE

$$\text{If } M=N^2 \Rightarrow O(TN^2) = O(10^3 \cdot 10^6)$$

$$\text{time} = \frac{10^{21}}{10^9} = (\text{ })$$
~~X~~ TLE

$$\text{If } M=\log N \Rightarrow O(T \log N)$$

$$O(10^3 \cdot \log_2 10^9) = O(10^3 \log_2 2^{\text{space}})$$

$$= 20 \cdot 10^3$$

~~Time = $20 \cdot 10^3$ h~~

Space

$$4 \text{ B} \cdot 10^{10} = 4 \text{ B} (2^{\text{space}}) \cdot 10$$

$$10^3 \text{ B} = 1 \text{ KB} = 4096 \text{ B}$$

$$10^6 \text{ B} = 1 \text{ MB}$$

$$10^9 \text{ B} = 1 \text{ GB}$$

~~Not OK~~

Time Complexity $O(T \times M) \leq 10^8$

Space Complexity $O(S) \leq 10^{6/7/8}$

$$\rightarrow O(S) \leq 10^8 \text{ B}$$

JAVA: Scanner — slow I/O.

Buffered Reader / Writer — fast I/O.

CE — Line no.

TLE → Infinite Loop

TLE → slow I/O.

TLE → $\frac{TC}{LC} \leq 10^9$ should be.

RTE → Divide By 0.

RTE → Index Out of Bound. (check $a[i], a[i+1], \dots$)

RTE → Null Pointer Exceptions

clctt.

cin, cout — slow

scanf, printf — fast

WA → Data Types

WA → Edge Cases → Find a test case where code doesn't work.

use custom inputs.

Expected op. (cal.)

Actual op. (By your code)

objects.

pointers.

Linked List

TLBS.

PGs

NZEC

Space limit < 10⁹ B

Segmentation Fault

Stack Overflow (Recursion)

* int pow(int a, int N).

$$a^3 = a^{x_1} \cdot a^{x_2} \cdot a^{x_3} \cdots a^{x_m}$$

$$\Rightarrow x_1 + x_2 + \cdots + x_m = N, x_i \in [1, 2, 4, 8, 16, 32, \dots], x_i \neq x_j$$

$$a^3 = a^4 \cdot a^4 \cdot a^1 \times \text{Not unique} \times$$

$$B = \begin{pmatrix} 3 & 2 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \quad a^3 = a^1 \cdot a^4 \cdot a^8 \Leftrightarrow \text{pair. of } 2 \times \text{each } x_i \text{ is unique.}$$

$$a^{x+y} = a^x \cdot a^y$$

$$28 \left(\begin{smallmatrix} 4 & 3 & 2 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{smallmatrix} \right) \quad a^8 = a^4 \cdot a^4$$

$$a^N = a^1, a^2, a^4, a^8, a^{16}, a^{32}, a^{64}, \dots$$

$$B \left(\begin{smallmatrix} 4 & 3 & 2 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{smallmatrix} \right)$$

4th bit.

0th bit - set

1st bit - unset

2nd bit - set

3rd bit - set

4th bit - unset

a initial

ans initial.

$x = x \cdot x$

$x = a,$

$ans = \{ \}$

only when set

$ans = ans \cdot x$

update

a^2

a^4

a^8

a^{16}

a^{32}

a^{64}

a^{128}

a^{256}

a^{512}

a^{1024}

a^{2048}

a^{4096}

a^{8192}

a^{16384}

a^{32768}

a^{65536}

a^{131072}

a^{262144}

a^{524288}

$a^{1048576}$

$a^{2097152}$

$a^{4194304}$

$a^{8388608}$

$a^{16777216}$

$a^{33554432}$

$a^{67108864}$

$a^{134217728}$

$a^{268435456}$

$a^{536870912}$

$a^{1073741824}$

$a^{2147483648}$

$a^{4294967296}$

$a^{8589934592}$

$a^{17179869184}$

$a^{34359738368}$

$a^{68719476736}$

$a^{137438953472}$

$a^{274877906944}$

$a^{549755813888}$

$a^{1099511627776}$

$a^{219902325552}$

$a^{439804651104}$

$a^{879609302208}$

$a^{1759218604416}$

$a^{3518437208832}$

$a^{7036874417664}$

$a^{14073748835328}$

$a^{28147497670656}$

$a^{56294995341312}$

$a^{112589990682624}$

$a^{225179981365248}$

$a^{450359962730496}$

$a^{900719925460992}$

$a^{1801439850921984}$

$a^{3602879701843968}$

$a^{7205759403687936}$

$a^{14411518807375872}$

$a^{28823037614751744}$

$a^{57646075229503488}$

$a^{115292150459006976}$

$a^{230584300918013952}$

$a^{461168601836027904}$

$a^{922337203672055808}$

$a^{1844674407344111616}$

$a^{3689348814688223232}$

$a^{7378697629376446464}$

$a^{14757395258752892928}$

$a^{29514790517505785856}$

$a^{59029581035011571712}$

$a^{11805916207002314344}$

$a^{23611832414004628688}$

$a^{47223664828009257376}$

$a^{94447329656018514752}$

$a^{18889465931203702904}$

$a^{37778931862407405808}$

$a^{75557863724814811616}$

$a^{151115727449629623232}$

$a^{302231454899259246464}$

$a^{604462909798518492928}$

$a^{1208925819597036985856}$

$a^{2417851639194073971712}$

$a^{4835703278388147943424}$

$a^{9671406556776295886848}$

$a^{19342813113552591773696}$

$a^{38685626227055183547392}$

$a^{77371252454110367094784}$

$a^{154742504908220734189568}$

$a^{309485009816441468379136}$

$a^{618970019632882936758272}$

$a^{1237940039265765873516544}$

$a^{2475880078531531747033088}$

$a^{4951760157063063494066176}$

$a^{9903520314126126988132352}$

$a^{19807040628252453976264704}$

$a^{39614081256504907952529408}$

$a^{79228162513009815905058816}$

$a^{158456325226019631810117632}$

$a^{316912650452039263620235264}$

$a^{633825300904078527240470528}$

$a^{1267650601808157054480941056}$

$a^{2535301203616314108961882112}$

$a^{5070602407232628217923764224}$

$a^{10141204814465256435847528448}$

$a^{20282409628930512871695056896}$

$a^{40564819257861025743390113792}$

$a^{81129638515722051486780227584}$

$a^{162259277031440102935560455168}$

$a^{324518554062880205871120910336}$

$a^{649037108125760411742241820672}$

$a^{1298074216255208823484483641344}$

$a^{2596148432510417646968967282688}$

$a^{5192296865020835293937934565376}$

$a^{10384593730441670587875889130752}$

$a^{20769187460883341175751778261504}$

$a^{41538374921766682351503556523008}$

$a^{83076749843533364703007113046016}$

$a^{16615349768706672940601422609232}$

$a^{33230699537413345881202845218464}$

$a^{66461399074826691762405690436928}$

$a^{132922798149653383524811380873856}$

$a^{265845596299306767049622761747712}$

$a^{531691192598613534099245523495424}$

$a^{106338238519722706819849104698848}$

$a^{212676477039445413639698209397696}$

$a^{425352954078890827279396418795392}$

$a^{850705908157781654558792837590784}$

$a^{170141181631556327911785665518168}$

$a^{340282363263112655823571331036336}$

$a^{680564726526225311647142662072672}$

$a^{136112945305245062329428532414544}$

$a^{272225890610490124658857064829088}$

$a^{544451781220980249317714129658176}$

$a^{108890356244196049863542825931632}$

$a^{217780712488392099727085651863264}$

$a^{435561424976784199454171303726528}$

$a^{871122849953568398908342607453056}$

$a^{174224569990713679781668521490612}$

$a^{348449139981427359563337042981224}$

$a^{696898279962854719126674085962448}$

$a^{139379655992570943825334817192496}$

$a^{278759311985141887650669634384992}$

$a^{55751862397028377530133926876992}$

$a^{111503724794056755060267853753984}$

$a^{223007449588113510120535707507968}$

$a^{446014899176227020241071415015936}$

$a^{892029798352454040482142830031872}$

$a^{1784059596704908080964285660063744}$

int pow(int a, int N)

$$x = a$$

if $i \leq \log_2 N$
ans = x^i
 $x = x * x$.

if (checkBit(N, i)).

$$\text{ans} = \text{ans} * x;$$

if;

int pow(int a, int N) {

$$\text{int } x = a, \text{ans} = 1;$$

for (int i=0; $i < 32$; i++)

if (checkBit(N, i))

$$\text{ans} = (\text{ans} * x) \% 1e9 + 7$$

$$x = (x * x) \% 1e9 + 7$$

return ans.

Time.C + $\frac{TC}{N}, 1$. If $N = 10^{18} \Rightarrow 10^9 \text{ sec}$
 $= 31.7 \text{ yrs.}$

a) $\log N, 1 \Rightarrow \text{if } N = 10^{18}$

$$\log_2 N = \log_2 10^{18} = 60 \text{ iterations}$$

$$\Rightarrow \text{Time} = \frac{60}{10^9} = \underline{\underline{60 \text{ ns}}}$$

Q1: 1 5 2 10 6 5 8 -6 8 1 2

Find the no. which comes only once.

1: 001

5: 101

2: 010

1) Take each element
compose with all others:

$$a1b1a = b.$$

$$a1b1c1b1a = c.$$

TC SC

2) XOR op. of all ele. $\Rightarrow N, 1$

TC SC

$N^2, 1$

Bonita force.

$a[N] = 1, 5, 2, \textcircled{10}, -6, 5, 8, -6, 8, 1, 2$.

✓ ~~Q1.~~ number Only Once ($\text{int}[] a, \text{int} n$)

```

    {
        int flag
        for (int i=0; i<N; i++)
        {
            if (flag == 0; i=0)
                for (int j=i+1; j<N; j++)
                {
                    if (a[i] == a[j])
                        flag = 1;
                    if (flag == 0)
                        return a[i];
                }
        }
    }

```

$$\begin{array}{r} 000 \\ 010 \\ \hline 010 \end{array}$$

$O(n^2) = \textcircled{2}$

	$\begin{array}{r} 010 \\ 011 \\ \hline 001a \end{array}$
	$\begin{array}{r} 100 \\ 100 \\ \hline 00 \end{array}$
	$\begin{array}{r} 101 \\ 101 \\ \hline 01 \end{array}$
	$\begin{array}{r} 111 \\ 110 \\ \hline 011 \end{array}$
	$\begin{array}{r} 100 \\ 010 \\ \hline 110 \end{array}$
	$\begin{array}{r} 101 \\ 101 \\ \hline 01 \end{array}$
	$\begin{array}{r} 111 \\ 011 \\ \hline 110 \end{array}$

Q2. Code:-

~~Ques~~ ans = 0

```

for (int i=0; i<N; i++)
{
    for (int j=0; j<N; j++)
    {
        ans = ans ∧ (a[i] + a[j]);
    }
}

```

$O(N^2)$, ~~Q1~~ reduce the complexity.

$a[] = a \ b \ c$

$$\Rightarrow (a+a) \wedge (a+b) \wedge (a+c) \wedge (b+a) \wedge (b+b) \wedge (b+c) \wedge (c+a) \wedge (c+b)$$

$$\Rightarrow (a+a) \wedge (b+b) \wedge (c+c).$$

Optimized :-

$ans = 0$

$\text{for (int } i=0; i < N; i++)$

$ans = ans \wedge (a[i] + a[i]);$

$\Rightarrow O(N), O(1)$

Ques 2

ans = 0

for (int i=0; i< N; i++)

 for (int j=0; j< N; j++)

 ans = ans + (a[i] & a[j])

a[N] = a b c d. $\alpha(a \& b)$

$\Rightarrow \underbrace{(a \& a)}_{\alpha} + \underbrace{(a \& b)}_{\alpha} + \underbrace{(a \& c)}_{\alpha} + \underbrace{(a \& d)}_{\alpha} + \underbrace{(b \& a)}_{\alpha} + \underbrace{(b \& b)}_{\alpha} + \underbrace{(b \& c)}_{\alpha} + \underbrace{(b \& d)}_{\alpha} + \underbrace{(c \& a)}_{\alpha} + \underbrace{(c \& b)}_{\alpha} + \underbrace{(c \& c)}_{\alpha} + \underbrace{(c \& d)}_{\alpha} + \underbrace{(d \& a)}_{\alpha} + \underbrace{(d \& b)}_{\alpha} + \underbrace{(d \& c)}_{\alpha} + \underbrace{(d \& d)}_{\alpha}$

$\Rightarrow \alpha \left[\underbrace{(a \& b) + (a \& c) + (a \& d)}_{\text{for } a \Rightarrow b, c, d} + \underbrace{(b \& c) + (b \& d)}_{\text{for } b \Rightarrow c, d} + \underbrace{(c \& d)}_{\text{for } c \Rightarrow d} \right]$

ans = 0.

for (int i=0; i< N; i++)

 for (int j=i+1; j< N; j++)

 ans = ans + (a[i] & a[j])

return ans;

i
f 2

3.

i
0 j
N
1 N-1
2 N-2
3 N-3
N 0

$$\begin{array}{r}
 110 \\
 100 \\
 1010 \\
 1100 \\
 \hline
 \end{array}$$

~~(N-1) + 10 1100 = 12 ✓~~

$= \frac{N(N-1)}{2} = \frac{(M)(N-1)}{2}$

$= O(N^2)$.

ans = 0.

5
(6)

*2
-12 ✓

$$\begin{array}{r}
 000 \\
 011 \\
 \hline
 011 \\
 \hline
 010 \\
 \hline
 001 \\
 \hline
 010 \\
 \hline
 010 \\
 \hline
 001 \\
 \hline
 \end{array}$$

i
j
2
3

$$\begin{array}{r}
 a=2 \quad 010 \quad b=3 \quad 011 \\
 b=3 \quad 011 \quad a=2 \quad 010 \\
 \hline
 001 \\
 \hline
 001 \\
 \end{array}$$

$\alpha(a \& b)$

$$\Rightarrow \alpha: a \quad b \quad c$$

$$\Rightarrow (a_1a) + (a_1b) + (a_1c) + 2(a_1b) + 2(a_1c) + 2(b_1a) + 2(b_1c) + 2(c_1a) + 2(c_1b) + 2(c_1c)$$

Optimized.

ans = 0

for (int i=0; i<N; i++)

 for (int j=0; j<N; j++)

 ans = ans + 2 * (arr[i] + arr[j])

i	j	ans
0	[1, N)	121
1	[2, N)	N-2
2	[3, N)	N-3
...
N-1	[N, N)	0

$$\Rightarrow \text{ans} = a_1 + a_2 + a_3 + a_4$$

final = 2 * (ans)

(25), (10), (42), (20)

16+8+1, 8+2, 32+8+2, 16+4

(N)+...+0.

(N)(N) $\frac{N}{2}$ $\frac{N(N)}{2}$

O(N²)

$$\Rightarrow 1(1) + 2(2) + 1(4) + 3(8) + 2(16) + 1(32)$$

32 16 8 4 2 1

\Rightarrow How to find No. of 1's, 2's, 4's, 8's, 16's, 32's
in a no. 25: 011001
10: 001010

$a \wedge b \Rightarrow 1$ if means

either a & b bits are set.

42: 101000

20: 010100

for (j=j=0; j<N; j++)

{ int c=0, j=0;
while (j < log₂N)

{ if (checkbit(N, j))

a: a₁ a₂ a₃ a₄ a₅ a₆ [a₁, a₃ have 1 in 3rd bit]
x x x x x x (8).



a₁ 1. any of a₂, a₄, a₅, a₆ will give 1 in 5th bit

a₃ 1. any of a₂, a₄, a₅, a₆ will give 1 in 5th bit set.

$$((a \times 4) \times 8) \times 2$$

$$(((a \times (N-4)) \times 2) \times 2)$$

$\text{if } g=0;$
while ($g \leq \log_2 N$)

2 C = for(all ele, or, N, 9).

$\text{ans} = \text{ans} + c * (\text{N} - c) * (1 \ll i) * 2$

$$\begin{array}{r}
 135 \\
 201 \\
 911 \\
 101 \\
 \hline
 3(1) + 1(2) + 1 = 9
 \end{array}$$

$$\begin{array}{r} \cancel{\text{quadratic}} \\ \text{ans} = 0. \end{array}$$

for $i = 0$ to 32.

c = for(a, N, i)

$$ans = ans + c * (N - c) * (1 \ll i) * \alpha.$$

$$\begin{array}{r} 0.10 \\ \underline{\times 001} \\ \hline 0.0001 \end{array}$$

$$O(32 \cdot N) = O(N)$$

$$10 \text{ sets} \cdot 10^{13} = 10^{14} \text{ bits per second}$$

* Triple Trouble:-

~~1st b/t X ✓ 2nd S Q~~

1st bit set: $\left(\frac{1}{N} \right) \left(\frac{2}{N} \right) \left(\frac{2}{N} \right) \dots \left(\frac{10}{N} \right) \dots \left(\frac{N}{N} \right)$

Print only once (int J a, m)
{ for(int i=0; i<N; i++) }

{ for (q=0; q<N; q++)

{ $\text{if } (\alpha[i] == \alpha[j])$

C++;

if (c == 1)

return after;

3

for i=0 to 32

C = for (0, N, 9)

$$\text{ans} = \text{ans} + ((\text{c}\%3) * (\text{L} \times \text{i}))$$

$\mathcal{O}(N^2), \mathcal{O}(1)$. st. ↓
 \downarrow

10:10'10

⑨ N.I.

~~yes = 0~~
~~no = 1~~

3 \approx (Ch 31)
setbit_1

1-to-1 \Rightarrow 1-to-6

* QW: 1 5 6 5 4 2 → rep=5, missing 3.

$\text{Age}_D = 1 \ 4 \ 4 \ 2 \Rightarrow 4 \text{ repeated, missing} \Rightarrow 3.$

- Array from 1 to N elements, find 1 repeated no., 1 missing no.

1) Brute force.

for $i = 1 \text{ to } N$.

count(9) $\xrightarrow{2 \rightarrow}$ repeating.

```
for( i=1 ; i<N ; i++ )
```

```
{   int c=0;
    for(int j=0; j<N; j++)
```

{ if ($i == \text{alg}[j]$)

C++;

$\{$ if ($c == 0$) return $a[?]$ - missing.

If ($c == 2$) return arr - repeating.

1 : 0001

4:0100

4: 0100

2: 0010

0011 ⇒ 3^o.

..... miss

Recursion

$$\text{sum}(N) = 1+2+3+\dots+N.$$

$$= s(N-1) + N$$

$$s(N) = s(N-1) + N.$$

/ taking the smaller instances of the problem.

④ int sum(int N)

```

    {
        if (N==1)
            return 1;
        else
            return sum(N-1) + N;
    }

```

$\begin{array}{rcl} & & H2+3 \\ & \nearrow & \downarrow \\ s(3) & = & 6 \\ & \downarrow & \downarrow \\ s(2)+3 & = & 6 \\ & \downarrow & \downarrow \\ s(1)+2 & = & 3 \\ & \downarrow & \downarrow \\ & & 3 \end{array}$

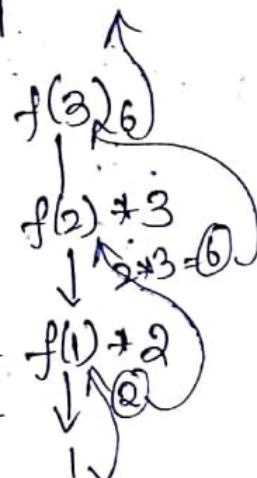
④ int fact(int N)

$$3 = 3 * 2 * 1$$

```

    {
        if (N==0)
            return 1;
        return fact(N-1) * N;
    }

```



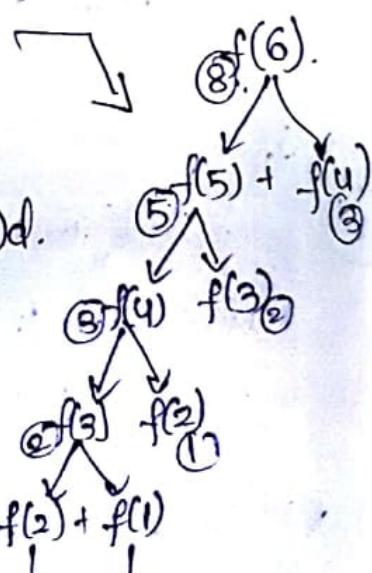
④ int fib(int N)

$$1, 1, 2, 3, 5, 8, \dots$$

```

    {
        if ((N==1) || (N==2))
            return 1;
        return fib(N-1) + fib(N-2);
    }

```



APSum: $a=1, d=2$
 $1, 3, 5, 7, 9, \dots$

$$t_n = a + (n-1)d.$$

Recursion

① Assumption

② Main Logic

③ Base Condition

int APSum(int a, int d, int N) $a, a+d, a+2d, \dots$

{

 if ($N==0$). return 0; $\Rightarrow O(1)$

 return $a+(N-1)*d + \text{APSum}(a, d, N-1); \Rightarrow T(N)$

$\boxed{T(N)=T(N-1)+1} \Rightarrow O(N)$

Complexity.

* int sum(int N)

{

 if ($N==0$) return 0; $\Rightarrow O(1)$

 return sum($N-1$) + N; $\Rightarrow T(N-1)$

{

$\boxed{T(N)=T(N-1)+1}$ Recurrence Relation

$$\begin{aligned} &= (T(N-2)+1)+1 = T(N-2)+2 \\ &= (T(N-3)+1)+1+1 = T(N-3)+3 \\ &\quad \vdots \\ &= T(N-K)+K \end{aligned}$$

$$T(0)=1$$

$$T(N-K)=T(0)=1$$

when $N-K=0$.

$$\underline{N=K}$$

$$T(0)+N$$

$$= 1+N$$

$$= \underline{\Theta(N)}$$

$$\boxed{O(N)}$$

Complexity.

* int fact(int N)

{

 if ($n==0$) return 1; $\Rightarrow O(1)$

 return $N \cdot \text{fact}(N-1); \Rightarrow$

const.

Recurrence

$\boxed{\text{Relation } T(N)=T(N-1)+1}$

$\Rightarrow O(N)$ time comp.

② Fibonacci:

int fib(int N)

{ if ($(N==1) \parallel (N==2)$) return 1;

$\Rightarrow O(1)$

return fib(N-1) + fib(N-2);

$\Rightarrow T(N-1) + T(N-2)$

3.

$$\boxed{T(N) = T(N-1) + T(N-2) + 1}$$

$$\begin{aligned} T(1) &= 1 \\ T(2) &= 1 \end{aligned}$$

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

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

$$= T(N-2) + 2T(N-3) + T(N-4) + 3$$

$$= (T(N-3) + T(N-4) + 1) + 2(T(N-4) + T(N-5) + 1) + (f(N-5) + f(N-6)) + 1 + 3$$

$$= T(N-3) + 3T(N-4) + 2T(N-5) + T(N-6) + 7$$

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

$$T(N) = T(N-1) + T(N-2) + 1 \Rightarrow T(N-1) = T(N-2) + T(N-3) + 1$$

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

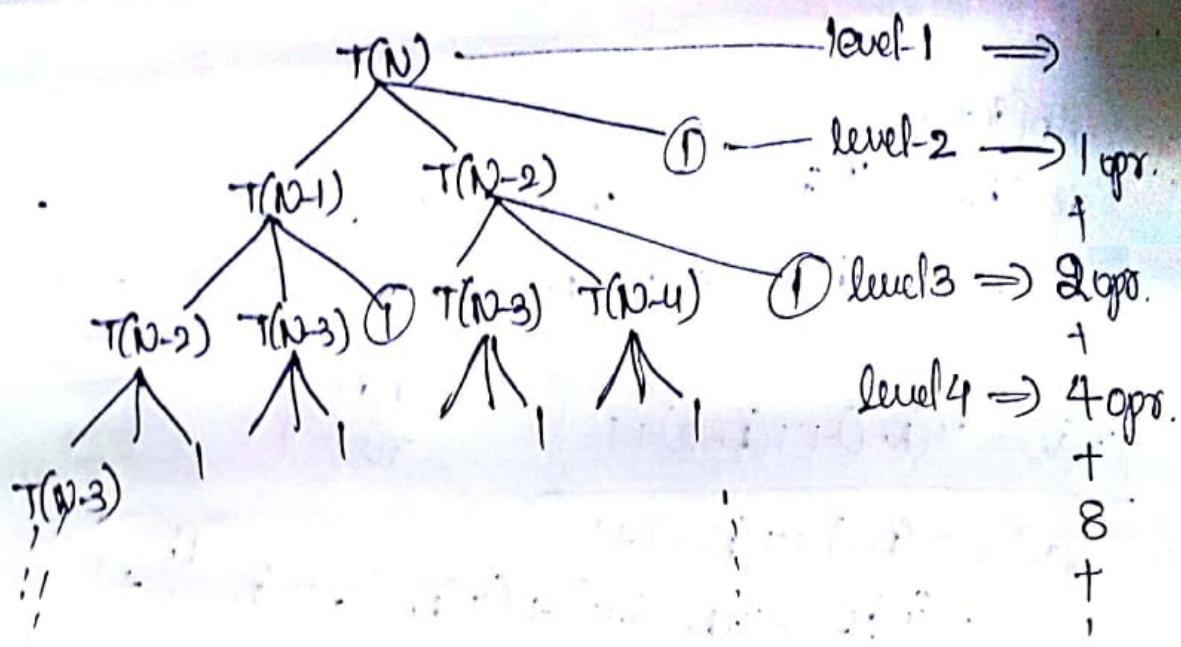
$$\rightarrow \text{if } N-K=2, \quad N=K+2 \Rightarrow \underline{K=N-2}$$

$$\begin{aligned} N-(N-2+1) \\ N-N+1. \\ \textcircled{1}. \end{aligned}$$

$$= T(2) + (N-2) T(1) + (N-3) T(0) + \dots + 2^{N-2-1} T(1) - (N-2+1)^2$$

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

$$\underline{\underline{O(2^N)}} \quad \Bigg\}$$



$$\begin{aligned}
 & T(1) \\
 & = T(N-(N-2)) \\
 & \quad T(N-1) \Rightarrow 1 \text{ opr} = 2^{0+1} \\
 & \quad T(N-2) \Rightarrow 2 \text{ opr} = 2^{1+1} \\
 & \quad T(N-3) \Rightarrow 2^2 \text{ opr.} = 2^{2+1}
 \end{aligned}$$

level n

$$2^n$$

$$1 + 2 + 4 + 8 + \dots + 2^n$$

$N-N+2$

$$T(N-(N-2)) = 2^{N-2} \text{ opr.}$$

$$\begin{aligned}
 & \frac{2^n}{2} (2(1) + (n-1)^2) \\
 & = 2^{n-1} + 2^{n-1}(n-1)
 \end{aligned}$$

Total: $1 + 2 + 4 + 8 + \dots + 2^{N-3}$

$$= 2^0 + 2^1 + 2^2 + 2^3 + \dots + 2^{N-3}$$

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

$$= 2^{N-1} - 1$$

$\underline{k=N-3}$

$$= 2^{N-2} - 1$$

$$= \frac{2^N}{4} - 1$$

$\Rightarrow \underline{\underline{O(2^N)}} \text{ Time Complexity}$

$$\textcircled{1}. T(N) = \alpha T\left(\frac{N}{2}\right) + 1 \Rightarrow O(N)$$

$$\textcircled{2}. T(N) = \alpha T\left(\frac{N}{2}\right) + N \Rightarrow O(N \log_2 N)$$

$$\textcircled{3}. T(N) = T\left(\frac{N}{2}\right) + 1 \Rightarrow O(\log_2 N)$$

$$\textcircled{4}. T(N) = T\left(\frac{N}{2}\right) + N \Rightarrow O(N)$$

$$\textcircled{5}. T(N) = T\left(\frac{N}{2}\right) + T\left(\frac{N}{4}\right) + N^2 \Rightarrow \underline{\underline{O(N^2)}}$$

$$a^{\log_a b} = b \quad \text{ToDo}$$

$$\log(a^{\log_a b}) = \log(b)$$

$$\log_a b \cdot \log_a c = \log_b c$$

$$\log_b^b = \log_b c$$

$$\textcircled{1} \boxed{T(N) = \alpha T\left(\frac{N}{2}\right) + 1} \Rightarrow O(N)$$

$$= \alpha \left(\alpha \cdot T\left(\frac{N}{2^2}\right) + 1 \right) + 1 = \alpha^2 \cdot T\left(\frac{N}{2^2}\right) + \alpha + 1 = \alpha^2 \cdot T\left(\frac{N}{2^3}\right) + \alpha^2 + \alpha + 1$$

$$= \alpha^K \cdot T\left(\frac{N}{2^K}\right) + (\alpha^K + \alpha^{K-1} + \dots + 1) \\ = \alpha^K \cdot T\left(\frac{N}{2^K}\right) + \frac{1(\alpha^K - 1)}{(2-1)} = \alpha^K \cdot T\left(\frac{N}{2^K}\right) + \alpha^K$$

$$\begin{aligned} N = 2^K \\ \log_2 N = \log_2 2^K \\ \frac{N}{2^K} = 1 \Rightarrow N = 2^K \\ K = \log_2 N \end{aligned} \Rightarrow \begin{aligned} N \cdot T(1) + N - 1 \\ \Rightarrow \boxed{O(N)}. \end{aligned}$$

$$\textcircled{2}. \boxed{T(N) = \alpha T\left(\frac{N}{2}\right) + N} \Rightarrow O(N \log N)$$

$$= \alpha \left(\alpha \cdot T\left(\frac{N}{2^2}\right) + \frac{N}{2} \right) + N = \alpha^2 \cdot T\left(\frac{N}{2^2}\right) + \alpha N + N$$

$$= \alpha^2 \cdot \left(\alpha \cdot T\left(\frac{N}{2^3}\right) + \frac{N}{2^2} \right) + N + N = \alpha^3 \cdot T\left(\frac{N}{2^3}\right) + 3N$$

$$= \alpha^K \cdot T\left(\frac{N}{2^K}\right) + KN.$$

$$= N \cdot T(1) + (\log_2 N) N = N + N \log_2 N$$

$$\rightarrow \boxed{O(N \log_2 N)}$$

$$\textcircled{3}. \boxed{T(N) = T\left(\frac{N}{2}\right) + 1} \Rightarrow O(\log N).$$

$$T(N) = \left(T\left(\frac{N}{2}\right) + 1\right) + 1 = T\left(\frac{N}{2^2}\right) + 2$$

$$= \left(T\left(\frac{N}{2^3}\right) + 1\right) + 2 = T\left(\frac{N}{2^3}\right) + 3.$$

$$= T\left(\frac{N}{2^K}\right) + K.$$

$$\frac{N}{2^K} = 1 \quad = T(1) + \log_2 N.$$

$$W = 2^K$$

$$K = \log_2 N$$

$$\Rightarrow \boxed{O(\log N)}$$

$$\textcircled{4}. \quad T(N) = T\left(\frac{N}{2}\right) + N$$

$$= \left(T\left(\frac{N}{2^2}\right) + \frac{N}{2}\right) + N = T\left(\frac{N}{2^2}\right) + \frac{N}{2} + N$$

$$= \left(T\left(\frac{N}{2^3}\right) + \frac{N}{2^2}\right) + \frac{N}{2} + N = T\left(\frac{N}{2^3}\right) + \frac{N}{2^2} + \frac{N}{2} + N$$

$$T(N) = T\left(\frac{N}{2}\right) + N$$

$$= T\left(\frac{N}{4}\right) + \frac{N}{2}$$

$$= T\left(\frac{N}{4}\right) + \frac{3N}{2}$$

$$= T\left(\frac{N}{8}\right) + \frac{7N}{4}$$

$$= T\left(\frac{N}{16}\right) + \frac{15N}{8}$$

$$= T\left(\frac{N}{32}\right) + \frac{(2^K - 1)}{2^K} * N$$

$$\begin{cases} N = 2^K \\ K = \log_2 N \end{cases}$$

$$= T(1) + \frac{(N-1)}{2^{K-1}} * N$$

$$= 1 + \frac{(N-1)2}{N} * N$$

$$= 1 + 2(N-1) \\ = 1 + 2N - 2 \\ = N(N-1)$$

$$= T\left(\frac{N}{2^K}\right) + \frac{N}{2^{K-1}} + \frac{N}{2^{K-2}} + \dots + \frac{N}{2^0}$$

$$= T\left(\frac{N}{2^K}\right) + N \left[1 + \frac{1}{2} + \frac{1}{2^2} + \dots + \frac{1}{2^{K-2}} + \frac{1}{2^{K-1}} \right]$$

$$= T\left(\frac{N}{2^K}\right) + N \left[\frac{1 - (1/2)^K}{1 - 1/2} \right]$$

$$= T\left(\frac{N}{2^K}\right) + 2N \left(1 - \frac{1}{2^K}\right)$$

$$= T(1) + 2N \left(1 - \frac{1}{N}\right)$$

$$= 1 + 2N - 2$$

$$= 2N - 1 \\ \Rightarrow \boxed{O(N)}$$

$$\begin{matrix} a = 1 \\ r = 1/2 \end{matrix}$$

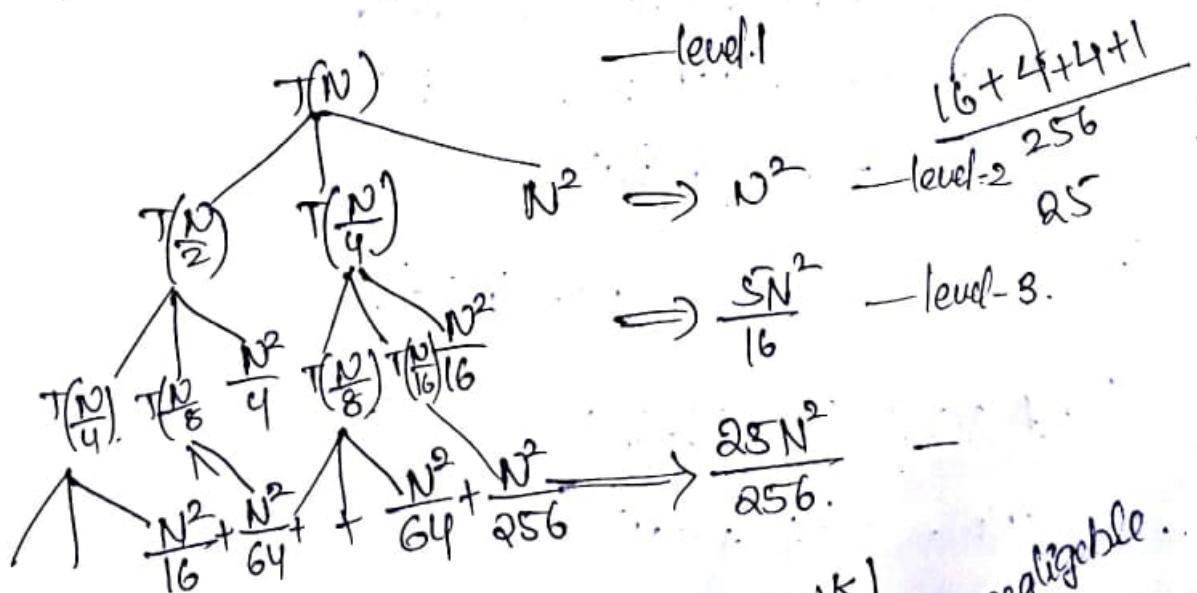
$$\begin{aligned}
 5. \quad T(N) &= T\left(\frac{N}{2}\right) + T\left(\frac{N}{4}\right) + N^2 \Rightarrow O(N^2) \\
 &= \left[T\left(\frac{N}{2^2}\right) + T\left(\frac{N}{2^4}\right) + \left(\frac{N}{2}\right)^2 \right] + \left[T\left(\frac{N}{2^3}\right) + T\left(\frac{N}{2^6}\right) + \left(\frac{N}{4}\right)^2 \right] \\
 &= T\left(\frac{N}{2^3}\right) + 2 \cdot T\left(\frac{N}{2^3}\right) + T\left(\frac{N}{2^6}\right) + \left(\frac{N}{2}\right)^2 + \left(\frac{N}{4}\right)^2
 \end{aligned}$$

$$T\left(\frac{N}{2^k}\right) = T\left(\frac{N}{2^3}\right) + T\left(\frac{N}{2^4}\right) + \sqrt{\left(\frac{N}{2^2}\right)^2}$$

$$T\left(\frac{N}{Q^3}\right) = T\left(\frac{N}{Q^4}\right) + T\left(\frac{N}{Q^5}\right) + \left(\frac{N}{Q^3}\right)^2$$

$$T\left(\frac{N}{Q^4}\right) = T\left(\frac{N}{Q^5}\right) + T\left(\frac{N}{Q^6}\right) + \left(\frac{N}{Q^4}\right)^2$$

$$\Rightarrow T\left(\frac{N}{Q^6}\right) + 3T\left(\frac{N}{Q^5}\right) + 3T\left(\frac{N}{Q^4}\right) + T\left(\frac{N}{Q^3}\right) + \left(\frac{N}{2}\right)^2 + \left(\frac{N}{2^2}\right)^2 + 8\left(\frac{N}{2^3}\right)^2 + \left(\frac{N}{2^4}\right)^2$$



$$\begin{aligned}
 & N^2 \left[1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \left(\frac{5}{16}\right)^3 + \dots + \left(\frac{5}{16}\right)^K \right] \\
 & = N^2 \left[\frac{1 - \left(\frac{5}{16}\right)^K}{1 - \frac{5}{16}} \right] = \frac{16N^2}{11} \left[1 - \left(\frac{5}{16}\right)^K \right]
 \end{aligned}$$

negligible.

$\Rightarrow \underline{\underline{O(N^2)}}$

To Do

1) $a^{\log_a b} = b$

2) Sum of n natural nos = $\frac{n(n+1)}{2}$

3) $\frac{N}{2}(2a + (n-1)d) = ?$ How?

1) $a^{\log_a b} = b$

$$\log(a^{\log_a b}) = \log b$$

$$\log_a b \cdot \log_a e = \log_b e$$

$$\frac{\log b}{\log a} \cdot \frac{\log a}{\log e} = \log_b e$$

$$\Rightarrow \log_b e = \log_b e$$

$$b=b$$

2) $1 \rightarrow 1$
 $1+2=3$ $\frac{2(3)}{2}=3$
 $1+2+3=6$

$$1+2+3+4+\dots+(n-1)+n$$

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

$$(n+1)+(n+1)+(n+1)+\dots+(n+1)+(n+1)$$

n times $(n+1)$ $\Rightarrow \frac{n(n+1)}{2}$
 added twice.

3) $a + (a+d) + (a+2d) + (a+3d) + \dots + (a+(n-1)d)$

$$\Rightarrow na + d + 2d + 3d + \dots + (n-1)d$$

$$= na + d(1+2+3+\dots+(n-1))$$

$$= na + d \cdot \frac{(n-1)n}{2}$$

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

MASTER's THEOREM:

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

$$f = \log_a b, \quad f(n) = N^c$$

1) $f < c \Rightarrow O(N^c)$

2) $f = c \Rightarrow O(N^f \log N)$

3) $f > c \Rightarrow O(N^f)$

$$\textcircled{1} \quad T(N) = aT\left(\frac{N}{2}\right) + b$$

$$a=2 \quad b=2 \quad t = \log_b^a = \log_2^2 = 1.$$

$I = N^0 = c = 0.$ $t > c \Rightarrow O(N^t)$

$\Rightarrow O(N)$

$$\textcircled{2} \quad T(N) = aT\left(\frac{N}{2}\right) + N$$

$$a=2 \quad b=2 \quad \log_b^a = \log_2^2 = 1.$$

$N! \Rightarrow c=1 \quad t=c.$

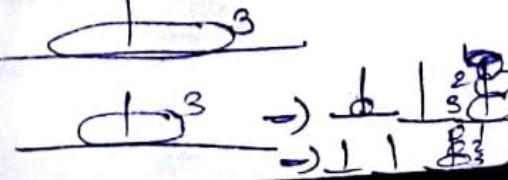
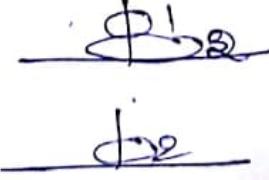
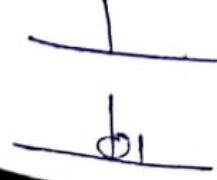
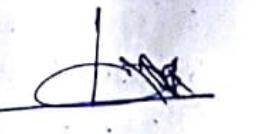
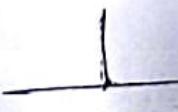
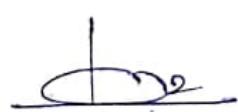
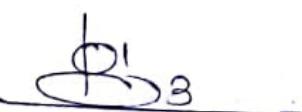
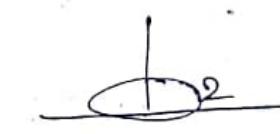
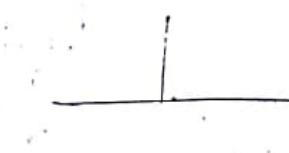
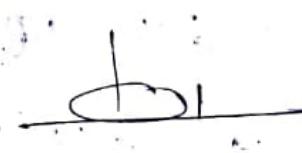
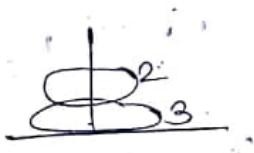
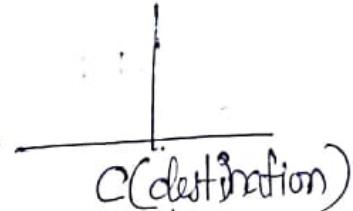
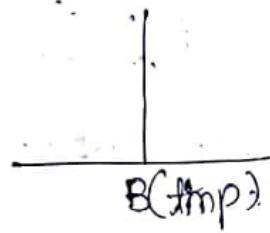
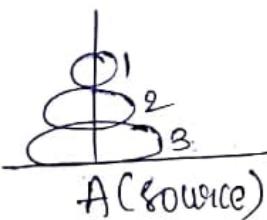
$\Rightarrow O(N^t \log N)$

$\Rightarrow O(N \log N)$

Towers of Hanoi

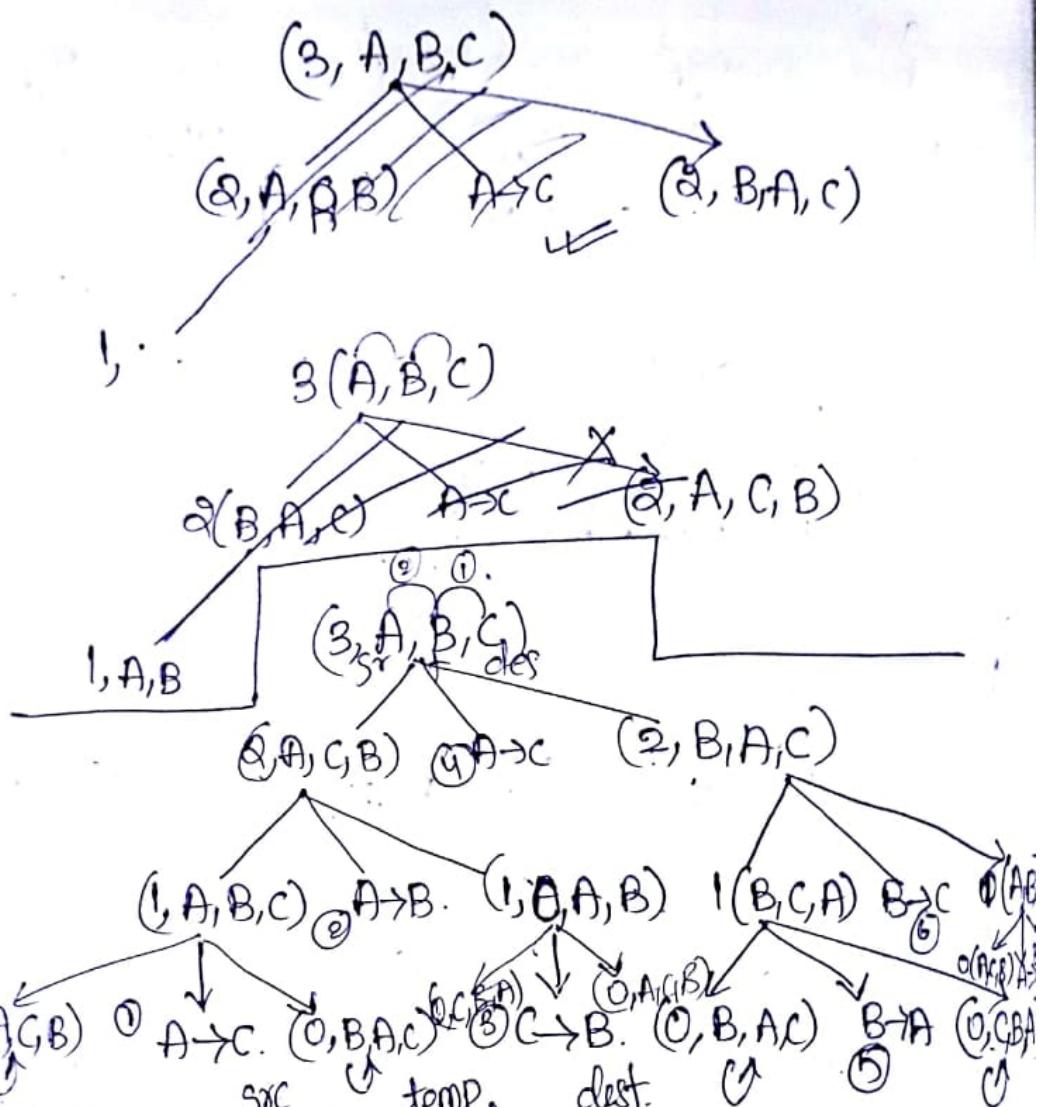
- only one disk at a time.
- smaller should be on bigger.

$N=3$



A (source), B (temp), C (destination).

- 1) A \rightarrow C.
- 2) A \rightarrow B
- 3) C \rightarrow B
- 4) A \rightarrow C
- 5) B \rightarrow A
- 6) B \rightarrow C
- 7) A \rightarrow C.



void TOH (int N, char src, char temp, char dest)

{

~~section~~ TOH (N-1, src, dest, temp)

cout ("Move " + N + " disk from " + src + " to " + dest);

~~section~~ TOH (N-1, dest, src, temp);

void TOH (int N, char src, char temp, char dest)

{

~~if~~ (N == 0) return;

~~return~~ TOH (N-1, src, dest, temp)

cout ("Move " + N + " disk from " + src + " to " + dest);

~~return~~ TOH (N-1, dest, src, temp);

};

$$\text{complexity: } T(n) = 2T(n-1) + 1.$$

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

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

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

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

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

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

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

$$= 2^n \cdot T(0) + 2^n - 1$$

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

$$= O(2^n)$$

$$\begin{array}{l} n-k=0 \\ \hline n=k. \end{array}$$

#func-calls \Rightarrow T on left, T on right, 1 main = 15. $= 2^{n+1} - 1$

$\prod_{i=0}^{N-1} i!$.

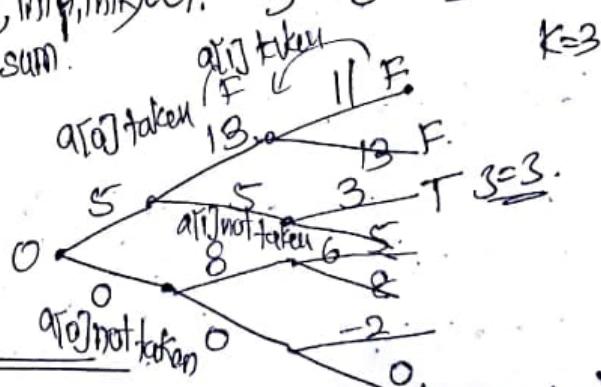
bool checkIfSumExists(int a[], int N, int sum);

if

int sum = 0;

if ($a[i]$ taken)

sum = sum + a[i];



bool checkIfSumExists(int a[], int N, int sum; int index, int k)

if ($\text{index} == N$) return sum == k;

return checkIfSumExists(a, N, sum + a[index], index + 1, k);

checkIfSumExists(a, N, sum, index + 1, k);

$T(N) = O(T(N-1)) + 1 \Rightarrow O(2^N, 1).$

Q3. b) $f(\text{int arr[], int N, int index, int K})$

{ if ($\text{index} == N$)
 return $K = 0$;
 return $f(\text{arr}, N, \text{index} + 1, K - \text{arr}[\text{index}]) ||$
 $f(\text{arr}, N, \text{index} + 1, K)$.
}

4) Reduce parameter index also, start from last.

from main start from last position) $5^0 8^1 2^2$

$f(\text{arr}, N-1, K) :=$

b) $f(\text{int arr[], int N, int K})$

{ if ($N == -1$)
 return $K = 0$;

return $f(\text{arr}, N-1, K - \text{arr}[N]) || f(\text{arr}, N-1, K)$;

}

Comp:- $O(2^N), 1.$

Valid Parenthesis

() ✓ \rightarrow length should be even.

()() ✓ \rightarrow no. of '(' = no. of ')'

(()) ✗ \rightarrow first should be '(' then ')'

()(()

(<) Lexicographical Order, '

$N=4$

(()) $\Rightarrow w_1$

() () $\Rightarrow w_2$

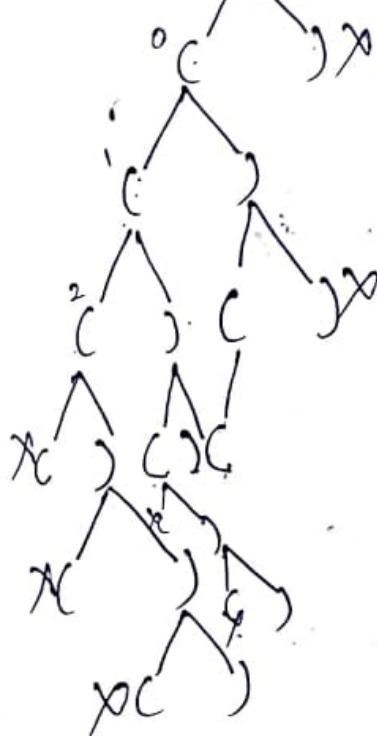
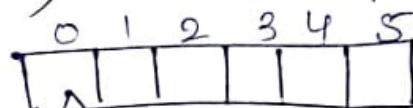
$N=6$

- 1) $((()))$ ✓
- 2) $((())())$ ✓
- 3) $(())()$
- 4) $((())())$ ✓
- 5) $(())()$

Lexicographic Order.

- ①. $(((()))$
- ②. $((())())$
- ③. $((())()$
- ④. $((())())$
- ⑤. $(())()$

char.



$$N=6 \Rightarrow 3\cdot(, 3\cdot)$$

put 'C' only if no. of Closes $\leq \frac{O}{2}$.

void wfparenthesis(char a[], int N,
 int index, int noOpen, int noClose)
 {
 if (index == N) pf(a); return;
 if (noOpen <= $\frac{N}{2}$)

{ pf("("); a[index] = '(';
 wfparenthesis(a[], N, index+1,
 noOpen+1, noClose);

{ if (noOpen > noClose)

{ a[index] = ')';
 wfp.(a[], N, index+1,
 noOpen, noClose+1);

void f(char a[], int N; int idx,
 int op, int cl) {

{ if (idx == N) o(1);
 print a[0];
 return;

{ if (op < N/2)
 a[idx] = '(';
 f(a, N, idx+1, op+1, cl);

{ if (op > cl)
 a[idx] = ')';
 f(a, N, idx+1, op, cl+1);

$$\begin{aligned} \text{idx} &= \text{op} + \text{c} \\ T(N) &= T(N) + 1 \\ \Rightarrow O(N) \end{aligned}$$

for $N=4 \Rightarrow$ w/o parenthesis $\Rightarrow 2$.

No.
4 → 2
6 → 5
8 →

Catalan
Numbers.

Read about it.

Given a list of words;

Check if the given pattern satisfies

with words.

dict: { abc, ab, x, xyx, z, bdc, xyz, d }

str₀: d/bdc/x/xyz/z/x

str_N: abc

str_N:

Q: dict: { ab, bc, abc } str₀: abcabc

bool f(Starting str, int N, int idx, List list)

{ If (idx == N)

return T;

{ for (int i = idx; i < N; i++)

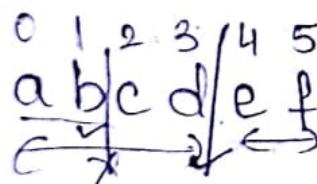
{ If (valid(str, i, p, list))

If (f(str, N, i + 1, list) == T)

return T;

else return false;

List: { ab, abc, ef }



f(str, 6, 0, ls)

= φ ∕= F

→ pdx = 2

p = 2 to 5

∅₃

→ pdx + 4
T → pdx = 6

pdx = 0
∅₁

f(str, 6, 0, ls)
at i = 1
f(str, 6, 2, ls)
(Cct.F)

pdx = 2
∅₂

∅₄
∅₅
∅₆
rel. F

pdx = 0
∅₃

f(str, 4, ls)
set T

Ques: array of rooms for coloured blocks.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----

Ques.
N=15.

colors.

Ques.
M=3.

0	1	2
4	2	3

lengths of index coloured blocks.

Print the ~~max~~ possibilities in which br. can be placed in
arr such that those length coloured blocks are placed in arr
with atleast -the distance of K in between.

eg. K=2.

empty spaces = 1. K times

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	-	-	-	-	-	-	2	2	2	-	-

⑦.

-3.

4

①.

②.

③.

5!

0	0	0	0	-	-	-	1	1	1	2	2	2	-	-
0	0	0	0	-	-	-	1	1	1	2	2	2	-	-

⑥.

6!

3/6.5.4!

7.4!

16/

=15

1	0	0	0	-	-	-	1	1	1	2	2	2	-	-
1	0	0	0	-	-	-	1	1	1	2	2	2	-	-

⑦.

⑧.

⑨.

1	1	1	1	-	-	-	1	1	1	2	2	2	-	-
1	1	1	1	-	-	-	1	1	1	2	2	2	-	-

1	1	0	0	0	-	-	1	1	1	1	-	-	2	2
1	1	0	0	0	-	-	1	1	1	1	-	-	2	2

⑩.

16/

=15

1	1	0	0	0	-	-	1	1	1	1	-	-	2	2
1	1	0	0	0	-	-	1	1	1	1	-	-	2	2

1	1	0	0	0	-	-	1	1	1	1	-	-	2	2
1	1	0	0	0	-	-	1	1	1	1	-	-	2	2

1	1	0	0	0	-	-	1	1	1	1	-	-	2	2
1	1	0	0	0	-	-	1	1	1	1	-	-	2	2

1	1	0	0	0	-	-	1	1	1	1	-	-	2	2
1	1	0	0	0	-	-	1	1	1	1	-	-	2	2

1	1	0	0	0	-	-	1	1	1	1	-	-	2	2
1	1	0	0	0	-	-	1	1	1	1	-	-	2	2

1	1	0	0	0	-	-	1	1	1	1	-	-	2	2
1	1	0	0	0	-	-	1	1	1	1	-	-	2	2

1	1	0	0	0	-	-	1	1	1	1	-	-	2	2
1	1	0	0	0	-	-	1	1	1	1	-	-	2	2

1	1	0	0	0	-	-	1	1	1	1	-	-	2	2
1	1	0	0	0	-	-	1	1	1	1	-	-	2	2

1	1	0	0	0	-	-	1	1	1	1	-	-	2	2
1	1	0	0	0	-	-	1	1	1	1	-	-	2	2

1	1	0	0	0	-	-	1	1	1	1	-	-	2	2
1	1	0	0	0	-	-	1	1	1	1	-	-	2	2

1	1	0	0	0	-	-	1	1	1	1	-	-	2	2
1	1	0	0	0	-	-	1	1	1	1	-	-	2	2

1	1	0	0	0	-	-	1	1	1	1	-	-	2	2
1	1	0	0	0	-	-	1	1	1	1	-	-	2	2

1	1	0	0	0	-	-	1	1	1	1	-	-	2	2
1	1	0	0	0	-	-	1	1	1	1	-	-	2	2

1	1	0	0	0	-	-	1	1	1	1	-	-	2	2
1	1	0	0	0	-	-	1	1	1	1	-	-	2	2

1	1	0	0	0	-	-	1	1	1	1	-	-	2	2
1	1	0	0	0	-	-	1	1	1	1	-	-	2	2

1	1	0	0	0	-	-	1	1	1	1	-	-	2	2
1	1	0	0	0	-	-	1	1	1	1	-	-	2	2

1	1	0	0	0	-	-	1	1	1	1	-	-	2	2
1	1	0	0	0	-	-	1	1	1	1	-	-	2	2

1	1	0	0	0	-	-	1	1	1	1	-	-	2	2
1	1	0	0	0	-	-	1	1	1	1	-	-	2	2

1	1	0	0	0	-	-	1	1	1	1	-	-	2	2
1	1	0	0	0	-	-	1	1	1	1	-	-	2	2

1	1

SORTING

	<u>Worst Case</u>	<u>Best Case</u>
1) Bubble Sort	$O(N^2), 1$	$O(N^2), 1$
2) Selection Sort	$O(N^2), 1$	$O(N^2), 1$
3) Insertion Sort	$O(N^2), 1$	$O(N), 1$
4) Merge S., Quick S., Radix S.; Heap S,		

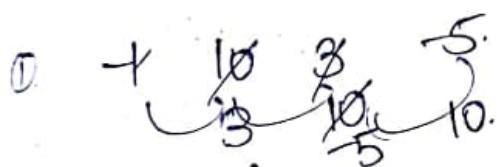
1) Bubble Sort:

void BubbleSort(int a[], int N)

```

    {
        for(int i=0; i<N; i++)
            {
                for(int j=0; j<N-i; j++)
                    {
                        if(a[j] > a[j+1])
                            swap(a[j], a[j+1]);
                    }
            }
    }

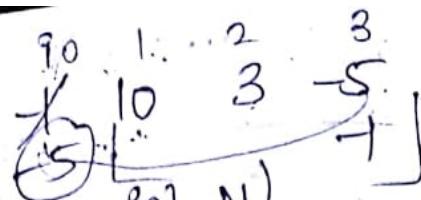
```



Maintain flag var.
If there are no swaps in
1st iteration, then there
will be no swaps in
remaining iterations.

2) Selection Sort:

void SelectionSort (int [] arr, int N)



{ for (int i=0; i<N; i++)

{ min = arr[i]; idx = i;

for (int j=i+1; j<N; j++)

{ if (arr[j] <= min)

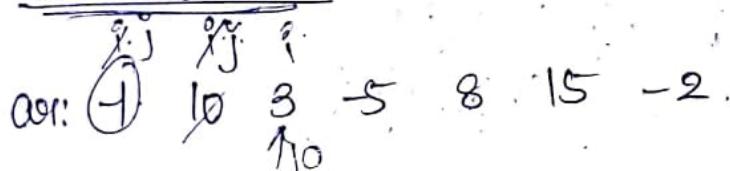
{ min = arr[j]; idx = j;

swap(arr[i], arr[idx]); swap(arr[i], arr[idx]);

}

}

⑤. Insertion Sort:



void InsertionSort (int [] arr, int N)

{

for (int i=0, i<N; i++)

{ int j = i-1, x = arr[i];

for (j>=0)

{ if (arr[i] < arr[j])

arr[j+1] = arr[j];

j--;

arr[j+1] = x;

}

for (int i=0; i<N; i++)

{ j = i-1, x = arr[i];

while (j>=0 && arr[j]

{

arr[j+1] = arr[j];

j--;

arr[j+1] = x;

}

Best Cases: When array is sorted. $\rightarrow O(N), 1$

$O(N^2), 1$

23/19

ele.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	n.v.
cnt:	1	5	2	7	10	2	5	2	2	10	1	2	3	5	2

$0 \leq a[i] \leq m+1$

$1 \leq a[i] \leq 10$. 0th voted 1
1st voted 5
voters 3 candidates

Find the elected person.

①. $\text{int } \text{cnt}[11] = \{0\}$ $\text{cnt}(1) = \#1's \Rightarrow 2$
 $\text{for } i=0; i < n; i++$ $\text{cnt}(2) = \#2's \Rightarrow 6$
 $\text{cnt}(10) = \#10's \Rightarrow 2$ } find max.

②. get solve ($\text{int } \frac{a[1]}{m+1}, \text{int } N, \text{int } m$)

{ $\text{int count}[11] = \{0\}$.
 $\text{for } (\text{int } i=0; i < n; i++)$
 $\{ \quad \text{count}[a[i]]++;$
 $\}$
 $\text{int max} = \text{count}[0]; \text{max} = 0;$
 $\text{for } (\text{int } i=1; i < m; i++)$
 $\{ \quad \text{if } (\text{count}[i] > \text{max})$
 $\quad \text{max} = \text{count}[i]; // \text{max} = i.$
 $\}$
 $\text{return max};$

Complexity: $O(N+m)$, Time: $O(m)$, Space

```
int ans=0;
for(int i=1; i<=m; i++)
{
    if(count[i]>count[ans])
        ans=i;
}
return ans;
```

After $\{5 \ 2 \ 7 \ 10 \ 2 \ 5 \ 2 \ 2 \ 10 \ 1 \ 2 \ 3 \ 5 \ 2$.

Using count array between $\text{count}[11]$

0	1	2	3	4	5	6	7	8	9	10	11
0	2	6	1	-	-	-	-	1	1	1	-

ans. ↓
 $= 0$ $N \ 1 \ 1 \ 2 \ 2 \ 2 \ 2 \ 2 \ 3 \ - \ -$

④ CountSort
void CountSort(int arr[], int N, int m)

{
 int T[] count = new int[m+1];

 for (int p=0; p<N; p++)
 count[arr[i]]++;

 int K=0;
 for (int i=0; i<N; i++) {
 arr[K+i] = count[i];
 K++;

 }
 for (int j=0; j<count[p]; j++) {
 arr[K+j] = count[p];
 }

}

}

Total iterations of innerloop = $O(2 + 61 \cdot 1) = N$.

Outerloop iterations = $m+1$.

$$O(m+1+N) = O(m+N)$$

Time Complexity = $O(m+N)$, $O(m)$ space.

— CountSort is better than BS, SS, IS :-
 $O(N+m)$.

— but difficult to use with negative elements $-10 \leq arr[i] \leq 20$
as indexes can't be negative but can use offset
indexes.

$-10 \leq arr[i] \leq 20 \Rightarrow$ can be stored

in count[31]
 $[a, b] \Rightarrow b - a + 1$ count array size

$$10^9 \leq arr[p] \geq 10^9 + 100$$

100 unique elements.

$$10^9 \Rightarrow 0$$

$$10^9 + 1 \Rightarrow 1$$

$$10^9 + 2 \Rightarrow 2$$

1

$$10^9 + 100 \Rightarrow 100$$

$$-10 \leq arr(p) \leq 20$$

element count index

$$-10 \Rightarrow 0$$

$$-9 \Rightarrow 1$$

$$-8 \Rightarrow 2$$

⋮

$$0 \Rightarrow 10$$

$$1 \Rightarrow 11$$

$$20 = 30$$

* size of count array
b-a+1
 $20 - (-10) = 30$

void countSort (int arr[], int N, int a, int b)

{ int[] count = new int[b-a+1];

for (int i=0; i<N; i++)

{ count [arr[i]-a]++;

}

int index=0.

for (int i=0; i<(b-a+1); i++)

{

for (int j=0; j<count[i]; j++)

arr[index+i] = a+i;

}

}

let R=b-a+1.

Comp: $O(N+R)$, $O(R)$
Time Space.

Count Sort:

1) Cannot Use : $R > 10^{10}$

2) Should not be used

when N and R are very far.

The diff = $N-R$ is very large.

then count sort take $(N+R)$ iterations.
which is more.

⇒ Should use only when $R \leq N$

- when count arr. size is comparable to N elements.

Ex. arr: -10 100 10000
count arr size = $b-a+1$
 $= 10000 - (-10) + 1$
 $= 10,011$
Size.

$N=3$ elements.

$O(N+R) = (3+10,011)$
~~more iterations~~

↓
otherwise
BS, SS, DS
takes $O(N^2)$

$3^2 = 9$ ito

which is

fine when

Comp to 10014

p78

Q) $\text{arr} = [18, 26, 15, 12, 9, 3, 19, 8, 13, 17, 35, 11, 93]$
 $0 \leq \text{arr}(i) \leq 10^9$
 $1 \leq k \leq 100$.
 Eg: $K=10$.

choose the subset from arr such that
 the sum of any two elements in the
 subset is not divisible by k .

Eg: we can't choose subset $\{18, 9, 12, 3\}$ as $18+12=30$.
 Eg: we can choose subset $\{18, 15, 9, 3, 19, 93\}$.

①. Brute Soln: Generate all subsets — $2^N \cdot N$ valid
Force: check if sum of pairs is div by 10 & not $= N$.

Comp: $O(2^N \times N \times N^2)$, $O(N)$.

Even if $N=30$

$2^{30} = 10^9 \times$ we
 need upto $\frac{1}{10}$

```

int subsetValidity(int arr[], int N)
{
    int subsetArr = new int[N];
    int idx=0; int max=0;
    for(int i=0; i<(1<<N); i++)
    {
        for(int j=0; j<N; j++)
        {
            if((i<<j)&i)!=0)
            {
                subsetArr[idx++] = arr[j];
            }
        }
        if(sumisValid(subsetArr, idx))
        {
            if(max < idx)
            {
                max = idx;
            }
        }
    }
    int k=0;
    for(int i=0; i<idx; i++)
    {
        subsetArr[k] = arr[i];
        k++;
    }
    idx=0;
}
    
```

```

boolean sumIsValid(int[] subsetArr, int idx, int k)
{
    for (int i=0; i<idx; i++)
    {
        for (int j=i+1; j<idx; j++)
        {
            if ((subsetArr[i] + subsetArr[j]) % k == 0)
                return false;
        }
    }
    return true;
}

```

$O(2^N \cdot N \cdot N^2)$, $O(N^3)$ complexity.

Input: 18 26 15 12 9 3 19 5 13 17 35 11 93 16 14 24 4

18 ✓ 12 any one. ②. k=10.

(3, 13, 93) ✓ 17. max 3, 13, 93. ✓ ①.

5 ✓ 15 ✓ 35. any one. ③.

16, 26 ✓ 4, 14, 24 take 24, 14, 24. ①.

(9, 19) ✓ 11 take max. no. of ele. 9, 19. ①. $O(N^2)$

Non-divisible Subsets., Hackerrank, Problem Solving.

=

$$\textcircled{2} \quad A_N : 3 \ 5 \ 12 \ 18 \ 30$$

$$B_m : -5 \ -2 \ 9 \ 11$$

Initially both A_N and B_m are sorted.

Get the complete 2 arrays. Get sorted combination.
N+m elements.

$$\text{Sorted } C_{N+m} : -5 \ -2 \ 3 \ 5 \ 9 \ 11 \ 12 \ 18 \ 30.$$

①. Merge Sort. $\Theta(N \log N)$.

①. Copy A_N to C_{N+m} — N time

Copy B_m to C_{N+m} — m time.

Sort C_{N+m} using BS, SS, IS $\Rightarrow (N+m)^2$.

Comp: $O(N+m + (N+m)^2)$, $O(N+m)$.

②. Copy A_N to C_{N+m} — N time.

Using insertion sort put the elements of B_m to C_{N+m}
— max elements to be shifted everytime = N.

for mele $\Rightarrow m \times N$.

Comp: $O(N + m \times N) \text{ or } O(N+m)$.

③

$$P_0 : \begin{matrix} & 1 & 2 & 3 & 4 \\ 3 & & & & \\ 5 & & & & \\ 12 & & & & \\ 18 & & & & \\ 30 & & & & \end{matrix}$$

$$P_m : \begin{matrix} -5 & -2 & 9 & 11 \\ 0 & 1 & 2 & 3 \end{matrix}$$

$$P_2$$

Comp: $O(N+m)$, $O(1)$.

No. of comparisons.

Compare P_1, P_2

put the smaller one in P_2

inc the $P_1 \neq P_2$ it

2. $\text{int} C$
 $\{$
 sortedCombination ($\text{int} A[]$ $\rightarrow N$, $\text{int} B[]$ $\rightarrow M$)

$\text{int } K=0;$ $\text{int } C_{NM} = \text{new int}[N+m];$
 $\text{int } p1=0;$
 $\text{int } p2=0;$

 while ($(p1 < N) \&& (p2 < M)$)

$\{$

 if ($A_N[p1] < B_M[p2]$)

$\{$

$C_{NM}[K++] = A_N[p1];$

$p1++;$

$\}$

 else if ($B_M[p2] < A_N[p1]$)

$\{$

$C_{NM}[K++] = B_M[p2];$

$p2++;$

$\}$

$\}$

 if ($p1 < N$)

$\{$

 while ($p1 < N$)

$\{$

$C_{NM}[K++] = A_N[p1];$

$p1++;$

$\}$

$\}$

 while ($p2 < M$)

$\{$

$C_{NM}[K++] = B_M[p2];$

$p2++;$

$\}$

 return $C_{NM};$

$\}$

* Initially A_{N+m} and B_m are sorted, with m empty slots in A_{N+m} . Merge B_m in A_{N+m} such that A_{N+m} becomes sorted.

$$\begin{array}{cccccc} A_{N+M} & : & 3 & 5 & 12 & 18, 30 & - & - \\ B_M & : & -5 & -2 & 9 & 11 & & \end{array}$$

①. Cptm Take Extra Space.

Put, sort & copy GUTM in ANFM.

Comp: $O(N+m+(N+m)^2)$, ~~$O(N+m)$~~ , $O(1)$

② Inception Sort. $O(N + m * N)$, ~~$O(n + m)$~~ (1)

③. several combination: $O(N+m)$, $O(m+N)$ if $m \ll N$ is true
 Note: $(n+1)^{nd}$. P. 33

④ $O(MN)$

Nel. $(M \times N) \times (N \times D)$. $C_{ij} = \sum_{k=1}^N A_{ik} B_{kj}$

$C_{11} = 11$

$A \cdot // \text{size } M \times N$

$B \cdot // \text{size } N \times D$

$C \cdot // \text{size } M \times D$

```

public void startFromLastChecking(int[] ANM, int[]):
{
    int p1=N-1, p2=M-1; int k=N+M-1; // pointing at last.
    while ((p1>=0) & & (p2>=0))

```

{ If ($A[p_1] > B[p_2]$)

$$\{ A[K--] = A[PI] \}$$

Pl--;

$\nexists (A[p_1] \times B[p_2])$

$$\{ \quad A[K--] = B[p_2]$$

{ pd --;

while ($pt \neq 0$)

 if ($pt == 0$)

 { while ($p2 > 0$)

 { $A[k-] = B[p2];$

 } $p2--;$

 }

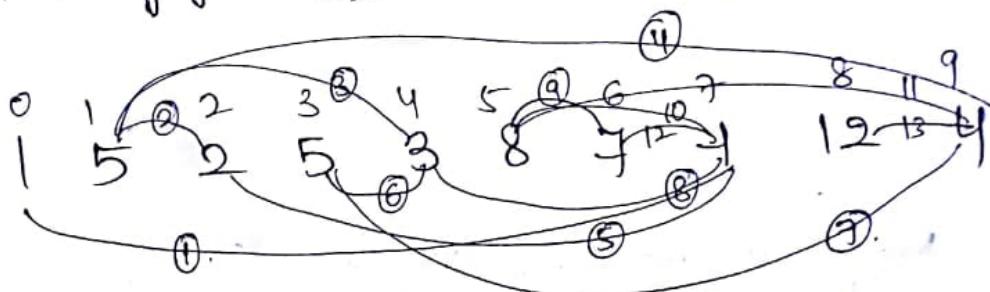
 }

$i=1 \quad j=4$

$\circlearrowleft 573$

① $a[10]=10$ 0 5 2 5 3 8 7 1 12 4

Count No. of pairs (i, j) such that $(i < j)$ & $a[i] > a[j]$.



Ans: 13.

int countNoOfPairs (int¹ arr, int N)

{ int count=0;

 for(int i=0; i<N; i++)

 { for(int j=i+1; j<N ; j++)

 { if (arr[i] > arr[j])
 count++;

 }

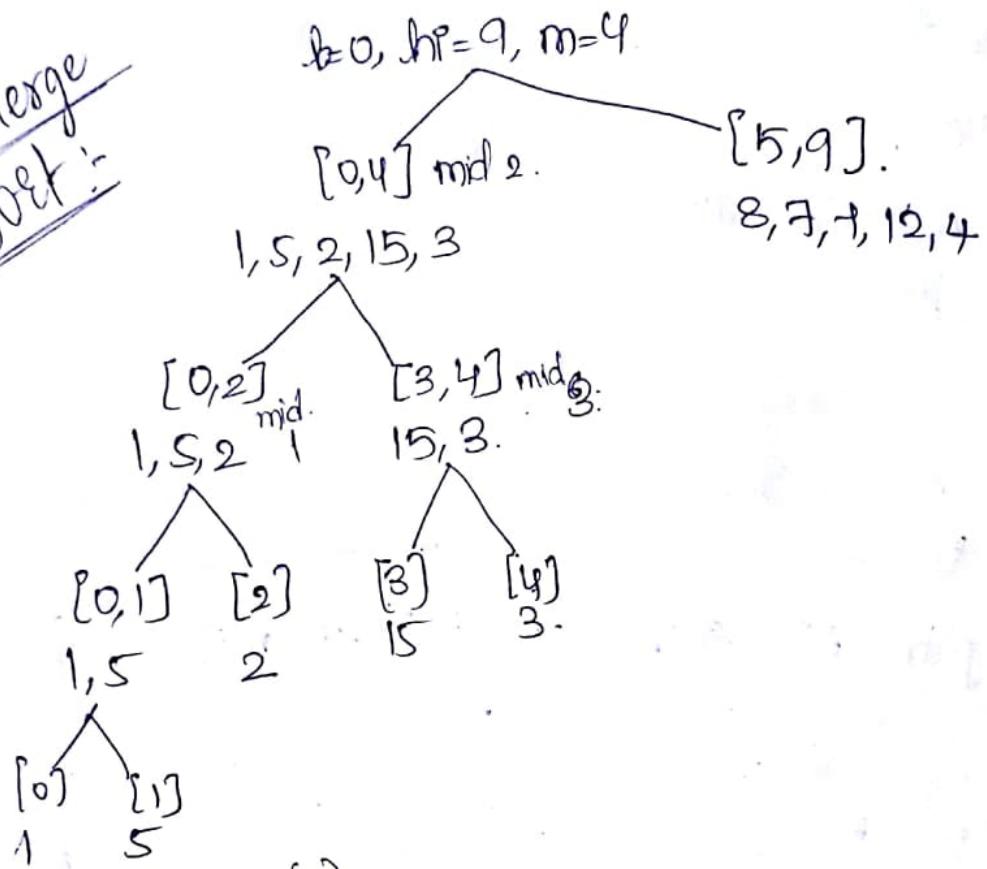
 return count;

}

\Rightarrow comp $O(N^2)$, $O(1)$.

0 1 2 3 4 5 6 7 8 9
 1 5 2 15 3 8 7 12 4

Merge Sort :-



$T(N)$

```

void MergeSort(int arr[], int low, int high)
{
    if (low == high) {
        c(1)
    }
    int mid = (low + high) / 2;
    MergeSort(arr, low, mid); —  $T(N/2)$ 
    MergeSort(arr, mid+1, high); —  $T(N/2)$ 
    Merge(arr, low, mid, high); —  $O(N)$ .
}
    
```

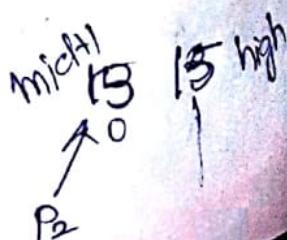
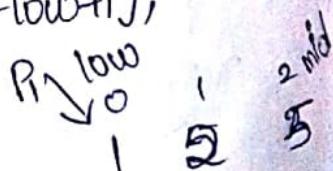
void Merge(int arr[], int low, int mid, int high)

int[] temp = new int[high - low + 1];

int p1 = low;

int p2 = mid + 1;

int k = 0;



```

while ((pl <= mid) && (pr <= high))
{

```

```

    if (arr[pl] < arr[pr])
    {

```

```

        temp[k++] = arr[pl];
        pl++;
    }

```

```

else if (arr[pl] >= arr[pr])
{

```

```

    temp[k++] = arr[pr];
    pr--;
    count++;
}

```

```

while (pl <= mid)
{

```

```

    temp[k++] = arr[pl];
    pl++;
}

```

```

while (pr <= high)
{

```

```

    temp[k++] = arr[pr];
    pr--;
}

```

```

for (int i=0; i < temp.length; i++)

```

```

    arr[low+i] = temp[i]; // O(n)
}

```

(or)

```

for (int i=low; i <= high; i++)
{
    arr[i] = temp[i-low];
}

```

$$T(n) = \alpha T\left(\frac{n}{\alpha}\right) + N$$

$\begin{array}{l} a=2 \\ b=2 \end{array}$

$$n^{\log_a b} = n^{\log_2 2} = n^1$$

$t=1$
 $c=1$

$$t=c \Rightarrow N \log N$$

$$\Rightarrow N \log N$$

Comp: $O(N \log N), O(N)$

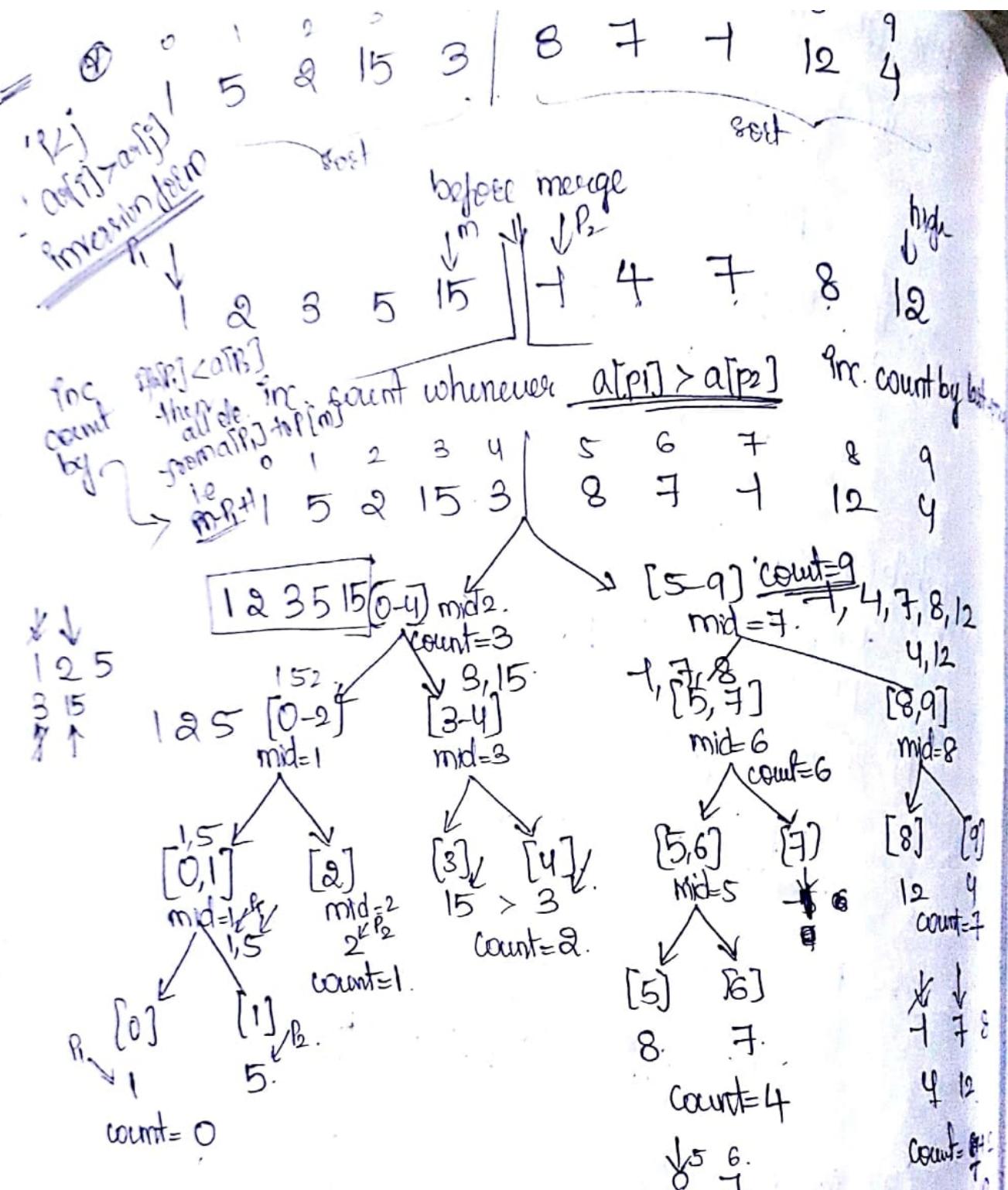
merge $O(N+m)$

$A_N B_m$

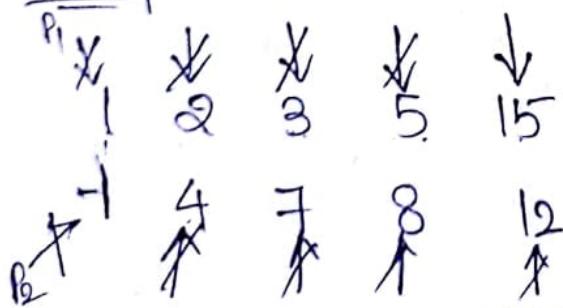
$A_{N/2} B_{N/2}$

$O\left(\frac{N}{2} + \frac{N}{2}\right)$

$O(N)$



last step - 1 2 3 4 5 7 8



$$\text{count} = 9 + 5 = 14 + 2 = 16 + 1 = 17 + 1 = 18 + 1 \leftarrow 19$$

ans

Comp: $O(N \log N)$

min count value :-

Q1: 1 3. 10 12 20. Ascending Order
Min=0.

max Count Value:

Q1: ~~$20 \ 12 \ 10 \ 3 \ 1$~~ $\underline{N=5}$.

$4+3+2+1 = 10$ elements on right are smaller

$$\frac{N(N-1)}{2} = \frac{5(4)}{2} = 10$$

$$0 \leq \text{ans} \leq \frac{N(N-1)}{8}$$

* Sum Of Pages

sum of 2 elements = k. $\begin{array}{l} \text{True} \\ \text{False} \end{array}$

$\partial g_N = -5 \ 3 \ 18 \ 12 \ 25 \ -6 \ 15 \ 9 \ +$

$$k = 6, 10, 15, 12, 28, 36$$

$$\alpha_1(i) + \alpha_1(j) = k \text{ and } i \neq j.$$

①. Brute Force

boolean sumOfPairs(int[] arr,
Comp. i) $O(N^2)$,
for $i = 0$ to N .
 for $j = i+1$ to N
 if ($arr[i] + arr[j] == k$).
 return T ;
 return F ;

Q. Sort the array:
N log N.

Comp.  $O(N \log N + N)$, time
 $O(N)$, space

If ($p_1 == p_2$)
return false

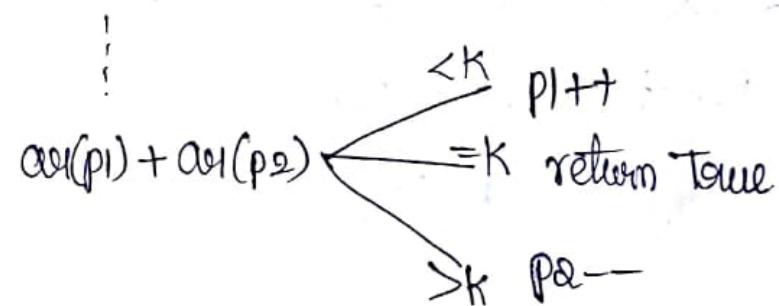
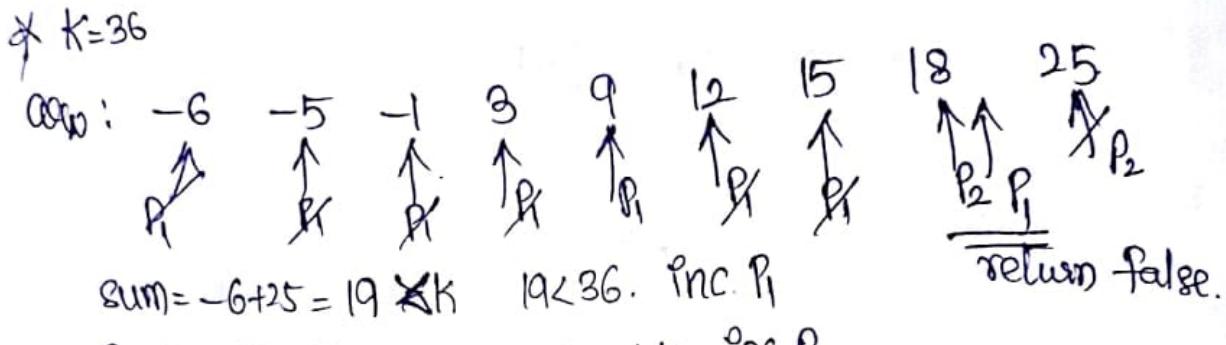
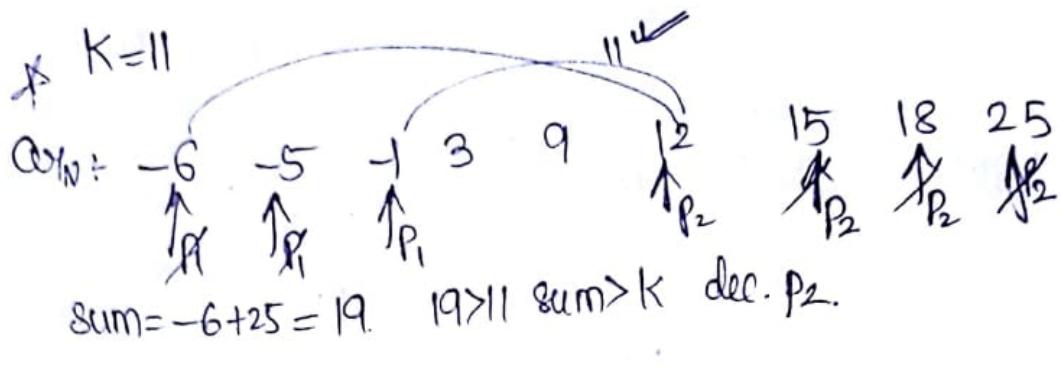
$P_1 = 0$; $P_2 = N-1$;
while($P_1 \leq P_2$) $\{$ $SUM = SUM + arr[P_1] + arr[P_2]$ $\}$

$$\{ p_f(\text{sum} > k) \text{ dec } p_2 \Rightarrow p_2 -$$

$p_f(\text{sum} > k) \rightarrow p_2 \Rightarrow P_2$

$\text{pp}(\text{sum} < k)$ $\text{qrc R} = \text{P1++}$
 $\text{pp}(\text{sum} == k)$ return true;

If $(g_{\text{un}}) = 1$)



If ($p_1 == p_2$) return False.

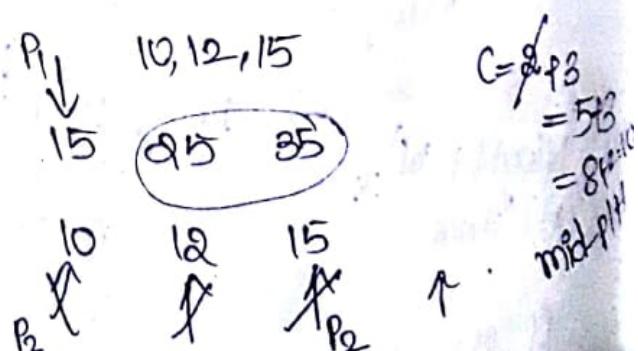
$K=11$

If $\sum = -6 + 25 = 19.$ $19 > 11$ dec p_2

for whatever values after -6, i.e greater than -6, sum will be still ≥ 19 .

to ignore these

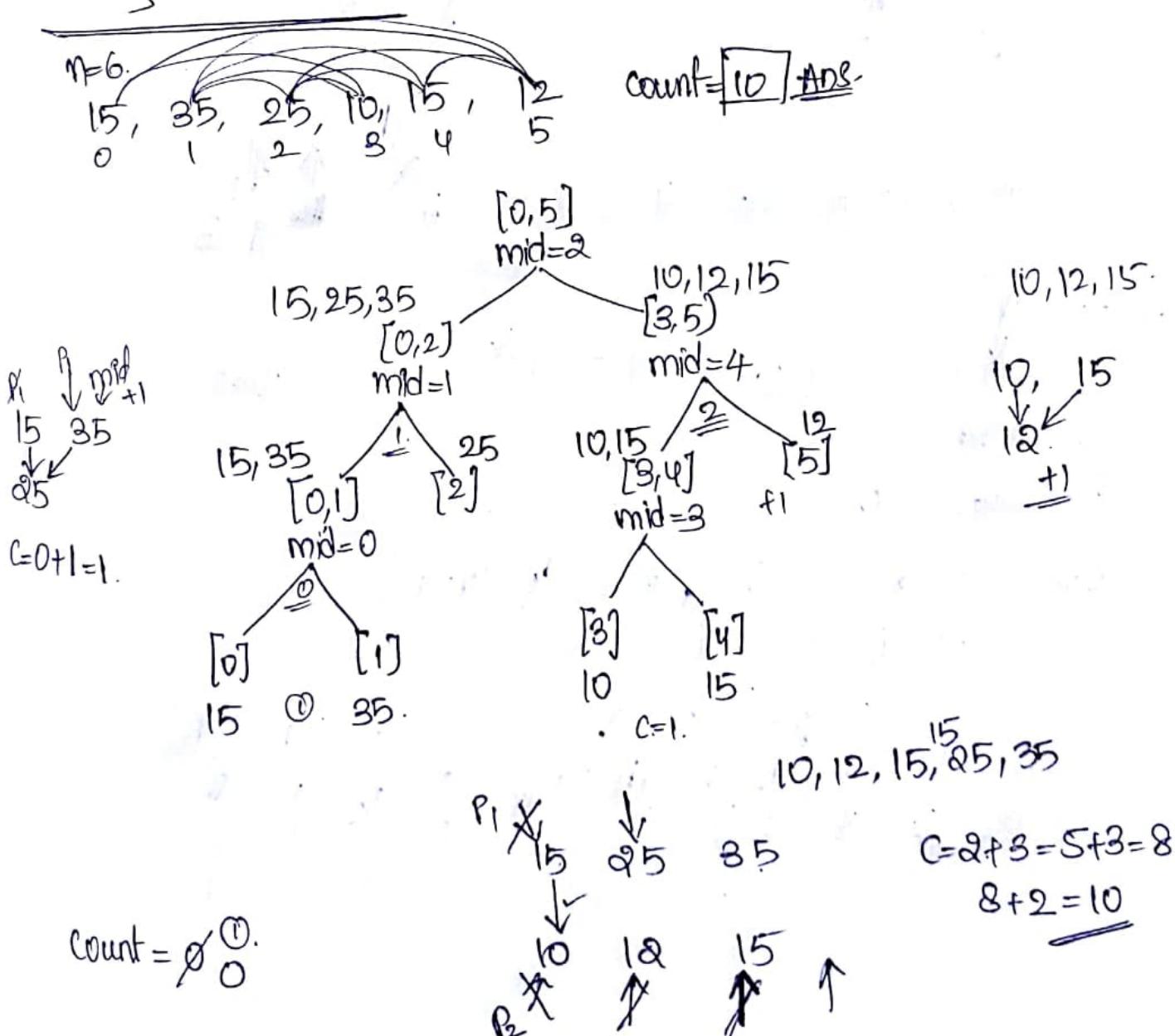
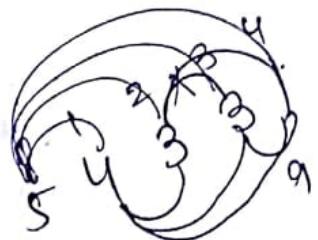
- ① Pair Sum
- ② Pair Difference
- ③ Triplet Sum.

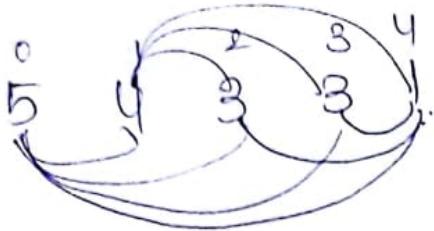


```

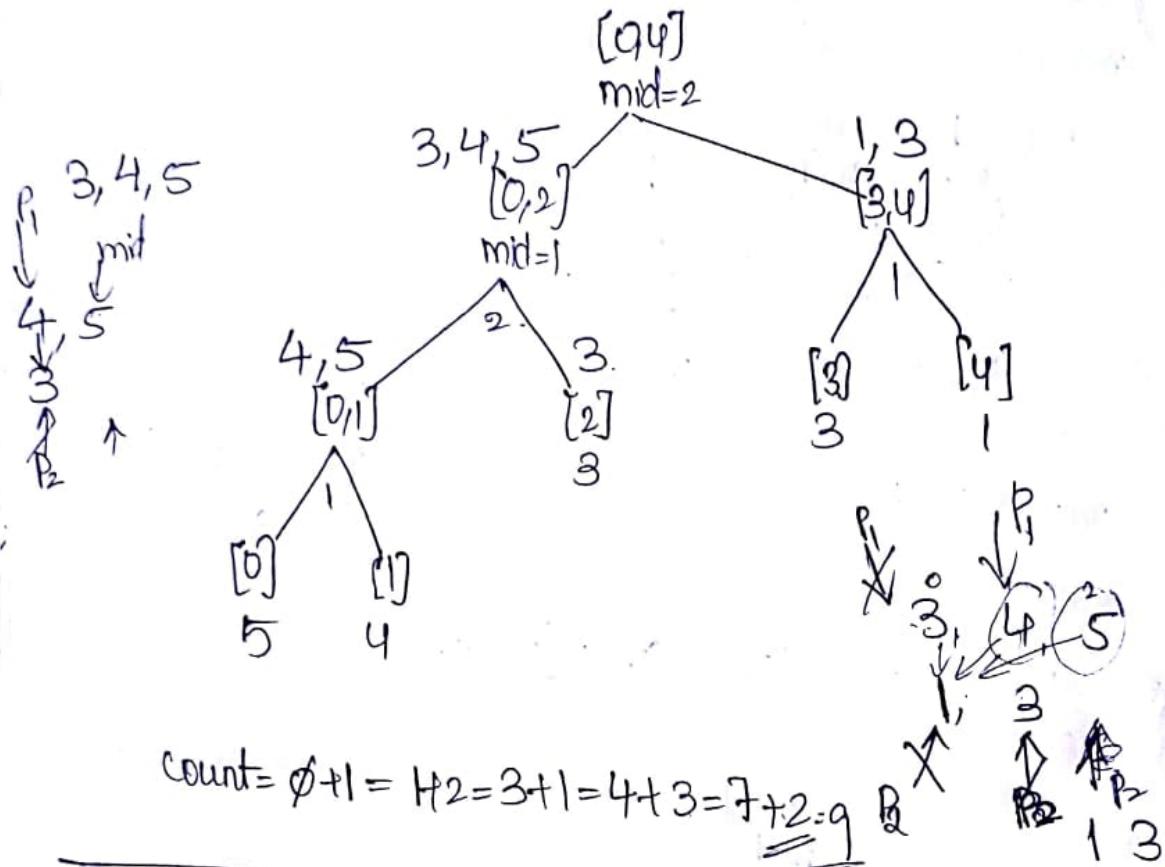
bool solveSum( int arr[], int N, int k)
{
    mergesort(arr, 0, N-1);
    // Point the Array;
    int p1=0, p2=N-1; int sum=0;
    while (p1 != p2)
    {
        sum = arr[p1] + arr[p2];
        if (sum > k) p2--;
        if (sum < k) p1++;
        if (sum == k) return true;
    }
    return false;
}

```

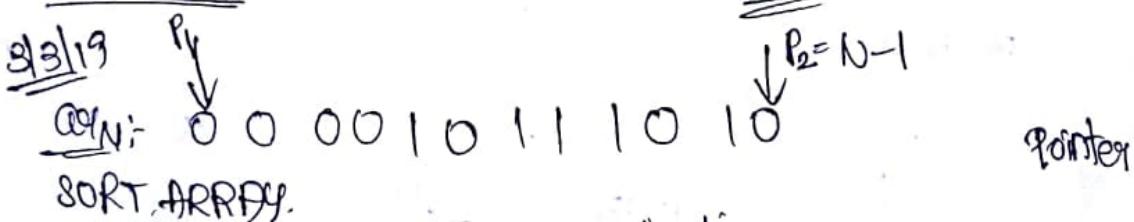




count = 9.



$$\text{count} = \emptyset + 1 = H_2 = 3 + 1 = 4 + 3 = 7 + 2 = 9$$



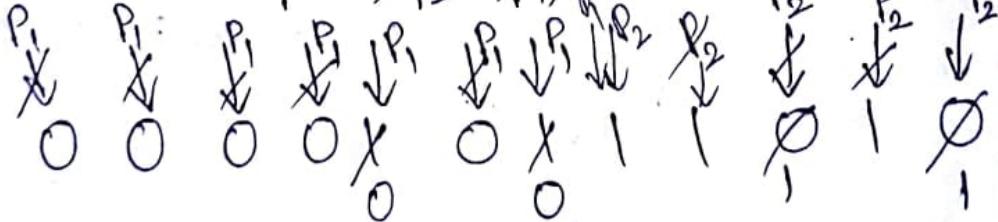
Count Sort. $CO=x$, $2N$ iterations
write in N iterations.

void sortZeroesOnes(int arr[], int n)

for (int i=0; i<n; i++)

{

 int p1=0, p2=N-1;



```

void setSortZeroOnes(int a[], int n)
{
    int p1=0, p2=N-1;
    while(p1 < p2)
    {
        if(a[p1]==0) p1++;
        if(a[p2]==1) p2--;
        if((a[p1]==1) && (a[p2]==0))
        {
            swap(a[p1], a[p2]);
            p1++; p2--;
        }
        if(p1==p2)
            break;
    }
}

```

Loop should run till $p1 < p2$.

```

void setSortZeroOnes(int a[], int n)
{
    int p1=0, p2=N-1;
    while(p1 < p2)
    {
        if((a[p1]==1) && (a[p2]==0))
        {
            int temp=a[p1];
            a[p1]=a[p2];
            a[p2]=temp;
        }
        if(a[p1]==0)
            p1++;
        if(a[p2]==1)
            p2--;
    }
}

```

① $\frac{P_1}{P_2} \frac{P_1}{P_2}$

② $\frac{P_1}{P_2} \frac{P_1}{P_2}$

③ $\frac{P_1}{P_2} \frac{P_1}{P_2}$

④ $\frac{P_1}{P_2}$

⑤ $\frac{P_1}{P_2} \frac{P_1}{P_2}$

⑥ $\frac{P_1}{P_2} \frac{P_1}{P_2} \frac{P_1}{P_2}$

⑦ $\frac{P_1}{P_2} \frac{P_1}{P_2} \frac{P_1}{P_2}$

2) 0

3) 10

4) 1

5) 0

6) 0 0000

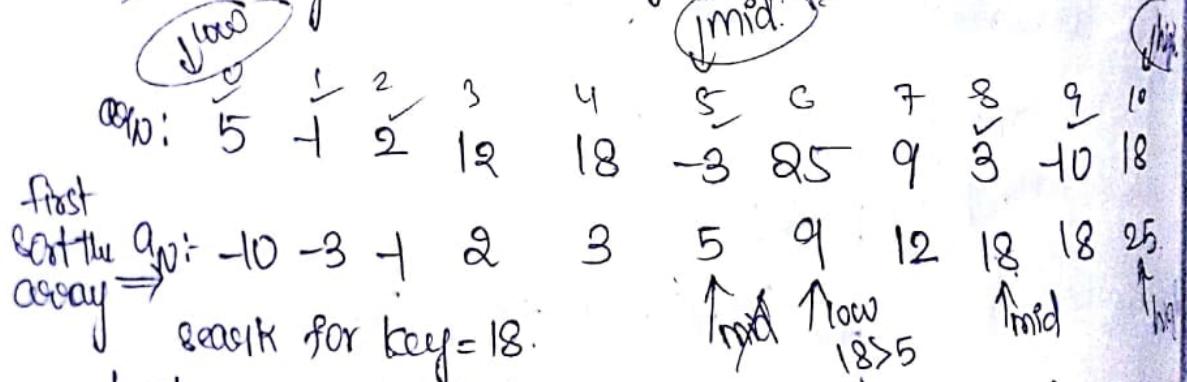
7) 1 111

* Quick Sort :- To Do.

SEARCHING:-

① Linear Search : $O(N), O(1)$

② Binary Search : $O(\log_2 N), O(1)$.



bool BinarySearch (int arr[], int n, int k)

{

int low=0, high=n-1,

while (low <= high)

{

mid = (low+high)/2;

if (arr[mid] == key)

return true;

Recursive

```

if (key > arr[mid])
    low = mid + 1;
else if (arr[mid] > key)
    high = mid - 1;
}
// while end.
return false;

```

Recursive:

```

bool binarySearch(int[] arr, int n, int k, int low, int high)
{
    int mid = (low + high) / 2;
    if (arr[mid] == k)
        return true;
    if (arr[mid] > k)
        return binarySearch(arr, n, k, low, mid - 1);
    else
        return binarySearch(arr, n, k, mid + 1, high);

    if (low > high)
        return false;
}

```

Base Condition.

```

3
bool binarySearch(int[] arr, int n, int k, int low, int high)
{
    int mid = (low + high) / 2;
    if (low > high) return false;
    if (arr[mid] == k) return true;
    if (arr[mid] < k)
        return binarySearch(arr, n, k, mid + 1, high);
    return binarySearch(arr, n, k, low, mid - 1);
}

```

$T\left(\frac{n}{2}\right)$

execute only 1 ref.

Binary Search $T(n) = T\left(\frac{n}{2}\right) + 1$

$$\text{conf} \quad n^{\log b} - n^{\log b} = n^0 = 1 = t.$$

$$n^c = n^0 \Rightarrow c = 1$$

$$t = c \Rightarrow \log N \rightarrow$$

Linear Search: $O(N)$, $O(1)$ unsorted array

Binary Search

Binary Search Complexity: $O(\log N)$.

Simplified Masters Theorem:

Q4 arr: 5 12 18 25 9 3 10 18

key=k	<u>Sorted Array</u>	<u>Unsorted Array</u>	<u>Unsorted Array</u>
Linear Search	$O(N)$, 1	$O(N)$, 1	$\# \text{keys}=1$
Binary Search	<u>Sorted array</u> $\log N$, 1	<u>to sort</u> $N \log N + O(\log N)$	<u>to sort</u> $N \log N + O(\log N)$

key=k	<u>Sorted Arr.</u>	<u>Unsorted Arr.</u>	<u>Unsorted</u>
Linear Search	$O(N)$, 1	$N, 1$	$\# \text{keys}=1$
Binary Search	$\log N$, 1	$N \log N + O(\log N), N$	$O(N), 1$

03, 1's

merge

sum of pairs

at back

mergesort

a-b=k

Two Pointer Technique:

Sorting - Quick Sort

- do not take extra space
- do not consider recursion
- code & implementation

* Qn: 5 + 8 12 18 -3 25 9 3 → 10 18 → first sort
 sum of pairs $a[i] + a[j] = k$, $i < j$. Use binary search & solve.
 $K = 10, -5, 23$

function sumOfPairs (int[] arr, int n, int k)

{

for (int i=0; i < n; i++)

{

int key = k - arr[i];

If (binarySearch (arr, n, key, $\underline{i+1}, \underline{n-1}$)) // will
 not work
 $\text{if } BS(g)$

return true;

}

return false;

}

$$a+b=k$$

$$b=k-a$$

$$\text{key.} = a+b.$$

$$b=k-a$$

as same element can be found so search from $i+1$ to $n-1$

First Sort the Array to apply Binary Search

Sorted → 10 → 3 → 8 3 5 9 12 18 18 25.

$$K = 10, -5, 23.$$

$$K = a+b$$

$$b = K-a$$

$$\begin{array}{l} K \\ 10 \\ -3 \\ \vdots \\ 5 \end{array} \quad \begin{array}{l} a \\ -10 \\ -3 \\ \vdots \\ 5 \end{array} \quad \begin{array}{l} b \\ 10 \\ 13 \\ \vdots \\ 5 \end{array}$$

① BS false

13 → F.

Comp.: 1. sorting first : $N \log N$

2. for N elements BinarySearch -

$$= N \log N. \log K$$

$$\Rightarrow O(N \log N + N \log N) = O(N \log N). O(2N \log N)$$

Comp: $O(N \log N)$, $O(1)$
 Time. Space

5 → True. $\text{if } BS(4)$

but this should be false as we have only one 5,
 $\text{so } BS(\underline{i+1}, \underline{n-1})$.

we can search on the right side as BS($i+1, n$).
We can also search on the left side as BS($0, i-1$)
but it takes more no. of iterations in for ($i=0$ to $n-1$)
If we go from $n-1$ to 0 then left side take less no. of iterations.

bool sumOfPairs (int arr, int n, int k)

{

 for (int i = 0; i < n; i++)

 if (BSR (arr, n, k - arr[i], 0, i-1))

 return true;

}

 return false;

3.

-10 -3 1 2 3 5 9 12 18 18 25

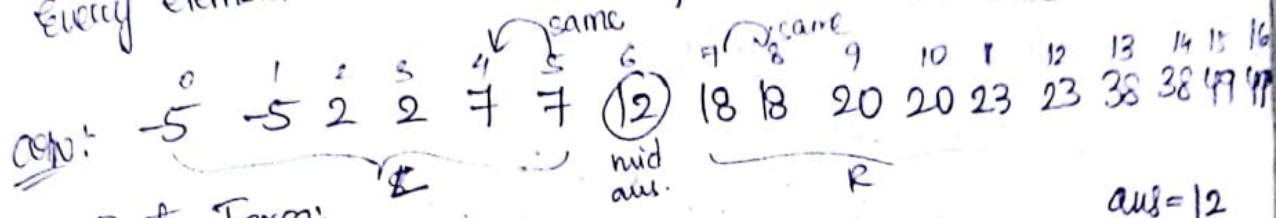
Key = 14

a b \downarrow check from right side.
-10 24 \rightarrow F
-3 17 \rightarrow F
1 15 F
2 12 - True ✓

Key = 14

a b. \downarrow check from left side

② Find the element which occurs only once in given sorted array.
Every element occurs twice except which occurs once.



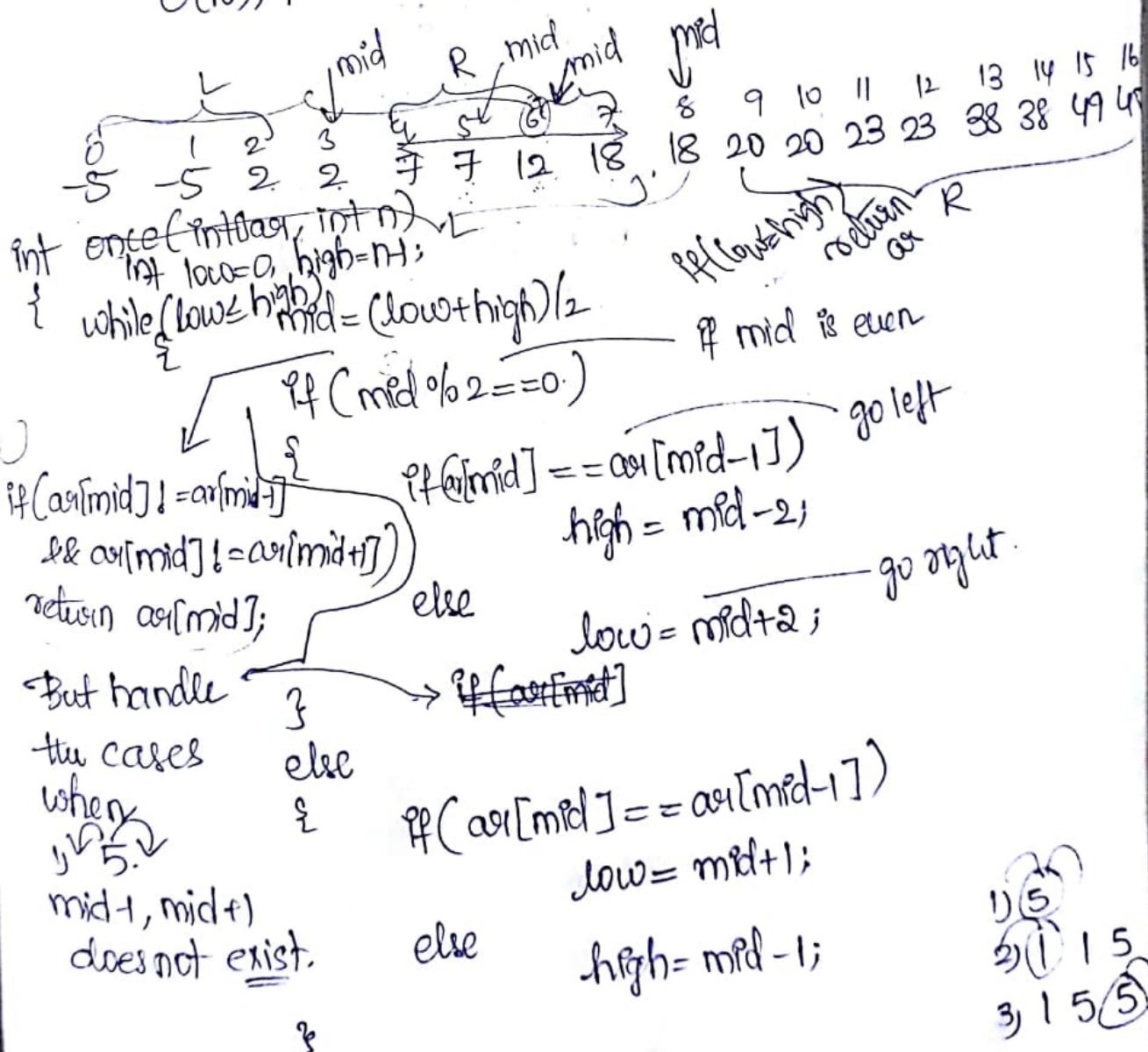
① Brute Force:

Compare $arr[p]$ with $arr[R]$. same skip.
 $O(N)$, 1 diff return.

1) $N, 1 \rightarrow arr[1] == arr[1+1]$
2) $N, 1 \rightarrow \text{XOR}$.

② XOR of all elements.

$O(N)$, 1



1) 5
2) 1 5
3) 1 5 5

int occursOnce (int[] arr, int n)

{
 int low=0, high=n-1;

 while (low <= high)

 {
 int mid = (low+high)/2;
 if ((arr[mid] != arr[mid-1]) && (arr[mid] != arr[mid+1]))

 return arr[mid];

 if (mid % 2 == 0)

 {
 if (arr[mid] == arr[mid-1])

 high = mid - 2;

 else
 low = mid + 2;

 }

 else

 if (arr[mid] == arr[mid-1])

 low = mid + 1;

 else

 high = mid - 1;

}

}

④ Finding floor:

N elements.

a[N]: 10 3 -5 12 20 9 -6 53 -15 7 15.

Ques: $x: \max_{i=0}^{N-1}(a[i]) \leq x$

<limits.h> Cpp.

S: $\frac{K}{-8} \Rightarrow -15$

No.of
queries: 18 \Rightarrow 15

-5 \Rightarrow -5

12 \Rightarrow 12

100 \Rightarrow 53

-25 $\Rightarrow -2^{31}$ (Integer.MIN_VALUE).

Java
Integer.MIN_VALUE
 $= -2^{31}$

①. Iterate & find the ele $< k \Rightarrow o(Q * N)$.

int findingFloor(int[] a, int n, int k)

{
 int floor = Integer.MIN_VALUE;

 for (int i=0; i<n; i++)

 {
 if ((a[i] <= k) && (floor > k))

 floor = a[i];

 }

 return floor;

};

②. Sort and check.

int findingFloor(int[] a, int n, int k)

{
 int low=0, high=n-1;

 mergeSort(a, low, high);

 modifiedBinarySearch(a, n, k, 0, n-1);

};

int modified Binary Search (int arr[], int n, int k, int low, int high)

{
 int floor = Integer.MIN_VALUE;

 while (low <= high)

 {
 int mid = $\frac{(low+high)}{2}$;

 if (arr[mid] < k)

 {
 low = mid + 1;

 if (arr[mid] < k) {

 if (arr[mid+1] > k))

 return arr[mid];

}

 if (arr[mid] > k)

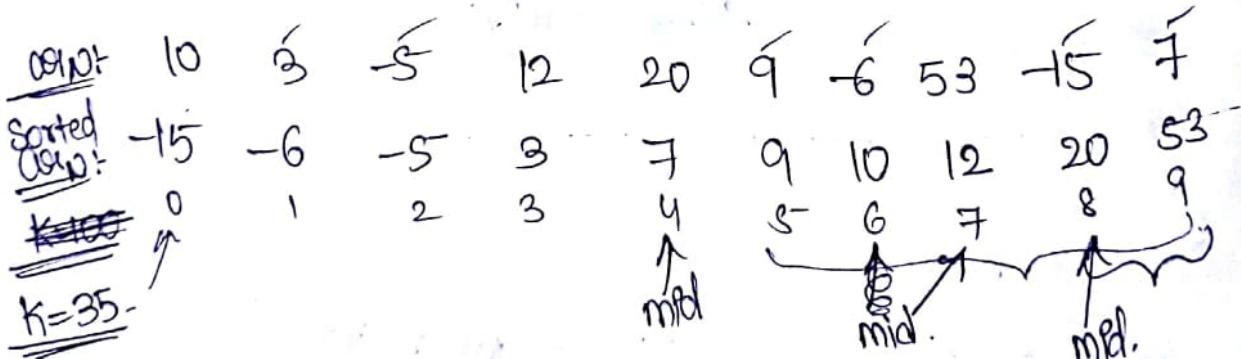
 {
 high = mid - 1;

}

 if (arr[mid] == k)

 return arr[mid];

}



```

int findingFloor(int arr[], int N, int x)
{
    int ans = Integer.MIN_VALUE;
    while (low <= high)
        if (arr[mid] > x) high = mid - 1
        else // (arr[mid] <= x)
    {
        ans = arr[mid];
        low = mid + 1;
    }
    return ans;
}

```

```

int findingfloor(int[] arr, int N, int x)
{
    int low = 0, high = N - 1; mergesort(arr); -N log N
    int ans = Integer.MIN_VALUE;
    while (low <= high)
    {
        int mid = (low + high) / 2;
        if (arr[mid] > x)
        {
            high = m - 1;
        }
        else
        {
            ans = arr[mid];
            low = mid + 1;
        }
    }
    return ans;
}

```

Comp: $O(N \log N + Q \log N)$, $O(1)$.

Q Given 2 arrays sorted. Finding Floor for Array of Queries

Array arr_N: -15 3 -5 -6 7 9 10 12 15 20 53

Ans: -18 18 -5 12 100 -25.

Key array other array: -25 -8 -5 12 18 100.
Queried array.

Sorted Arrays:

Two Pointers
arr_N: -15 -6 -5 3 7 9 10 12 15 20 53

arr_Q: -25, -8, -5, 12, 18, 100.

ans_Q: -18, -5, 12, 18, 100.

void findingFloorForArr_Nof2Queries (int arr_N, int arr_Q) {

int ans = Integer.MIN_VALUE; P1=0, P2=0;

while (P1 < N && P2 < Q)

{ if (arr_N[P1] > arr_Q[P2])

ans_Q[P2] = ans;

P2++;

while (arr_N[P1] ≤ arr_Q[P2])

ans_Q[P2] = arr_N[P1];

P1++;

if (arr_Q[P2] > arr_N[P1])

{ ans_Q[P2] = ans_Q[P2-1];

P1++;

P2++;

```

int p1=0, p2=0 value=Integer.MIN_VALUE;
while(p1 < N && p2 < Q)
{
    if (arr(p1) <= arr(p2))
        ans(p2) = arr(p1);
    p1++;
}
else
    p2++;

```

3)

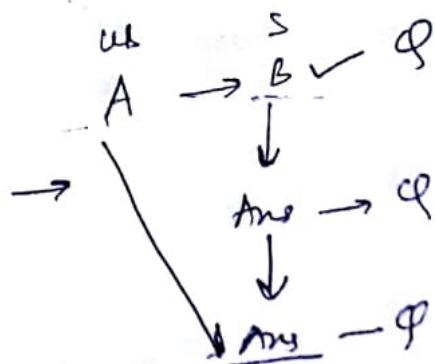
1) sorting

$N \log N + Q \log Q + N + Q + Q \log Q$, $3Q$

copy & sorting
query array
into Cr.
array.

barray copy
into Cr. array.
ans arr.

sorted query array



④ Frequency Sort :-

$\text{arr}_N = 1 \ 5 \ 1 \ 10 \ 12 \ -1 \ 3 \ 5 \ 5 \ 1 \ 5 \ -6 \ 18 \ 5 \ 83 \ -1$

$\alpha : \text{count}(x)$

$$3 \Rightarrow 1$$

$$5 \Rightarrow 5$$

$$-1 \Rightarrow 2$$

$$12 \Rightarrow 1$$

$$-4 \Rightarrow 0$$

Time Space

① $O(N), 1$

② Count Sort

$$R = x = \text{high value} - \text{low value} + 1$$

$N + O, R$ not always best.

⑤ Sorted, Binary Search, linearCount BS(5)

$\text{arr}_N = -6 \ -1 \ -1 \ 1 \ 1 \ 1 \ 3 \ 5 \ 5 \ 5 \ 5 \ 5 \ 5 \ 10 \ 12 \ 18 \ 23$

Sort: $N \log N + O(\log N) + N$, 1.

sort: \uparrow
Binary
Search

\uparrow
Count++

\uparrow
left & right.

(Linear count)

sorted: $-6 \ -1 \ -1 \ 1 \ 1 \ 1 \ 3 \ 5 \ 5 \ 5 \ 5 \ 5 \ 5 \ 10 \ 12 \ 18 \ 23$

\uparrow
 $\text{low} = \text{mid} + 1$

$P_1 \uparrow$
BinarySearch1
BS1 to find
first occurrence
of 5.

\uparrow
go left mid.
 \uparrow
 $\text{ans} = \text{mid}$
 \uparrow
 $\text{high} = \text{mid} - 1$

$P_2 \uparrow$
BS2 to find
last occurrence
of 5.

for BS2 go right

~~ans = mid~~

~~high =~~

\uparrow
 $\text{ans} = \text{mid}$

\uparrow
 $\text{low} = \text{mid} + 1$

$$P_1 = \text{BS1}(\text{arr}, N, x)$$

$$P_2 = \text{BS2}(\text{arr}, N, x)$$

$$\text{Count} = P_2 - P_1 + 1;$$

If ($\text{ans} == -1$)

```

int BS1(int arr[], int N, int x)
{
    int low=0, high=N-1, ans=-1;
    while (low <= high)
    {
        int mid=(low+high)/2;
        if (arr[mid] > x)
            low=mid+1;
        else if (arr[mid] < x)
            high=mid-1;
        else // (arr[mid] == x)
        {
            ans=mid;
            high=mid-1; // BS1 []
        }
    }
    return ans;
}

```

0 1 2 3 4 5
 | 5 5 5 5 5 5

↑
 mid.
 ans=mid

Q. array of N, - } sorting. 4) $N \log_2 N + N + Q \times \log_2 N$, N
 Array of Queries. - }

Given Get the frequency count in an array of queries
 $a_{1:N}$: with each unique element.

Given $a_{1:N} = -6 -1 1 1 1 3 5 5 5 5 10 12 18 23$

Return
 Query A_N: -6 -1 1 1 3 5 10 12 18 23
 Array
 frequency B_N: 1 2 3 1 5 1 1 1 1

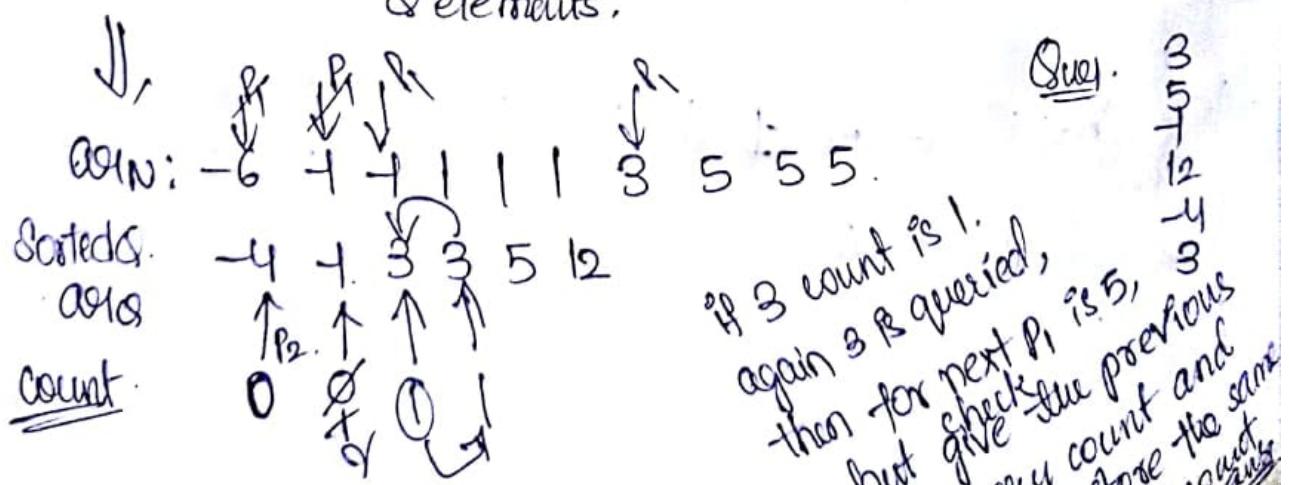
A_N : given sorted array.

B_N : sorted unique elements frequency count.

```
void sortedQueryFrequency(int[] AN, int[] BN)  
{    int x = AN[0], A[0] = x, B[0] = 1, pdx = 0  
    for (int i = 1; i < N; i++)  
    {        if (AN[i] == x)  
        {            B[pdx]++;  
        }        else  
        {            x = AN[i];  
            pdx++;  
            A[pdx] = x;  
            B[pdx] = 1;  
        }    }}
```

$N \log N$ - sort array
+ $Q \log Q$ - sort query
+ $N + Q$ - Two pointers concept
+ $Q \log Q$ - Binary search for
 Q elements.

Time:



Minimizing the Absolute Difference:

AN: 15 10 2 25 43
BN: 7 18 90 5 35

CN: 29 12 19 40 30 \Rightarrow sort 12 19

$$\text{Find } \min(\max(A(i), B(j), C(k)) - \min(A(i), B(j), C(k)))$$

Brute Force: Consider All 3,3 combinations.

(1) Brute force: $O(N^3)$, 1

(2) sorting AN — fix ele. {combinations.

sort CN: —

12 15 18 19

$$x = \min(A(i), B(j))$$

$$y = \max(A(i), B(j))$$

find ele from C st. $x \leq \text{ele} \leq y$

3 in b/w x & y

for every fixed ele, binary search for C ele: $\log N$

$$\Rightarrow O(N \log_2 N + N^2 \cdot \log_2 N), 1.$$

$x \leq z \leq y$

BS, find floor of inc

& find ceil of inc.
take the least diff.

(3). Sort all the three arrays (pointers)

AN: $\downarrow P_1$ $\downarrow P_1$ $\downarrow P_1$ $\downarrow P_1$ $\downarrow P_1$
2 10 15 25 43
 $\downarrow P_2$ $\downarrow P_2$ $\downarrow P_2$ $\downarrow P_2$
5 7 18 20 35
 $\downarrow P_3$ $\downarrow P_3$ $\downarrow P_3$
12 19 29

$\min = 10$
7 5 4
8 > 5
New update

$$x = \max(2, 5, 12) = 12, y = \min(2, 5, 12) = 2, \text{diff} = 12 - 2 = 10$$

always inc min pointer. to get the min. diff.

$3N \log N + 3N$.

Sorting 3 arrays. for traversing all.

Q) Given all positive elements with no duplicates.
Find the first missing positive element.

Ex. ①. $\text{arr}[N]: 1 \ 2 \ 3 \ 4 \ 5 \ 8 \ 9 \ 12 \ 15$
ans: 6 is missing;

②. $\text{arr}[N]: 1 \ 2 \ 3 \ 4$ ans: 5

③. $\text{arr}: 1 \ 2 \ 3 \ 4 \ 5 \ 6 \neq 18$ ans: 1.
1 missing.

Min ans: 1, max ans: $N+1$ - the; $[1, N+1]$

① $O((N+1) * N)$, 1, $N+1$ elements, linear search each. N .

② $O((N+1) * \log_2 N)$, 1. Binary search for $N+1$ elements.

③. At index i , should have $i+1$.

for ($i=0; i < N; i++$)
{ If ($\text{arr}(i) \neq i+1$)

 return $i+1$;

return $N+1$;

- $O(N)$,

0 1 2 3 4 5 6 7 8
1 2 3 4 5 8 9 12 15

④. int findFirstPositiveMissing (int arr[])

{ int ans = N, low = 0, high = N-1;
while (low <= high)

{ mid = (low + high) / 2;

if (arr[mid] == mid + 1)

 low = mid + 1;

else

 ans = mid;

 high = mid - 1;

} return ans + 1;

0 1 2 3 4 5 6 7 8 9 10 11 12 13
 8 10 11 12 13
 8 10 11 12 13

Q) find First Positive Missing (int a[7])

{ ans = -1, low = 0, high = N-1;

while (low <= high)

{ mid = (low+high)/2

if (a[mid] == mid+1)

{ ans = mid;

low = mid+1;

{

else

{ high = mid-1;

{

return ans+2;

~~O(logN), 1~~

* Give array of all positive & negative elements.

Find the first positive elements.

① Linear search for N+1 ele. ✓ $O((N+1)N), 1$

② Binary search for N+1 ele ✓ $O((N+1) \log N), 1$

0 1 2 3 4 5 6 7 8 9 10
 -5 -2 1 1 2 3 4 7 10 11 12

ans = 5.

2) -10 -9 -2 0 1 2 5 6 7. ans = 3.

3) -2 -1 3 4 ans = 1.

To Do:

~~1~~

④ Find the Square Root

①. int sqrt(int N) will not go till N

{ for (int q=1; q<=N; q++)

{ if (q*q == N)

 return q;

Stop when q is found.
at \sqrt{N} place.

N=25

Comp: $O(\sqrt{N})$, O(1)

min. ans = 1, max ans = sqrt

②. int sqrt(int N)

{

int low=0, high=N, ans=1.

while (low <= high)

{

mid = (low+high)/2;

if ((mid*mid) == N)

 return mid;

else if ((mid*mid) > N)

{

 high = mid-1;

}

else

{

 ans = mid, store ans.

 low = mid+1;

}

N=25

low high mid

0 25 13

0 12 6

4 5 3

5 5 4

5
ans.

low high mid

0 25 13

0 12 6

0 5 2

Comp
 $\Rightarrow O(\log N)$, O(1)

chart()
→ check
mid*mid=N.

If there are N ele.

then binary search: $\log N$

If there are $N \log N$ ele.

\Rightarrow binary search: $\log(\log N)$

Q) Job: N Tasks each take some time given in array

$1 \ 5 \ 8 \ 2 \ 13 \ 18 \ 12 \ 7 \ 8 \ 9 \ 10$

Given: K no. of workers. Each worker is assigned only tasks. Find the min time in which all the tasks get complete.

K=4?

g. $\begin{array}{|c|c|c|c|} \hline 1 & 5 & 8 & | & 2 & 13 & | & 1 & 8 & 12 & | & 1 & 5 & 3 & 6 \\ \hline w_1 = 14 & w_2 = 15 & & w_3 = 21 & & & w_4 = 15 & & & & & & & & & \\ \hline \end{array}$

$\max(\{14, 15, 21, 15\}) = 21$. Ans. 21. (But not minimal).

$$C_{\max} = 21, \text{ ans} = 21$$

\downarrow

$\begin{array}{|c|c|c|c|} \hline 1 & 5 & 8 & | & 2 & 13 & 1 & | & 8 & 12 & | & 1 & 5 & 3 & 6 \\ \hline 14 & & & 16 & & & 20 & & & & & 15 & & & & \\ \hline \end{array}$

Ans = 20 (minimal) $\Leftarrow \text{ans} = \min_{21, 20} (\text{ans}, C_{\max})$

⇒ 20.

$\begin{array}{|c|c|c|c|c|c|} \hline 1 & 5 & 8 & 2 & 13 & 1 & 8 & 12 & 1 & 5 & 3 & 6 \\ \hline \textcircled{1} 14 & \textcircled{2} 15 & \textcircled{3} 9 & \textcircled{4} 13 & \textcircled{5} 14 & & & & & & & \\ \hline \end{array}$

Ans = 15. (Minimal) \Leftarrow

Brute Force

Out of $N-1$ positions, chooses $k-1$ positions. \Rightarrow partitions.

k nested loops

$$\text{ans} = \min(\text{ans}, C_{\max})$$

$1 \ 5 \ 8 \ 2 \ 13 \ | \ 8 \ | \ 12 \ | \ 5 \ 3 \ 6$
 maximum possible answer = sum of all ele.
 minimum possible answer = max(all ele).

	low	high	mid.
13.	65	89	
13	38	25	

$$\frac{13+38}{2}$$

2
14
2
14

int findMinTime(int arr[], int k)

{
 int ans =

 int low = max(arr), high = sum(arr);

 while (low <= high)

{

 mid = (low + high) / 2;

 if (valid(arr, N, k, mid))

{

 ans = mid;

 high = mid - 1;

{
 else

{

 low = mid + 1;

{

} return ans;

Complexity: $O(N \log S)$, $O(1)$.

valid(arr, N, k, m) \rightarrow iterates through all the partitions and checks if the sum \leq mid, counts the no. of partitions. deck
back
front
valid

If no. of partitions $< k \Rightarrow$ valid
else \Rightarrow invalid.

$O(N)$

BS.
O(log S)
for finding
sum of
all ele.

$O(\log S)$

Given array of house positions. Q1. N.
k friends. minimize the distance between friends.

\Rightarrow Minimize the maximum distance between 2 friends.

Object: Goal: Path: minimize Distance Between friends (int cost[])

```
int ans=0;
for( int i=0; i<N-K; i++)
{
    sum=0; max
    for( int j=i; j<i+K; j++)
        sum=sum+a[j];
    if( sum>ans)
        ans=sum;
}
```

3 return ans;

$$O((N-K+1) * K), 1$$

```

int diffMin( int a[], int k )
{
    int ans;
    for( int i=0; i<n-k; i++ )
    {
        ans maxdiff=0;
        for( int j=i; j<i+k; j++ )
        {
            diff = a[j] - a[j+1];
            if( diff > maxdiff )
                maxdiff = diff;
        }
        ans = min( ans, maxdiff );
    }
    return ans;
}

```

④ Maximise the ^{min} distance between k friends:

arr: 5, 11, 15, 23, 24, 29, 40, 48, 52, 59, 65, 69, 73
 $k=6$.

$$\min = 4.$$

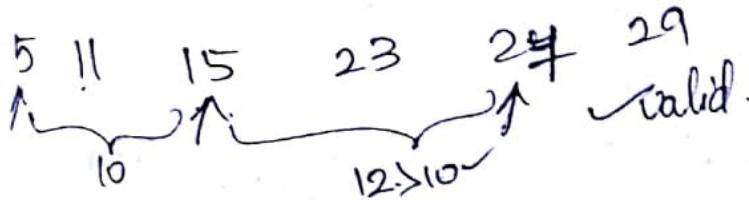
maximise this \min .

1) $N \times k - 1$ If fixed first & last pos. fixed.
 \uparrow \uparrow \uparrow
 k partitions diff ? \uparrow
 $N-2$

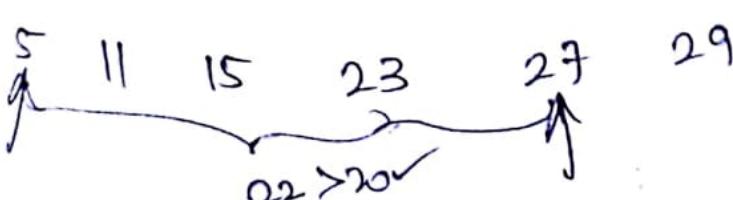
max the Min Distance (int a[], int k) e.g. ans=10

```
int ans = c;
int low = 0; // min diff b/w Two Adjacent Elements;
int high = a[N-1] - a[0];
while (low <= high)
{
    mid = (low + high) / 2;
    if (valid(a, N, K, m))
    {
        ans = mid;
        low = mid + 1;
    }
    else
    {
        high = mid - 1;
    }
}
```

e.g. ans=10. k=3



ans=20 k=3



valid:

diff b/w selected positions should be \geq mid, each time and if $k=0 \& k < 0$ valid else Not valid.

cannot place third friend.
hence not valid.

* Find the Median of Two Sorted Arrays such that
 $N+m = 0$

A_N: -5 -2 4 18 25

B_m: -8 -1 0 7 10 15 23 31

-8 -5 -2 1 0 4 7 10 15 18 23 25 31
Median.

① Brute Force.

Merge the sorted array into C_{N+m}:

$\Rightarrow O(N+m)$, $O(N+m)$.

② Take two pointers,

inc pointer of the smaller element

inc count till $(N+m)/2$

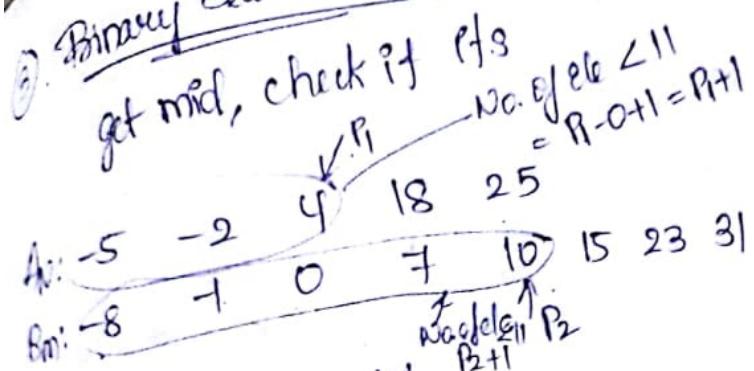
ans = count.

$\Rightarrow O(N+m)$, $O(1)$

```
int median(int a[], int b[])
{
    int p1=0, p2=0;
    while(p1<a.length && p2<b.length)
    {
        if(a[p1]<=b[p2])
        {
            p1++;
            count++;
        }
        else
        {
            p2++;
            count++;
        }
    }
    return a[count];
}
```

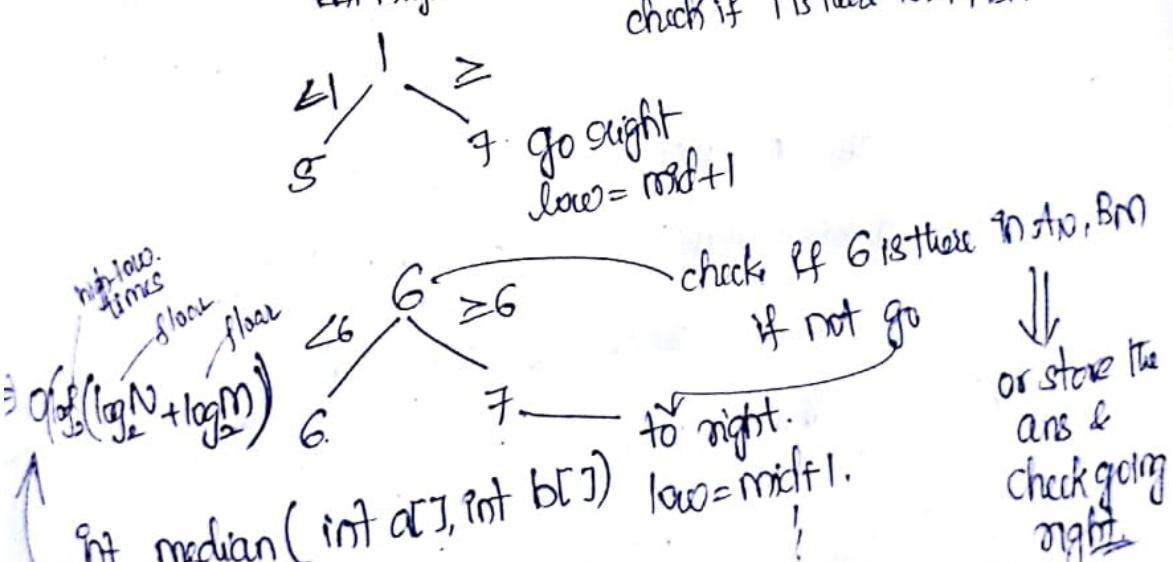
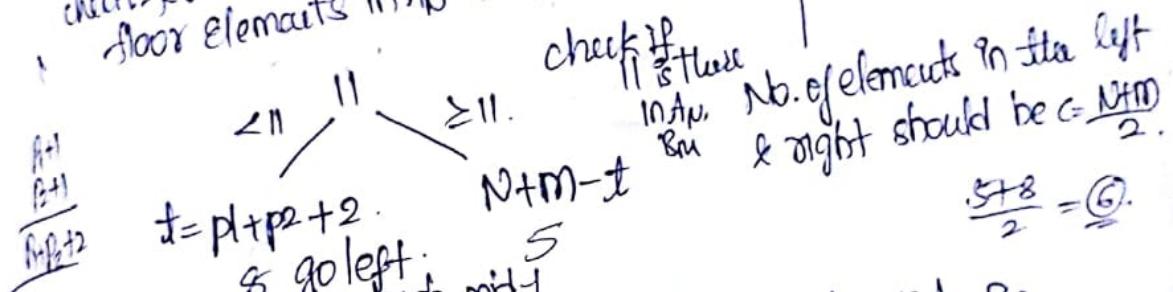
Binary Search Usage

get mid, check if P1



check for 11.
floor elements present
in A₀ & B₀.

low	high	mid
-8	31	11
-8	10	1
2	10	6
7	10	8



int low = min(A[0], B[0]);
int high = max(A[N-1], B[M-1]), K = $\frac{N+M}{2}$

while (low <= high)

{
 mid = (low + high) / 2;

 less = max(LessEle, floor(A, N, mid) + floor(B, M, mid));

 if (less == K)

 {
 ans = mid;
 go right // check if its there in A or B
 if yes return
 else go right
 continue search.

 if (less < K)

 low = mid + 1;

 if (less > K)

 high = mid - 1;

④ Median for the arrays with repeated numbers.

$$\frac{N+M}{2} = \underline{\underline{100}}$$

Eg. $N+M=201$.

If $\underbrace{-3 -3 -3}_{20 \text{ ele} \leq 3}, \quad \boxed{181}$

→ go right & search.

If $\underbrace{0 0 0 0}_{101 \text{ element}} \quad \boxed{100 \text{ ele}}$

store 0 as ans : $ans=0$.

and continue searching
left as mid can be in
those 101 elements.

① Jobs : Cabinet Partitioning, HR-SI.

② Houses - friends, Max the min dist; SPOJ : Aggressive Cows.

③ Median: InterviewBit - Searching - Median of N arrays.

④ Minimizing Absolute Diff : IB. - 2 pointer.

⑤ Sqrt - IB - Searching.
Obst - HR-SI.

HASHING

JAVA: { Unsorted keys } Hash Map, Hash Set { Unsorted }

{ Sorted keys } Tree Map, TreeSet { Sorted }

C++: Unordered-Map, Unordered-Set
Regular-Map, Regular-Set.

Python: Dictionaries, Sets. {}

cannot update keys, but we can update values / <key, value> unique

Set

<key> // Mathematical set
unique only unique data

(To-Do)
Exercises

arr: 1, 5, 2, 1, 1, 15, 18, -6, -6, 1, 18, 5, 20, -1, -1.

- Insert all elements in.
- 1) Unsorted Set - print (unsorted)
 - 2) Sorted Set - print (sorted out)
 - 3) HashMap, Set - ele, freq. { Unsorted }
 - 4) TreeMap, Sets - sorted ele, freq.

java:- docs.oracle.com
cpp: cplusplus.com
python: python.org

C $\xrightarrow{\text{C++}}$ #include <stdio.h>
 C $\xrightarrow{\text{Java}}$] #include <math.h>

C $\xrightarrow{\text{C++}}$ #include <cmath.h>

C $\xrightarrow{\text{C++}}$ #include <cmath.h>

10/3/19

List of Phone Numbers: 10-digit:-

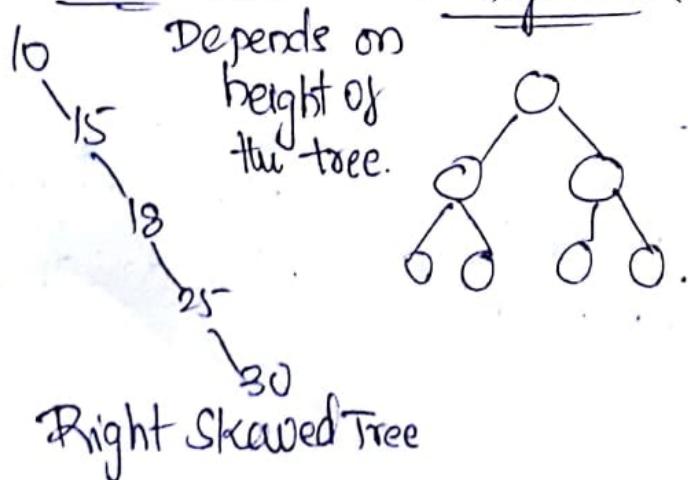
<u>Solution</u>	<u>Insert(α)</u>	<u>Search(α)</u>	<u>Delete(α)</u>
1) Unsorted Array	$O(1)$	$O(N)$	$O(N)$
2) Sorted Array	$O(N)$ Insertion Sort	$O(\log N)$ Binary Search	$O(N)$
3) Unsorted Single Linked List	$O(1)$ at last pointer.	$O(N)$ Linear Search	$O(N)$
4) Binary Search Tree.	$O(H)$	$O(H)$	$O(H)$
H=Height	H=N (worst) Right/ Left Skewed.	H= $\log N$ (Avg) Balanced BST AVL Red-Black.	
5) Hashing (DAT).	$O(1)$ (Direct Access Table) count=T	$O(1)$ count==T or F	$O(1)$ count=F

But we need $\text{bool count}[10^0] = \{F\}$. \Rightarrow space = $10^{10} = 10\text{GB}$.

— DAT has wastage of space.

Mapping bigger element to smaller index can get rid of this problem.

BST Worst Case: $O(N)$. BST Avg. Case: $O(\log N)$



Hashing: $\text{hash}(x) = x \% 10$ → Size of Hash Table
 Hash Value
 Hash Key

Hash Function.	0	1	2	3	4	5	6	7	8	9	
Table N=10	55		53	14	45	4	27	126	84		
Collision											

Insert - $I(x)$
 Search - $S(x)$
 Delete - $D(x)$

Data :- 55, 45, 14, 27 (4), 126, 84, 55... hash(x) = (x+1) % 10
 i starts with 0 & checks empty slot.

Observation :- Two different hash keys gives the same hash value.

Collision Resolutions :-

- 1) Separate Chaining : Insert element at next available empty index
- 2) Open Addressing :

(i) Linear Probing : Insertion: We start from index first. We continue searching until we find the key or we find the empty slot.

While deleting, we just delete the element, but this gives problem while searching as the deleted item slot becomes empty and searching stops at the empty slot.

To overcome this, mark the deleted item slot as 'deleted slot', and searching should be continued when deleted slot is found.

avg {
 (i) $I(x)$
 (ii) $S(x)$
 (iii) $D(x)$
 } - $O(P)$ } P - No of Probes.
 All three should ~~not~~ have empty slots
 i.e. #EmptySlots $\neq 0$,

- Insertion can't be done when #Empty slots $\#E = 0$
- Searching continues until the empty slot or element is found, if no. of empty slots are '0', then issues searching.
- Deletion depends on searching.
- If we just double the hash table size, then the no. of empty slots is N, everything becomes possible.

$Size(HT) = N \propto N$ double it.
 50% Empty slots
 If we put N ele. in $2N$ slots.

50% slots are empty.
 Some space is wasted.
 but time is saved.

$$M=100.$$

$$\text{hash}(x) = (x+i) \% m$$

Linear Probing

0	1	2	3	4	5	6	7	8	9	10	11	12	13
			103	503	1203	803	88	4	104	504	1204	804	1503

Problems of Linear Probing:-

(3, 103, 503, 1203, 803) - 1-cluster of collisions.

(4, 104, 504, 1204, 804) - 1-cluster of collisions.

But both of these clusters combine and cause a bigger cluster of collisions, and all the other to be inserted elements should go through 10 no. of collisions.

(2) Quadratic Probing. If $m=100$.

$$\text{hash}(x) = (x+i^2) \% 100$$

5, 103, 503, 1203, 803, 4, 104, 504, 1204, 804, 1503

$$\downarrow \quad 3 \rightarrow 4 \rightarrow 7 \rightarrow 12 \rightarrow 19 \rightarrow 5 \rightarrow 8 \rightarrow 13 \rightarrow 20 \rightarrow 29 \rightarrow 28$$

$$\begin{aligned} (103+0^2) \% 100 \\ (103+1^2) \% 100 \\ (103+2^2) \% 100 \\ = 4. \end{aligned}$$

Here the number of collisions decrease.

$$503: \quad 503 \% 100 = 3 \text{ occupied}$$

$$(503+0^2) \% 100 = 3 \text{ occupied}$$

$$(503+1^2) \% 100 = 4 \text{ occupied}$$

$$(503+2^2) \% 100 = 7 \text{ empty}$$

put 503 in 7.

$\underbrace{}_{0^2}, \underbrace{}_{1^2}, \underbrace{}_{2^2}$

$$I(x) = P_1$$

$$S(x) = P_{-1}$$

$$D(x) = P_{-2}$$

Avg Case: $O(1)$

(3) Double Hashing:

— uses two hash functions.

$$h_1(x) = (x+i) \% m \quad \approx \text{some func.}$$

$$h_2(x) = (x+i^2) \% m \quad \approx \text{some func.}$$

$$\boxed{h(x) = (h_1(x) + h_2(x)) \% M} \quad \text{Double Hashing.}$$

$$\text{Let: } h_1(x) = x \% m$$

$$h_2(x) = \text{sum of digits}(x)$$

$$\begin{array}{c} \downarrow \\ 3, 103, 503, 1203, 803, 4, 104, 504 \\ \downarrow \quad 6 \rightarrow 7 \rightarrow 1, \rightarrow 9 \rightarrow 14 \rightarrow 8 \rightarrow 24 \rightarrow 13 \end{array}$$

$$\begin{array}{l} (x+h_2) \% m \\ (3+3) \% 100 \\ = 6 \end{array}$$

$$\begin{array}{l} (x+h_2) \% m \\ 3+8 \% 100 \\ = 11 \end{array}$$

$$\begin{array}{l} (x+h_2) \% m \\ 3+6 \% 100 \\ = 9 \end{array}$$

$$\begin{array}{l} (x+h_2) \% m \\ 3+11 \% 100 \\ = 14 \end{array}$$

$$\begin{array}{l} 4+4 \% 100 \\ = 8 \end{array}$$

$$\begin{array}{l} 4+5 \% 9 \\ = 9 \end{array}$$

7 collisions

$$4+4 \% 5 = 29.$$

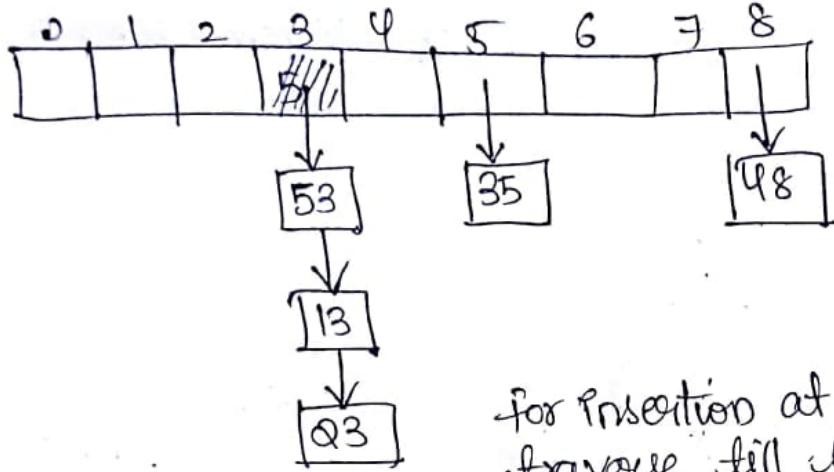
$$\begin{array}{l} (h_1(x) + h_2(x)) \% m \\ (4+4 \% 5) \% 100 \end{array}$$

$$504 \rightarrow 4+9 \% 100 \quad \text{collide} \quad 4+9 \% 100$$

a) Separate Chaining:

- Insert the element at index position, if collision create new linked list.

53, 35, 13, 48, 23

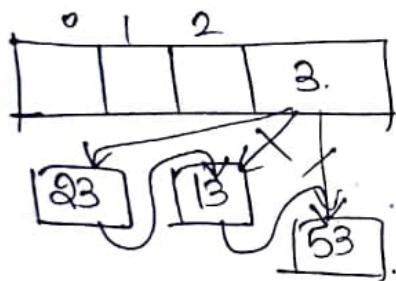


$$I(x) = O(1)$$

$$S(x) = O(L) \quad L = \text{length}$$

$$D(x) = O(L) \quad L = \text{length}$$

for insertion at last, we need to traverse till the tail last,
instead we can insert at first
and link its pointer to previous



Ideal Hash Function:

- 1) Uniformly Distributed keys over hash table.
- 2) Hash function should be simple compute.

$$\begin{aligned}
 h(x) &= \text{sum of digits} \\
 &= ax^1 \\
 &= ax^p \\
 &= px
 \end{aligned}
 \quad \left. \begin{array}{l} \text{OK.} \\ \text{takes time to} \\ \text{compute } h. \end{array} \right\}$$

$h(x)$ should be simple.

Duplicates:

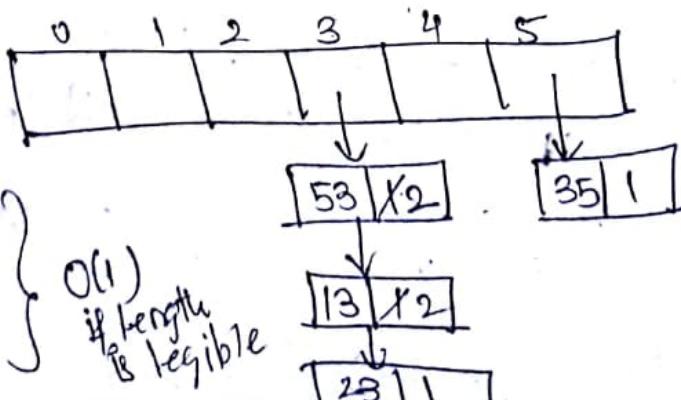
53, 35, 13, 23, 13, 53

Maintain key, & count.

$$I(x) = O(L) \quad \left. \begin{array}{l} \text{Search/Insert} \\ \text{if length is negligible} \end{array} \right\}$$

$$S(x) = O(L)$$

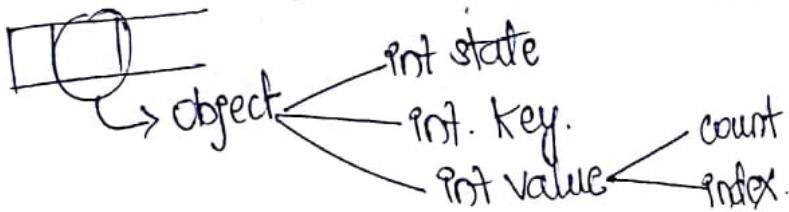
$$D(x) = O(L)$$



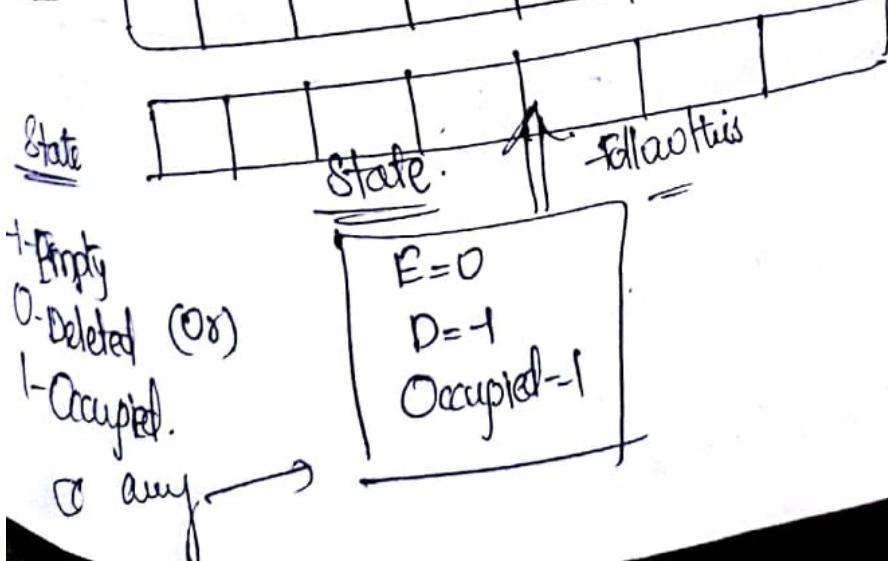
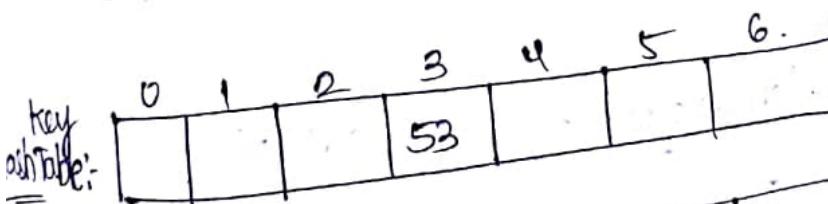
Linear Probing : Handling Duplicates.

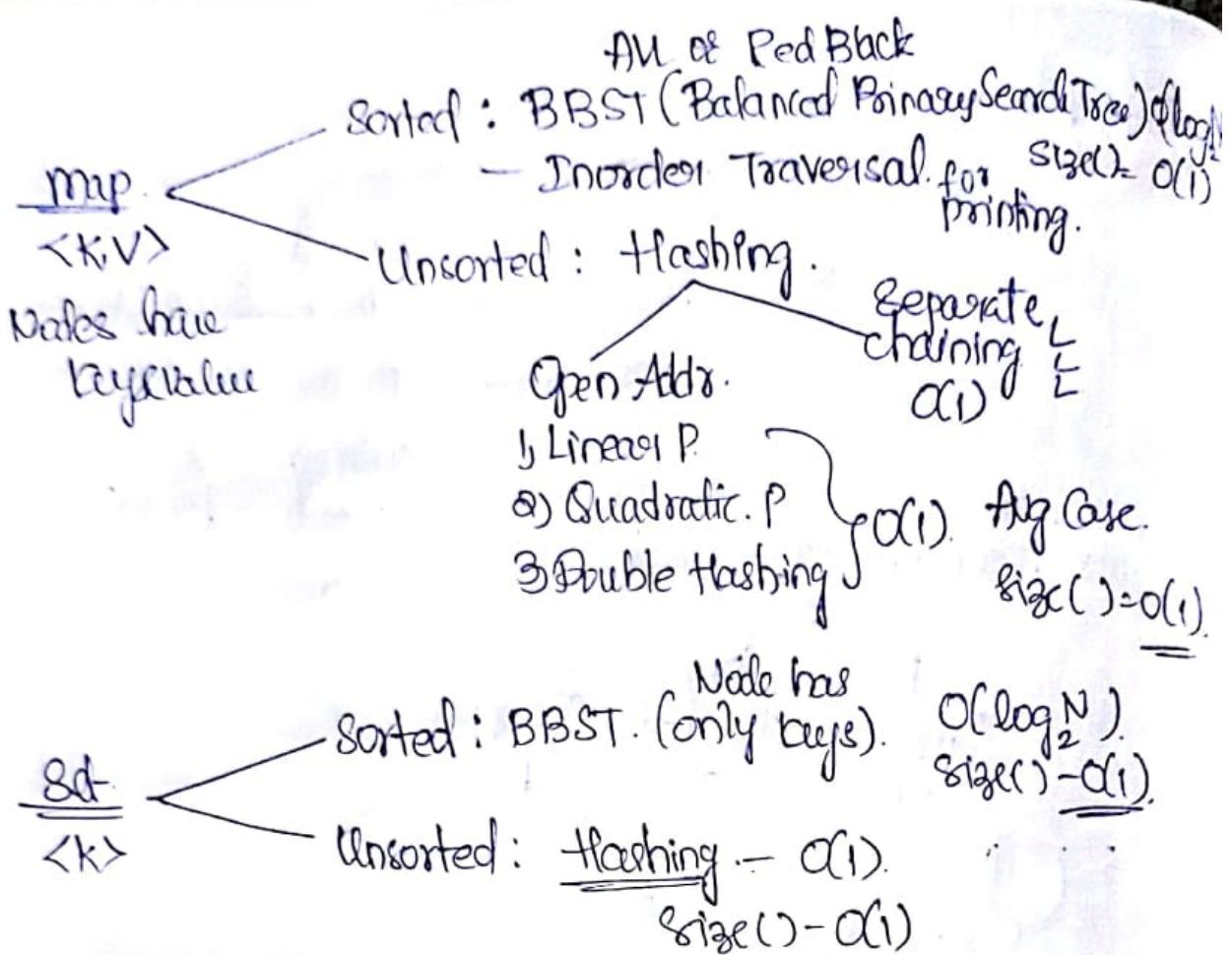
- Maintain count array
- Insertion - Increment count.
- Deletion - decrement count if > 0 .
- Searching - Check the count.
 - count = 0 absent
 - ≥ 1 present.

Also maintain state Table.



I(53), I(5), I(13), I(53),





- Set can be implemented using map.
- Map cannot be implemented using set.

Map
|
Set.

④ HashMap Implementation for Open Addressing :

- A $\text{ht}[m]$ [Object type]. A has
 - key
 - value
 - state
 - 0 empty
 - 1 Ddt
 - 1 Occupied
- int key[m], value[m], state[m];
 - constructor
 - int size();
 - bool search(int x);
 - void insert(int x);
 - void insert(int x);

array's Implementation

class HashTab {

 private {
 public HashTab() { m = 1000; }

 int key[] = new int[m];

 int value[] = new int[m];

 int state[] = new int[m];

 int count = 0;

 value = {0};

 state = {0};



} main()

 {
 x = sc.nextInt();

 map.insert(x); HashMap map = new HashMap();

 map.search(x);

 map.delete(x);

}

void insert(int x).

{

 int hashIndex = x % m;

 while (state[hashIndex] == 0)

 {
 hashIndex++;

 key[hashIndex] = x;

 value[hashIndex]++;

 state[hashIndex] = 1;

 } 0 - Empty

 } 1 - Deleted

 } 1 - Occupied

} out of
 while loop.

 hashIndex++;

 } end of if } .

```

void insert (int x):
{
    int i=0;
    int hashIndex = (x % M);
    while (i < M)
    {
        if (state[(x+i) % M] == 0) || state[(x+i) % M] == -1
        {
            hashIndex = (x+i) % M;
            break;
        }
        i++;
    }
    key[hashIndex] = x;
    value[hashIndex]++;
    state[hashIndex] = 1;
}

```

Do also.
 If element is
 already present
 update value
 count.

```

boolean search (int x)
{
    int i=0;
    int hashIndex = (x + i) % M
    while (i < M)
    {
        if (key[hashIndex] == x)
            return True;
        i++;
    }
}

```

```

void insert(int x)
{
    int idx=0;
    for(int i=0; i<4; i++)
    {
        idx = (x + i * p) % M; occupied
        if (state[idx] == 0 && key[idx] == -1)
        {
            value[idx]++;
            break;
        }
        if (state[idx] == 0 empty || state[idx] == -1) deleted
        {
            key[idx] = x;
            value[idx] = 1;
            state[idx] = 1; occupied.
            break;
        }
    }
}

```

$\Rightarrow N+Q, N$.

④ Sum Of Pairs:

$a[1]: 5 \ 12 \ -6 \ 8 \ 25 \rightarrow 18.$

$$a[i] + a[j] = k, i \neq j.$$

boolean sumOfPairs(int a[], int k)

{

 HashMap<Integer, Integer> map = new HashMap<>();

 for (int i=0; i < a.length; i++)

{

 if (!map.contains(a[i]))

 map.put(a[i], 1);

 else

 map.put(a[i], map.get(a[i]) + 1);

}

 for (int i=0; i < a.length; i++)

{

 int a = a[i];

 int b = k - a;

 if (map.contains(b))

{

 if (a == b && map.get(b) >= 2)

 return true;

 if (a != b)

 return true;

}

 return false;

K=19

<u>a</u>	<u>b</u>
5	14
12	7
-6	25 ←

<u>a</u>	<u>b</u>
15	19
12	12
<u>0</u>	

Hash Map -

O(N+N), O(N)

insert, [↑] for searching till b
Delete.

hash set:

arr: 5 12 -6 8 25 -9 18 12

Ex 24. a b Hash Set. (Only left side ele. to a.)

5 {} False (Not Found)

12 {5} False

-6 {5, 12} False

8 {5, 12, -6} F. 16 not in {5, 12, -6}

25 {-1} {5, 12, -6, 8} F

-9 {5, 12, -6, 8, 25} F

18 {5, 12, -6, 8, 25, -9} F

12 {5, 12, -6, 8, 25, 18} F. True.
found.

this 12 & prev. 12 are different
since hashsets have
unique keys.

1 ∈ fix a (= arr[1])

1 ← Compute b.

1 ← Search b in HashSet.

 └ Found return True

 └ Not found, Insert a in HashSet.

⇒ O(1)

$$\begin{cases}
 a+b=k \\
 a-b=k \\
 a+bc=k \\
 a+b=c \\
 abc=cd
 \end{cases}
 \text{problem}$$

Contiguous Maximum Subarray Sum :-

$a[N] = 5, 4, -10, 6, 10, -3, 4, 7, 5, 2, -25, 10, 4, 1, 2, 7, 1$

Max sum = 23.

- ① Brute force: Generate all subarrays.
Find other sums of subarrays.
Update maximum sum.

int maxSum(int arr[])

#N #SubArrays

No. of SubArrays: $\boxed{a} \Rightarrow 1 \Rightarrow 1$

$\boxed{a b} \Rightarrow 2 \Rightarrow 3$

$\boxed{a b c} \Rightarrow 3 \Rightarrow 6$

$\boxed{a b c d} \Rightarrow 4 \Rightarrow 10$

$\boxed{a b c d e} \Rightarrow 5 \Rightarrow 15$

$$\boxed{\text{No. of Subarrays} = \frac{N(N+1)}{2}}$$

sub length
 area
 #1 = 5
 #2 = 4
 #3 = 3
 #4 = 2
 #5 = 1. }
 $\left. \begin{matrix} 5+4+3+ \\ = 15. \\ \frac{N(N+1)}{2} \end{matrix} \right\}$

int maxSum(int arr[])

{ int sum=0; max = Integer.MIN_VALUE;
for (int i=0; i<arr.length; i++)

{ sum=0; j=i+1;

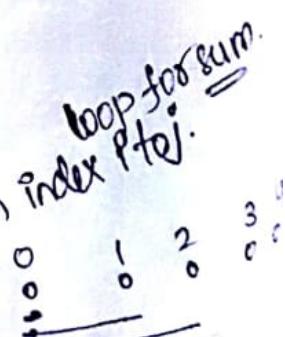
for (int j=0; j < arr.length; j++)

{ sum += arr[j];

if (max > sum)

{ max = sum; }

return max;



```

int maxSum(int arr[], int N)
{
    int sum=0, ans=Integer.MIN_VALUE;
    for(int i=0; i<N; i++)
    {
        for(int j=i; j<N; j++)
        {
            int s=0;
            for(int k=i; k<=j; k++)
                s=s+arr[k];
            ans=max(ans, s);
        }
    }
    return ans;
}

```

Bonita Force.

$O(N^3), O(1)$.

$\frac{9}{5}$ $\frac{4}{5}$	$\frac{5}{4}$ $\frac{-10}{5}$	$\frac{6}{-10}$ $\frac{10}{-3}$	$\frac{-3}{4}$ $\frac{4}{-1}$
--------------------------------	----------------------------------	------------------------------------	----------------------------------

ans is the forev sum got.

Carry forward
the forev.sum.

```

int maxSum(int arr[])
{
    int ans=Integer.MIN_VALUE;
    for(int i=0; i<N; i++)
    {
        int s=0;
        for(int j=i; j<N; j++)
        {
            s=s+arr[j];
            ans=max(ans, s);
        }
    }
    return ans;
}

```

* Generating Subsets can be only done on array with unique elements.

arr: 5 12 -1 4 5 13
 ~~dup~~

SubArrays: $\frac{N(N+1)}{2}$

Subsets: X No duplicates should be there.

Subsets for unique elements 2^N . Let subset be {5, -1, 13} = {13, -1, 5}

Subsequences: - Should maintain order.

- can contain duplicates. 5 -1 13 → {-1, 13, 5}

No. of Subsequences: 2^N .

∅ @ 1 → 1+∅=2

∅ @ ∅ @ ∅ → 2+1=4.

∅ @ ∅ @ ∅ @ ∅ → a b c abc
 ab bc ac ∅

2^N

∅ is also subset, subsequence.

#Subsets = 2^N

#subArrays = $\frac{N(N+1)}{2}$

#subseq. = 2^N

* Count the number of Non-Decreasing Subsequences.

① Brute force:- Generate all subsequences.

- check if subseq. is non-dec order.

```
int nonDecSubSeq(int arr[], int N)
```

```
{ for (int i=0; i<(1<<N); i++)
```

```
{ for (int j=0; j<N; j++)
```

```
    if ((1<<j)&i) b=0;
```

```
    if (j>0) prev=j;
```

```
    if (arr[prev] > arr[j]; count++;
```

```

int nonDecSubSeq( int arr[], int N )
{
    int count = 0;
    for( int i=0; i<(1<<N); i++ )
    {
        int ok = 0;
        int prev = Integer.MIN_VALUE;
        for( int j=0; j<N; j++ )
        {
            if(((1<<j)&i)!=0)
            {
                if( prev != 10 && prev != j )
                {
                    if( arr[j] > arr[prev] )
                        ok = 1;
                }
                prev = j;
            }
            if( ok == 1 )
                count++;
        }
    }
    return count;
}

```

12 13
prev

```

int solve( int arr[], int N )
{
    int c = 0;
    for( int i=0; i<(1<<N); i++ )
    {
        if( check(i, arr, N) )
            c++;
    }
    return c;
}

```

```

bool check( int i, int arr[], int N )
{
    int prev = Integer.MIN_VALUE;
    for( int j=0; j<N; j++ )
    {
        if(((1<<j)&i)!=0)
        {
            if( arr[j] < prev )
                return false;
            prev = arr[j];
        }
    }
    return true;
}

```

2 5 ✓
5 2 X

Ques.

FLASHING :-

$$A_N : -1 \quad 10 \quad 2 \quad 18 \quad 12 \quad 25$$

$$B_N : 6 \quad 12 \quad -3 \quad 10 \quad 23$$

Find $A \cup B$ and $A \cap B$.

$$A \cup B = \{-1, 10, 2, 18, \dots\}$$

$$A \cap B = \{10, 12\}$$

```

void unionIntersection(int An[], int Bn[])
{
    HashMap<Integer, Integer> map = new HashMap();
    for(int i=0; i < N; i++)
    {
        if(!map.contains(An[i]))
            map.put(An[i], 1);
        else
            map.put(An[i], map.get(An[i]) + 1);
    }
}

```

Same to put all elements of B. in map.

Count = 2
Intersection
all union.

```

for(int i=0; i < map.size(); i++)
{
    if(map.get(i) ≥ 2)
        cout("intersecting");
}

```

```

cout("union");
}

```

find length of Subarray (longest subarray) which elements
 can be rearranged in a normal sequence of numbers.
 (Ans): 10, 15, 13, 16, 14, 9, 8, 11, 2, 6, 5, 3, 1, 4, 3, 2, 8,
 ↓
 13, 14, 15, 16.
 in sequence
 length = 4

1, 2, 3, 4, 5, 6
 in sequence
 length = 6
 (Ans),

Brute Force
all the subarrays - $O(N^2)$

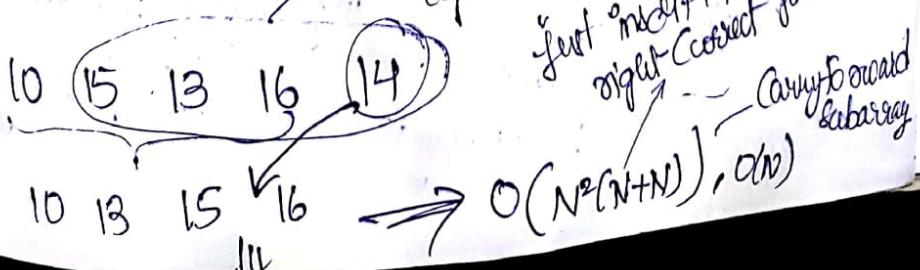
Sort the subarrays - $O(N \log N)$ &
check if the subarray has contiguous sequence N.

$\Rightarrow O(N^2(N \log N + N)) \propto N$ space for new.

for (i = 0 to N) → carry forward sum

for (j = i to N) → copy subarray from i to j, set it, and check if it has contiguous elements.
 { if (valid(arr, i, j))
 { ans = max(ans, j - i + 1); }

\rightarrow Instead of sorting all the subarrays, just insert the next element in the right position using insertion sort
 Neomparison method & just shift the other elements as required.



A subarray with max 15, min 12 with length 3 is invalid
possible as: 14 15 13 12

X can't put any
so subarray is invalid
as length of $\frac{\max - \min}{\text{of elements}} = 15 - 12 + 1 = 4$

Subarray is from i to j .

check if length of subarray == $\max - \min + 1$
 $j - i + 1 == \max - \min + 1$.
then the subarray is valid.
else not.

First. subArraySeq (int A[], j)

```
{ int ans=0;
    for( int i=0; i<N; i++ )
        for( int j=i+1; j<N; j++ )
            {
                M=max(A[i], i, j);
                m=min(A[i], i, j);
                if( j-i+1 == M-m+1 )
                    ans = max(ans, j-i+1);
```

} returns ans;

Comp:- $O(N^2(N+1), 1)$ \Rightarrow carry forward subarray max, min.
Subarray to find max, min if subarray i to j .
only max M, min m storing in var not in array)

5. ~~not~~
invalid
+1
8/
2
)

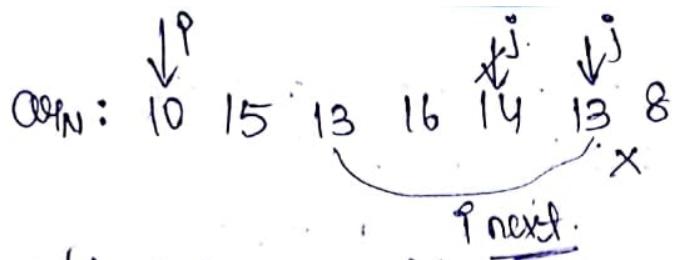
④ forward Max M and min m, instead of taking a subarray and again calculating max & min.
 $O(N^2(1+1)) = O(1) \rightarrow$ array forward max, min

```
int subArrayEq(int A[])
{
    int ans = 0;
    for (int i = 0; i < N; i++)
    {
        int Max = A[i], min = A[i];
        for (int j = i + 1; j < N; j++)
        {
            min = min(m, A[j]);
            Max = max(M, A[j]);
            if (j - i + 1 == Max - min)
                ans = max(ans, j + 1);
        }
    }
    return ans;
}
```

⑤ Array contains duplicates.
Get the length of max subarray whose sum which has difference between the unarranged elements as 1.

6 5 3 14 3 5 6.
6 5 3 4 5 6 valid. But if taken
diff 1 1 1 1 1 diff: 1 1 X
All diff. should be 1.

OK ③ X can have 11, 12, 12, 12, 15.
OK



③ $O(N^2(N+1+j))$, $O(1)$

subarrays single search min max update comp

$a[j]=14$, check from i to $j-1$
 if we find 14 in that subarray
 then break resembling that till
 subarray is not valid.
 we already have the same
 element in the subarray, diff = 0

④ Task Take HashSets
 To check if the elements are not duplicates. req. diff = 1

$\Rightarrow O(N^2(1+1+i))$, $O(N)$
 using HashSets

Print SeqDiff1(int arr[], int N)

or ans = 1 set add
 $j=i+1$
 set

int ans = 0;

for (int i=0; i < N; i++)

{ int M = arr[i], m = arr[i];

Hashset<Integer> set = new HashSet<>();

for (int j=i+1; j < N; j++)

{ if (set.contains(arr[j]))

break;

set.add(arr[j]);

M = max(M, arr[j]);

m = min(m, arr[j]);

If ($M-m+1 == j-i+1$)

ans = max(ans, j-i+1);

2 x d² n².

shash
pe

② Array \rightarrow Subarray max length whose elements are non-decreasing
should have diff 0 or 1.

10 15 13 16 14 13 8 11 2 6 5 3 1 4 3 1 2 20

1 2 2 3 4 5 6 valid

Hashset: 1, 2, 3, 4, 5, 6.

Size of Hashset == M-m+1

then valid ✅

$$\begin{aligned} M &= 6 \\ m &= 1 \\ M-m+1 &= 6-1+1 \\ &= 6 \end{aligned}$$

seqDiff0or1 (int arr[], int N)

{ int ans = 0;

for (int i=0; i<N; i++)

{ int M = m = arr[i];

Hashset<integer> hset = new HashSet<>();

for (int j=i+1; j<N; j++)

{ hset.add (arr[j]);

M = max (M, arr[j]);

m = min (m, arr[j]);

if (M-m+1 == hset.size())

ans = max (ans, j-i+1);

}

return ans;

$\Rightarrow O(N^2(1+1))$, $O(N)$

* Given an array of size N , find the no. of distinct elements in the window of size K .

$$\text{Q}_1 \text{ Q}_N : 10 \underbrace{12}_{2} \underbrace{18}_{3} \underbrace{15}_{3} \underbrace{12}_{2} \underbrace{8}_{2} 3 \quad 9 \quad 9 \quad 10 \quad 12$$

$K=4$

① : `int[]` distinct Elements in Window (Prt. `arr[1]`, Prt. `arr[2]`)

```

    {   print]; distinct = new print[n-k+1]; int l=0;

```

for (int q=0; q<N^{k+1}; q++)

```
{ HashSet<Integer> hs = new HashSet<()>;
```

for(int j=0; j<k; j++)

? $\text{he}.\text{odd}(\text{adj});$

distinct[l++] = hb.size();

$$\Rightarrow d(N-k+1) * k \bigr) d(k)$$

No. of windows \uparrow \rightarrow Insertions in hashset

② Casey forward HashSet.
etc
delete first, print last ele.

②. Delete first, then last.
Con : But in HashSet, if we delete one ele, all its

electro first, then water

②. Delete first, then last.
Con : But in HashSet, if we delete one ele, all its

duplicates are also deleted and the ~~subset~~
returns the wrong ans. // so use hashmaps.
 β Hashset problem

$$\text{q. } \begin{array}{r} 10 \\ \times 2 \\ \hline 20 \end{array} \quad \begin{array}{r} 10 \\ + 15 \\ \hline 25 \end{array} \quad \begin{array}{r} 25 \\ - 12 \\ \hline 13 \end{array} \quad \begin{array}{r} 13 \\ \times 2 \\ \hline 26 \end{array}$$

(3) should be

③ should be

$$\{10, 2, 12, 15\}$$

but gives a
as 12 & 13 drop off

of distinct

O 1 8

1, print n, print
l=0;

et <> c;

Use Hash Maps:-

- 1. Use forward Hash Maps.
- 2. maintain count value of each ele.
- 3. If count == 0, remove from Hash Map.
- 4. Else for removing just dec. the count value.

distinct Elements In Window (int arr[], int N, int k)

```
void distinctElementsInWindow (int arr[], int N, int k) {
    HashMap<Int, Int> map = new HashMap<>();
    for (int i=0; i<N-k+1; i++) {
        for (int j=i; j<i+k; j++) {
            if (!map.contains(arr[j])) {
                map.add(arr[j], 1);
            } else {
                map.add(arr[j], map.get(arr[j])+1);
            }
        }
    }
}
```

if

et size

problem

; 15)

Ans 2nd row
s deep all
also deleted

Only one Hash Map.
 $= O(K + (N-K))$, $O(K)$

STRINGS.

ASCII values:-

'a' = 97, 'b' = 98, ... 'z' = 122
 'A' = 65, 'B' = 66, ... 'Z' = 90
 '0' = 48, '1' = 49, ... '9' = 57

④ Print ASCII values:

```
for(int i=0; i<128; i++)
    printf("%d, %c\n", i, i);
```

⑤ No. of occurrences of characters:-

String: a y m n q m z t y n n d c c d c

void noOfOccurrences(String str)

```
{ int c[26] = {0};
    for(i=0; i<N; i++)
        c[str[i]-a]++;
}
```

In case of storing Count Array is better than Hashmap and TreeMap sorted as in treehash searching takes $O(\log N)$ but in count array direct index access $O(1)$

⑥ (a) Find the first repeating character in String:

(b) Find the repeating in substrings

String: a y m n q m z t y n n d c c d c

(a) $y = \text{ans}$.
 already T
 ans = m

(b) $m = \text{ans}$.

P) use count int array if count == 2 // ans. is that ele.
 P) use boolean array, make first time 'T', if second time
 appears & already previously is T, then that is ans.
 char repeatingOne(string str)
 {
 bool cnt[26] = {F};
 for (int i=0; i<N; i++)
 {
 if (c[str(i)-'a'] == T)
 return str[i];
 c[str(i)-'a'] = T;
 }
 }

dc

Q) Find the length of longest substring which is palindrome.
 str: a g m n q n m y y m n d c c d c
 ans = 6

0. check palindrome property for all substrings $\frac{N(N+1)}{2}$

0. check palindrome property for all substrings $O(1)$

$\Rightarrow O(N^2 * N)$, check palindrome property for substring.

substrings

int maxPalindromeSubstring (string str)

{ if ans == 0)
 for (int i=0 to N)

{ for (j=i to N)

{ if (isPalin(str, i, j) == T)

ans = max (ans, j-i+1);

ans =

HashMap
 $O(\log N)$

dc

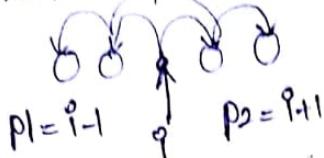
} return ans;

Max longest Palindrome Substring:

②. Pointers

odd Palindrome

1 center. If same.



max N centers in even string of N
Expansion on left & right sides. N.

$$\Rightarrow N+N.$$

even Palindrome

2 centers.



(N-1) centers.

N checkings

$$(N-1) \times (N)$$

$$O(N^2), O(1)$$

Complexity $O(N \times N + (N-1) \times (N))$, $O(1)$
 $\Rightarrow O(N^2), \underline{O(1)}$

int LongestPalindromeSubString(string str)

{

for (int i=0; i<N; i++)

{
 $p_1 = i-1$, $p_2 = i+1$

 while ($p_1 \geq 0$ && $p_2 < N$ && $s[p_1] == s[p_2]$)

p_1--
 p_2++

 ans = max (ans, $p_2 - p_1 + 1$);

}

}

$p_2 - p_1 + 1$

check if B is present in A as substring.

A: a b s m a s t c d
B: s m a r t

Time: $O(N \cdot m) + m$ O(1) Brute Force.

KMP Algorithm: $O(N+m)$, $O(m)$. \rightarrow Top Codes. (Do it)

Rabin Karp String Matching Algorithm: $\Rightarrow O(N+m)$, $O(N)$

Using hash functions:

$$h(s_1) = h(s_2)$$

$$h(\text{"smart"}) = s+t+m+a+r+t$$

$$h(stri) = \sum_{i=0}^{N-1} stri(i)$$

$$1) h(B_m) = s+t+m+a+r+t = h_B \Rightarrow O(M)$$

$$2) h(A_{m+1}) = a+b+s+m+a = h_A \Rightarrow O(m) \quad (N-m+1)$$

3) Compare $h_A = h_B \Rightarrow O(1)$ If equal return else next substring of AN.

$$4) h_A = b+s+m+a+r = h_A - 'a' + 'r'. \Rightarrow (N-m) \text{ times}$$

$\Rightarrow O(N+m)$ Time Complexity., $O(1)$ Space Comp.

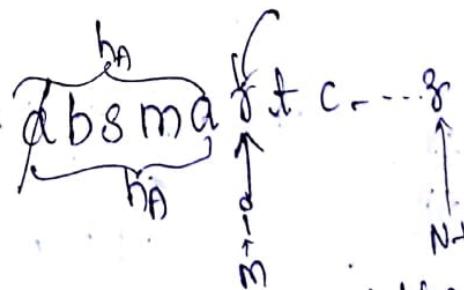
$$1) h_B = \sum_{i=0}^{m-1} B(i).$$

$$2) h_A = \sum_{i=0}^{m-1} A(i)$$

$$3) h_A = h_B$$

$$4) h_A = h_A - 'a' + 'r'$$

$$5) h_A = h_A - A(i-m) + A(i)$$



allow! ordering is not taken care smart has same hash as smart

2) Flaw $h(s_1) == h(s_2)$ but s_1 and s_2 are not same.

$$\text{eg: } \frac{a}{q_7} \frac{c}{q_7} == \frac{b}{q_8} \frac{b}{q_8}$$

So change the hash function to order to overcome this.
Hash functions $h(s_1)$:

$$\text{eg: } \sum_{i=0}^{N-1} (\text{str}(i))^2 \Rightarrow \text{will not work with } a b c \\ c a b. X$$

$$2) \sum_{i=0}^{N-1} (i+1)^2 \text{str}(i) \text{ will not work with } a b c \\ c d d c \\ X \text{ same hash value}$$

$$3) \sum_{i=0}^{N-1} (i+1) \text{str}(i) X \begin{matrix} a & b & c & c & b \\ a & c & b & b & c \end{matrix} \text{ fails}$$

$$4) h(\text{str}) = \sum_{i=0}^{N-1} \text{str}(i) * p^{i+1} \quad \text{where } p \text{ is the prime} \\ \text{inc power. eg } p=23, 59, 103 \text{ or any one.}$$

$$A: \overbrace{a \ b \ s \ m \ a}^i \ g \ t$$

$$B: \underbrace{s \ m \ a \ g \ t}_m$$

p and prime no.

$$1) h_B = s*p^0 + m*p^1 + a*p^2 + g*p^3 + t*p^4$$

$$2) h_A = a*p^0 + b*p^1 + s*p^2 + m*p^3 + g*p^4$$

$$3) \text{ compare } h_A == h_B$$

$$4) h_A = \frac{h_A - a*p^0 + g*p^4}{p} = b*p^1 + s*p^2 + m*p^3 + a*p^4 + g*p^5$$

for all
 $N-m+1$ elements.

$$h_A = b*p^1 + s*p^2 + m*p^3 + a*p^4 + g*p^5 \quad \text{inc power of prime.}$$

$$h_A = (h_A - A[i-m]*p) / p + A[i]*p^M$$

$$\downarrow \quad \sum_{i=M}^{N-1} h_A = (h_A - A[i-m]*p) [p + A[i]*p^M]$$

If m is large size
 If $m=100 \rightarrow$ then if $p=2$, then $g^{100} \rightarrow$ overflow
 $\Rightarrow 2^0 \bmod 10^{100}$
 Let $K=10^{100}$
 But we cannot distribute $\% K$
 over division by p . as division is not distributive over $/p$.
 So change the hash function such that it does not contain any
 change $\sum_{i=0}^{N-1} h_A((h_A - A[i-m]*p) / p + A[i]*p^i) \% K$
 $\Theta(-p^i) \% K$ Inverse Modulo.

Use dec powers of prime:-

$\forall i \sum_{j=0}^{N-1} h_A = (h_A - A[i-m]*p) / p + A[i]*p^i$ gives issue.
 $\forall i \leq m$ so use hash func. $h(\text{str}) = \sum_{i=0}^{10-1} \text{str}(i) * p^{N-i}$ dec powers.

A: $\begin{cases} a \\ b \\ s \\ m \\ a \end{cases}$ ↓
 $\begin{array}{ccccc} a & b & s & m & a \end{array}$ ↓
 B: $\begin{array}{ccccc} s & m & a & s & t \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 0 & 1 & 2 & 3 & 4 \end{array}$

$$h_B = s*p^5 + m*p^4 + a*p^3 + s*p^2 + t*p$$

$$h_A = a*p^5 + b*p^4 + s*p^3 + m*p^2 + a*p$$

$$\therefore \text{Compare } h_A = h_B$$

$$h_A = b*p^5 + s*p^4 + m*p^3 + a*p^2 + s*p$$

$$\forall i \sum_{j=0}^{N-1} h_A = (h_A - A[i-m]*p^m) * p + A[i]*p$$

$$\Rightarrow \forall i \sum_{j=0}^{N-1} h_A = ((h_A - A[i-m]*p^m) * p + A[i]*p) \% K$$

$$\Rightarrow \left(\forall i \sum_{j=0}^{N-1} h_A = ((h_A - A[i-m]*p^m + A[i]) * p) \% K \right)$$

RABIN KARP ALGORITHM: ROLLING HASH

pwsi[m+1]; k = 10^9 + 7, p = ??

$$\text{if } \forall i \text{ such that } \text{pwsi}[0] = 1, \quad \sum_{i=1}^m \text{pwsi}(i) = (\text{pwsi}[m-1] * p) \% k$$

$$\text{h}_B = 0; \quad \sum_{i=0}^{m-1} h_B = (h_B + B(i) * \text{pwsi}[m-i]) \% k$$

$$\text{h}_A = 0; \quad \sum_{i=0}^{m-1} h_A = (h_A + A(i) * \text{pwsi}[m-i]) \% k$$

Compare $h_A == h_B$

$$\sum_{i=m}^{N-1} h_A = ((h_A - A[i-m] * \text{pwsi}[m] + A[i]) * p) \% k$$

But \rightarrow

$$\rightarrow h_A = (h_A + k) \% k$$

Repeat from step 2,

$$\Rightarrow O(m+m+m+1+N-m), O(m).$$

$$\Rightarrow \underline{O(N+m)}, O(m) \text{ complexity.}$$

① Collisions can happen if taken primes are less.

So take bigger primes \Rightarrow e.g. 29, 59, 163.

② Double Hashing with different prime numbers.

$\text{pwsi}_1[m+1], \text{pwsi}_2[m+1]$. $p_1 \& p_2$ primes.

1) h_{B1}, h_{B2}

2) $h_A; h_{A2}$

3) Compare $h_{A1} == h_{B1}$; $h_{A2} == h_{B2}$

4) Compare h_A & h_{A2} . Repeat.

1/1/19
 0 1 2 15 -3 12 18 -6 9 3 4 7
 5
 ② CON^o
 N=10
 i j : $\sum_{k=i}^j \text{val}(k)$
 0 ≤ i < j ≤ N
 8 | 0 5 → 27
 matrix
 3 5 → 37
 4 8 → 22
 0 2 → 22
 Subarray from flag: sum.

```

① Brute force long
{
    long sum=0;
    for( int k=i; k<=j; k++)
        sum += arr[k];
    return sum;
}

```

→ QXN, 1

2) Cumulative Sum Method

$$\text{Q3) } \frac{0}{5} \quad \frac{1}{2} \quad \frac{2}{15} \quad \frac{3}{-3} \quad \frac{4}{12} \quad \frac{5}{18} \quad \frac{6}{-6} \quad \frac{7}{9} \quad \frac{8}{4} \quad \frac{9}{1}$$

99N: 5 + 22 19 31 49 43 52 56 55.
18: 5 + 22 19 31 49 43 52 56 55.

$$CS(5) - CS(2) = 87 \quad CS(j) - CS(i-1)$$

$$4. 8 \Rightarrow CS(8) - CS(3) = 87 \quad \text{if } i=0 \Rightarrow CS(i)$$

$$0.2 \Rightarrow CS(2) = 22$$

```
long subArraySum (int arr[], int n, int i, int j)
```

long cs = new CSEnJ; (CSEnJ=容器);

```
for(int i=1; i<N; i++)
```

$$CS[i] = CS[i-1] + cost(i);$$

$\text{if } (i == 0) \text{ return } \text{set}[j];$

the section $CS[j] - CS[j-1]$;

To get rid of if else condition when $i=0$,
we do. $CS[j] - CS[i] + arr[i]$.
We start the array indices from 1.

$$\begin{aligned}CS[0] &= 0 \\CS[1] &= arr[1] \\CS[2] &= arr[2] \\CS[3] &= arr[3]\end{aligned}$$

Cumulative Sum \approx Prefix Sum
(Mathematics) (Computer Science Proj.)

② $arr[0 \dots 9] = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$

Q. $\downarrow i \ j \ x$
 $0 \leq i \leq j \leq N$

$\forall k \in [i, j] \quad arr(k) = arr(k) + x$ // Add x to all elements from index i to j .

Queries

1 6 10
2 8 -3
3 9 2
0 1 4

// Find the sum of Array Elements
 $\begin{array}{cccccccccc}0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\4 & 4 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\7 & 7 & 7 & 7 & 7 & 7 & 7 & 7 & 7 & 7\end{array}$

① Sum of array. For each query iterate N times then sum

② Point the Array $\Rightarrow Q \times N + N, 1$

1 6 10 $\Rightarrow 10$ is added $(6-1+1) = 5$ times $\Rightarrow 10 \times 5 = 50$

4 8 -3 $\Rightarrow -3$ is added $(8-4+1) = 5$ times $\Rightarrow -3 \times 5 = -15$

3 9 2 $\Rightarrow 2$ is added $(9-3+1) = 3$ times $\Rightarrow 2 \times 3 = 6$

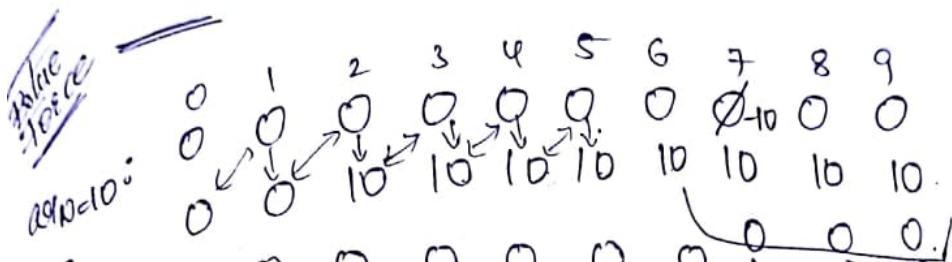
0 1 4 $\Rightarrow 4$ is added $(4-0+1) = 2$ times $\Rightarrow 4 \times 2 = 8$

Sum $\Rightarrow 51$

HR Practice

void arrayAfterManipulations (int arr[], int n, int q, int i, int j, int)

{
 for (int



arr[i]:
arr[6]:
arr[7]:
arr[8]:
arr[9]:
CS:
ori:

frem 2 all
but we had
till 6 only.
so put -10 at
arr(7) to)
get CS as
deg.

put root CS(p) and calculate CS(p+1)=CS(p)+deg(CH1).

- | | | | |
|---|---|----------|--|
| 1 | 9 | α | after 10 at arr(2), -10 at arr(7) & take prefixsum. |
| 2 | 6 | 10 | after -3 at arr(4), 3 at arr(9) & take prefixsum |
| 4 | 8 | -3 | if add 2 at arr(7), -2 at arr(10) but arr(10) not
there leave .ps |
| 7 | 9 | 2 | add 4 at arr(0), -4 at arr(2) p.s. |
| 0 | 1 | 4 | |

\Rightarrow add α at arr(i), add $-\alpha$ at arr(j+1) \Rightarrow then prefix sum
 $\text{if } j = N \text{ can do at last}$

void arrayAfterManipulation (int arr[], int N, int Q, int i, int j, int)

{

 for (int q=0; q<Q; q++)
 { if (j=N)
 arr[q] = arr[q] + i[q];
 arr[

query comp.
 $Q+N, 1$

	0	1	2	3	4	5	6	7	8	9
AgeN:	0	0	0	0	0	0	0	0	0	0
	4				-3					
	12	6	10	6.						3

1) 4 8 -3

3) 7 9 2

4) 0 1 4

	0	1	2	3	4	5	6	7	8	9
AgeP:	4	0	6	0	-3	0	0	-8	0	3
	4	9	6	0	10	7	7	4	-1	2

PrefixSum: 4 4 10 10 7 7 4 -1 2 \Rightarrow Ans ②

Element value as product of all elements except that element.

$$\text{① } \text{Euler's formula: } e^{j\theta} = \cos(\theta) + j\sin(\theta)$$

$$\text{P}(\text{D}) = \frac{2}{910} = 0.0022$$

Wk: 200
120:

$$\text{at } \overline{\text{force}}: N_1 = P / \sin(\theta) \Rightarrow N_1$$

\Rightarrow $bq(i) = 1/c(i)$, $bq(i) = 1/c(i)$ \rightarrow i, i

Q: $P = \prod_{q=0}^{\infty} \cos(q)$, // This doesn't work when $\cos(q)=0$.

N. 1 So handle it.

→ N, calculating p, calculate it except O.

while calculating, except for that everything is 0.

ff single 2010 - 2 0 1 6 -8 0.
0. 2 0 0 0 0 0 br

g 0.96
all obs in br.

multiple zeros in
e.g. $\text{cor: } \begin{matrix} 0 & 0 & 2 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$
 $\text{by: } \begin{matrix} 0 & 0 & 0 & 0 & 0 \end{matrix}$

3).  product of ele on left product of ele: on right.

2 5 1 6 → 8
↑ ↑ ↑ ↑ ↑
a₂-p+1

$\text{def } P = 13$

while ($pv = 0$)

leftP = leftP * arr[pi];

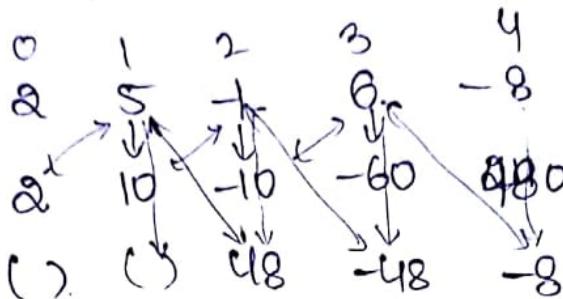
one at $q \Rightarrow cor(p) = leftP * rightP$

ans at 9 $\Rightarrow \text{cov}(P) = \text{left } P \rightarrow \dots$

$i^{\text{th}} \rightarrow (0 \text{ to } i-1) * (i+1 \text{ to } N-1)$



$a[i]$:



Prefix $P_P(i)$:
Product
Suffix $S_P(i)$:
Product.

Prefix Product, $P_P(i) = \prod_{j=0}^{i-1} a[i]$

(PP)

$$P_P(0) = a[0]$$

$$P_P(i) = \prod_{j=1}^{N-1} P_P(j) * a[i] \quad \text{--- N ele array}$$

or we can simply

use PP var but
update pp each

* for
all add

sol ①.

Suffix Product, $S_P(i) = \prod_{j=i}^{N-1} a[j]$

(SP)

$$S_P(N-1) = a[N-1]$$

$$S_P(i) = \prod_{i=N-2}^0 S_P(i+1) * a[i] \quad \text{--- N}$$

Answer b(i):

$$b[i](0) = S_P(1)$$

$$b[i](N-1) = P_P(N-2)$$

$$bi = P_P(i-1) + S_P(i+1)$$

Co

Comp.

$N, 2N$

To reduce space from $2N$ to 1

Instead of using arrays for prefix & suffix product

use only variables pp and sp & update them
execution time.

Count
 $\rightarrow M_4$

$$pp = \text{as}(0);$$

$$\bigvee_{i=1}^{N-2} \text{bs}(i) = pp + \text{sp}(i+1)$$

$$pp = pp + \text{as}(i);$$

$$\text{bs}(N-1) = pp;$$

Q) Ans: a b c d a g z a m n f a d x y
d d a t y z 6 ways.
not sorted.

Ans: d a f a

* find the smallest substring length of A which contains all characters of B.

* D. Brute force:- Generate all substrings - N^2 .

check if that substring contains all characters of B.

Complexity: $N^2(m+N)$, Not m .

— should not match the same char. again.

— or use hashsets.

Complexity: $N^2 + (m+N)$, N/m . List of indexes matched.
(in hashset).

Optimize this. copy substring.

Q). Count Array for B, A $CB(i)$ $CA(i)$

The frequency of each character in A \geq freq. in B

Count array B

for all substrings

Count array A

Count array B

Count array A

$CA(i) \geq CB(i)$

for all characters 0 to 25.

$\rightarrow M + N^2(N+26), 82$ space $CA(i), CB(i)$

1) Create count array of B. $CB[]$, $ans = \infty, N+1$
 2) for($i = 0$ to N)
 { for($j = i$ to N)
 { $CA[26] = 0$ // count array for every character
 for($k = i$ to j)
 { $CA[A[k] - 'a']++$; }
 if($\forall i \in [i, j] CA[i] \geq CB[i]$) \Rightarrow $ans = \min(ans, j-i+1)$; }
 }
 return ans.

3). Carry forward count array
 \Rightarrow update count array of A with incoming character
 $\Rightarrow O(m + N^2(1+26))$, $O(52)$. $\frac{CA + CB}{26} = 52$
 \uparrow update count array of A.
 \Rightarrow int $ans = \infty, N+1$
~~CB[26] = {0}~~
 $\forall i \in [0, N]$, $CB[B[i] - 'a']++$
 for(int $i = 0$; $i < N$; $i++$)
 {
 $CA[26] = \{0\}$;
 for(int $j = i$; $j < N$; $j++$)
 {
 $CA[A[j] - 'a']++$;
 if(valid(CA, CB))
 $ans = \min(ans, j-i+1)$;
 }
 }
 return ans.

```

bool valid(int CA[], int CB[])
{
    for (int i=0; i<25; i++)
    {
        if (CA(i) < CB(i))
            return false;
    }
    return true;
}

```

$\rightarrow O(m + N^2(1+26))$, $O(52)$

④ BINARY SEARCH:

```

int smallestSubstring(string A, string B, int N, int M)
{
    low = M, high = N, ans = -1;
    Create CB[M]
    count array of B.
    while (low <= high)

```

mid = (low + high) / 2;

If (valid(CB, A, N, mid))

ans = mid;

high = mid - 1;

else

low = mid + 1;

return ans;

A: a lc d a a t

B: d a t a

low = 4

high = 6

mid = 5

Binary Search: $\log_2(N-M+1)$

In each call valid function - $O(1)$

$\rightarrow O(N * \log_2(N-M+1))$, $O(52)$ compare

(a) If $CA(i) \geq CB(i)$ return true

\rightarrow valid(mid)
 If all length of
 substrings of
 length mid are
 valid, then
 $ans = mid$
 go left.

\rightarrow In each
 update count array;
 say dec val of current
 by inc val of previous
 character

boolean valid (int CB[], String A, int N, int mid)

{

int[] CA = new int[26];

for (int i=0; i<mid; i++)

{

 CA for (int j=i; j<mid; j++)

{

 CA[j]++; // CA[A[j]-'a']+;

}

~~Brute Brute Force~~

~~bool~~ valid (int CB[], string A, int N, int mid)

{

 CA[26] = 0;

 for (int i=0; i<mid; i++)

{

 CA[A[i]-'a']++;

}

 if (check(CA, CB)) return true;

 for (int i=mid; i<N; i++)

{

 CA[A[i-mid]]--; // remove prev ele.

 CA[A[i]-'a']++; // Add the cur. ele

 if (check(CA, CB))

 return true;

}

 return false;

}

boolean check(CA, CB)

{

 if (CA(i) ≥ CB(i)), return true;

 return false;

 if (CA(i) < CB(i))
 return false;
 return true;

OPERATORS

$$\begin{aligned} a \& b &= a \% 2 \\ a \& l &= a + (1 - a \% 2) \end{aligned}$$

left shift $a \ll i = a * 2^i$
right shift $a \gg i = a / 2^i$

$$a \& a = a$$

$$a \& 0 = a$$

$$a \& b \& a = b$$

⑥ Power of 2
long power (long N)

if return $1L \ll N$

⑦ check if power of 2, given a.

if $((a \& (a - 1)) == 0)$

return true;

if $(a == 0)$ return false;

⑧ Bad checkBit (int N, int i)

return false;

a). return $(1 \ll i) \& N$;

b). return $(N \gg i) \& 1$;

c). return $((N \gg i) \% 2)$;

⑨ Set or no. of ones of no. of set bits.

Algorithm: If we want to find the number of set bits in a number, we can do it by using the above method.

For example, if we want to find the number of set bits in the number 10, we can do it by using the above method.

For example, if we want to find the number of set bits in the number 10, we can do it by using the above method.

For example, if we want to find the number of set bits in the number 10, we can do it by using the above method.

For example, if we want to find the number of set bits in the number 10, we can do it by using the above method.

For example, if we want to find the number of set bits in the number 10, we can do it by using the above method.

For example, if we want to find the number of set bits in the number 10, we can do it by using the above method.

For example, if we want to find the number of set bits in the number 10, we can do it by using the above method.

For example, if we want to find the number of set bits in the number 10, we can do it by using the above method.

For example, if we want to find the number of set bits in the number 10, we can do it by using the above method.

For example, if we want to find the number of set bits in the number 10, we can do it by using the above method.

For example, if we want to find the number of set bits in the number 10, we can do it by using the above method.

For example, if we want to find the number of set bits in the number 10, we can do it by using the above method.

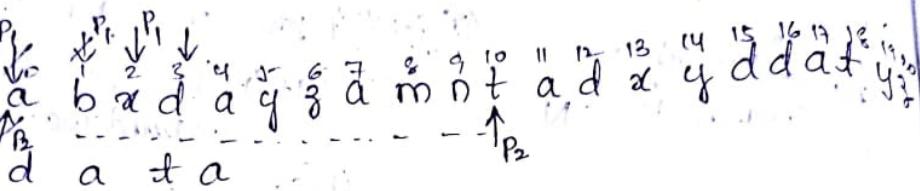
For example, if we want to find the number of set bits in the number 10, we can do it by using the above method.

For example, if we want to find the number of set bits in the number 10, we can do it by using the above method.

17/3/19

(2)

Ques: Two Pointers Soln for Smallest Substring length
of A containing all characters of B. ans=10

A: 
 $p_1 \downarrow$ $p_1 \downarrow$
 $a \ b \ c \ d \ a \ g \ g \ a \ m \ n \ t \ a \ d \ x \ y \ d \ d \ a \ t \ y$
 $p_2 \downarrow$ \uparrow
 $d \ a \ t \ a$

- First initialize pointers, p_1, p_2 at start 0, $ans = -1$
 - Increment pointer p_2 until till the elements of B are in A i.e the the substring is valid, update $ans = p_2 - p_1 + 1$
 - Inc. p_1 and check if the substring from p_1 to p_2 is valid.
 If it is valid, we can inc. p_1 , check validity from p_1 to p_2 .
 But if it is not valid, there is no use of checking the substrings from p_1 to p_2 as if length 10 is not valid, then lengths 9, 8, 7... containing those will obviously be not valid.
 - When inc p_2 , we update count array CA with incoming char.
 - when dec p_1 , we update count array CA with outgoing char.
- $CA[A(p_2) - 'a']++;$ only compare with $'a'$
- $CA[A(p_1)]--;$

$\Rightarrow O(N * 26 + N), O(52) \Rightarrow \underline{O(N), O(1)}$
 \uparrow
 p_2 can move at max till N.

Efficient.

code H

④ $A_N: p^1 \ p^2 \ p^3 \ p^4 \ p^5 \ p^6 \ p^7 \ p^8 \ p^9 \ p^{10} \ p^{11} \ p^{12} \ p^{13} \ p^{14} \ p^{15} \ p^{16} \ p^{17} \ p^{18}$
 $B_M: x \ c \ s \ m \ a \ g \ t \ y \ z \ k \ q \ n \ t \ e \ r \ n \ a \ r \ t \ x \ z \ c \ d$

Q ↓ $p \ j \ k \ l$ * substring $(A, p, j) = \text{substring}(B, k, l)$
 $0 \leq p \leq j \leq N$ $p \ j \ k \ l$
 $0 \leq k \leq l \leq M$ $2 \ 6 \ 10 \ 14 \Rightarrow \text{True}$
 $5 \ 12 \ 7 \ 14 \Rightarrow \text{False}$
 $10 \ 15 \ 3 \ 8 \Rightarrow \text{True}$.

① Brute force: Match M with N indexes.

$$\rightarrow O(Q + \underbrace{N/m}_{\min}), O(1)$$

② Hash functions If $s_1 == s_2$ then $h(s_1) == h(s_2)$

good hash function

$$\text{hash}(s[i]) = \sum_{i=0}^{N-1} s[i] * p^{i+1}; p \text{ is big prime}$$

Eg. for $x \ c \ s \ m \ a \ g \ t$.
 ① power.

$$\begin{aligned} \text{HA}(i) &= \sum_{j=0}^i A(j) * p^{j+1} & \text{② } \text{HA}(0) = A(0) * p \\ \text{hash value till } i^{\text{th}} \text{ char} & & \sum_{i=1}^N \text{HA}(i) = (\text{HA}(i-1) + A(i)) * \text{pwr}(i) \\ & & \text{③ } \text{HB}(0) = B(0) * p \\ & & \sum_{i=1}^M \text{HB}(i) = (\text{HB}(i-1) + B(i)) * \text{pwr}(i+1) \end{aligned}$$

④ $V_1 = \text{HA}(j) - \text{HA}(p-1) \quad V_1 = (V_1 + k) \% K$
 seq. $\Rightarrow V_1 = s * p^3 + m * p^4 + \dots + t * p^7$

⑤ $V_2 = \text{HB}(l) - \text{HB}(k-1) \quad V_2 = (V_2 + k) \% K$
 seq. $\Rightarrow V_2 = s * p^{11} + m * p^{12} + \dots + t * p^{15}$

⑥ $d = |j-k|$ absolute diff. of pow.

diff in powers

either $V_1 = \frac{V_2}{d}$

⑦ If $p < k$: $V_1 = V_1 * \text{pwr}[d]$
 else $V_2 = V_2 * \text{pwr}[d]$

$d = |k-j|$

⑧ $V_1 * d = V_2$

long array

- 1) Powr array of size $\max(N, M) + 1$ $\rightarrow N, M, \text{Pwr}[0..J=1]$ \leftarrow long pwr[N];
 $\forall i = 1 \dots N \quad \text{Pwr}[i] = \text{Pwr}[i-1]^p$
- 2) HA $\rightarrow N$ ②. N
 3) HB $\rightarrow M$ ③. m
 4) V1 $\rightarrow l$ ④. Q
 5) V2 $\rightarrow l$ ⑤. P
 6) d = l - k - 1
 7) Compare.

$$\Rightarrow O(N + N + M + Q), O(N + M)$$

— p should be bigger youme - to avoid collisions.

— or else, next go for double hashing.

⑥. if ($i \geq 1$)
 $V_1 = HA(j) - HA(i-1)$, $V_2 = HB(l) - HB(k-1)$
 else $V_1 = HA(j)$ else $V_2 = HB(l)$

$$V_1 = (V_1 + k) \% k \quad ; \quad V_2 = (V_2 + k) \% k$$

⑦. $d = l - k - 1$

⑧. if ($i < K$),
 $V_1 = V_1 * \text{pwr}[d]$,
 else $V_2 = V_2 * \text{pwr}[d]$

⑨. if ($V_1 == V_2$)

return True;

else
 return False;

~~do~~ ~~gt~~

Q8

Find the length of longest palindrome string:

A_N: a b c x y z y z y x c p q p d

① Brute force: All substrings, check palindrome

$$\Rightarrow O(N^2 \times N), O(1)$$

② Hashing: $\text{hash}(s) = \text{hash}(\text{rev}(s))$

p_5, p_6, p_7, p_8, p_9 } equal.

p_8, p_7, p_6, p_5 } $\neq p_9$

Forward p_1, p_2, p_3, p_4, p_5 } equal.
Hash a b c d e
Backward p_5, p_4, p_3, p_2, p_1 } Hash

$$ap^1 + bp^2 + cp^3 + dp^4 + ep^5 = e^{p_1} + d^{p_2} + c^{p_3} + b^{p_4} + a^{p_5}$$

should be equal.

in palindrome

FH($p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8$)
a b c x y z y z y x c p q p d z

{ valid func. $f(x, y)$.

Query $x=4, y=8$

$$FH(p) = FH(p-1) + \text{stai}(p) * p^{p+1}$$

handle $p=0$, $FH(0) = F(H(-1))$

$$BH(p) = BH(p+1) + \text{stai}(p) * p^{N-p}$$

smallest $p: x+1$

$$BH(p) = BH(i) + \text{No need}$$

if $i=N$, $BH(i) = BH(i+1)$

$$f = FH(y) - FH(x-1) : x+1 = sf$$

smallest $p: N-y$

$$b = BH(x) - BH(y+1) : N-y = sb$$

O(1)

$d = |\text{sf} - \text{sb}|$
 if ($\text{sf} < \text{sb}$)
 $f = f * p^d$
 else
 $b = b * p^d$
 if ($f == b$) return true;
 return false;

} But here we still generate substrings and for each substring check if the substrings & reverse of theirs have same hash values. So comp. $O(N^2)$; $O(1)$
 $\Rightarrow O(N^2); O(1)$

⑤. Binary Search with Flipping

valid : If $\text{ans} = 11$, length string is valid ✓
 If $\text{ans} = 10$, length valid ✓
 If $\text{ans} = 9$, length valid ✓
 If $\text{ans} = 8$, length valid ✓
 If $\text{ans} = 7$, length valid ✓
 If $\text{ans} = 6$, length valid ✓
 If $\text{ans} = 5$, length valid ✓
 If $\text{ans} = 4$, length valid ✓
 If $\text{ans} = 3$, length valid ✓
 If $\text{ans} = 2$, length valid ✓
 If $\text{ans} = 1$, length valid ✓
 If $\text{ans} = 0$, length valid ✓

Print longestPalindrome(string A)

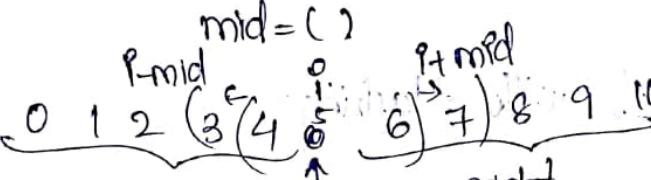
```

low=0, high=N-N%2
while (low <= high)
  mid = (low+high)/2
  if (valid(mid))
    ans = mid;
    low = mid+1;
  else
    high = mid-1;
}
  
```

Valid func takes substrings of length mid.

```
boolean valid1 (mid)
{
    for (int i = mid; i <= N-1; i++)
        int x = i - mid + 1;
        int y = i;
    return Valid(x) y
}
```

$\Rightarrow O(N + N + N + 2N \log_2 N)$, $O(3N)$
↑
Power func. forward hash backward hash valid Binary Search
for even, odd lengths.

④ \downarrow $mid = 2$

Binary Search on length of expansion.
If () valid \Rightarrow expand left & right.
min possible expansion will be minimum of left & right lengths
if (valid1 (2))
{ ans = mid + 1;
low = mid + 1; else { high = mid;
} }
if () not valid \Rightarrow compress left & right.

⑤ Manacher's Algorithm (N, N)
(on Text-Code)

```

int longestPalindrome(string s) {
    int low=0, high= min(i, N-i-1);
    while (low <= high) {
        mid = (low+high)/2;
        if (check(s, mid, i+mid)) {
            ans = 2*mid + 1;
            low = mid+1;
        } else {
            high = mid-1;
        }
    }
    return ans;
}

```

Given Array.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Query Type.

1	$q \Rightarrow arr(q) = 1 - arr(q)$
2	$q : \min_{j=0}^{N-1} q-j $
3	$arr(j) = 1$

// find the closest distance to the nearest 1.

Query Type. index.

1	5 \Rightarrow (go to index 5 \Rightarrow 1-0 \Rightarrow 1. distance from index 8 to closest 1 i.e $5 \Rightarrow dist = 3$)
2	8 \Rightarrow 3 ans. distance from index 8 to closest 1 i.e $5 \Rightarrow dist = 3$
1	10 \Rightarrow 1-0 = 1
2	8 \Rightarrow 8 to 10 index ans = 2
2	4 \Rightarrow ans 1.
1	5 \Rightarrow 1-0 = 0
2	4 \Rightarrow 6 (if it itself has 1, no others, so dist = 0)
2	10 \Rightarrow 0
1	10 \Rightarrow 1-0 = 0
2	7 \Rightarrow No 1 in whole array \Rightarrow ans = -1.

23/3/19

① Brute force:
 Query1 \Rightarrow 1 } QXN, N
 Query2 \Rightarrow N }

② Use list - store the indexes in the list which have 1 at the
 If the element again due to query 1 becomes zero,
 remove it from the list. — N Space.

- Search the nearest index which has 1 i.e. closest.
- linear search — $O(N)$.

\Rightarrow 1 Query \rightarrow N / } QXN, N
 & Query \rightarrow N }

If ele. already there
 remove it from list

③. Sorted list maintenance. (Tree Set) — Space N.

Find min. floor. or min. ceil.
 (log N) (log N)

Query1 \Rightarrow N

Query2 \Rightarrow log N

Java - TreeSet
 C++ - set

④. Sorted set - BBST - Tree Set

5, 12, 15, 20, 23

Search - $O(\log N)$

Insert - $O(\log N)$

Delete - $O(\log N)$

Query1 \Rightarrow log N

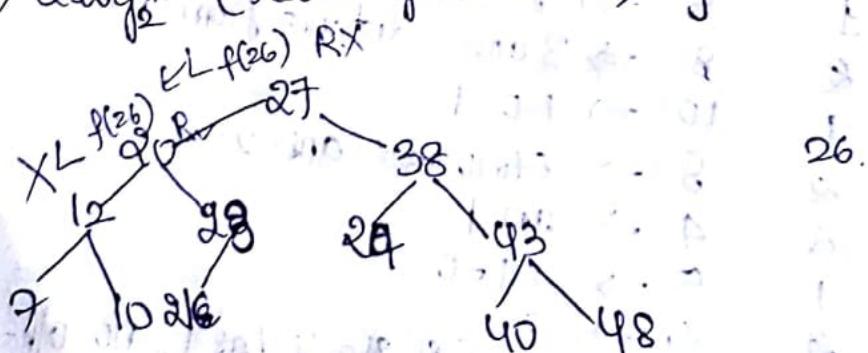
Query2 \Rightarrow N

(Take diff. of each key, get min.)

⑤. Balanced Binary Search Tree. (Height = $\log N$ always)

Query1 \Rightarrow log N (sorted set - BBST) / Ele found delete
 / not found add }

Query2 \Rightarrow 2 log N (Search floor & ceil.) } QXN, N



② Valid Subarray (largest) with equal no of zeroes & ones.

CB :- 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
CB :- 0 0 1 0 1 0 0 0 1 1 0 0 1 0 0

①. Brute Force: Find all subarrays - $O(N^2)$

Check if it is valid - $O(N)$

No need to ^{check} space for each subarray - $O(1)$
 just iterate from i to j

Comp :- $O(N^2 * N), O(N)$

②. without storing subarrays just maintain count of 0s & 1s.

— Comp: - $O(N^2), O(1)$ — Carry forward count.

Valid subarray for $i=0$ to N

$c[2] = \{0\}$

for $j=1$ to N

$c[ar[j]]++$

if ($c[0] == c[1]$)

$ans = \max(ans, j-i+1)$

③. Using Binary Search.

If $ans=6$: check all subarrays

Should be definitely said if $ans=6 \vee 4 \vee 2 \rightarrow$ search for more
 $low=mid+2$
 $ans=mid$

Ans fails if not. If $ans=12 \times 14 \times 16 \times$ go left $\Rightarrow high=mid-2$

wrong assumptions for each binary search call valid function
 $(\log N)$ (N)

$\rightarrow O(N \log N), O(1)$

ValidSubArray (int ar[], int n)

```

ans =
low = 0, high = N
while (low <= high)
{
    mid = (low + high) / 2
    if (valid())
        {
            low = mid + 2;
            ans = mid;
        }
    else
        high = mid - 2;
}

```

doesn't work if
 $ans=6$.

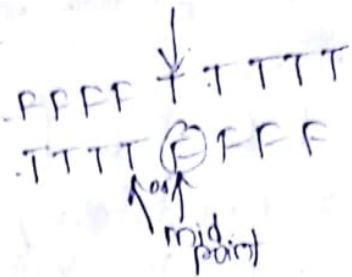
$0 [1 0 0 0 0] 1] 0$
 but 8 ✓ valid

BS doesn't work
 always

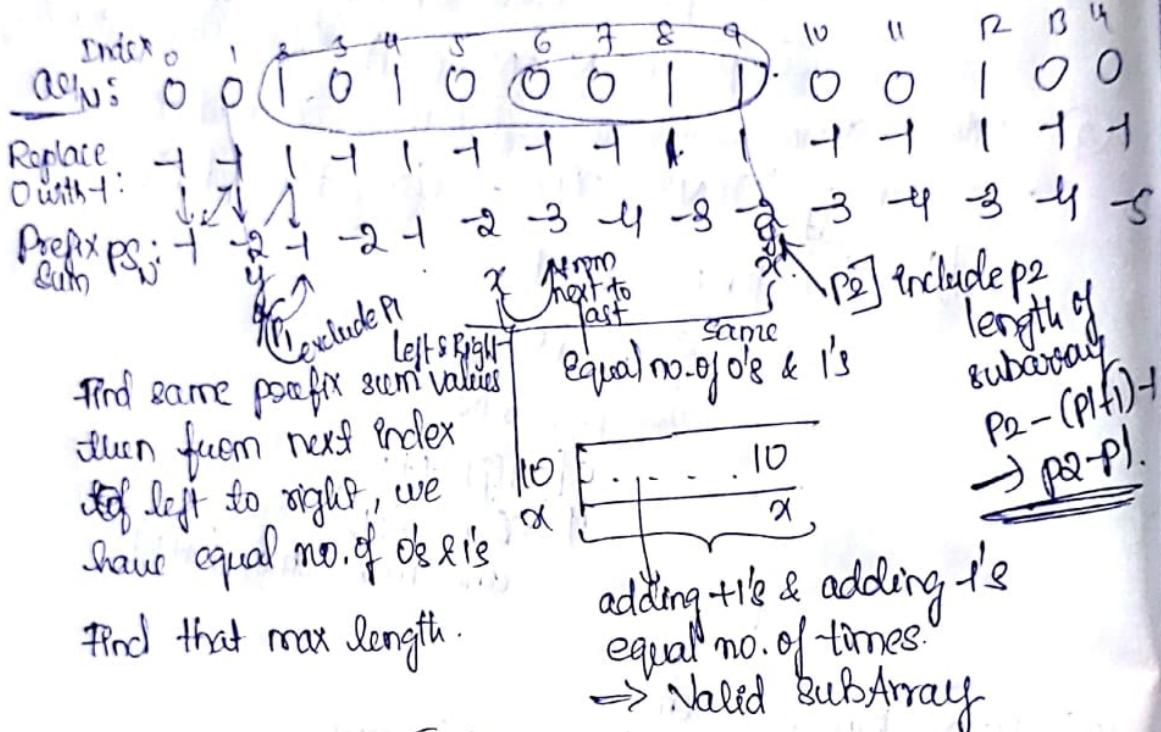
Be well work only if

valid func

valid func



(a point till which
all are P, & then
from there all T)
(Og)



Maintain Hash Maps <First Index> & <Last Index>

If not present, insert in both Hash Maps.

If already present, update only the last index.

Traverse only one from left to right

in ps. e.g.

<First Index>

<Last Index>

Take diff Li-Fi = update ans with max ele.

-1 0 0 0 0

0 1 2 4

-2 1 1 1

1 2 5

Right is front sum creating Hash Map getting length from it.

Complexity: $N + N + N + N, 3N$ — prefix sum + Hash.

\Rightarrow Or instead of last index hash map, use only a variable & update it always.

④ Rotate the given array 'd' no. of times.

Given: $\begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 39 & 1 & 4 & 7 & 8 & 10 & 15 & 19 & 23 & 30 & 34 \end{matrix}$ $\begin{matrix} 11 & 39 \\ 34 & 34 \end{matrix}$

$d=1 \Rightarrow 39 \ 3 \ 1 \ 4 \ 7 \ 8 \ 10 \ 15 \ 19 \ 23 \ 30 \ 11$

$d=3 \Rightarrow \begin{matrix} 80 & 34 & 39 & 1 & 4 & 7 & 8 & 10 & 15 & 19 & 23 \\ \text{sorted} & d-1 & \text{sorted} & & & & & & & & \end{matrix}$

Now, we are given an array after 'd' no. of rotations,

find if the key 'k' is present or not.

e.g.: $k=30 \rightarrow \checkmark \text{True}$
 $k=28 \rightarrow \times \text{False}$.

If 'd' is known index \rightarrow array b/w 0 to $d-1$ is sorted.

array b/w 0 to $d-1$ is sorted.

array from d to $n-1$ is sorted.

Search for key in any one of the two arrays using Binary Search.

Problem — $d=?$ (d is unknown) — ?

① Linear Search — $O(N) \cdot O(1)$

Ques: If d is not known \rightarrow is key present or absent?

Ans: First find d , then BS for key if any of 2 arrays \rightarrow $O(d-1) + O(n-d)$

Given: $\begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 19 & 23 & 30 & 34 & 39 & 1 & 4 & 7 & 8 & 10 & 15 \end{matrix}$

left $\quad \quad \quad$ mid $\quad \quad \quad$ right
 $a[0] > a[n-1]$ or $a[mid] > a[n-1]$ R. Right.

if (mid \in Left) // go right.

low = mid + 1

else // if mid \in Right Side // go left

high = mid - 1

mid = mid

high = mid - 1

mid \in left side

when $a[mid] > a[0]$ or $a[mid] > a[n-1]$

mid \in right side

when $a[mid] < a[0]$ or $a[mid] < a[n-1]$.

$\text{arr: } 19 \ 23 \ 30 \ 34 \ 39 \rightarrow 3 \ + \ 4 \ 7 \ 8 \ 10. \ 15$

left right

~~W~~ boolean findKey (int arr[], int key)

```

    {
        // Left side
        // Right side
        if (low == 0)      high = n-1
        while (low <= high)
        {
            mid = (low+high)/2;
            if (mid < left)
                if (key > arr[low] && key < arr[mid])
                {
                    ans = mid;
                    high = mid-1;
                }
                else
                {
                    ans = mid;
                    low = mid+1;
                }
            else // if (mid < right) // mid to n-1
                if (key > arr[mid] && key < arr[n-1])
                {
                    ans = mid;
                    low = mid+1;
                }
                else
                {
                    ans = mid;
                    high = mid-1;
                }
        }
        return false;
    }
}

```

(*) Find if the key is present in row wise sorted & column wise sorted matrix.

Rect $N \times M$:	10	8	5	7	10	skip row.	let key = 15.
	5	1	6	8	11		
	2	5	7	10	12		
	3	8	15	18	20	for 26	
	5	10	20	23	24	strip column.	
	13	18	25	38	9	out of array return false.	

- ① Brute Force: $N \times M$, $\Rightarrow N \times \log M$, start from top left.
- ② Binary Search in each Row $\Rightarrow N \times \log M$, start from top left.
- ③ Binary Search in each column $\Rightarrow M \times \log N$, start from top right or bottom left.

② Mat. \Rightarrow Check if key is present in completely sorted matrix.

Check if key is found									8<10
Row	0	1	2	3	4	5	6	7	8
Neat $M \times N$:	0 - 8	- 5	- 2	4	7	8	23 ¹⁰	87 ¹¹	
1	10 ⁶	12 ⁷	13 ⁸	20 ⁹					
2	29 ¹²	30 ¹³	(32) ¹⁴	36 ¹⁵	41 ¹⁶	49 ¹⁷			
3	52	53	58	60	67.	70			
4	72	75	78	91	93	97			

- ① Brute Force : $N \times M$, l.
 - ② Using BS: $N \times \log_2 M \approx M \times \log_2 N$
 - ③. $\lceil (M+N) / 2 \rceil$, l. } $O(N \log_2 M)$
 - ④. Linear search to find in row - $O(N)$. BS - $O(\log_2 M) \Rightarrow$ will give the row num to search in
 - ⑤. Finding floor of first column — $O(\log_2 M)$
 - ⑥. finding ceil of key in last col }

thus binary search for key in that particular row - $O(\log_2 N)$.

$\Rightarrow (\log_2 M + \log_2 N)$, l.

⑥. Consider A as a complete sorted array.

B.C. wet sorted array. mid.

ele index \Rightarrow 14
 32 \Rightarrow mid \Rightarrow (2, 2)
 mid/M, mid % M

③. Nat_{Nxm}: Find the max no. of ones in each row ^{all sorted rows}

	0	1	2	3
mat _{Nxm} :	0	0	0	0
0	0	0	0	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	1
0	0	0	0	0

$$m - \text{idx} \\ = m - \text{idx} \\ (\text{idx}, m)$$

Stand at top of bridge
1 egg!
update date
→ PC one
Continue in ~~area~~
It seen 0
ans some
no update
in down

row-if all continue
col-if 0 continue
same col

①. NxM, 1

② Using BS: $N \log M$, 1

MN

int findMaxNoOfIs (int arr[])

{ if ans=0;

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

ans = max (ans, BinarySearchToFindNthOne(arr, n-1))

۳

return ans;

3. Int BinarySearchToFindNoOfChesInRowi(arr[], int i).

၁၃

~~print~~ idx = M;

int low=0, high=M-1;

while (low <= high)

$$\{ \quad \text{mid} = (\text{low} + \text{high}) / 2;$$

If (~~a[i]~~ a[i][mid] == 1)

$$qdx = m'd;$$

high = mid - 1;

18

else
}; $low = mid + 1;$

3

3. $M = \text{idx};$

(3) $(1+m)$ Top Right & Bottom Right

int s1
{

④ Sum of OR of all SubArrays

$\text{arr} = \{5, 2, 10, 7, 12, 22, 13, 18, 1\}$

$$\text{ans} = 5 + 5|2 + 5|2|10 + 5|2|10|7 + \dots - - - - -$$

$$+ 2 + 2|10 + 2|10|7 + \dots - - - - -$$

①. Brute-force: Generate all subarrays. - calculate sum of ORs.
 $\frac{N(N+1)}{2}$ - $O(N^2)$, $O(1)$

$$\begin{array}{c} 2 | 0 \\ 0 | 5 : 10 | 1 \\ \hline 10 | 1 \end{array} \quad \begin{array}{c} 5 : 10 | 1 \\ 2 | 0 | 0 \\ \hline 10 | 1 | 0 | 0 \end{array} \quad \begin{array}{c} 5 | 10 | 1 \\ 2 | 0 | 0 | 10 \\ \hline 10 | 1 | 0 | 10 \\ \hline 11 | 11 \end{array}$$

Brute Force:

$$\text{ans} = 0$$

for $i = 0$ to N

$$c = 0$$

for $j = i$ to N

$$c = c | \text{arr}(j)$$

$$\begin{matrix} 8 & 7 \\ 8 & 7 \\ 7 & \end{matrix}$$

return ans;

②. Check for Bit positions.

$\text{arr} = \{5, 2, 10, 7, 12, 22, 13, 18, 1\}$

For all bit positions check for how many no.s its set.

Let $b=3 \Rightarrow$ value of bit pos $= 2^b = 2^3 = 8$.

$$(N-i) * 8 + 3$$

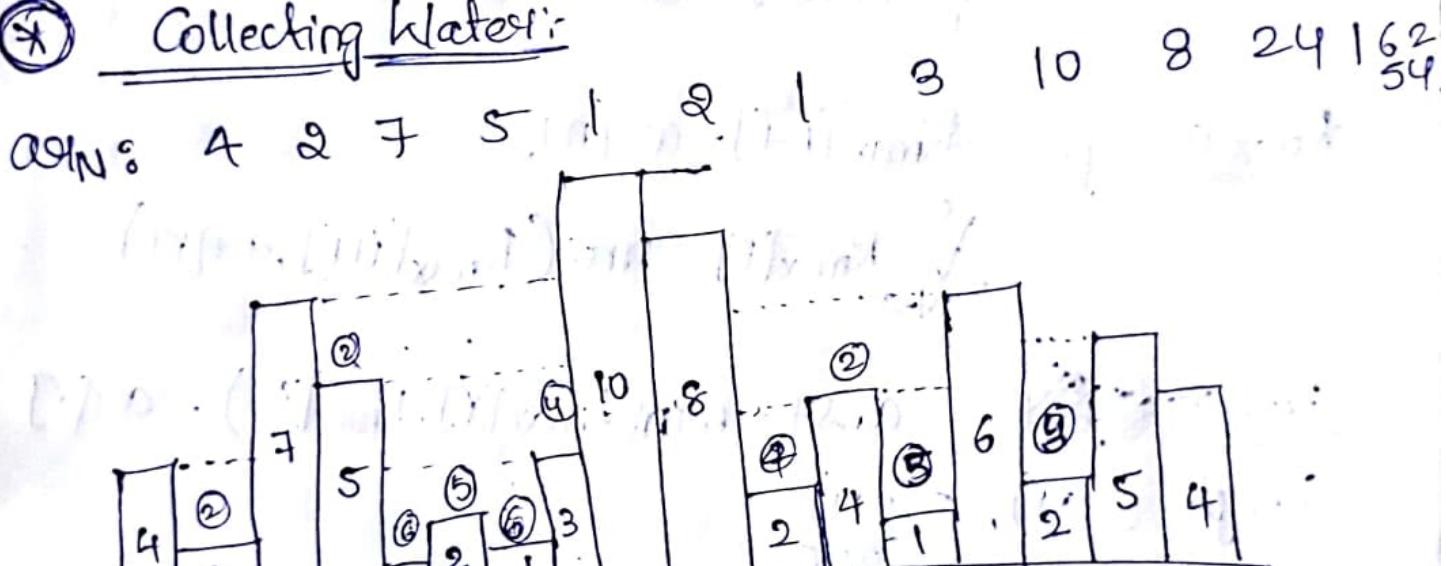
$\overbrace{\text{bitpos}}^{\text{value}}$ $\overbrace{\text{N}-i+1}^{\text{(N.i)}}$

```

int sumOfORofSubarray(int arr[], int N)
{
    for b = 0 to 32
        c = 0
        for i = 0 to N // for all ele. of array.
            if (checkBit(arr[i], b))
                ans = ans + (N - i) * 2^b * (c + 1)
                c = 0
            else
                c++
    }
    return ans;
}

```

④ Collecting Water:



int collectingWater(int arr[], int n)

{

 int ans = 0;

 for (int i=0; i<n; i++)

 {

 Lmax = max(arr[i], Lmax);

 Rmax = max(arr[i], Rmax);

 ans += min(Lmax, Rmax) - arr[i];

}

 return ans;

}, O(N^2), O(1)

Lmax array: Lmax[0] = arr[0];

$\bigvee_{i=1}^{N-1}$ Lmax[i] = ~~max~~ max(Lmax[i-1], arr[i])

Rmax array: Rmax[N-1] = arr[N-1];

$\bigwedge_{i=N-2}^0$ Rmax[i] = max(Rmax[i+1], arr[i])

Complexity: ~~O(N^2)~~: ans += min(Lmax[i], Rmax[i]) - arr[i]

Complexity: O(N), O(2N)
Space.

Instead of taking array for Lmax[i], take a var. lmax
update it all the times.

lmax = 0.

$\bigvee_{i=0}^{N-1}$ lmax = max(lmax, arr[i])

ans += min(lmax, Rmax[i]) - arr[i]

Complexity: O(N), O(N)

Space.

⑧ Find the first positive missing element:

Eg. arr_N: 5 2 8 -3 1 4 5, 100 7 -6 \Rightarrow ans=3

Eg. arr_N: 4 1 3 2 \Rightarrow ans=5

Eg. arr_N: 2 5 1 2 3 7 \Rightarrow ans=4. ans $\in [1, N+1]$

① Brute Force: check for ~~all elements~~. Search for ele. Not in array
 \Rightarrow ~~all~~ ~~ele~~ ~~check~~ $O(N^2)$

② Sort Array: 1 2 3 4 if all tve, then check if index has index+1 ele. or not
 $N \log N$

-6 -5 -2 3 4 If -ve, skip repeating numbers.
 count no. of -ve.

③ \downarrow idx = pidx - count of -ve.
 If arr[idx] \neq arr[pidx] return idx+1

Iterate & check.

Comp $\Rightarrow O(N \log N + N), O(1)$

③ Using HashSets:

Put ele in hashset - N

Search for 1 to N+1 numbers - N+1.

hashset space = N

\Rightarrow Comp $O(N + N + 1), O(N)$.

④ Count Array: ans always lie in (1 to N+1)
 ans $\in [1, N+1]$

Result Count Array

Bool cnt[N+2] = {F}

Iterate and populate count \Rightarrow N

Find first ele. pidx where cnt[pidx] = F \Rightarrow N

Space N+2 size cnt array \Rightarrow N+2

Comp $\Rightarrow O(N + N), O(N + 2)$

space Optimize

Take cnt[N+1] = {F}

If all are F then ans is

cnt[N+2] '0' based index $c(0) = 1 \times N + 2$
 $c(1) = 2 \times N + 2$

Count array: Space N.

bool cnt[N] = {F}

for i=0 to N

{ if (arr[i] ≥ 1 & & arr[i] ≤ N)
 cnt[arr[i]-1] = T.

// find first false idx, ans = idx + 1.

(5) → 1) Iterate and make all -ve ele. & ele. > N+1 as ∞

→ At idx p, go to $(a[i]-1)$ index.

→ make the ele there as -ve if it
is tve, let it be like that if it
was already negative indicating
that index+1 is present.

→ Iterate through, find the ^{first} positive element's index.

answer is index+1 or

If all are -ve ans = N+1.

Eg. 0 1 2 3 4 5 6 7 8 9
 go to p = 5, make it -ve.
 if not already. 2 8 -8 100 4 -5 100 7 -8
 ∞ ∞ ∞ ∞

Ele. ans $\in [1, N+1]$ ans $\in [1, n]$

int firstPositiveMissingNum(int arr[], int n)

{ int ans = n+1;

1) for (int i=0; i<n; i++)

{ if (arr[i] ≤ 0 || arr[i] > n)

 arr[i] = Integer.MAX_VALUE;

3

idx	num
0	1
1	2
2	3

2) `for (int i=0; i<n; i++)`
 {
 int ~~x = arr[i]~~^{abs};
 if ($i \leq n$) $\rightarrow x = x - 1$;
 if ($x \geq 0$ if $x < 0$)
 {
 if ($a[x] > 0$)
 {
 a[x] = -a[x];
 }
 }.

3) `for (int i=0; i<n; i++)`

{ if ($a[i] > 0$)

{ ans = $i + 1$;

}; break;

return ans;

}

0	1	2	3	4	5	6
2	∅	X	∅	100	2	5
↓	-6	∞	∞	∞	∞	∞

first positive idx

ans = idx + 1

N = 7

(*) Ch

Prt
f

Ques 18
① Find the frequencies of all elements from 1 to N.
 $arr: 3 \ 5 \ 1 \ 2 \ 4 \ 12 \ 6 \ 8 \ 5 \ 7 \ 9 \ 2 \ 1 \ 10 \ 11$

$$1 \leq arr[i] \leq N$$

$$0 \Rightarrow 5$$

$$1 \Rightarrow 2$$

$$2 \Rightarrow 1$$

$$3 \Rightarrow 0$$

② Brute Force: Count sort, Hash Maps

③ Comp: - $O(N), O(1)$

$arr: 3 \ 5 \ 1 \ 2 \ 4 \ 12 \ 6 \ 8 \ 5 \ 7 \ 9 \ 2 \ 1 \ 10 \ 11$
+100 +100 +100 +100
90+3+2 = 100 $y_{max} = 1$
add 100 add 100 add 100.

idx Ele. crd.
0 1 $\Rightarrow ()$
1 2 $\Rightarrow ()$
2 3 $\Rightarrow ()$

(*) Nu

90
2

Add something greater than N like 100 or 1000 etc
N+1 or N+2 etc

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

void findFrequency(int arr[], int N)

{ int x = N+1; x+=1; --

for (int i=0; i<N; i++)

{ int idx = arr[i] % x - 1;

arr[idx] = arr[idx] + x;

W int freq = new int[N];

for (int i=1; i<N+1; i++)

{ freq[i] = arr[i-1] / x;

}

for
{ i=1

(*) Check if a number is prime:

```

int boolean isPrime(int N)
{
    if (N <= 1) return false;
    for (int i=2; i <= sqrt(N); i++)
    {
        if (N % i == 0)
            return false;
    }
    return true;
}
    
```

(*) Number of prime numbers from 1 to n , including n .

```

int NoPrimes (int N)
{
    int count=0;
    for (int i=2; i <= N; i++)
    {
        if (isPrime(i)) --> N
        count++;
    }
    return count;
}
    
```

\Rightarrow Comp.: $O(N + \sqrt{N})$, O(1).

$$\begin{aligned}
 & P: 2, 3, 4, 5, \dots, \sqrt{n} \\
 & H: \sqrt{2}, \sqrt{3}, \sqrt{4}, \sqrt{5}, \dots, \sqrt{n} \\
 & \text{Total} = \sqrt{2} + \sqrt{3} + \sqrt{4} + \sqrt{5} + \dots + \sqrt{n}
 \end{aligned}$$

Total = sum of sqrt of first n natural no. ≈ 1 .

```

int NoPrimes (int N)
{
    bool count[N+1] = { false } // assume all as true
    // all are prime
    count[0] = false;
    count[1] = false;
}
    
```

```

for (int j=2; j <= N; j++)
{
    if (count[j] == false)
        for (int i=2; i <= N+1; i = i+j)
            if (count[i] == true)
                count[i] = false;
}
    
```

```

    if (count[i] == true)
        count[i+j] = false;
}
    
```

$n=13$ count[14]

2 3 4 5 6
7 8 9 10 11 12 13

for primes (int N)

{ for $p \leq N$; $j=2$

for $c=0;$

$p(j) = p(j-1)$

for (int $i=2$; $i \leq N$; $i++$)

{ if ($p(i) == 1$)

 c++

 for (int $j=(2*i)$; $j \leq N$; $j+=i$)

$p(j) = 0;$

return c;

0	2	3	5	7	11	13	17	19	23	29	31	37	41	43	47	53	59	61	67	71	73	79	83	89	97	
9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62
63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89

Coupli- $O\left(\frac{N}{2} + \frac{N}{3} + \frac{N}{5} + \dots + \frac{N}{P}\right) = N \cdot \left(\frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \dots\right)$ P is the largest prime less than n .

$$P \Rightarrow P = \sqrt{N}$$

$$P \Rightarrow P \approx \sqrt{N}$$

\Rightarrow it will skip prev 0 to 4 checking as already checked with lesser primes.

$$\text{Lippe of it: } \frac{N}{5} - 4$$

$$\Rightarrow O\left(\sqrt{N} \left(\frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \dots + \frac{1}{P} - (P-1) \right)\right)$$

Largest prime less than or equal to \sqrt{N} .

int primes(int N)

bool p[N+1] = {T};

p[0] = p[1] = F;

for (int i=2; i<=sqrt(N); i++) // Use only primes to
cancel the multiples

if (p[i] == T) instead of considering all

for (int j=i*i; j<=N; j=j+i)

p[j] = F;

int count = 0;

for (int i=0; i< N+1; i++)

if (p[i] == T)

count++;

return count;

$\Rightarrow O(\sqrt{N} \left(\frac{1}{2} + \frac{1}{3} - 2 + \frac{1}{5} - 4 + \dots + \frac{1}{p} - (p-1) \right))$.

* Calculate the no. of primes in given range [a, b]

$0 \leq a \leq b \leq 10^{10}$ Range = $b-a+1$

$b-a \leq 10^6$

e.g. [3, 10] $\Rightarrow 3$

[51, 7263] \Rightarrow ()

EXTENDED

SIEVE.

\Rightarrow first form a list L of all prime numbers till \sqrt{b} .

def list L = {2, 3, 5, ... p }

L(i) = prime no.

Range [a, b]

$i = \text{index of list}$

$L[i] = \text{prime no.}$

$L = \{2, 3, 5, \dots\}$

$\frac{0}{51} \quad \frac{1}{52} \quad \frac{2}{53}$

```

int primesInRange (int a, int b)
{
    bool p[b-a+1] = {0}; // init. p[i] = false
    for (int i=0; L[i] <= b; i++)
    {
        if (a % L[i] == 0)
        {
            a++; // skip multiples of L[i]
        }
        else
        {
            p[i] = true;
        }
    }
    int count = 0;
    for (int j=a; j<=b; j++)
    {
        if (p[j-a])
        {
            count++;
        }
    }
    return count;
}

```

// go through P[] get count of True.

return count;

Precomputations:

Sieve

functions.

NCR

Fibonacci

Prime Number

Factorials = $O(N)$

$$1 \leq T \leq 10^5$$

$$1 \leq N \leq 10^6$$

for each fact., $O(T \times N) = 10^{11} \cdot 10^6 \geq 10^{17}$ Time Out

Precompute all. $\rightarrow O(N + T) \cdot 10^6 \leq 10^{16}$

④ Game Theory

→ 2 Player Game

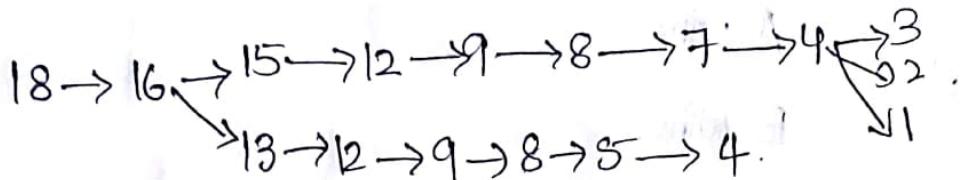
→ Alternate Turns.

A vs B.

→ ① N coins, Moves - 1, 2, 3. players A, B.

A, B can pick any of 1, 2, 3

wins if after your lifting $\frac{\text{no. of coins}}{2}$ left are
loses if after your lifting $\frac{\text{no. of coins}}{2}$ left are



Game theory: W = win, L = loss.

Optimal playing give the other person multiples of moves: 1, 2, 3. N: 0 1 2 3 4 5 6 7 8 9 10 11 12 13

Optimal: L w w w L w w w L w w w L

If A starts and get himself multiple of 4 i.e. $(N+1)$

then A loses, B wins & viceversa.

A starts: If $(N \% 4 == 0)$

B wins.

else

A wins., N is no. of coins.

win: $\frac{3}{4}$ moves leads to L_{optimal}
lose: All moves leads to W

② Optimal vs Greedy.

N: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

greedy: L w w w L L L w L L L L L L L

Optimal: L w w w L W W W W W W W W W

If Greedy starts and $N=1, 2, 3, 7$.

Greedy wins
if Optimal starts and $N=4$

In all other cases, greedy will lose & optimal wins.

Greedy always picks max value coin

Greedy vs Greedy

N: 0 1 2 3 4 5 6 7 8 9 10 11 12 13
 Greedy: L W W W L L L W W W L L L W.

If A always starts

$$\text{if } (N \% G == 1 \text{ or } 2 \text{ or } 3)$$

A wins

else

B wins.

win: A moves leads to L
lose: A moves leads to W

② N coins - Moves = 1, 2, 4, 8, 16, 32, ... powers of 2

There are N coins, each player can pick any power of 2 number of coins.

Greedy vs Greedy: Picks max pow of 2 in N

N: 0 1 2 3 4 5 6 7 8 9 10 11 12 13

G1: L W W L W L L W W L L W

No pattern

Optimal vs Optimal: moves = 1, 2, 4, 8, 16, 32

N: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

O: L W W L W W L W W L W W L W W

If A starts.

$$\text{if } (N \% 3 == 0)$$

B wins

else

A wins.

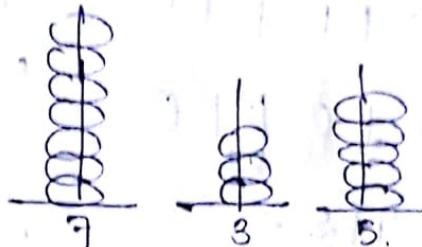
If pattern is found \Rightarrow time can do in O(1)

But if pattern is not found, then construct game table - O(N^2) time
and then access N from computed game table

4. Snuke VI/8 Optimal: moves: 1, 2, 4, 8, 16, 32

N: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
 G1: L W W L W L W W L L W L L L L
 O: L W W L W W W W W W W W W W W W

④ Name of Piles.



In 1 move can pick any no. of piles from any, but only from that then.

In order to win

give the opponent the situation such that XORs of given become 0.

XORs of given become 0.

so 4 moves XOR=0.

If for move we give other person XOR=0, then we will win

	7	3	5	7: 111 → 6: 110
→	6	3	5	3: 011 3: 011
	6	3	4	5: 101 5: 101
	6	2	4	001 000
→	5	2	4	000
	5	2	4	000
→	5	1	4	000
	5	1	4	000
→	3	1	2	3: 011 3rd 01100101 1: 001 4th 0001101 2: 010 5th 1011111
	3	1	2	0.0.0
				0.0 X 0 X 0 X 0

make no. of 1's to even

e.g. 1's \Rightarrow 3 $\xrightarrow{4 \text{ inc } 1's}$ $\xleftarrow{2 \text{ dec } 1's}$

Come up with move such that you make XOR as 0.

take away 22 from 3rd pile to make XOR 0

-
time

	1010110	64 32 16 8 4 2 1
1st	0100111	\oplus 010101010101
2nd	011001001	1111001
3rd	0001101	0001111
4th	1011111	1111111
5th	00X0X10	0101010

from 3rd: $-16 - 4 + 2 \Rightarrow$

we can add. do XOR from left side.

& only when we have removed greater no. of piles already.

X0X0X0X0X0X0

$\downarrow -64 + 32 - 8 - 4 + 1.$

NIMs Game

Win: \exists move leads to Lose lose $\text{XOR} = 0 \wedge$ more $\text{XOR} \neq 0$

Lose: \forall moves lead to win win $\text{XOR} \neq 0 \exists$ move $\text{XOR} = 0$

* Grundy Numbers- GD

* Find Real World Applications for each of the data structures

7	8	5
\Rightarrow	7	2 5
6	2 8	\Rightarrow 6 th : 110 8 th : 010
\Rightarrow	6	2 4 0 1, 8 th : 10X0
5	8 4	$\frac{00X0}{}$
\Rightarrow	5	1 4
4	1 0 1	remove 1 from 5 th
3	0 0 1	5 th : 101
2	1 0 0	8 th : 0101
1	0 0 0	4 th : 100

6: 110
2: 010
4: 100
000

- piles size denotes moves.

opp. got XOR^0
so whatever he does helps.
Remove 1 from 2th.

i.e. can pick up any of no. of piles from pile size.

denote by 0 and 1 won up

④ Subset sum Problem

check if the given sum exist in any of subsets sum.

①: Get all subset & check: $2^N \times N$, 1

②: 2^N , 1 Using recursion.

But if $N=40$, then $2^N = 2^{40} = (2^{10})^4 = (10^3)^4 = 10^{12} > 10^8$ TimeOut

③. Partition array & generate subsets, for that partition.

		A[0:N]	B[N:1]
Ex:	5	5 + 3 ⇒ sum 0	10 7 8 → sum
Subsets:	5	5 → 5	10 → 10
	1	1 → 1	7 → 7
R = N(0)	3	3 → 3	8 → 8. P: B[N-1]
	5, 1	5, 1 → 4	-10, 7 → -3
	5, 3	5, 3 → 8	-10, 8 → -2
	1, 3	1, 3 → 2	7, 8 → 15
	5, 1, 3	5, 1, 3 → 7	-10, 7, 8 → 5

$$a + b = k$$

Complexity: $2 * 2 * \frac{N}{2} + \text{Sum of Pairs}$ then Sum of Pairs.

left half Subsets for half calc. sum of subset elements $N = 2^{\frac{N}{2}}$ elements.

- 1) Brute force: $N, 1 \rightarrow (2^{\frac{N}{2}})^2 = 2^N$
- 2) Sorting & 2pointers: $N \log N, 1 \rightarrow 2^{\frac{N}{2}} \log_2 \frac{N}{2} \rightarrow O(N \cdot 2^{\frac{N}{2}})$
- 3) Sorting & Binary Search: $N \log N, 1 \rightarrow O(N \cdot 2^{\frac{N}{2}})$

$2^{\frac{N}{2}} + 2^{\frac{N}{2}}$
↑ insert in HM
↑ compare & search in sec HM.

{ 4) Hash Map

{ 5) Hash Set

$\frac{N}{2} 2^{\frac{N}{2}}$
 $N \cdot 2 + 2 +$
sums

Take sum of pairs of hashing \rightarrow then $\underbrace{Q \times Q^{\frac{Nb}{2}}}_{\text{subset sep.}} + \underbrace{Q^{\frac{Nb}{2}} + Q}_{\text{Hashing}}$

Sum of Pairs -

a) $Q^{\frac{Nb}{2}} \cdot Q^{\frac{Nb}{2}} = Q^N$. Brute-force

b) $N \cdot Q^{\frac{Nb}{2}} + Q^{\frac{Nb}{2}} + Q^{\frac{Nb}{2}}$ Sorting + 2 pass

c) $N \cdot Q^{\frac{Nb}{2}} + Q^{\frac{Nb}{2}}, N^{\frac{Nb}{2}}$ Sorting + Binary Search.

d) Hash map $\{ Q^{\frac{Nb}{2}}, Q^{\frac{Nb}{2}} \}$

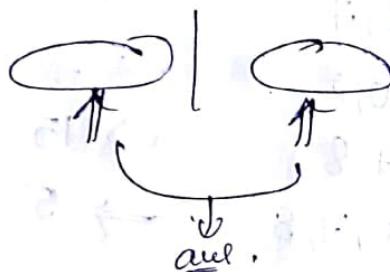
e) Hash Set $\{ Q^{\frac{Nb}{2}} + Q^{\frac{Nb}{2}} \}$

$$\Rightarrow \boxed{N \cdot Q^{\frac{Nb}{2}} + Q \cdot Q^{\frac{Nb}{2}}}$$

If $N=40$

$$40 \cdot 2^{20} + 2 \cdot 2^{20} \approx 10^6 < 10^9$$

Meet in the Middle (mHM)



$Q^N \Rightarrow \frac{Q^N}{Q} = Q^{\frac{Nb}{2}}$ will not reduce complexity, $Q^{\frac{Nb}{2}}$ will reduce comp.

$N^q \Rightarrow \frac{N^q}{2} = N^{\frac{q}{2}}$ will not reduce comp. $N^{\frac{q}{2}} = N^2$ will reduce comp.

$$a+b+c+d = k$$

$$(a+b) = k - (c+d)$$

$$\underbrace{N^2}_{N^2} = \underbrace{N^2}_{N^2}$$

$$N^2 \Rightarrow \underline{N^2}$$

INTERVIEW PUZZLES

→ 5 slots.

$$\underline{0} \quad \underline{1} \quad \underline{2} \quad \underline{3} \quad \underline{4}$$

freq. $\begin{matrix} 4 \\ 0 \end{matrix} \times$ $\begin{matrix} 1 \\ 1 \end{matrix} \checkmark$ $\begin{matrix} 3 \\ 2 \end{matrix}$ $\begin{matrix} 2 \\ 3 \end{matrix}$ $\begin{matrix} 1 \\ 4 \end{matrix} \times$) 4 should occur once.
 No. $\begin{matrix} 0 \\ 0 \end{matrix}$ $\begin{matrix} 1 \\ 1 \end{matrix}$ $\begin{matrix} 2 \\ 2 \end{matrix}$ $\begin{matrix} 3 \\ 3 \end{matrix}$ $\begin{matrix} 4 \\ 4 \end{matrix}$
 0 should occur 4 times. X

$$\underline{0} \quad \underline{1} \quad \underline{2} \quad \underline{3} \quad \underline{4} \quad \underline{\frac{3}{4}} \times$$

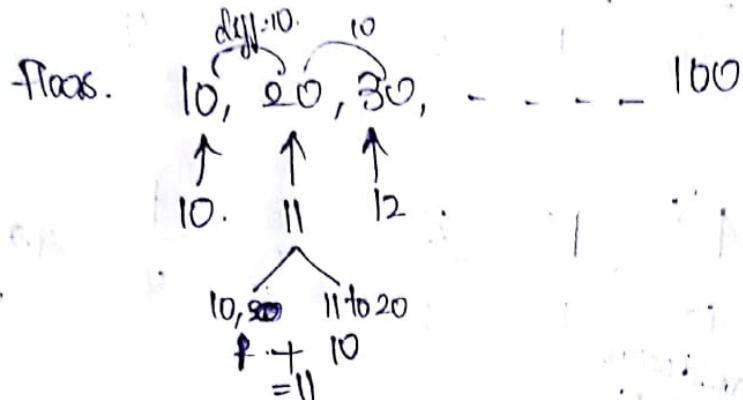
$$\underline{0} \quad \underline{1} \quad \underline{2} \quad \underline{3} \quad \underline{4} \quad \underline{\frac{0}{4}} \times$$

$$\underline{0} \quad \underline{1} \quad \underline{2} \quad \underline{3} \quad \underline{4} \quad \underline{\frac{0}{4}} \times$$

freq. $\begin{matrix} 2 \\ 0 \end{matrix} \times$ $\begin{matrix} 1 \\ 1 \end{matrix} \checkmark$ $\begin{matrix} 0 \\ 2 \end{matrix}$ $\begin{matrix} 0 \\ 3 \end{matrix}$ $\begin{matrix} 0 \\ 4 \end{matrix} \times$
 num. $\begin{matrix} 2 \\ 0 \end{matrix} \times$ $\begin{matrix} 1 \\ 1 \end{matrix} \checkmark$ $\begin{matrix} 0 \\ 2 \end{matrix}$ $\begin{matrix} 0 \\ 3 \end{matrix}$ $\begin{matrix} 0 \\ 4 \end{matrix} \times$

Answer is H

- (*) 100 floors and 2 eggs. ~~in abt 10 min~~
 Find the floor at which from where dropping of egg
 doesn't break. Do it with min Egg drops.



10 → 19 → 27 → 34 → 40 → 45 → 49 → 52 → 55 → 58

$$\frac{x(x+1)}{2} \geq N$$

$$x=10 \Rightarrow \frac{10(11)}{2} = 55 \nleq 100 \times$$

$$x=11 \Rightarrow \frac{11(12)}{2} = 66 \nleq 100 \times$$

$$x=12 \Rightarrow \frac{12(13)}{2} = 78 \nleq 100 \times$$

$$x=13 \Rightarrow \frac{13(14)}{2} = 91 \nleq 100 \times$$

$$x=14 \Rightarrow \frac{14(15)}{2} = 105 \geq 100 \checkmark$$

14 attempts.

30/3/19

STACKS & QUEUES

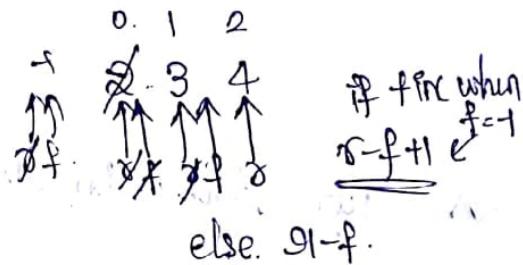
- Stacks. $t = -1$ initially.
- LIFO | FILO
 - push(x) $\Rightarrow arr[++t] = x$
 - pop() $\Rightarrow t--$
 - top() / peek() $\Rightarrow arr[t]$
 - size() $= t + 1$

- Queues: $q_i = -1, f = -1$
- FIFO | FILO
 - enqueue(x) $\Rightarrow arr[++q_i] = x$
 - dequeue(x) $\Rightarrow f++$
 - front() / rear() $\Rightarrow arr[f], arr[q_i]$
 - size() $= q_i - f$

Arrays / Linked List implementation.

Queue

- enqueue $\Rightarrow arr[++q_i] = x$
- dequeue $\Rightarrow f++$
- front() $\Rightarrow arr[f+1]$
- rear() $\Rightarrow arr[q_i]$
- size() $\Rightarrow q_i - f$



Stack: $t = -1$

- push - 70 ele. $t = 70$.
- pop - 30 ele $t = 40$.
- push - 50 ele $t = 90$

(In range).

Queue:

- enqueue - 70 ele. $f = 70$
- dequeue - 30 ele. $f =$

Circular Queue:

- enqueue(x): $arr[(q_i + 1) \% N] = x$

Dequeue(): $f = (f + 1) \% N$

front(): $arr[(f + 1) \% N]$

rear(): $arr[q_i]$

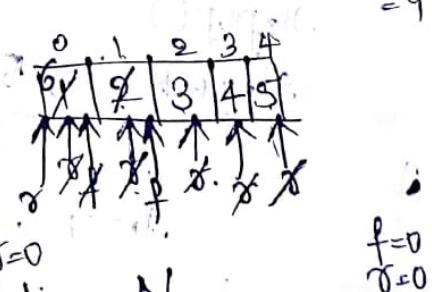
size(): $(q_i - f + N) \% N$

$N = 100$, will not work.

$f = 39$

$q_i = 70 \Rightarrow (70 + 50) \% 100 = 20$.

$E(71), D(40), size = 31, F(50) \Rightarrow size(81)$.



$N + f - q_i$

$f = 4$, size gives 0, but should be 100.

Find the size of stack.

```

count = 0;
while (prev[i] != -1)
{
    count++;
    i = prev[i];
}

```

```


    if idx = i;
    while (idx != -1)
    {
        count++;
        idx = prev[idx];
    }
    return count;
}


```

⇒ Instead maintain size array

push - inc value
pop - dec value

Size array:
 - prev - N
 - size - m
 - t - m
 - Empty list - N } $\& N + \alpha m$

Maintain empty list as stack or queue or array list.

⑧ Stack Operations along with getMax():

Stacks: 10, 3, -1, 15, getMax(), ↑, pop, getMax(), 10, peek(), ↓

push()

pop()

top()

size()

getMax()

LIFO:

normal Stack	10	3	-1	15	10

max Stack	10	10	10	15	12

top() of max stack

compare & push

max stack

Stack: push(x), pop(), top(), size(), getMax(), getMin(),
 for avg: maintain sum: $\text{push} \rightarrow \text{add } x \text{ to sum}$ $\text{getAvg}()$
 $\text{pop} \rightarrow \text{sub. from sum}$

$$\text{Avg} = \frac{\text{sum}}{\text{size()}}$$

for getMax() \rightarrow max stack.

for getMin() \rightarrow min stack.

① Implement Queue using Stacks:

- can use only stack operations.
- use only stack::inbuilt func. Stack<Integer> s = new Stack();
 & push()
 & pop()
 & top()
 & size().

Enqueue()

Dequeue()

front()

rear()

size()

①

4

X

8, 7, 10

12, 10, 7

4, 10, 7

7, 4, 3

7

3

S1

S2

(ON) for every dequeue, shifting
 as all the elements from S1 to S2.
 time shifting from S2 to S1.

- ②. for dequeue, let S2 be like that. & deque from S2
 if S2 becomes empty, then move S1 to S2

3, 10
4, X
X, 7
S1

X, 4
7, 4
10, 3
S2

- Q) enqueue: push into s1.
 If s2 is empty, push all from s1, pop from s1, push to s2.
- Q(N) Dequeue: pop from s2.

O(1) size(): $s1.size() + s2.size()$

O(N) front(): $s2.top()$, if s2 is empty, shift all elements from s1 to s2, then pop from s1, push to s2.

O(1) enque(x): whenever we enqueue something, update deque.

O(1) dequeue(x): if not like above

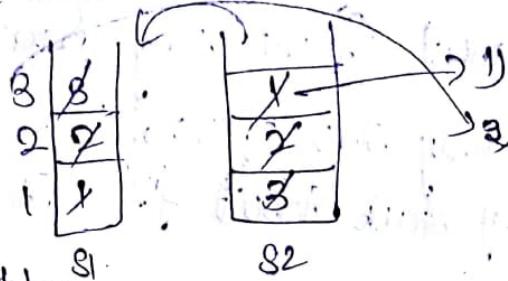
else $\downarrow O(N)$

1) front()?

2) read()?: $s1.top$

3) front()?

shift to s2



goes on $O(N)$

① Asymptotic Analysis:

② Amortized Analysis:

E(0) = $O(1)$

D(j) = $O(N)$

S(i) = $O(1)$

F(i) = $O(N)$

R(C) = $O(1)$

Ignore
Costly but
Rare Operations.

Dequeue is costly only
when s2 is empty so,

ignoring costly opr.
is considered in

Amortized Analysis.

③ Growth no. of operations (4)
on each element.

Life Cycle: Next 4 operations

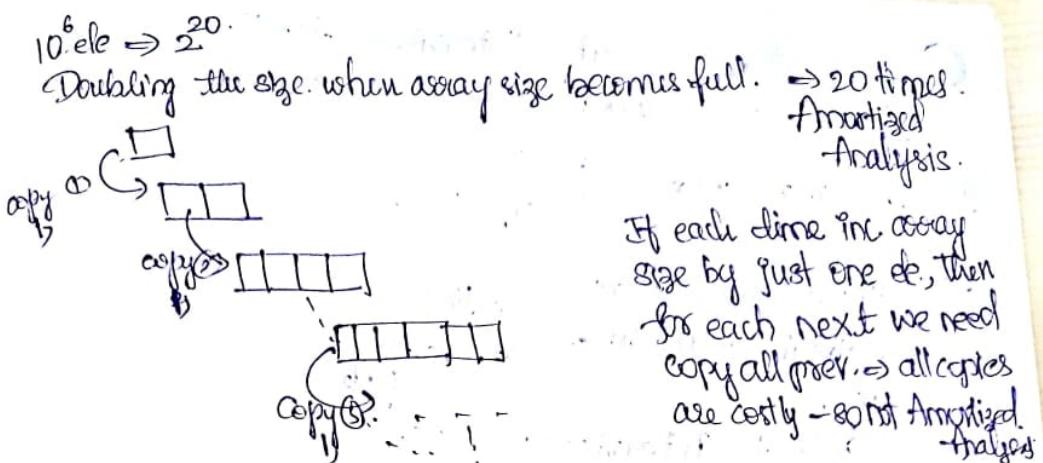
No. of opr = $4N \Rightarrow O(1)$

Python - [] :-

Java: ArrayList < >

C++: Vector < >

push all ele from s1 to s2
from s1, push to s2
all ele from s1 to s2
update array.



If each time inc. array size by just one ele, then for each next we need copy all prev. \Rightarrow all copies are costly - so not Amortized Analysis.

Amortized Analysis $\Rightarrow 20 \text{ times}$.
If no. of ele to be copied ie no. of copies: $(2^0 + 2^1 + 2^2 + \dots + 2^9) + 2^{20}$

$\Rightarrow O(1)$

iterations is counted.
 $\Rightarrow 2^{20} = 2^{10} \cdot 2^{10} \Rightarrow 2N$.

no. of insertions $\Rightarrow O(1)$.

\Rightarrow Always maintain some 25% to 50% space to do push & pop efficiently.

\times Checkout libraries for:

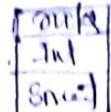
- sort
 - binary search
 - hashing
 - stacks / queues
 - heaps
 - min/max / sum
 - pairs
- for a store pair as $a \xrightarrow{w} b$
 $\langle b, w \rangle$
neighbor weight

operations (4)
push / pop
top / bottom

$\Rightarrow 4 \text{ op each}$

$\Rightarrow O(1)$

④ Que: stock Interview stocks
Ans: → stocks Interview Smart

① use extra space  POP.

⑤ without extra space: at least
 Maintain 2 pointers <at. last>

Smart Interviews stocks

i) stock - interview - strings

→ total first rev. word by word

ii) stocks - Interviews - Smart

⑥ Valid/Invalid Parenthesis: time, space

1 { () [] { () } } ⇒ use stack $O(N), O(N)$

2 (())

count = 0;

3 (() (()))

4 $\{ \Rightarrow \text{count}++$

5 $\}) \Rightarrow \text{count}--$

optimize for
single type of
brackets

At any point of time, if count becomes negative, return false.

Eg) (

return false.

① Find the first smaller element on the right side:

else -1.

arr: arr: 3 7 10 8 12 15 9 2 5 8 6 3

Ans: arr: 2 2 8 2 9 9 2 1 3 6 3 +

② Brute Force: $O(N^2)$, $O(1)$

```
for i = 0 to N
    bsr[i] = -1
    for j = i+1 to N
        if (arr[j] < arr[i])
            bsr[i] = arr[j]
            break;
```

③ Maintain Stack:

arr: 3 7 10 8 12 15 9 2 5 8 6 3
2 2 8 2 15 9 9 2 1 3 6

10	8	2	15	9	9	2	1	3	6
7	2	5							
3									

If ele is greater than top of stack
push it, else check

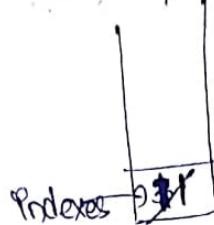
Instead of pushing elements in stack, push indexes into stack

```
void firstSmallerEleRt(int arr[])
{
    stack<int> s;
    for (int i=0; i<N; i++)
    {
        while (s.size() > 0 &&
               arr[i] < arr[s.top()])
        {
            // pop and populate
            bsr[s.top()] = arr[i];
            s.pop();
        }
        s.push(i);
    }
}
```

Complexity: $O(N), O(N)$

Start from right side.

0	1	2	3	4	5	6	7	8	9	10	11
3	7	10	8	12	15	9	2	1	3	6	4
2	2	8	2	9							



Indexes $\boxed{10}$ \times cannot

No need of indexes if
start from right as
replicate the same pos.

$6 > 3 \Rightarrow$ put 3 at idx 10.
push 6.

$8 > 6 \Rightarrow$ put 6 at idx 9.
push 8.

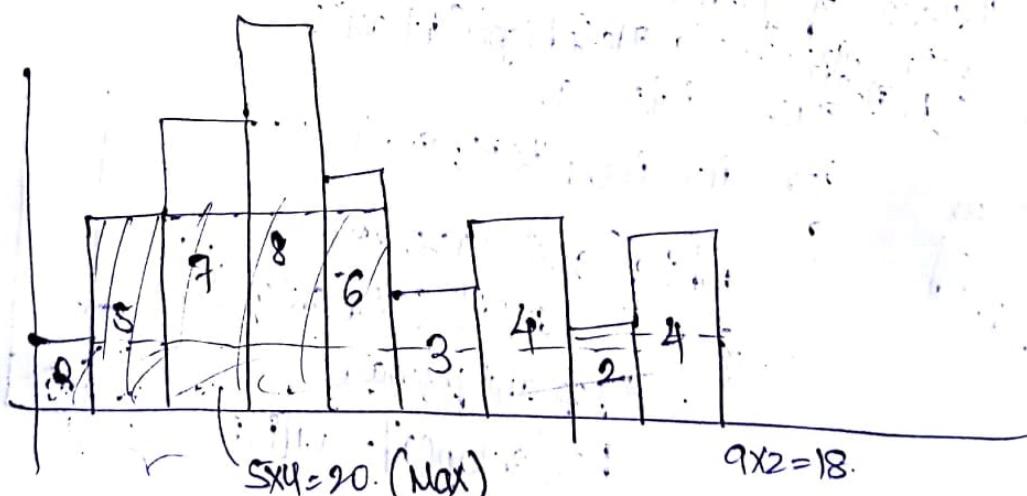
$5 < 8 \Rightarrow 8$ can't be ans.
pop 8.
can't be ans, pop

$5 > 3$

(Copy $O(N)$, $O(N)$)

* Smaller elements on the left $\textcircled{3}$ $O(N), O(N)$

* Maximum Area of the Histogram.



113

$\text{m}_i \text{ dy}$
 $\geq \text{topc}_i$

1

$\times 10$.

9

be an

us; pop 6

for every building, iterate towards left until find lesser ele, P
iterate towards right until we find lesser.

$$\text{Area} = (\rho_2 - \rho_1) - 1 \rightarrow \omega_1[?]$$



$$P1 \quad (12 - (-1) - 1) = 2$$

$$(p_2 - p_1 - 1) \neq 0 \text{ or } 1$$

(12-7-1) *3

$$(p_1 \xrightarrow{e} p_2)$$

↓
first smaller
ele. index on
left

-if none $p1 = -1$

first smaller
ele index on
right.

if none, p2 = N.

①. $O(N^2), O(1)$

②. $O(N), O(N)$

↑
on
2 stalks

* Find the max element in the window of size k.

1, 12, 10, 7, 9, 5, 3; 1, 2, 6.

① Brute force: $O((N-k+1) * k)$, $O(k)$

Time complexity: $O(N \cdot R \cdot T)$ — not optimized.

• If $k = \frac{N}{2}$ then $O(N^2) -$ (int N, int k)

```

    for( int q=0; q<N-K; q++)
        ans = ans + a[q];
}

```

$\inf \sin x = -\infty$ as $\sin x = \pi$

```
for (int j=0; j<HK; j++)  
    ans = max(ans, a[j]);
```

$$\text{ans} = \max($$

sout(Caus);

② Use sorted hash map \Rightarrow Tree Maps:

5 1 12 10 7 9 5 3 1 2 6

Copy

$\Rightarrow O(N \cdot \log k), O(k)$

5 → 1
1 → 1
12 → 1
10 → 1
{
} {
} {
} {
} {
}

Sorted one
Tree
Map.

12 → 1
5 → 1
10 → 1
12 → 1

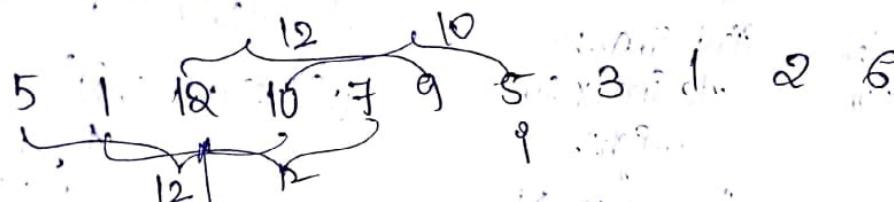
get first ele
 $\Rightarrow \max$

Sorted order map
of. Key
 $\Rightarrow \log k$

\log_2

I
D
S
log

③



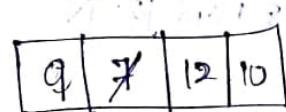
front()

If ele. is at
(q[x]) remove
(q[x]) q[t]

5, 1, 12, 10, 7

front

back



front back

small put.
big remove sm.
small

ans
dq.pop()

popBack()

pushBack()

popBack()

pushBack()

popFront()

pushFront()

popFront()

pushFront()

popBack()

pushBack()

popBack()

pushBack()

popFront()

pushFront()

Double Ended Queue : DEQUE

Double Linked List

for Nele.

O(1)

back()

front()

pushBack()

pushFront()

popBack()

popFront()

Comp:

1.1

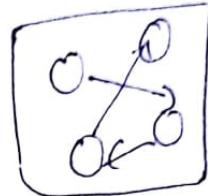
LINKED LIST

→ Singly }
 → Doubly } Circular.



→ Arrays
 → contiguous
 → Direct Access

→ Linked List
 → any memory loc.
 → should have to iterate through to access an element



class Node

{ int data;

Node next, prev;

}

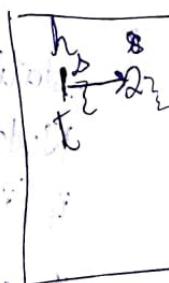
Node CreateNode(int x)

{

Node n = new Node();
 n.data = x;
 n.next = NULL;
 n.prev = NULL

return n;

}



* Create List: SLL 1 → 2 → 3 → 4 → ... N

return head

Node CreateList(int N)

{

Node head = createNode(1); Node temp = head;

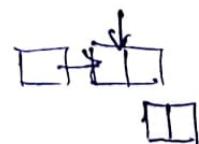
for(int i=2; i<=N; i++)

{ Node second = createNode(i);
 head.next = second;

~~second~~ head = second;

}

return temp;



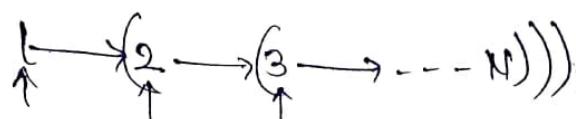
```

1. Node createList (int N)
{
    Node head = createNode(1); ①
    Node sthead = head; ②
    for (int i=2; i<=N; i++) ③
    {
        head.next = createNode(i); ④
        head = head.next; ⑤
    }
    return sthead; ⑥
}

```

Recursive Code:

- Assumptions
- Main Logic
- Base Condition.



```

1. Node createList (int N)
{

```

```

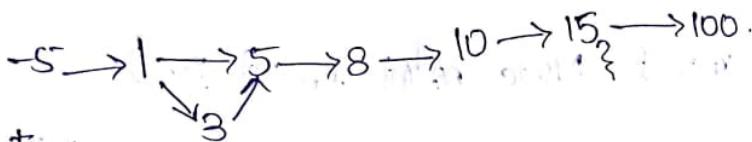
    static int i=1; ①
    Node head = createNode(i); ②
    if (i==N) ③
        return head;
    i++; ④
    head.next = createList(N); ⑤
    return head; ⑥
}

```

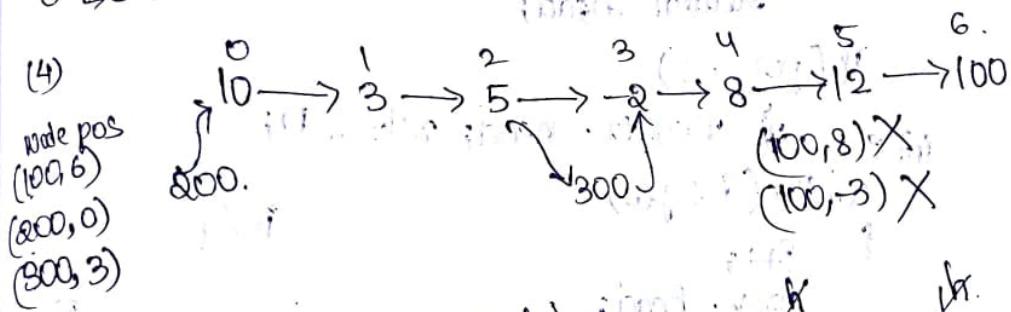
$i=1, 2, \dots, N$

Diagram showing the recursive call stack for $N=3$. The stack consists of three nodes labeled (1), (2), and (3). Each node has a self-loop arrow pointing to its 'next' pointer. The first node (1) has a self-loop arrow pointing to its value field, which contains '1'. The second node (2) has a self-loop arrow pointing to its value field, which contains '2'. The third node (3) has a self-loop arrow pointing to its value field, which contains '3'.

- ① void print(Node h);
 - ② int size(Node h);
 - ③ void printReverse(Node h);
 - ④ Node insertAtPosition(Node h, int x, int pos)
 - ⑤ Node insert(Node h, int x)
- ↳ Sorted Linked List.



empty
Or one node
Or, Or two.

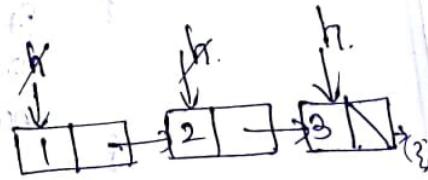


① void print(Node h)

```

while(h!=null)
{
    cout(h.data);
    h=h.next;
}

```



② int size(Node h)

```

int size=0;
while(h!=null)
{
    size++;
    h=h.next;
}
cout(size);

```

③. void printReverse(Node h)

```

{
    if(h==NULL)
        return;
    printReverse(h.next);
    cout(h.data);
}

```

④. Node insert(Node head, int x, int pos);

```

{
    Node temp = head; Node m = createNode(x);
    if(pos < 0 || pos > size(h))
        return head;
    if(pos == 0)
        m.next = head; return m;
    int i=0;
    while(i < pos)
    {
        i++;
        prev = head;
        head = head.next;
    }

```

```

    m.next = prev.next;
    prev.next = m;
}

```

~~sol~~ Node insert(Node h, int x, int pos)

```

{
    if(pos < 0 || pos > size(h))
        return h;
    Node temp = h;

```

```

    Node m = createNode(x);
    if(pos == 0)
    {
        m.next = h;
        return m;
    }
}

```

}

```

while(--pos)
    h=h.next;
n.next=h.next;
h.next=n;
return temp;
}

```

⑤. Sorted Singly Linked List:

3. $5 \rightarrow 10 \rightarrow 13 \rightarrow 18$ ↗
so.

```

Node insert (Node head, int x)
{
    Node n = createNode(x);
    Node temp = head;
    if (head == NULL || x < head.data)
    {
        n.next = head;
        return n;
    }
    while (h.next != null && h.next.data < x)
        h = h.next;
    m.next = h.next;
    h.next = n;
    return temp;
}

```

⑦ Node deleteAll (Node h, int x)

{
 if (pos < 0 || pos >= size(h))

 return h;

 Node th = h;

 if (th == null)

 Node deleteAll (Node h, int x)

{

 if (h == null)

 return h;

 Node th = h;

 while (h.next != null)

{
 if (h.next.data == x)

 h.next = h.next.next;

 else

 h = h.next;

}

 if (th.data == x)

 return th.next;

 return th;

⑧ void distinct (Node h)

{
 if (h == null)

 return;

 while (h.next != null)

{
 if (h.next.data == h.data)

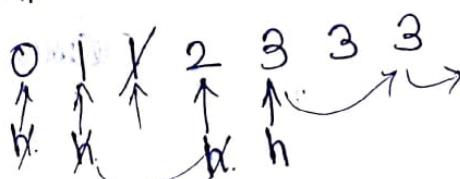
 h.next = h.next.next;

 else

 h = h.next;

}

}



⑥ Iterative Node reverse (Node h)

```

Node p=NULL;
while(h!=NULL)
{
    Node t=h.next;
    h.next=p;
    p=h;
    h=t;
}
return p;

```

Recursive

```

↓
Node reverse(Node h)
{
    if(h==NULL || h.next==NULL)
        return h;
    Node t=reverse(h.next);
    h.next.next=h;
    h.next=null;
    return t;
}

```

\downarrow
 $h \rightarrow (2 \rightarrow (3 \rightarrow (4)))$

② Merge Sort using LinkedLst.

o/p: $l_1: -5 \rightarrow 3 \rightarrow 10 \rightarrow 12 ?$

o/p: $l_2: 8 \rightarrow 5 \rightarrow 7 \rightarrow 13 \rightarrow 15 \rightarrow 18 ?$

Diagram illustrating the merge process:

- Initial state:
 - h_1 : $8 \rightarrow 5 \rightarrow 7 \rightarrow 13 \rightarrow 15 \rightarrow 18$
 - h_2 : $-5 \rightarrow 3 \rightarrow 10 \rightarrow 12$
- Merge step:
 - Compare h_1 .data ($8 < 10$) and h_2 .data ($-5 < 8$).
 h_1 is chosen as the current node.
 d is set to h_1 .
 - h_1 .next is set to h_2 .
 - h_2 .data is copied to d .
 - h_2 .next is set to h_2 .
 - t is set to h_2 .
 - h_1 is updated to h_1 .next (5).
 - t is updated to t .next (3).
 - h_2 is updated to h_2 .next (10).
- Final state:
 - l_1 : $-5 \rightarrow 8 \rightarrow 5 \rightarrow 7 \rightarrow 10 \rightarrow 12$
 - l_2 : $13 \rightarrow 15 \rightarrow 18$

Node Merge (Node h_1 , Node h_2)

{
 Node $d = \text{createNode}(-1)$; // create Dummy Node.

 Node $td = d$;

 while ($(h_1 \neq \text{null}) \& (h_2 \neq \text{null})$)

 {
 if (h_1 .data $<$ h_2 .data)

d .next = h_1 ;

$h_1 = h_1$.next;

 }

 else

d .next = h_2 ;

$h_2 = h_2$.next;

 }

$d = d$.next;

 }

 if ($h_1 \neq \text{null}$)

 {
 d .next = h_1 ;

 }

 else

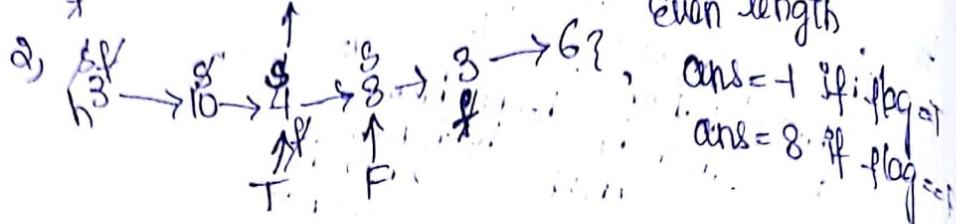
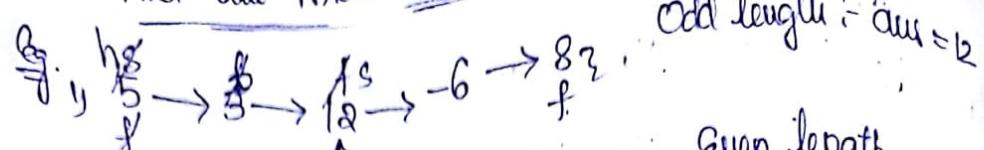
 {
 d .next = h_2 ;

 }

 return td .next;

}

* Find the middle Node in the given Linked List.



Node findMid (Node h, bool flag)

{ Node slow = fast = h;

if (h == NULL) return NULL;

return NULL;

while (fast.next != NULL && fast.next.next != NULL)

{ slow = slow.next;

fast = fast.next.next;

}

if (fast.next == NULL || flag == T)

return slow;

return slow.next;

join

*. SORT A SINGLELINKED LIST:

①. Bubble Sort

②. Selection Sort

③. Insertion Sort

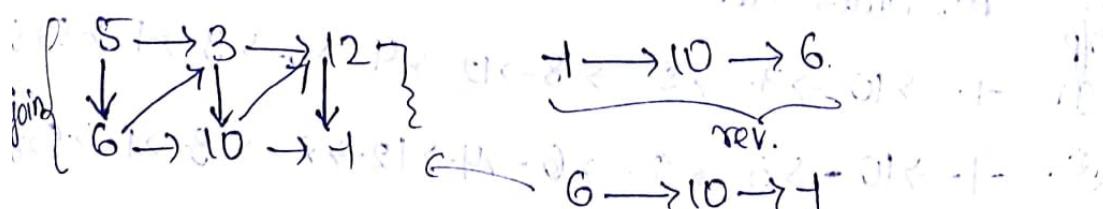
④ MergeSort in Linked Lists;

```
Node MergeSort(Node h)
{
    if (h==null || h.next==null)
        return h;
    Node mid = findMid(h, T);
    Node t = mid.next;
    mid.next = null;
    return merge(mergeSort(mid), mergeSort(t));
}
```

⑤ Form a linkedList with $\text{first} \rightarrow \text{last} \rightarrow \text{second} \rightarrow \text{secondlast} \rightarrow \dots$

Eg) ① I/P: $5 \rightarrow 3 \rightarrow 12 \rightarrow 1 \rightarrow 10 \rightarrow 6$
O/P: $5 \rightarrow 6 \rightarrow 3 \rightarrow 10 \rightarrow 12 \rightarrow 1$

② I/P: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$
O/P: $4 \rightarrow 5 \rightarrow 2 \rightarrow 3 \rightarrow 1$



O/P: $5 \rightarrow 6 \rightarrow 3 \rightarrow 10 \rightarrow 12 \rightarrow 1$

void firstLastLinkedList(Node h)

```
{
    if (h==null || h.next==null)
        return;
}
```

```
    Node m = findMid(h, T);
    Node t = m.next;
    m.next = null;
}
```

```
    join(h, reverse(t));
}
```

```

void join(Node h1, Node h2)
{
    Node temp = h1;
    Node t1, t2;
    if(h2 == null)
        return;
    while(h1 != null & h2 != null)
    {
        t1 = h1.next;
        t2 = h2.next;
        h1.next = h2;
        h2.next = t1;
        h1 = t1;
        h2 = t2;
    }
    if(h1 == null)
        printList(temp);
}

```

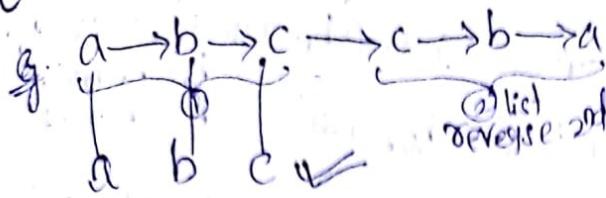
Q. I/p:

 O/p:

- ①
- ②

6/1/19

Check Palindrome:



- 1) Find mid
- 2) Break into two lists.
- 3) Reverse the 2nd list.
- 4) Compare character by character
- 5) Check if they are palindromes

① SLP. $h_1: 5 \rightarrow 3 \rightarrow 8$
 $h_2: 7 \rightarrow 5$

$$x = 538, y = 75 \quad \text{SLP.}$$

$$z = x + y \quad \text{OLP.}$$

$$z = 613.$$

OLP. $6 \rightarrow 1 \rightarrow 3$

$$\begin{array}{r} 1 \\ 5 \\ 3 \\ 8 \\ + \\ 7 \\ 5 \\ \hline 6 \\ 1 \\ 3 \end{array}$$

Adding starts from last until digit.

Hence reverse the lists first.

Then start adding & taking carry to the next.

If ans. taken in new nodes // space comp.

else ans. update in same any one of the list.

ii) Ans. will be the reverse of $8 \rightarrow 3 \rightarrow 5$
 got ans.

$$\begin{array}{r} 5 \\ 7 \\ + \\ 3 \\ \hline 1 \\ 6 \end{array}$$

iii) while updating ans, if
 we add the next ele.
 at the first positions of need, no need to reverse them.

Node add (Node h_1 , Node h_2)

{ return rev (addingLinkedLists (rev(h_1), rev(h_2))) }.

Node addingLinkedLists (Node h₁, Node h₂)

```
Node head = createNode ((h1.data + h2.data) % 10);  
h1 = h1.next; // h1 = h1.next;  
while (h1 != null && h2 != null) {  
    Node next = createNode ((h1.data + h2.data) % 10);  
    h1 = h1.next; // h1 = h1.next;
```

for:

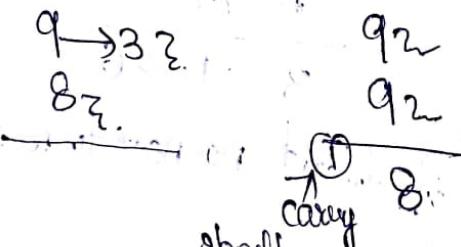
Node addingLinkedLists (Node h₁, Node h₂)

```
int carry = 0;  
Node h = null;  
while (h1 != null && h2 != null) {  
    data = (h1.data + h2.data + carry) % 10;  
    carry = (h1.data + h2.data + carry) / 10;  
    Node next = createNode (data);
```

Append at
end so that new.next = head;
no need of head = new;
rev. at end.

} return head;

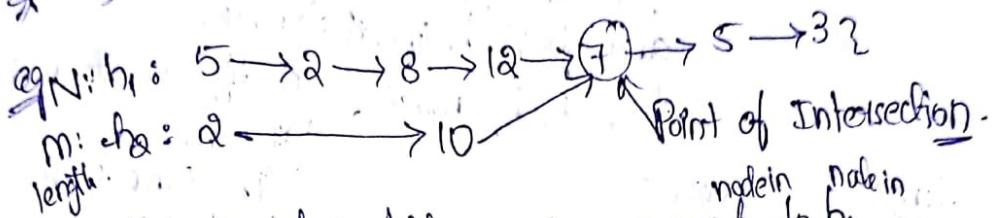
Handle:



should check new node for carry, even after
both lists gets exhausted.

Given 2 linked lists.
 some node of 2nd linked list points to some data of 1st
 linked list. i.e. that point forms the point of intersection
 of 2 linked lists.

- * Q. Do the 2 given linked list intersect? Yes No
- * Q. What is the point of intersection?



- ① Match Based on Address. from every h_2 to h_1 .
- Copy: $O(N \cdot M)$, $O(1)$
 Time Space.

$h_2: 2 \rightarrow 10 \rightarrow 7 \rightarrow 5 \rightarrow 3$

② HashSet<Node>

Put one linked list into HashSet.

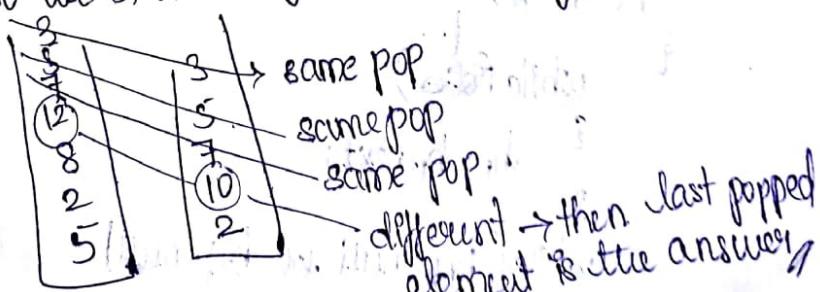
Search for the other nodes of other linked list.

Copy: $O(N + M)$, $O(N)$

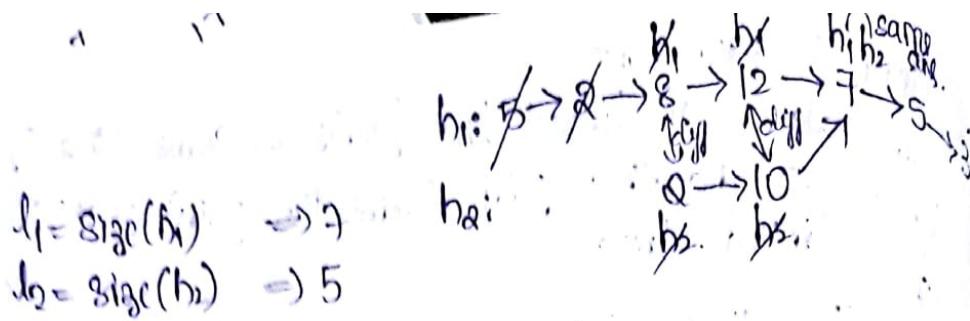
Insert check Hashset.

- ③ * Reversing linked list is not correct as in reversing 7 should point to 12 & 10 which is not possible.

* So use stacks & go with reversing kind.



Copy: $O(N + M)$, $O(N + M)$



$d = |h_1 - h_2|$ → & skip 2 nodes from the bigger ll, and point h_1 to next.ele.node, check h_1, h_2 nodes → if diff. go with h_2 → if same → first same will be the ans.

<u>length 7</u>	<u>length 5</u>	<u>length 6</u>	<u>length 6</u>	<u>length 6</u>
<u>1st 11</u>	<u>1nd 11</u>	<u>1st 11</u>	<u>1nd 11</u>	<u>1st 11</u>
3rd	1st	1st	1nd	1st
4th	2nd	2nd	2nd	2nd
5th	3rd	3rd	3rd	3rd
6th	4th	4th	4th	4th
7th	5th	5th	5th	5th

Compare:

Comparing length of both lists.

Time complexity: $O(N + M)$

Space complexity: $O(1)$

Node pointOfIntersection (Node h₁, Node h₂)

Node point of view

§ 9.1.1. $\text{init } h_1 = \text{shape}(h_1);$ $\text{shape}(h_1) = \text{shape}(h_1)$

Int. $\Delta Q = \sin \theta c (\Delta x);$ Int. $\Delta Q = \sin \theta c (\Delta x);$

~~get d = Math.abs(l1-l2);~~

if $d = \text{Math.abs}(l_1 - l_2)$
ff ($l_1 > l_2$) $d = l_2 - l_1$

{ whole ~~for~~

? $h_1 = h_1.\text{next};$

while ($b_1.b = \text{null}$ && $b_2.b = \text{null}$)

$$\{ \quad g_f(b, b_1 = -b_2)$$

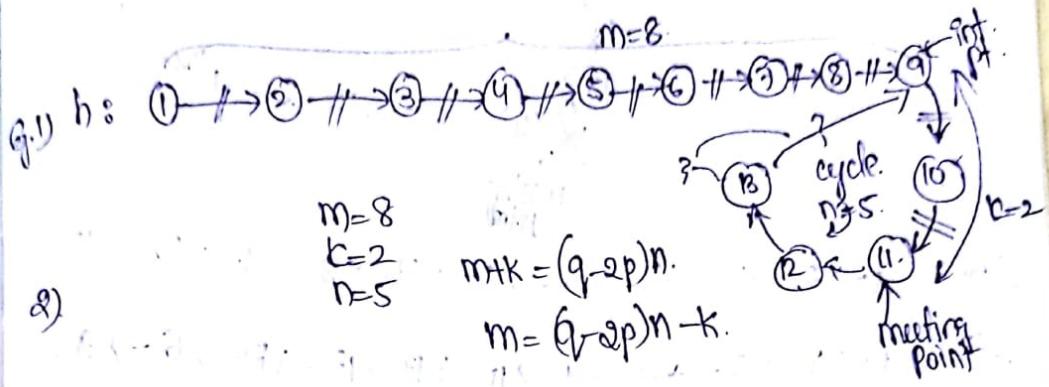
return hi;

~~else~~
{} h₁ = h₁.next;

3 $h_2 = h_2 \cdot \text{next};$

like { }

- (a) 1) Detect Cycle \rightarrow search in the left side hash set
 if found in prev
 \Rightarrow cycle
 else insert into hash set.
 2) Remove Cycle \rightarrow make this last.next link as null.



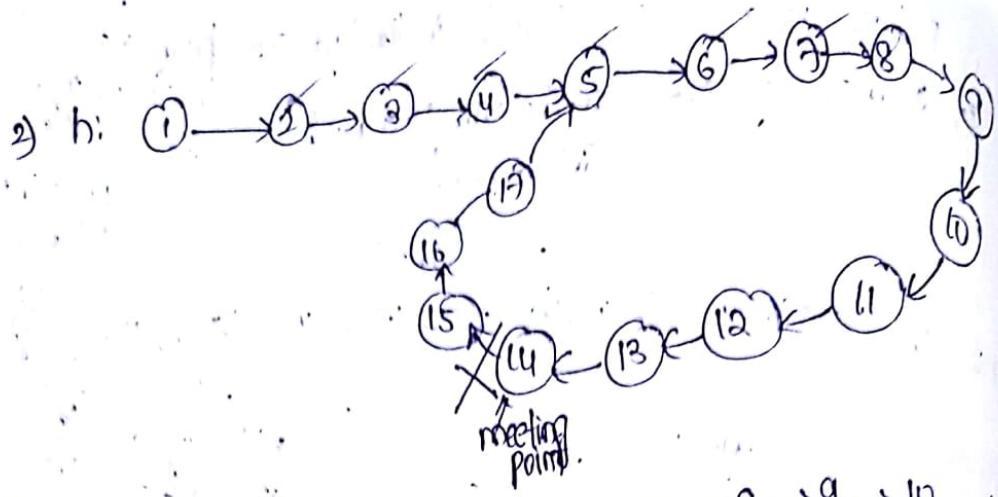
①. Hash Set: $O(N), O(N)$.

```
hashSet<Node> s;
while(h != null)
{
    if(s.contains(h) == true) // found cycle.
    {
        p.next = null; // remove cycle
        break;
    }
    else
    {
        s.insert(h);
        (p = h).next();
        h = h.next;
    }
}
```

②. Slow & fast Pointers:

Slow: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow 11$
 Fast: $1 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 9 \rightarrow 11 \rightarrow 13 \rightarrow 10 \rightarrow 12 \rightarrow 9 \rightarrow 11$

same cycle



slow: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow 11$

fast: $1 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 9 \rightarrow 11 \rightarrow 13 \rightarrow 15 \rightarrow 17 \rightarrow 6 \rightarrow 8$

~~head~~
slow $\rightarrow 12 \rightarrow 13 \rightarrow 14$ meeting point
fast $\rightarrow 10 \rightarrow 12 \rightarrow 14$ meeting point

void cycleDetection(Node h)

```
{
    slow = h;
    fast = h;
    while (fast != null && fast.next != null)
```

```
    {
        slow = slow.next;
```

```
        fast = fast.next.next;
```

```
        if (slow == fast)
```

```
            return slow; // meeting point
```

```
}
```

```
}
```

e.g. (f). meeting point (11). Break 11.next $b_1 \downarrow b_1 \downarrow h_1$
 $b_1 = 11 \rightarrow 12 \rightarrow 13 \rightarrow 14 \rightarrow 15 \rightarrow 16 \rightarrow 17 \rightarrow 8 \rightarrow 7 \rightarrow 6 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1$ same.

$b_2 = 5 \rightarrow 12 \rightarrow 13 \rightarrow 14 \rightarrow 10 \rightarrow 11$

$11 - 5 = 6$

use previous method of removing cycle & make p.next
 $O(N), O(N^2)$

$O(N)$, $O(N)$ space
either continue into same ll. in cycle detection
or else continue into same ll. in cycle detection

void cycleDetectionAndRemoval(Node h)

```

→ {
    slow = h;      h1 = h;
    fast = h;
    while (fast != null & fast.next != null) {
        slow = slow.next;
        fast = fast.next.next;
        if (slow == fast)
            h2 = slow;
    }
}
  
```

// $h_1 = h$, $h_2 = \text{slow}$; meeting point

while ($h_1 \neq h_2$) {
 h1 = h1.next;
 p = h2;
 h2 = h2.next;
}

p.next = null;

}

frequency of visit

Removing cycle
without extra
space
 $O(N), O(1)$

$$\begin{aligned} \text{slow} &= m + k + pn & s &= m + k + pn \\ \text{fast} &= m + k + qn & f &= m + k + qn \\ 2s &= f \Rightarrow f & & \end{aligned}$$

$$m + k + pn = 2(m + k + pn)$$

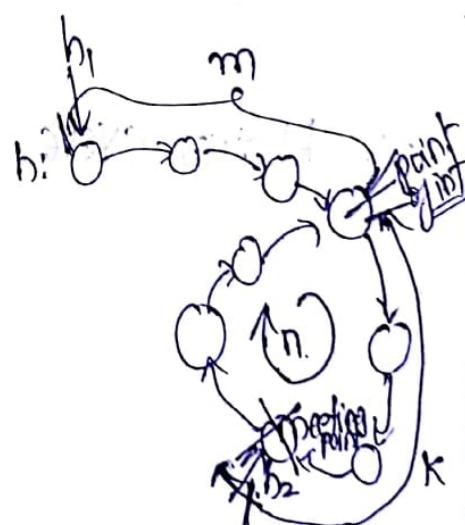
$$m + k + qn = 2m + 2k + 2pn$$

$$2pn + qn + m + k = 0$$

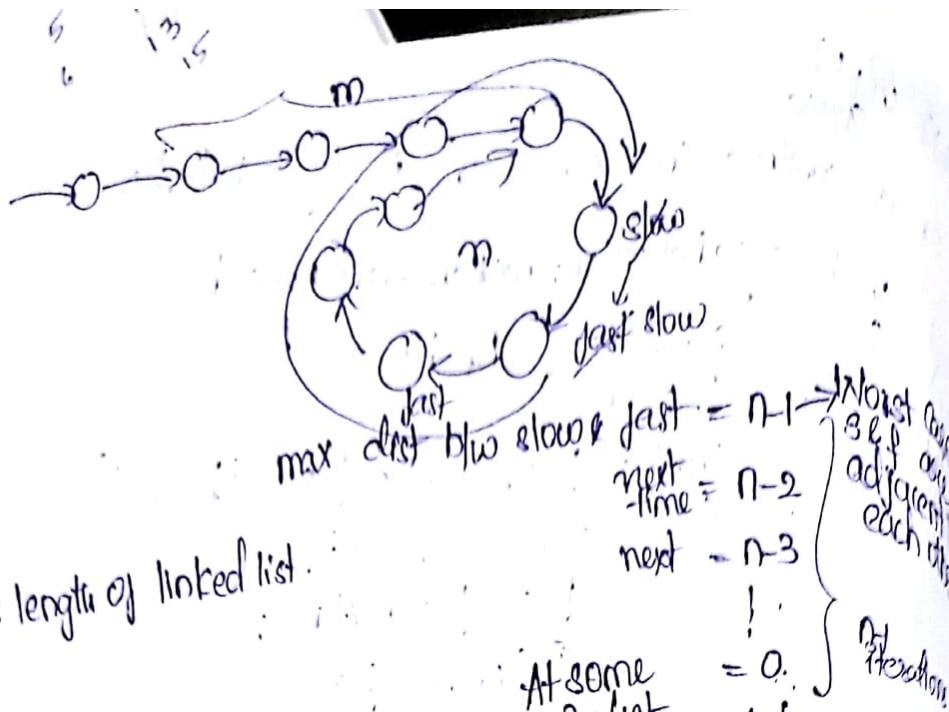
$$\boxed{m + k = (q - 2p)n}$$

$$m = (q - 2p)n - k$$

fast moves twice the no. of nodes as slow moves



n : loop size



slow will def. move m .
 Worst case: $(n-1)$ iterations slow

Compl: $(m+n-1)$

$O(N+N)$, $O(1)$ space.

\downarrow
 \downarrow
 meeting point
 of slow
 & fast

N iterations

$\text{meeting point} = N$
 $\text{worst cycle length}$
 $\text{is linked list length } \geq N$

* FLOYD CYCLE DETECTION ALGORITHM *

Initial step

Step 1: Initialize

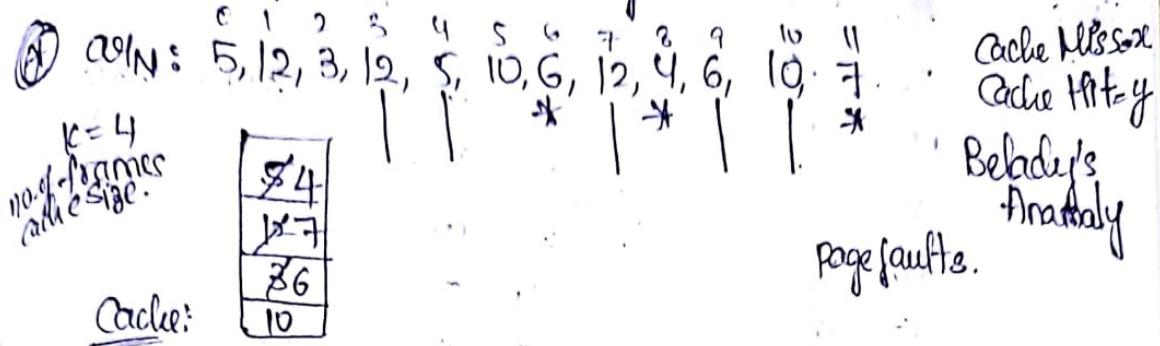
Step 2: Initialize

Step 3: Initialize

Step 4: Initialize

OS → Page Replacement Techniques: FIFO, LIFO, MRU, LRU, LFU.

Tagging → Virtual Memory.



Implementation of LRU Cache :: Least Recently Used Cache.

Hash Set. (4×4)

12
5
3
X
Don't use hash set.
as it doesn't store
index & values.

Hash map

ogn
PN, idx

K
5
12
3

10
5

To optimize
maintain
2 maps

If $\text{a}[\text{i}]$ is Not present \rightarrow insert $\text{a}[\text{i}]$ in Hashmap & its index in HM.

If its all present \rightarrow then update its index in HM.

If its not present, and cache K is full,
then iterate through HM & find the min. index,
remove that $\text{a}[\text{index}]$, ele and index from HM,
and insert the new got ele. $\text{a}[\text{i}]$ in HM with idn.

①. Soln1.

a) Present:

②. Soln2.

lock

2 HMs.

① PN idx

② idx sorted

b) Not Present Full : K

lock

Not full: 1.

lock

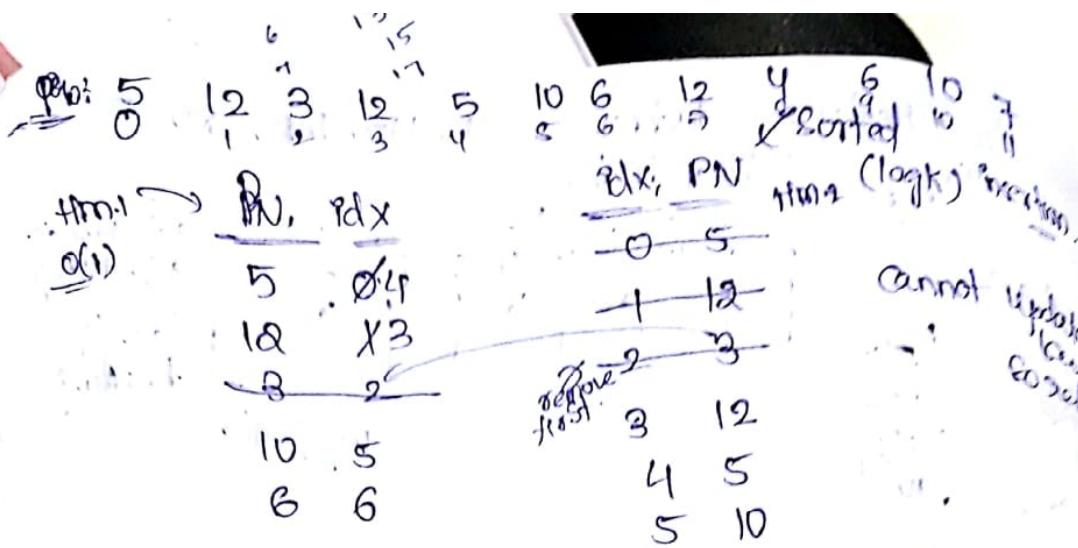
for Nele:- $N \times K$

Time $O(N \times K)$, Space $O(K)$

3. N. $\log_2 K$ e. for Nele.

Time $O(N \log_2 K)$

Space $O(2K) \Rightarrow O(K)$



②. Soln: $\Rightarrow O(N \log_2 K)$, $O(2K) = O(K)$ space
 time: $O(N \cdot K)$

① $O(N \cdot K)$, $O(K)$

② $O(N \cdot \log_2 K)$, $O(K)$

③ Use Linked List:

Dummy



Till size becomes k, search & insert.

In search, if not found → insert at last

In search, if found; remove from there
 and insert at last.

Once size becomes k, remove first, search, insert.

Optimized:

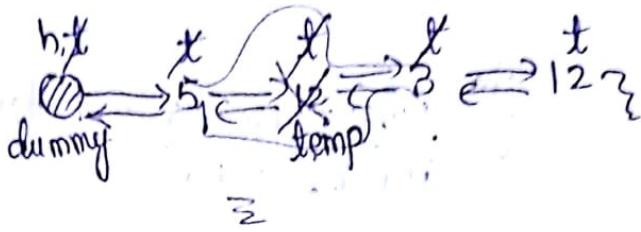
④ Maintain Hashmaps with either approaches:

Soln:

- a) store address of previous node
- b) Copy & delete next node.
- c) Doubly Linked List.

④ Hashmap & Doubly Linked List

key node
 {5: ⑤}
 12: ⑫
 3: ③



1, 1, 1.
 $\Rightarrow O(N), O(K)$. Key: Data
 Value: Node.

①.	②	③	④.
resent.	1	\log_k	k
not full	k	\log_k	k
not full	1	\log_k	k
$O(NK)$ $O(K)$ Hashmap	$O(N\log_k)$ $O(k)$ Sorted Hashmap.	$O(NK)$ $O(k)$ Linked List.	$O(N)$ — Time $O(K)$ — Space Hashmap ④ Doubly Linked List.

④ Hashmap, Double Linked List:

```
void LRU (int p[], int N, int k)
```

```
{
    Hashmap<int, Node> hm;
    Node head = createNode(1);
    Node tail = head;
    for(int i=0; i<N; i++)
    {
        if(!hm.contains(p[i]))
        {
            Node n = createNode(p[i]);
            tail.next = n;
            n.prev = tail;
            tail = n;
            hm.put(p[i], n);
        }
    }
}
```

if (hm.size() == k+1)
 // remove first ele. from linked list.

else
 Node temp = createNode(p[i])
 if (d != temp)

~~if (hm.size() == k+1)~~
~~// remove first ele. from linked list.~~
 ~~Node temp = createNode(p[i])~~
 ~~if (d != temp)~~

~~hm.remove(d);~~
 ~~hm.add(temp);~~

~~d = temp;~~

~~hm.add(d);~~

③ Clone (copy) a linked list:

gfp: S 10 3 -8 11

```

graph LR
    gfp["gfp: thm"] --> 5((5))
    5 --> 10((10))
    10 --> 3((3))
    3 --> -8((-8))
    -8 --> 11((11))
  
```

clone
new linked list
looks like: same
pref. given ll.

Node cloneList(Node head)

Node temp = newhead;
Node dummyhead = createNode(-1);

while(head != null),

```
newhead.next = createNode(head,data);
```

`head = head.next;`

`newhead = newhead.next;`

return ~~nexthead~~.next; temp.next;

class Node

④ clone the following: *int data;*

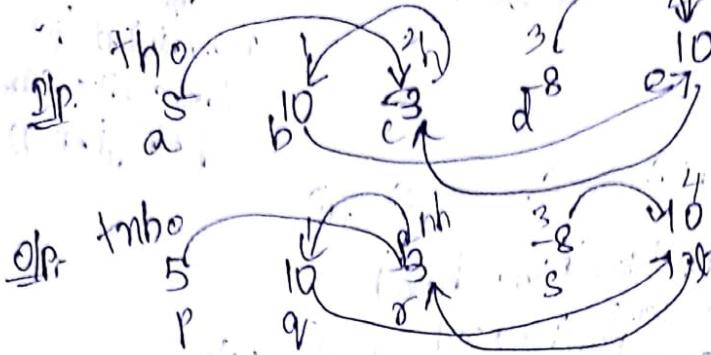
Node next;

Mode random;



~~Op: "5"~~ ~~To Bi~~ ~~X nh.random = h.random; X~~
~~wrong.~~ should find its own linked list random =

// First create Linked list & then search for Random elements in the parent linked list.



I don't
only on data
I rely on
abilities.

Com

Space
periods.

② Co

(2)

Node clone(int N, int pf[]; Node h)

th. starting

tnh. starting

// first find the position of random node in h list

// find the address of node with pos in nh list

for(i=0 to N)

{ int pos = findPosition(th, h.random); // optim.

Node temp = findNode(tnh, pos); // optim.

nh.random = temp; // nh.random = findNode(tnh, pos);

h = h.next;

nh = nh.next;

O(N²), O(N²)

// Optimize:

m1

a, 0

b, 1

c, 2

d, 3

e, 4

get pos.

0, p

1, q

2, r

3, s

4, t

get node

th, n

nh, m

pos, i

random, j

O(N+N)

traversing
linked
list

inserting
new
node

removing
old
node

Comp.
O(N)

Node clone()

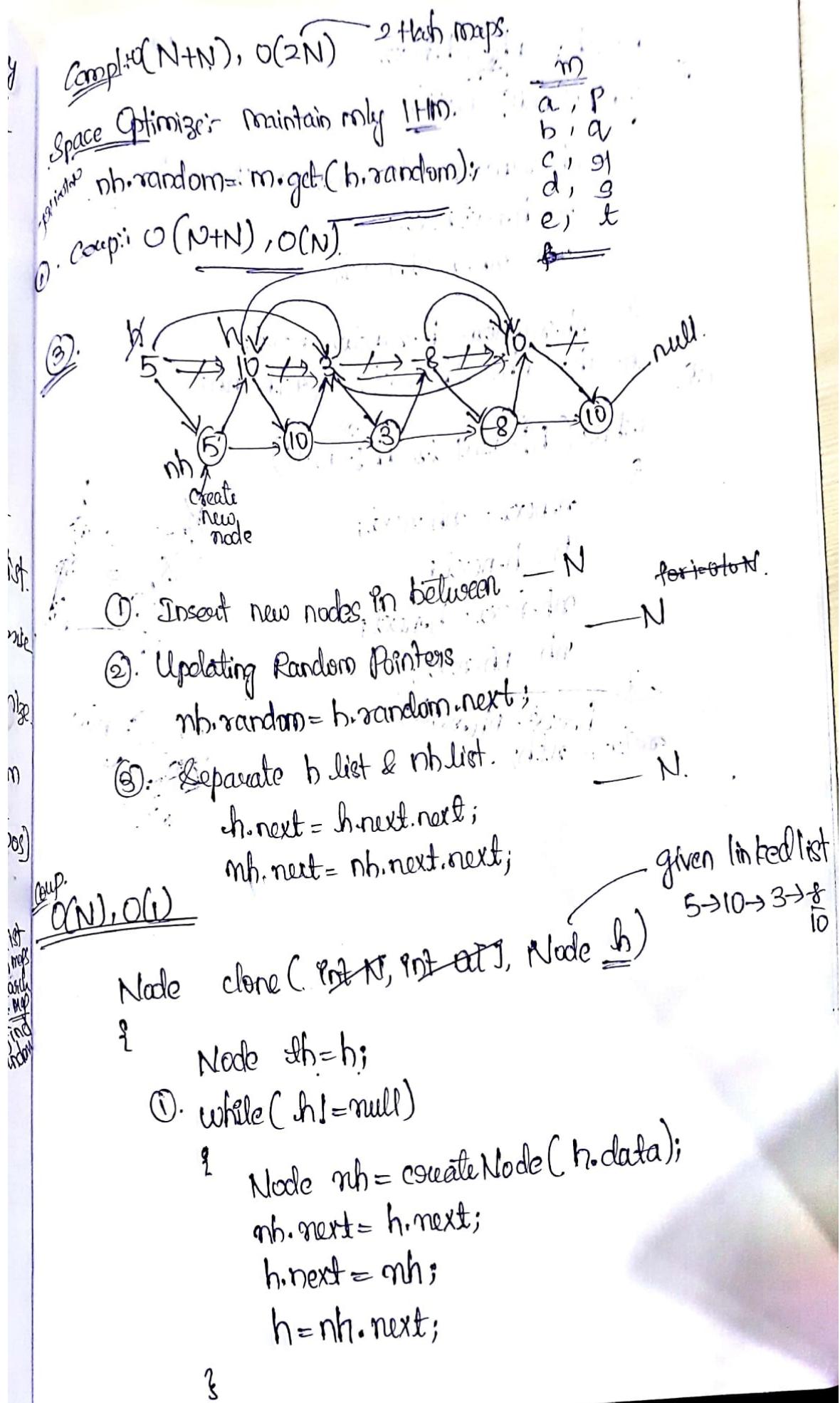
{ for(i=0 to N)

nh.random = m2.get(m1.get(h.random));

h = h.next;

nh = nh.next;

3



①. $h = th; nh = th.next;$

while ($nh.next == null$)

{

$nh.random = h.random.next;$

$h = h.next.next;$

$nh = nh.next.next;$

$nh.random = h.random.next;$

$h = th; nh = th.next; Node nth = nh;$

while ($h.next == null \& nh.next == null$)

{

$h.next = nh.next;$

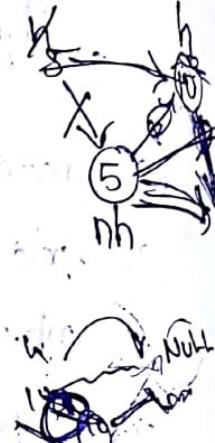
$h = h.next;$

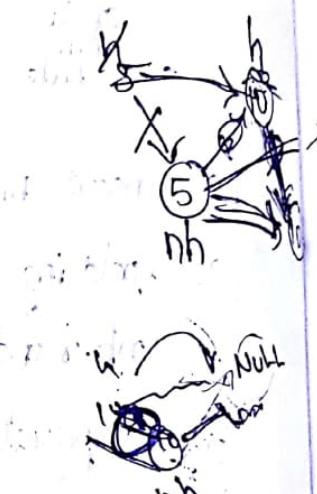
$nh.next = h.next;$

$nh = nh.next;$

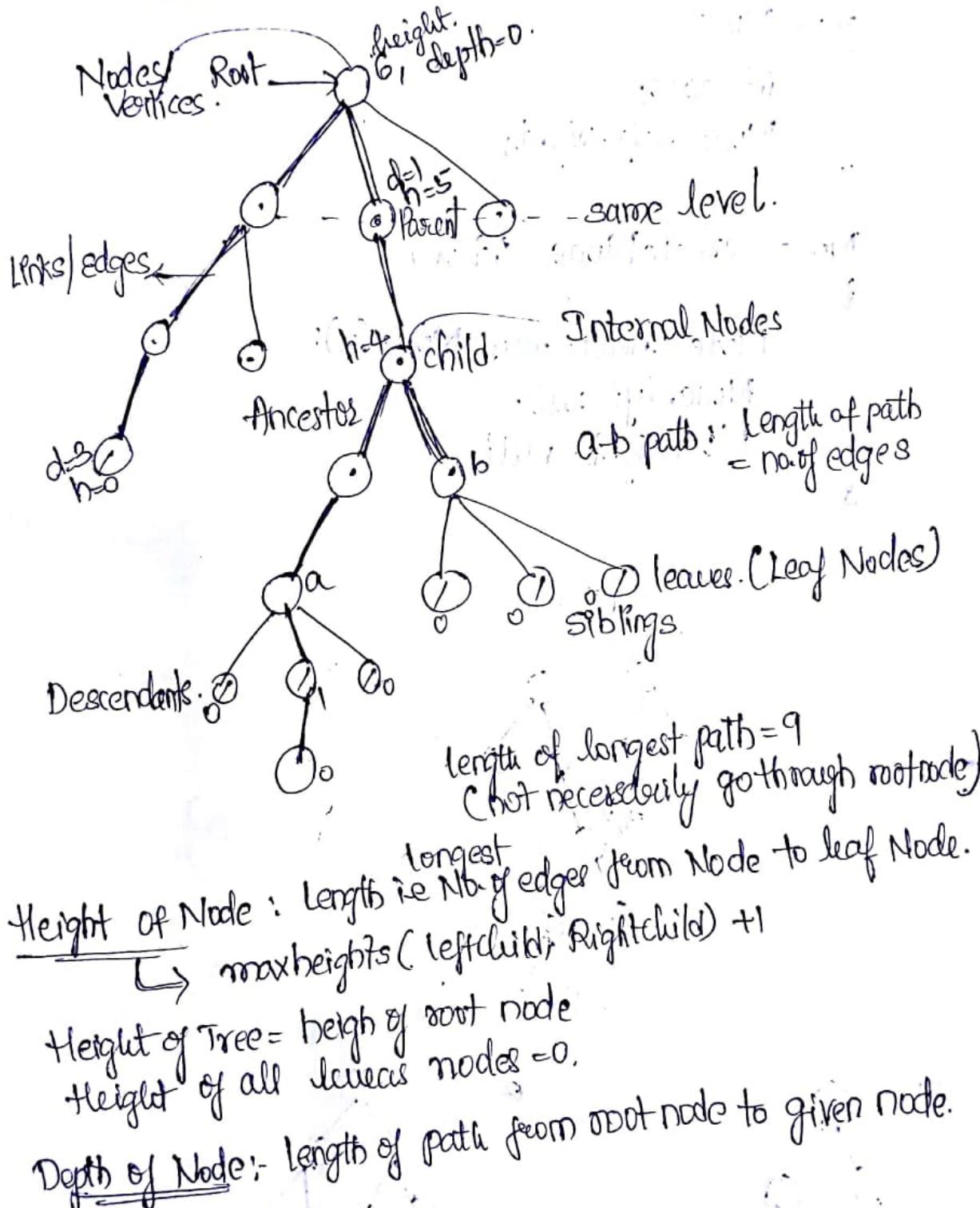
$h.next = nh.next;$

$return nth;$





TREES

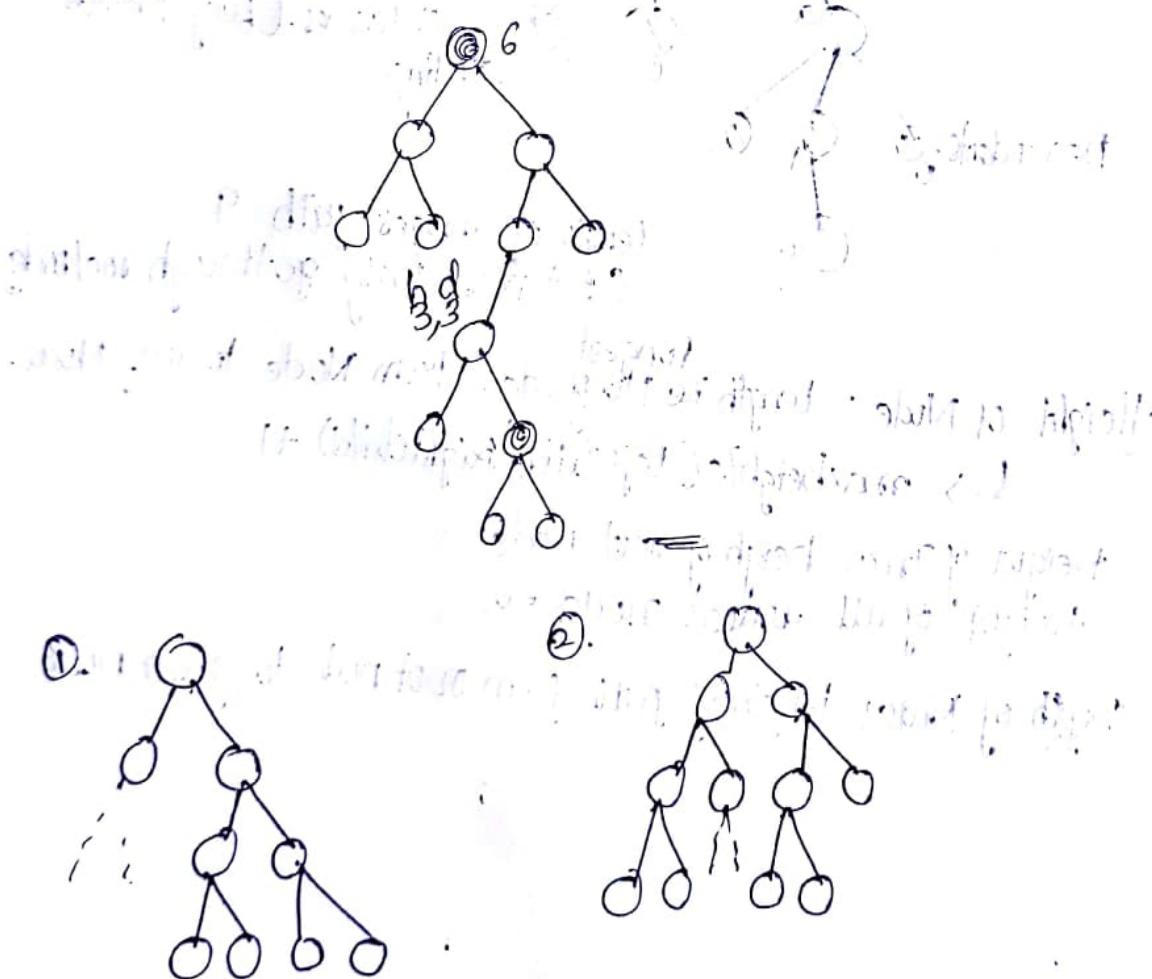


Binary Trees:

```
class Node
{
    int data;
    Node left, right;
}
```

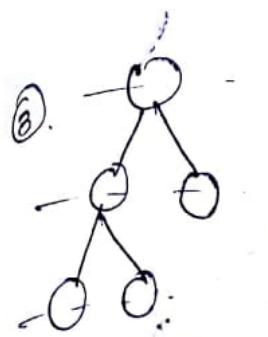
```
Node createNode( int x )
```

```
Node head = new Node(x);  
Node.left = null;  
Node.right = null;
```

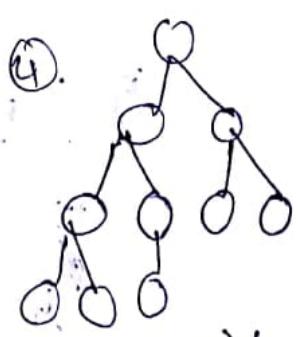


FBT ✓
CBT X

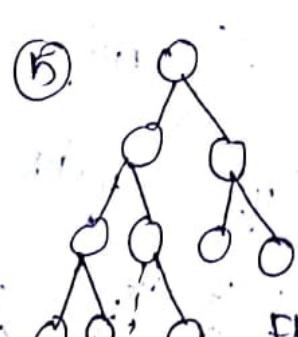
FBT ✓
CBT X



FBT ✓
CBT ✓



FBT X
CBT ✓

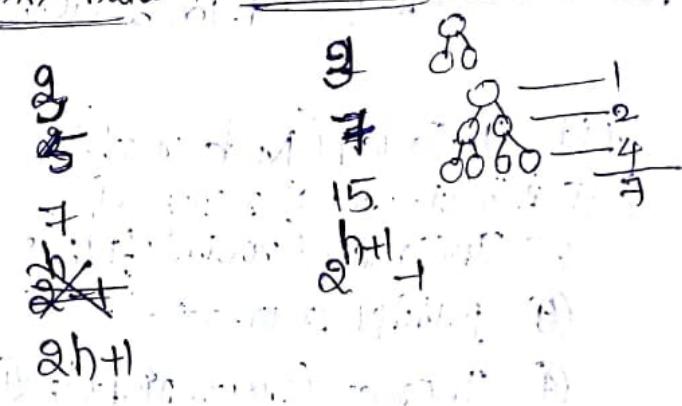
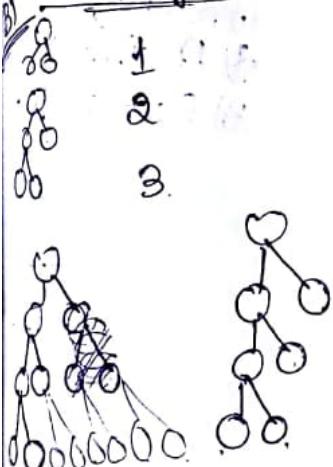


FBT X
CBT X

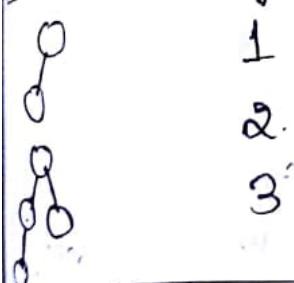
exactly.

- 5) Full Binary Tree: Each node should have 0 or 2 children.
 6) Complete Binary Tree: Fill the complete level, first from left to right and then go to next level.

Given Height = H : , min. nodes, max. nodes req.



BT. Given Height = H

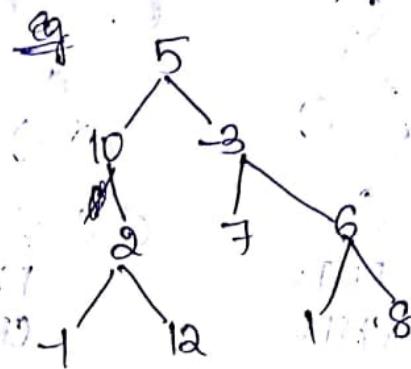
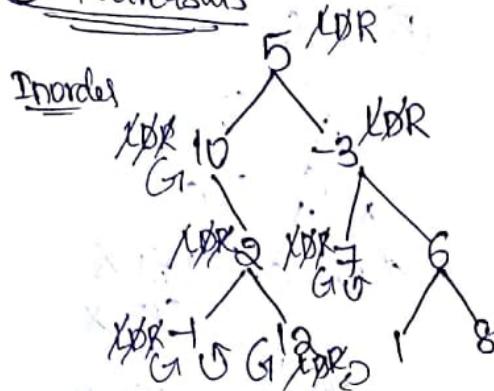


	<u>min</u>	<u>max</u>
1	2	3
2	4	7
3	8	15
	2^H	$2^{H+1} - 1$

(Full) FBT	<u>min</u>	<u>max</u>
	$2H + 1$	$2^{H+1} - 1$
CBT (Complete)	2^H	$2^{H+1} - 1$

Height = H

Traversals



Tree Traversals: i) Pre-order, ii) In-order, iii) Post-order.

i) Pre Order (DLR): 5, 10, -1, 2, 12, -3, 7, 6, 1, 8

ii) In Order (LDR): -1, 5, 10, 2, 12, 7, -3, 1, 6, 8.

iii) Post Order (LRD): -1, 12, 2, 10, 7, 1, 8, 6, -3, 5.

Assumption:
by Main logic
3) Base condition

Void InOrder (Node root)

- 1) If (root == null) return;
- 2) InOrder (root.left);
- 3) cout << root.data;
- 4) InOrder (root.right);

3

Pre Order: ① → ② → ③ → ④

In Order: ① → ② → ③ → ④

Post Order: ① → ② → ④ → ③

① int size(Node root); // no. of nodes = 10.

② int sum (Node root);

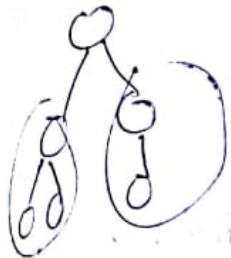
③ int maxValue (Node root);

④ int height (Node root);

⑤ ... fillDepth (- - -)

⑥ void levelOrder (Node root)

eg..
data depth
5, 0
10, 1 -3, 1
class Node {
 int data, depth;
 Node left, right;
};
5
10, -3, 6
2, 7, 1
1, 12, 1, 8



size of tree = 1 + size of left tree + size of right tree;

①. int size(Node root)

```
{
    if (root == null) return 0;
    return 1 + size(root.left) + size(root.right);
```

}

②. int sum(Node root)

```
{
    if (root == null) return 0;
    return root.data + sum(root.left) + sum(root.right);
```

}

③. int maxValue(Node root)

```
{
    if (root == null) return INT-MIN-VALUE;
    return max(maxValue(root.left),
               max(maxValue(root.right), root.data));
```

④. int height(Node root)

```
{
    if (root == null) return 0;
```

Bottom

Up
Approach

return 1 + max(height(root.left), height(root.right)), 3, 2, 1, 0

if return 0
then problem
height

return

3

correct
int height(Node root)

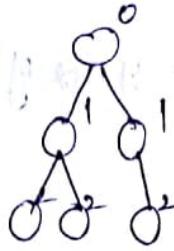
```
{
    if (root == null) return -1;
```

return max(height(root.left), height(root.right)) + 1;

to find the sum

⑤ Depth Filling:

Top-Down
Approach.



Initial root node
depth will be 0.

void fillDepth(Node root, int depth)

{
 if (root == null) return;

 root.depth = d;

 fillDepth(root.left, d+1);

 fillDepth(root.right, d+1);

}

Sols for Most of the Problems:-

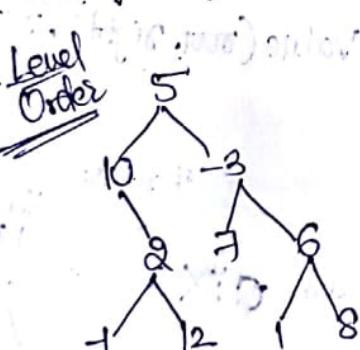
1) Traversal + 2) Approaches

Top Down

Bottom up

Inorder pre post level

⑥ Level Order



10 → 3

2 → 3

1 → 2 → 6

8 → 1 → 8

Use Queue: 5, 10, -3, 2, 3, 1, 2, 6, 8

Ans: 5, 10, -3, 2, 3, 1, 2, 6, 8

pop remove 5, push 5's left & right

For this, use markers in between.

marker

Queue: $\emptyset \rightarrow 10 \rightarrow \emptyset \rightarrow 7 \emptyset \rightarrow 10 \rightarrow \emptyset \rightarrow \emptyset \rightarrow \emptyset \rightarrow \emptyset \rightarrow \emptyset$

Ans: 5 ln

10 → 3 ln

2 7 6 ln

-1 12 1 8 ln

ln ln ln

infinite

when you pop

- put ln in ans.

- push • at last queue.

But push the marker only
when the queue is not empty.

Queue < > q;

q.size()

q.front()

q.insert()

q.pop();

- If queue is empty -

- don't push the marker.

2) Use Queue Size:

Queue: $\emptyset \rightarrow 10 \rightarrow 2 \rightarrow 7 \rightarrow 6$

Queue Size = 1: Iterate 1 time: 5 ln.

Queue Size = 2: Iterate 2 times: 10 → 3 then ln

Queue Size = 3: Iterate 3 times: 2 7 6 ln.

Every time we pop an
ele from queue,
we insert its
children in the
queue.

3) Use depth:

Queue: (5,0) (10,1) (3,1) (2,2) (7,2) (6,2) (-1,0), (12,3)
(1,3), (8,3)

currentDepth = 0 \neq 3

(10,1) → 0. put ln. update
current Depth.

If depth == currentDepth
just pop.

If depth < currentDepth

pop
ln

and update currentDepth

5 ln

10 → 3 ln

2 7 6 ln

-1 12 1 8 ln.

Q) find the sum.

①. Marker:

Queue<Node> q;

q.push(root)

Queue<Node> q;

void levelOrder(Node root)

{

 q.push(root); q.push(null);

 while (q.size() != 0)

{

~~if (q.pop())~~

 if (q.front() != null)

 if (q.pop());

 if (root.left) q.insert(root.left);

 if (root.right) q.insert(root.right);

 if (q.front() == null & q.size() != 0)

 print(in);

 q.push(null);

}



ab

ba

3. Marker

front().

Take in some val.

② Queue Size

Queue < Node > q;

void levelOrder (Node root)

{ q.push (root);

while (q.size () != 0)

{ int n = q.size ();

while (n--)

{ pf (q.pop ());

pf ("In");

if (root.left) q.insert (root.left);

if (root.right) q.insert (root.right);



③ Use Depth

Queue < Pair < Node, int > > q; // (node, currentDepth)

q.push (new Pair (root, 0));

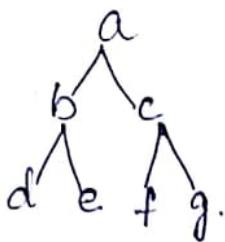
int currDepth = 0;

while (q.size () != 0)

{ pair p = q.front ();

=

② find the sum



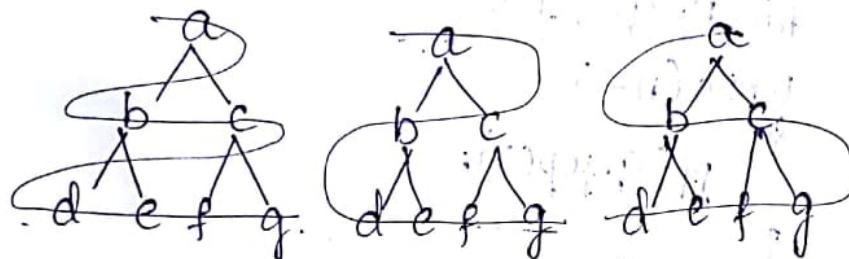
1) level Order

left to right:

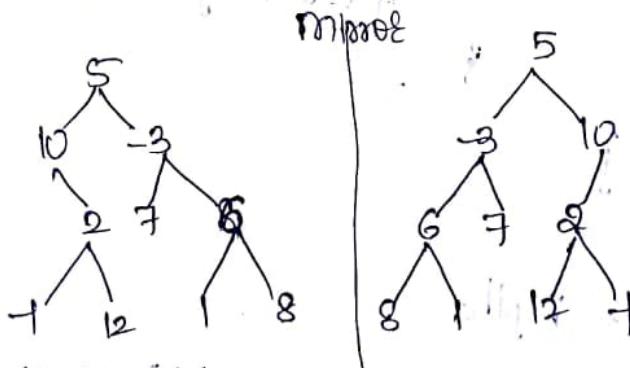
right to left: swap insertion

bottom to top:

top to bottom: swap insertion



④ Mirror:



- No new tree creation.
- Swap. left & right children at every node.

void mirror (Node root)

{ swap (root.left, root.right);

mirror (root.left);

mirror (root.right);

}

DLR

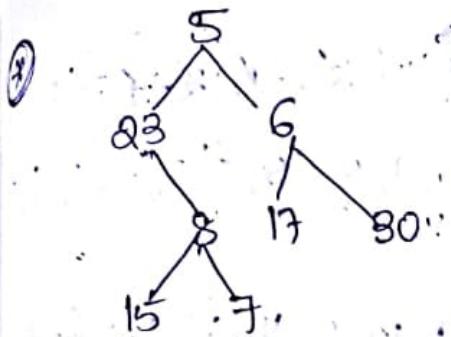
LRD

LDL

RDR

DRL

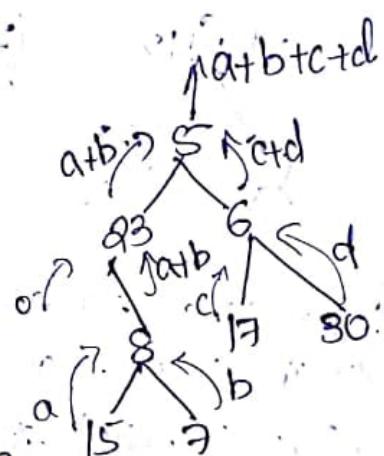
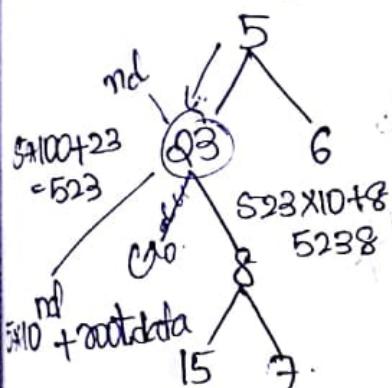
RLD



Sum of all the paths from root to leaves.

$$\text{Ans: } 523815 + 52387 + 5617 + 5630$$

$n = \text{no. of digits}$



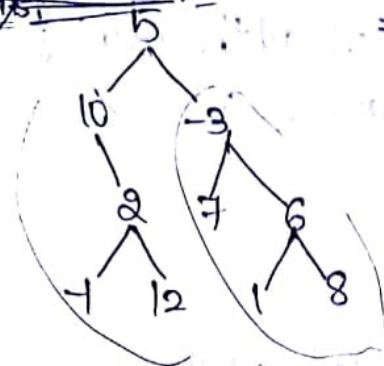
```

int solve(Node *root, int v)
{
    if (root == null) return 0;
    v = v * pow(10, digits(root.data)) + root.data;
    if (!isLeaf(root))
        return v;
    return solve(root.left, v) + solve(root.right, v);
}

```

④

Find Post Order.



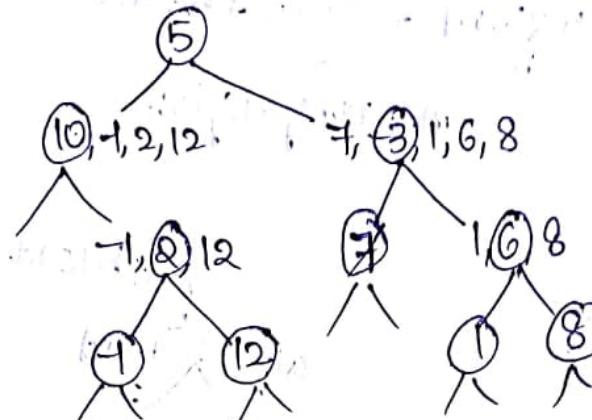
Ques:

Pre_N: (5) 1, 2, 3, 4, 5, 6, 7, 8

In_N: 10, -1, 2, 12, 5, 7, -3, 1, 6, 8

Post_N: ?

-1, 12, 2, 10, 7, 1, 8, 6, -3, 5



Start with pre-order ele. as root.

- find root in In-order array, left to root will belong to left subtree & right side elements to right subtree
- go on.

first find P in in[0-q]

low high

P	low	high	idx.	left call	right call
0	0	q	4	low, pdx+1, high	pdx+1, high

Search pre[0] in in[0-q] found 5 at idx 4.
make pre[0] be in[pdx] as
go left low, idx-1

p. is static var.

taken as

return null Node.

don't inc p here

5

pre[1] search in in[0-3]
get pdx = 0.

↓ form Tree

then Post Order

$\leftarrow (\text{root}, \text{right});$

① Create Tree & then Post Order

Node formTreeFromPreIn(int in[], int pre[], int low, int high)
static p=0;
if (low > high) return null;

int pdx = search(pre[p], in, low, high);

Node root = createNode(pre[p]);
p++;

root.left = formTreeFromPreIn(in, pre, low, pdx-1);

root.right = formTreeFromPreIn(in, pre, pdx+1, high);
return root;

}

void PostOrder(Node root); // gives postorder from formed tree

int pdx;

② Post Order Without creating tree:

void postOrder(int in[], int pre[], int low, int high)

static p=0;

if (low > high) return null;

int pdx = search(pre[p], in, low, high);

p++;

postOrder(in, pre, low, pdx-1);

postOrder(in, pre, pdx+1, high);

printf("%d ", in[pdx]);

}

II belong to
right subtree

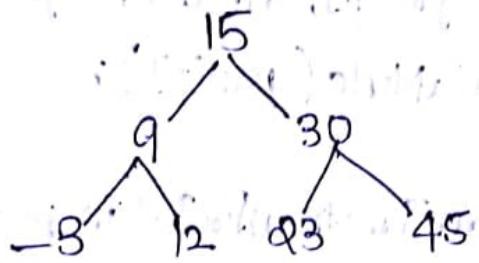
inc p
right call
pdx+1, high

I found 5 at idx 4
& in[pdx] as
go left inc p
low, idx-1

in[0-3]

= 0.

All ele. in left Subtree < Data < All ele. in Right Subtree



Insert 21. \Rightarrow

15.right = insert(15.right, 21)

30.left = insert(30.left, 21) \leftarrow

23.left = insert(23.left, 21)

create Node 21

as 23.left is null.

Insert:

① Node insert (Node root, int k)

{

if (root == null) return createNode(k);

if (k < root.data):

root.left = insert(root.left, k) \leftarrow

else

root.right = insert(root.right, k);

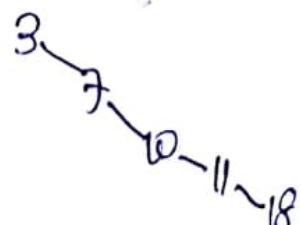
return root;

3.

Best Case $H = \log_2 N$

Comp: $O(H)$ \swarrow Worst Case $H = N$. Right/Left Skewed.

Skewed.



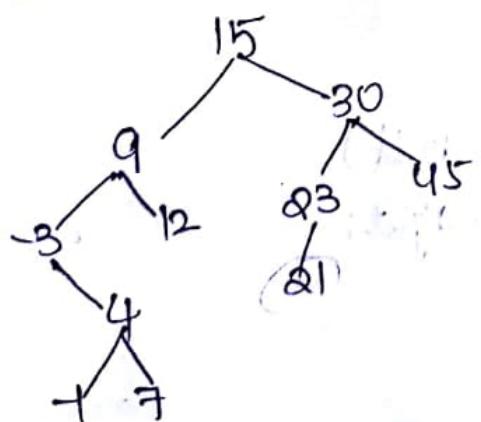
② Search

```
bool search(Node *root, int k)
{
    if (root == null) return false;
    if (k < root.data)
        return search(root.left, k);
    else if (k > root.data)
        return search(root.right, k);
    else // k == root.data
        return true;
}
```

may go to BST

```
Read T // No. of Test Cases
for (int i = 0; i < T; i++)
    Read N // No. of nodes;
    Node *root = null;
    loop(N)
        Read x // Data
        root = insert(root, x);
```

③ Delete:



Delete:

- case ①. Leaf Node : - 1 case ②. Single Child : - 3
- case ③. Both Child.
 - replace with
max in left subtree
 - or min in right subtree

1) find the sum

```
Node delete(Node root, int k)
{
    if (root == null) return root;
    if (k < root.data)
        root.left = delete(root.left, k);
    else if (k > root.data)
        root.right = delete(root.right, k);
}
```

Case 1: if (isLeaf(root)) // root.left == null & root.right == null
return null;

Case 2: if (root.left == null):
 return root.right;
if (root.right == null):
 return root.left;

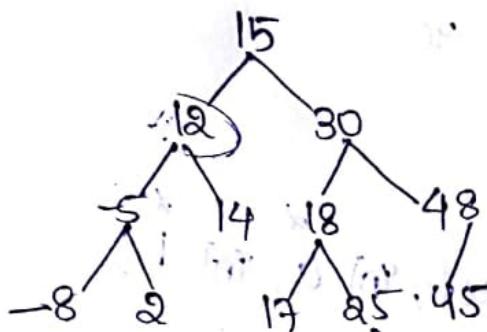
Case 3: if (root.left != null & root.right != null).
 root.data = findMax(root.left);
 root.left = delete(root.left, root.data);
}
return root.

int findMax(Node root)

```
{ int max;
    while (root.right != null)
        root = root.right;
}
```

return root.data;

Given Priority Tree, check if it is a BST.



① Do Inorder Traversal:

If got ascending order \Rightarrow BST
else \Rightarrow Not a BST.

— requires array extra space.

— or else just use a variable & check if var \rightarrow is $<$ current value.

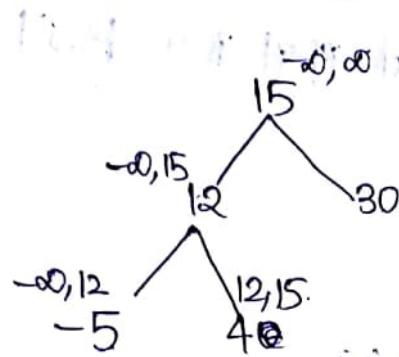
if ($\text{groot}.\text{data} > p$).
 $p = \text{groot}.\text{data}$.

else
f

do it.

```
bool checkBST(Node *root) {  
    static p = root.data;  
    if (checkBST(root.left));  
        if (root.data  $\leq$  p)  
            p = root.data;  
    static int P = INT_MIN;  
    if (checkBST(root.left) == F)  
        return P;  
    if (root.data > p)  
        p = root.data;  
    else  
        return P;  
    if (checkBST(root.right) == F) return F;  
    return T;
```

β) find the sum



(root.left) + size

bool isBST(Node root, int a, int b)

{

if (root == null) return true;

return root.data > a && root.data < b

&& isBST(root.left, a, root.data);
&& isBST(root.right, root.data, b);

}

if (isBST(root)) {
 print("true");
}

else

(new Node).left = left;

(new Node).right = right;

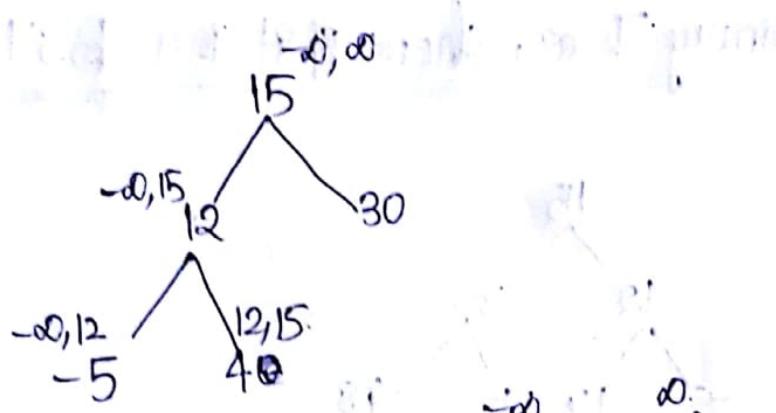
(new Node).data = data;

return (new Node).data;

else if (isBST(left)) {
 print("true");
}

else if (isBST(right)) {
 print("true");
}

else {
 print("false");
}



bool isBST(Node root, int a, int b)

{ if (root == null) return true;

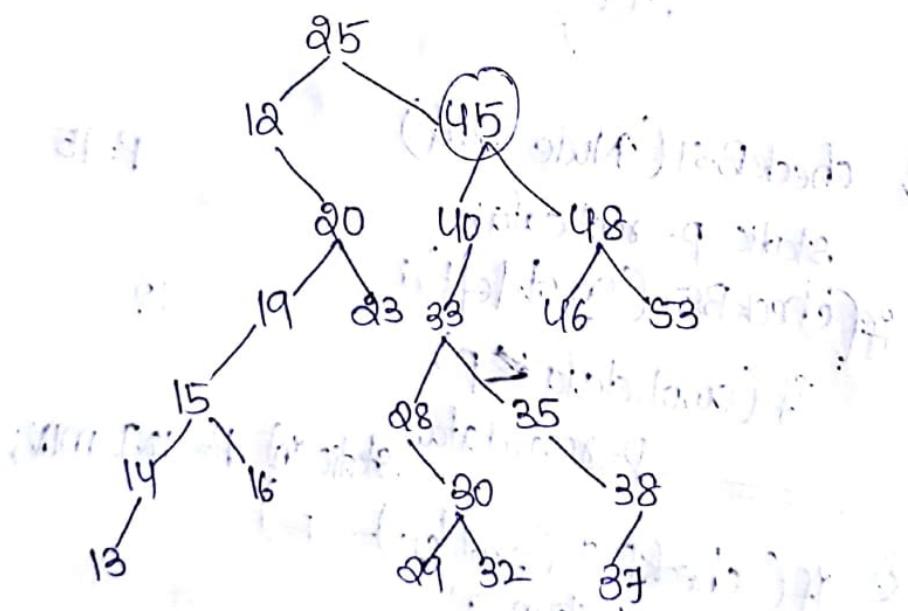
if (root.data > a && root.data < b)

&& isBST(root.left, a, root.data);

&& isBST(root.right, root.data, b);

Q4.

④ Turn the given BST such that only the nodes in range [15, 35] should remain.



```

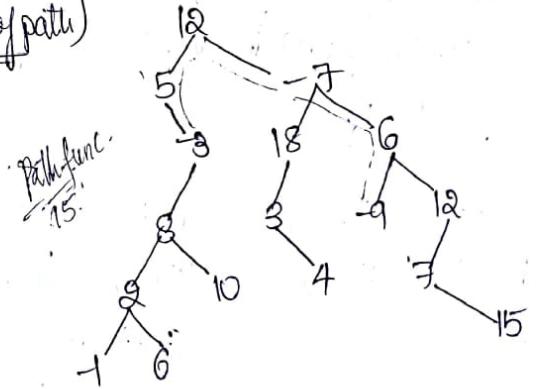
Node Trim( Node root, int a, int b)
{
    if (root == null)
        return null;
    if (root.data < a)
        return Trim (root.right, a, b);
    if (root.data > b)
        return Trim (root.left, a, b);
    root.left = Trim (root.left, a, b);
    root.right = Trim (root.right, a, b);
    return root;
}

```

- ④ for duplicate elements to be inserted in BST, then maintain the count of elements in the Node structure itself.

- ⑤ Find the length of the path in the given Binary Tree (BT)

<u>Node</u>	<u>Node</u>	<u>ans. (length of path)</u>
<u>a</u>	<u>b</u>	
-3	9	5
1	10	3
15	7	4



Eg. 15. → 7

Root to 15 path: 12, 7, 6, 12, 7, 15

Root to 7 path: 12, 7

$$\text{size}(1) + \text{size}(2) \\ = 4 + 0 \\ = 4 \text{ ans.}$$

Paths().

→ form ArrayList < Node > and form the path.

while searching down go on searching in preordred,

if ele. found add to the list, else no.

For next ele. in path if size is non zero add to

```

void path(Node root, Node a, ArrayList<Node> alist)
{
    if (root == a)
    {
        alist.add(a);
        return;
    }
    path(root.left, a, alist);
    if (alist.size() != 0)
    {
        alist.add(root);
        return;
    }
    path(root.right, a, alist);
    if (alist.size() != 0)
    {
        alist.add(root);
        return;
    }
}

// find the paths of node a, and node b.
Get the paths in reverse ways.

```

e.g. a, b:

alist: $\rightarrow 1, 2, 3, 4, 5$ size = 5
 blist: $\rightarrow 10, 8, 7, 6, 5$ size = 5

Start comparing from last same, remove diff. stop &
 $ans = size(alist) + size(blist)$

int length(Node root, Node a, Node b)

```

{
    ArrayList<Node> alist, blist;
    path(root, a, alist);  $\xrightarrow{N}$ 
    path(root, b, blist);  $\xrightarrow{N'}$ 
    return size(alist) + size(blist);
}

```

Given a Binary Tree, find the maximum sum across any path (a,b) in the Binary Tree.

Q. Brute Force

Take all the pairs of Nodes $\Rightarrow N^2$.
Find the sum.

Find the ~~number~~ of Nodes $\Rightarrow N^2$.

and find the sum of ele. in nodes.

e.g. $\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{matrix}$ sum of ele. In nodes.

$$\begin{array}{r} 4 \ 2 \ 8 \ -3 \ 8 \ 1 \\ 10 \ 8 \ -3 \ 5 \ 1 \\ \hline \end{array}$$

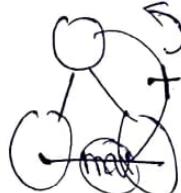
$$\text{Sum} = (1+2)+(8+10) = ()$$

Update max of sums got. \rightarrow last common removed.

Comp: $O(N^3)$, $O(N)$.

②. for all nodes,

Consider each node as root



int solveMaxSumPath(Node root)

for all ele. of tree. N .
~~max (f(root, data + f(root, left)) + f(root, right))~~
~~solveMaxSumPath(root, left); solveMaxSumPath(root, right)~~

int f(Node root) — O(N)

? if (root == null) return 0;

$\text{height}(\text{root}) = \max(\text{root.data} + \max(\text{f}(\text{root.left}), \text{f}(\text{root.right})), 0)$;

$O(N^2)$, $O(1)$.

For All pairs in N -nodes call solve.

```
int solve( Node root )
```

{ Time }

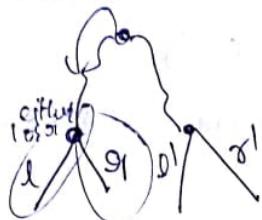
```
return max( root.data + f(root.left), root.data + f(root.right) );
```

```
solve( root.left ), solve( root.right ) );
```

3.

Complex.: $\rightarrow O(N^2), O(1)$.

③.



```
int ans = -∞;
```

```
int solve( Node root )
```

{

```
if( root == null )
```

```
return 0;
```

```
l = solve( root.left );
```

```
r = solve( root.right );
```

```
int ret = max( root.data, root.data + l, root.data + r );
```

```
ans = max( ans, ret, root.data + l + r );
```

```
return ret;
```

}

Complex.: $O(N), O(1)$

Instead of computing value again & again.

Given
Find &
source *

e.g.

Count :

①. Bottom

Find

H

Complex. :-

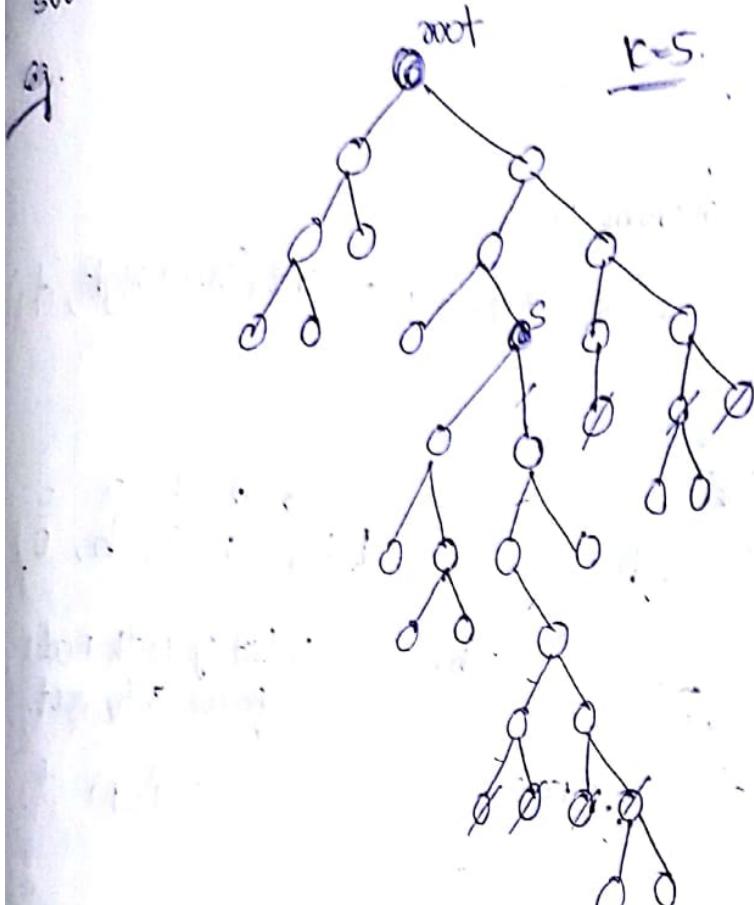
②.

Give

dist

e.g.

Given root node, source node, K-distance.
 find the number of nodes which are at K-distance from source node.



Count no. of ~~all~~ nodes with distance k from source node.

① Brute Force:

Find path of all the other nodes from source.

If length of path == k, count++

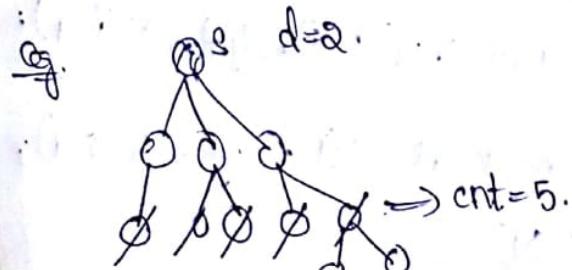
Complexity: $O(N^2)$, $O(N)$.

Time Space.

②

Given Node, find the ^{cnt} of ~~all~~ number no. of nodes at distance d in its subtree.

Ex. s $d=2$.



1) Use level order. = get to know the no. of nodes at level d.

2) $\text{int cnt(Node sroot, int d)}$

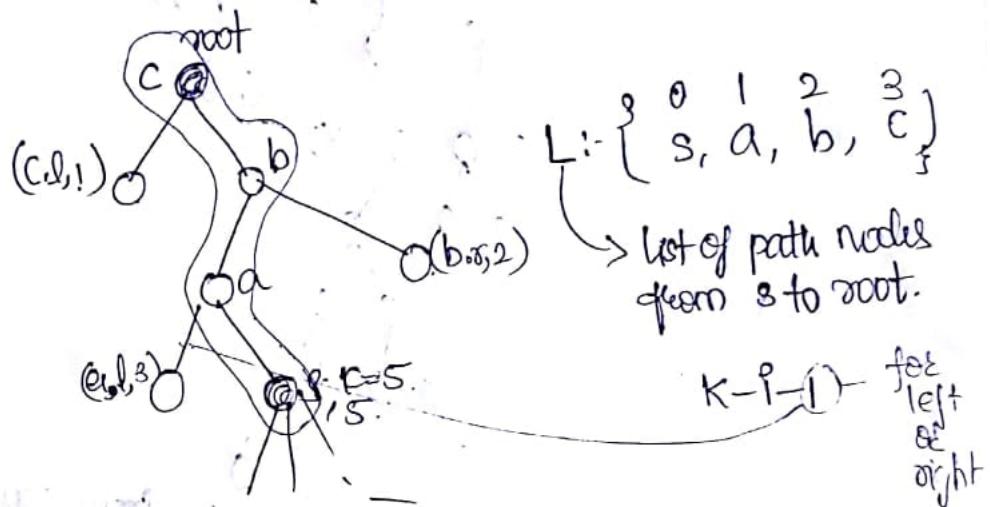
{
 if (sroot == null)
 return 0;

 if (d == 0)
 return 1;

 return cnt(sroot.left, d-1) + cnt(sroot.right, d-1);

}

Each time
current counts
only a part of
tree, not
all N nodes
all the time.



$\boxed{\text{cnt}(s, 5) + \text{cnt}(a, 1, 3) + \text{cnt}(b, 1, 2) + \text{cnt}(c, 1, 1)}$

L \Rightarrow path list b/w s and root.

L: {
 s, a, b, c
} call path(root, s), source node

$\text{int no. of Nodes at Distance k (Node root, Node s, int k)}$

{
 int ans = cnt(s, k); List L; // path from s to root N.

 for (i=1; i <= L.size(); i++)

 {
 if (L(i).left == L(i-1))

 ans = ans + cnt(L(i).right, k-i-1);

 else

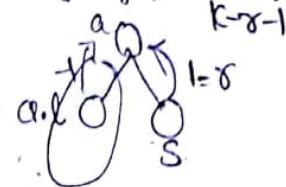
 ans = ans + cnt(L(i).left, k-i-1);

3

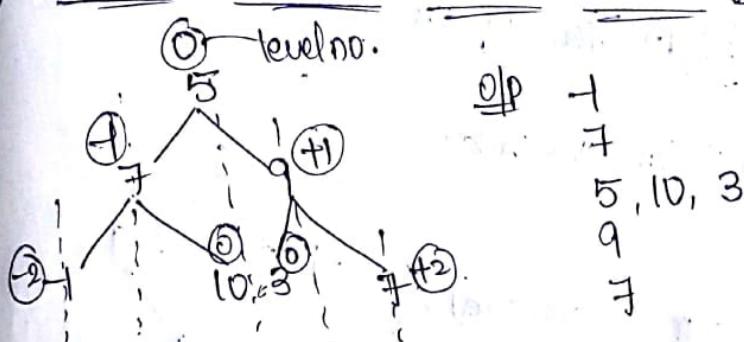
3

Complexity: O(N), O(N)

Instead of storing nodes of path in list, while getting path in Postorder itself check if the node is found or not if found return 1, and then call $\text{so_cnt}(\text{o.left}, \text{k}-1)$.



Point the Nodes in the vertical order fashion:-



Map flash map with <level no., <list of Nodes in that level>>

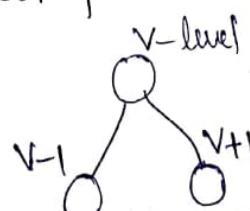
e.g. <2, {1, 3}> or. TreeMap to sort the level no.

<1, {7}>

{0, {5, 10, 3}}

<1, {9}>

<2, {2}>



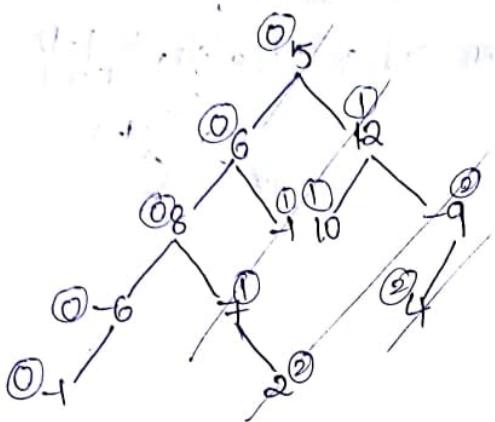
TreeMap<Integer, Node> map

map.get(v).add(groot);

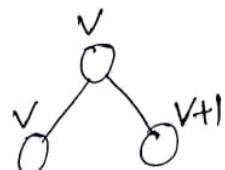
f(groot.left, v-1)

f(groot.right, v+1)

④ Point all the tree nodes in Diagonal Order fashion.



5, 6, 8, -6, 1
12, 1, 10, 9
-9, 1, 2
4.



map.get(v).add(root)

f(root.left, v)

f(root.right, v+1).

⑤ Left-View & Right-View of the Tree:

Left-View. (First ele. in the level order).

5, 6, 8, -6, 1, -7, 1

Right-View. (Last ele. in the level order)

5, 12, -9, 4, 12, 13, 1

Top-View:

6, 8, 12, 7, 3, 2

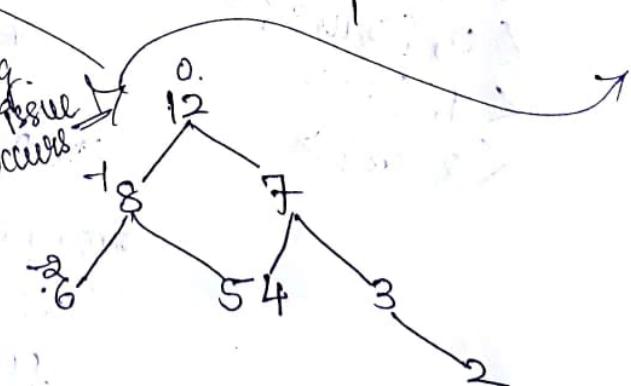
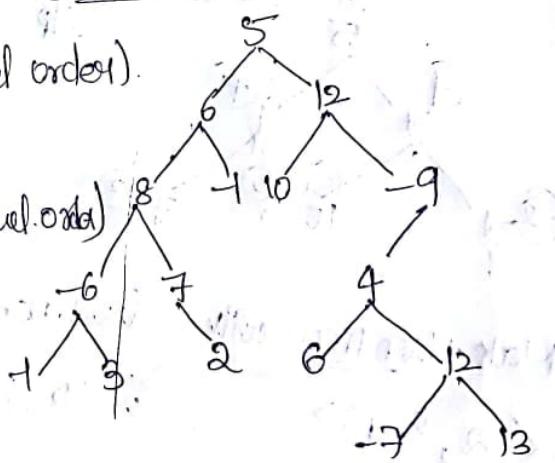
If taken vertical map ordering.

0 → 12, 5, 4, then issue occurs

1 → 8

-2 → 6, 1

(12, 5, 4, 8, 6, 1)

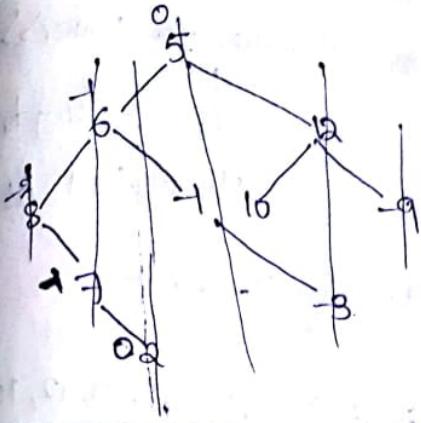


PostOrder

, -6, 1

, 7

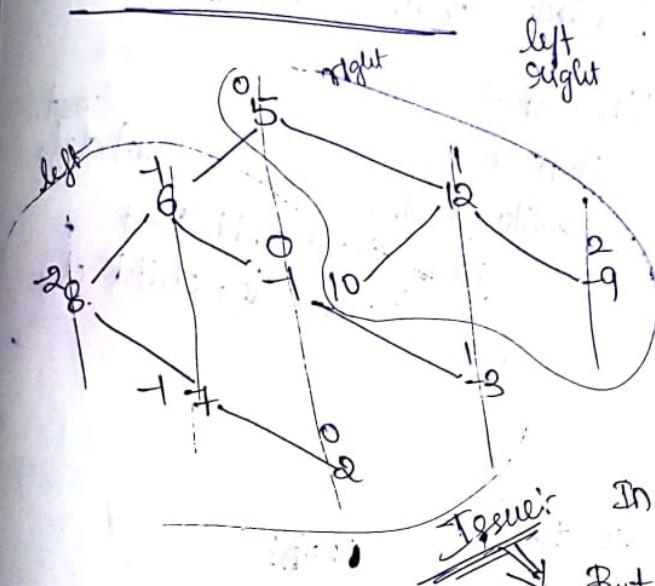
2



{ 0 : [5]

-1 : [6]

-2 :



0 : 5, 2

-1 : 6, 7

-2 : 8

-3 : -3

2 :

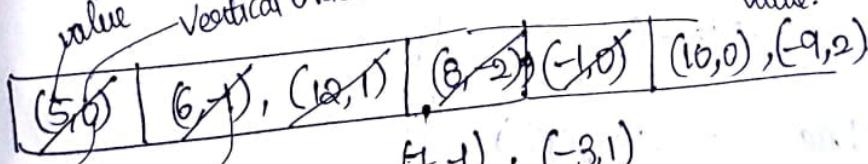
In level 1: 12 should come first
then -3.

But as we are traverse
in postorder fashion in
vertical ordering, then -3
comes first before 12 as
-3 is in left subtree & 12 in right

So,

- I) Maintain level order queue,
along with vertical order no.: Map < Integer, Integer >
value vertical order

queue.



(7, -1), (-3, 1)

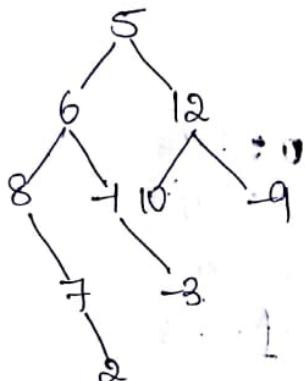
0 : 5
-1 : 6, 11
1 : 12

-2 : 8
<int, int>

Push in hash map
only if it is first
not there in map.

Q) OR Hashmap < Vertical Order, < Value, level Order >
 sort according to level order.
 Point first Node.

Point Pre-Order Using Iterative Fashion



Pre Order: 5, 6, 8, 7, 2, -1, -3, 12, 10, -9

Use stack.

push root.
 while popping root
 push right child first
 & then left child. cli.

```

class PreOrder
{
    stack<integer> stack;
    PreOrder (Node root)
    {
        if (root != null)
            stack.push (root);
        // push the root.
    }
    bool hasNext()
    {
        return stack.size () != 0;
    }
    Node getNext()
    {
        Node ret = stack.pop();
        if (ret.right != null)
            stack.push (ret.right);
        if (ret.left != null)
            stack.push (ret.left);
        return ret;
    }
}
  
```

main:

```

PreOrder ob = new PreOrder();
while (ob.hasNext())
{
    point (ob.getNext().data)
}
  
```

, levelOrder >>

↑
it according to level
order first. Node.

code
function

, -1, -3, 12, 10, -9

root.
print data

↓ first
+ child. ch.

ob=new PreOrder()

hasNext()

do.getNext().data

class PreOrder

Stack< Integer> stack;

PreOrder (Node root)

{ if (root != null)

{ stack.push(root);

bool hasNext()

{ return stack.size() != 0;

Node getNext()

{ Node n = stack.pop();

if (n.right != null) stack.push(n.right);

if (n.left != null) stack.push(n.left);

return n;

}

3.

Main

PreOrder ob = new PreOrder(root);

while (ob.hasNext())

Print(ob.getNext().data);

Do It.

Iterative: 1. Inorder Traversal

2. Postorder Traversal

3. Morris Traversal.

④ SDLL \Rightarrow BST
 (Sorted Doubly Linked List) \Leftrightarrow (Balanced Binary Search Tree)

↳ Brackets
 $1 \Rightarrow 3 \Rightarrow 8 \Rightarrow 12 \Rightarrow 15 \Rightarrow 18 \Rightarrow 25$

void convertTOBST(Node h)

{

Node root = findMid(h);

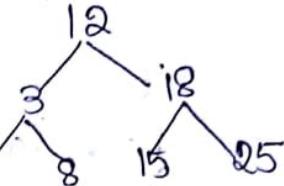
h = root.next;

// now break the links;

root.prev = findMid(h);

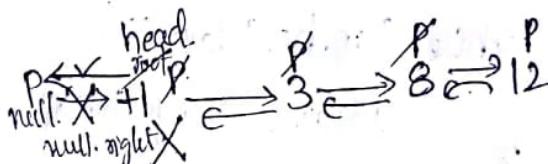
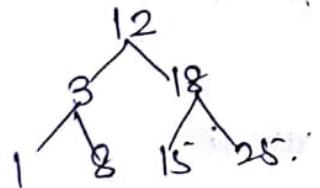
root.next = findMid(h');

}



⑤ BST \Rightarrow SDLL

Double
 Inorder: Sorted Linked List.



void convertToSDLL(Node root)

{

Node p = null;

convertToSDLL(root.left);

if (p != null)

p.right = root;

else

head = root;

root.left = p;

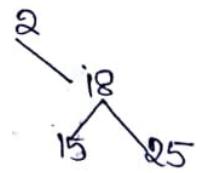
p = root;

convertToSDLL(root.right);

}.

Find Floor and Ceiling in BST:

ref Search Tree



floor(17) \Rightarrow $10 < 17 \Rightarrow$ go right. ans = 10.
 $18 > 17 \Rightarrow$ go left.
 $15 < 17 \Rightarrow$ ans.

ans = Integer.MIN_VALUE;
 int floor(Node root, int key)

```

{
    if (root.data <= key)
    {
        ans = root.data;
        floor(root.right, key);
    }
    floor(root.left);
}
return ans;
  
```

List.

P
2

July 19

Solution

Unsorted Array
Sorted Array

Insert(x)

$\frac{1}{N}$

delMin()

$\frac{N}{N}$

$N(1)$
if asc, shifting takes N
if desc, del. last. 1

$\frac{N}{N}$

H

$\log_2 N$

getMin()

$\frac{N}{1}$

N

H

1

(Min)
(Max)

(Unsorted Single LL)

Binary Search Tree(BST)

CBT, \Rightarrow Node \leq child

$\log_2 N$

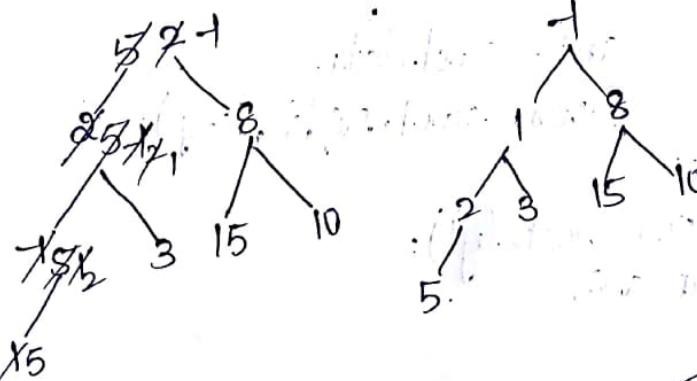
CBT (Complete Binary Tree).

a) Node \leq child Nodes.

Heaps

arr: Form the Binary Tree such that the above 2 conditions satisfy.

arr: 5 2 8 1 3 15 10 ~~7 1~~



Complete Binary Tree(CBT)

H Height
N Nodes.

$\frac{N}{2^H}$ (min)
 $\frac{N}{2^{H+1}}$ (max)

$$N = 2^H$$

$$N = 2^{H+1} - 1$$

$$H = \log_2 N$$

$$N + 1 = 2^{H+1}$$

$$\log_2(N+1) = H+1$$

$$H = \log_2(N+1)$$

$\text{Insert}(x) \Rightarrow O(\log N)$

$\text{getMin()} \Rightarrow O(1)$

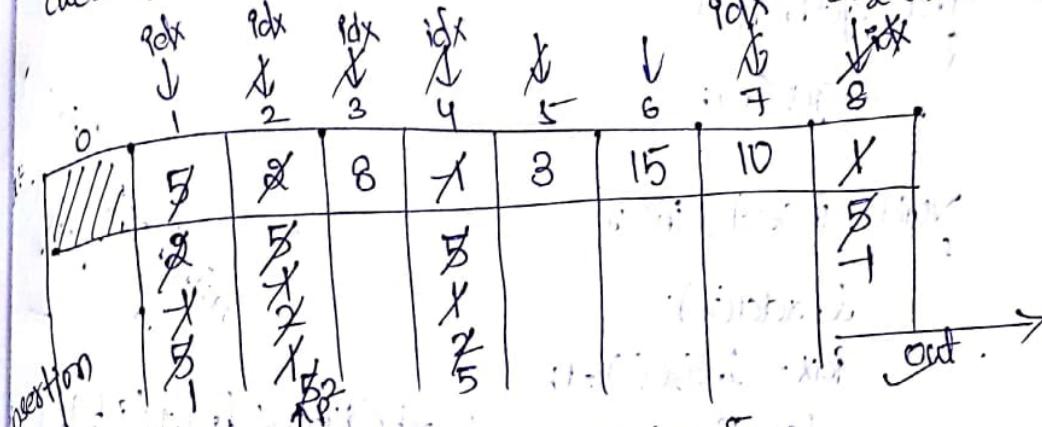
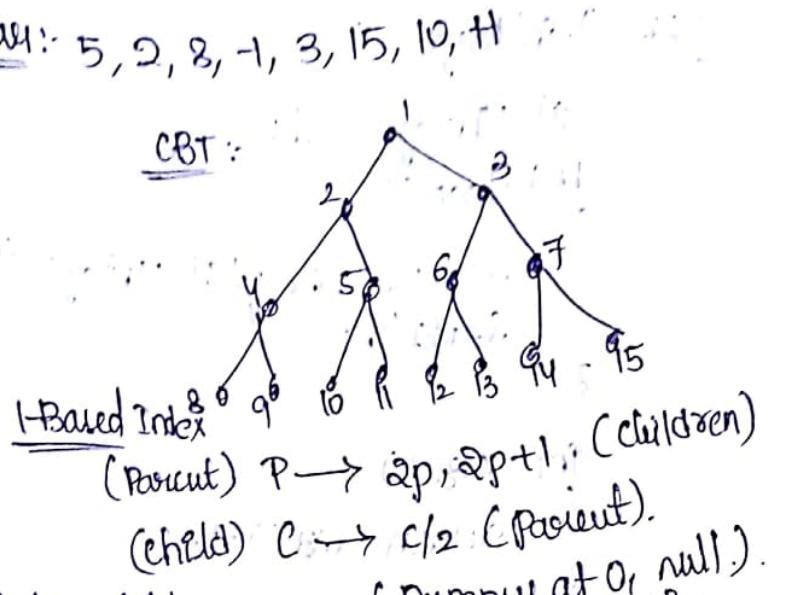
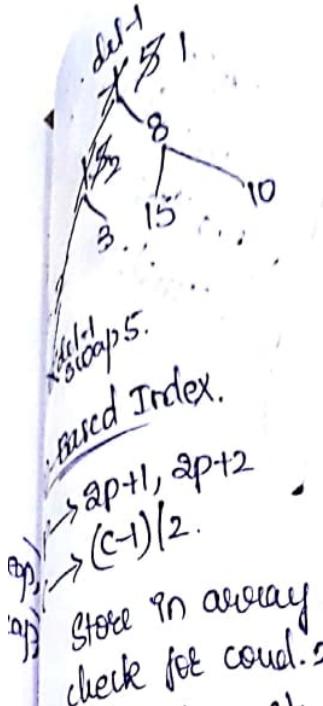
$\text{delMin()} \Rightarrow O(\log N)$

for deletion

swap with the last node, form tree

And then from root go on checking for condition node \leq child.

swap accordingly
root with min(left child right child)



Insertion: -1, 5, 2, 8, 3, 15, 10; 5.

After Deletion: 1, 2, 8, 5, 3, 15, 10. after deleting min.

a[0]
node > child
so swap

while inserting,
check with parents.
if cond2. is not satisfied,
swap, then.

while deleting, swap arr[i], arr[idx]
idx -

then focus first root,
check with its children
get min of two children
- check for cond2) NodeLchild
- swap according.
- go with the min child next
Continue checking. -

```

class MinHeap {
    List<Integer> l;
    MinHeap() {
        l = new ArrayList<Integer>();
        l.add(0);
    }
    int size() {
        return l.size() - 1;
    }
    int getmin() {
        return l.get(1);
    }
    void insert(int x) {
        l.add(x);
        int idx = l.size() - 1;
        while (idx > 1 && l.get(
            idx / 2).swap(l.get(idx))
            idx = idx / 2;
    }
}

```

TODO

- Generate Tmp of Help
- 1) NextHeap
- 2) Templates
- Use Boost

```

    ✓ void delmin()
    {
        swap(l.get(1), l.get(l.size() - 1));
        l.remove_back();
        int idx = 1;
        int c1 = a * idx;
        int c2 = a * idx + 1;
        while(idx < l.size() && minchild = min(l.get(c1), l.get(c2)))
        {
            swap(l.get(idx), l.get(minchild));
            idx = minchild;
        }
    }

```

Given the array, find the first k smallest elements:

arr: 10 15 -3 8 12 7 18 20 1 35
 $k=4 \rightarrow$ ans: -3 8 7 1

1. Brute Force: Sort and get first k elements.

Complexity $O(N \log N)$, $O(1)$ check sort
- No extra space

2. Use MinHeap:

$O(N \log N + k(1 + \log N))$, $O(N)$ heap space.

Insertion of
N ele in minHeap

getmin
del
min

$\xrightarrow{\text{Comp}}$ $O(N \log N) \cdot O(N)$

Insertion: $\log_2 + \log_2 + \log_2 + \dots + \log_2 \Rightarrow \log_2 N$

3. Use MaxHeap of k elements.

10, 15, -3, 8 | 12, 7, 18, 20, 1, 35
 $k=4$ MaxHeap

12 comes
getmax() = 15
 $= 15 > 12$ delmax

k size
Heap

$O(k \log k + (N-k)(1 + \log k + \log_2))$, $O(k)$ k size
heap
Insert.
coming ele.
if it is less
than getmax().

$\xrightarrow{\text{Comp}}$ $O(N \log k), O(k)$

(10) 15, 23, 8, 12, 7, 18, 20, 1, 35

-3, 8, 7, 1, 10 15, 12, 18, 20, 35

Best Case $\Rightarrow O(N)$ Partitioning takes $> O(N)$ time.
when we take left ele as pivot.

Worst case $\Rightarrow O(N^2)$, $T(N) = T(N-1) + N$

Take smallest ele. as pivot,
then 0 no. ele. on left of pivot
and $(N-1)$ ele. on right side.

① pivot $(N-1)$ ele

ele. again $O(N-1)$ ele
do O.S.

So, $T(N) = T(N-1) + N$
 $\Rightarrow O(N^2)$.

Worst Case: $T(N) = T(N-1) + N \Rightarrow O(N^2)$

Best Case: $T(N) = \alpha T\left(\frac{N}{2}\right) + N \Rightarrow O(N \log N)$

Average Case: $T(N) = T\left(\frac{9N}{10}\right) + T\left(\frac{N}{10}\right) + N \Rightarrow O(N \log N)$.

Quick Select:

~~we will not consider bigger~~ Aug:

~~90% less (20%)~~ $T(N) = T\left(\frac{9N}{10}\right) + N \Rightarrow O(N)$

~~10%~~ \Rightarrow Quick Select: $O(N), O(1)$

Q. Complexity: $O(N), O(1)$

To Do
1) Heap-Sort
2) Median of Medians

element should be at max k distance away from pk
actual position:

Given a k-sorted array,

Ans: arr: 10 5 2 12 1 10 8 6 16 14 7 11 8 13 12
Ans: arr: 1 2 5 6 8 10 10 12 13 15
k-sorted array, $k=3$

① Brute Force: Simply sort the array.
 $\rightarrow O(N \log_2 N), O(1)$.

② Use min-Heap.

1) Insert $(k+1)$ ele in heap: $\Rightarrow (k+1) \log_2(k+1)$

2) For $(N-k)$ \Rightarrow del. min ele., insert new ele.

\therefore ~~10, 8, 16, 15~~ $\rightarrow O(N \log_2(k+1)), O(k+1)$

Ans: -1, 2, 5, 6, ...

③ Given an row-wise sorted matrix :-

Output the completely sorted order:

matrix: -5 2 8 12 18 20
3 7 10 15 20 23
-8 -6 -1 0 7 12
7 10 18 53 97 123
0 18 23 129 35 42
Ans: -8, -6, -5, -1, 0, 2, 7, 7, 8, ...

④ Brute force: Copy into array & sort the array.

Copy: $\Rightarrow O(Nm + Nm \log_2 Nm), O(Nm)$.

② Merging Row by row:

$$m(2+3+4+\dots+N)$$

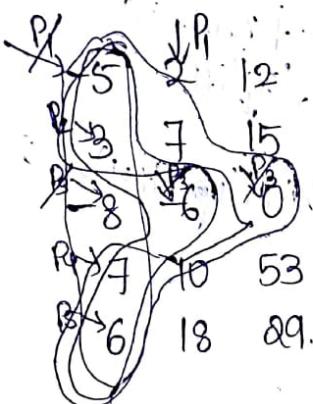
$$= m\left(\frac{N(N+1)}{2} - 1\right)$$

$$\therefore = mN^2 \quad \text{space: } Nm$$

Complex: $O\left(m\left(\frac{N(N+1)}{2} - 1\right)\right)$, $O(Nm)$

③ Take pointers:

Take pointers at start of all the rows, get the min ele, inc the pointer of that particular row, and then check again the min ele. In the next all



→ N pointers $\rightarrow N$ pointers $\rightarrow N$ pointers $\rightarrow \dots$

Complex: $O(N * Nm)$, $O(N)$

Forget min for all
of N ptr ele. \rightarrow Nm ele.

④ Use MinHeap: get min $\rightarrow O(1)$

First column in minheap Insert new small $\rightarrow O(\log N)$ Delete min one $\rightarrow O(\log N)$

Comp. $O(Nm + \log N)$, $O(N)$

Heap space: Nm

minHeap

minHeap<A>

sort the elements based on the data, and for next ele to check.

A

int data;
int slowIndex;
int colIndex;

fr. 'colIndex' in the particular slowIndex, hence minHeap<A> template as object A: A{ int d, g; }

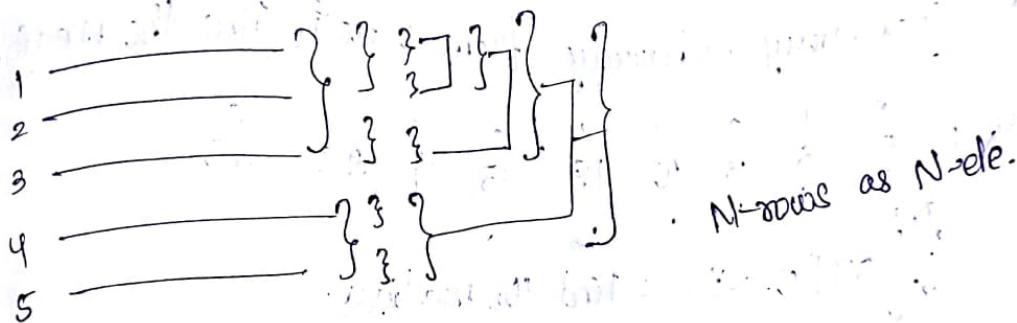
(ii) use pair.

MinHeap<Pair<int, Pair<int, int>>>

iv

③ Using Merge Sort,

taking, considering one ele. as one row.



$$T(N) = \alpha T\left(\frac{N}{2}\right) + NM$$

splitting
N-rows.

merging all ele.

Cmplx: O(

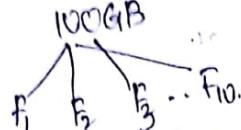
)

To do: complex

④ Sort a 100GB file of numbers.

We have only 1GB of RAM.

First,



10 chunks of 1GB each.

Each chunk take into 1GB RAM, sort it, push back in Hard Disk.
→ Like that for 10 files - 10 chunks; we have each one sorted.

— Then take the first parts of all the chunks into RAM,
sort it use merge heap, remove def. min. of all the
parts of chunk, then put the next part of that chunk
in RAM & so on.

⑤ For every subarray from 0 to i, find the median.

arr:	0	1	2	3	4	5	6	7	8	9	10
	5	8	3	10	18	15	9	25	20		

N
arr [0 → i] : find the median.
i = 0

Index
0 → 0 SubArray → 5 med.

After 5, 5, 5, 5, 8, 8, 9, ... 0 → 1 → 5 med

0 → 2 → 5, 8 → 5 med

0 → 3 → 3, 5, 8, 10 → 5 med

① Brute force:

For all subarrays from 0 to N.

There are N subarrays. For each subarray, sort it & get the middle element as median.

$O(N + N \log N)$, $O(N)$

↑
Subarrays
Rooting.

for copying & sorting them

Insertion Sort:

$O(N^2)$, $O(1)$

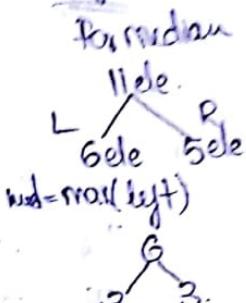
sort & result median.

No. of smaller elements in left, No. of greater ele. in Right.

a) $\text{arr}(i) \in L \& R$.

b) $\text{size}(L) - \text{size}(R) = [0, 1]$.

Median will be max ele. from the left



ex: $5, 8, 3, 10, 18, 15, 9, 7, 2, 1, 3, 3$

ans: $5, 5, 5, 8, 8, 9, 10, 18, 15, 12, 7, 3, 3$

MaxHeap (Left)

(Right)

5, 3, 8 8, 10, 18

5 - insert in left, size ok.
8 - $> \max(\text{left})$ ie 8
insert in right.

$\frac{L}{5} + \frac{R}{8}$. med = $\max(\text{left})$

med = 5.

When new ele comes, follow 2 cases.

if ele $> \max(\text{left})$,

$3 \Rightarrow 3 < \max(\text{left})$

3 < 5

then insert in Right side, else insert in left side.

check size = size(L) - size(R)

size = 1 - 0 E $[0, 1]$
med = $\max(\text{left}) = 5$.

if size E $(0, 1)$

$10 \Rightarrow 10 > \max(\text{left})$

ans med = $\max(\text{left})$.

insert in right.

else if size $\notin (0, 1)$

size = 2 - 2 = 0 E $(0, 1)$
med = $\max(\text{left}) = 5$.

Remove min (right) from

right side & put it in the

left side. till

\Rightarrow ans med = $\max(\text{left})$

$\log_2 N$, $O(N)$

$\log_2 N/2$

$18 \Rightarrow 18 > \max(\text{left})$
Insert in Right Side.

$5, 3, 8, 8, 10, 18$

size = $8 - 3 = 5 \notin (0, 1)$

del min from Right

insert in left \Rightarrow med = 8

- Given N-words $1 \leq \text{length(word)} \leq m$.
 Find the number of words which start with Query, going through words.

\downarrow ans.
 da \Rightarrow 3 data
 dark
 date
 ate \Rightarrow 3 stem
 sleep
 stu \Rightarrow 1 student
 dec \Rightarrow 2

stem, sleep
 data, sleep
 dark, sleep
 smart, drawer
 smooth, slack
 steep, student
 algo

① Brute Force:

For every query, check N words, at max m letters. Comp
 If equal \Rightarrow Inc. count
 Complexity $O(Q \cdot N \cdot m)$, $O(1)$. $\rightarrow O(Q \cdot N \cdot m), O(1)$

② Sort all the words & do binary search space.

letter words
 $\Rightarrow O(m \tilde{N} \log N + Qm * \log N)$, $O(1)$. algo.

$$p_1 = 0, p_2 = N$$

for (q) $p = 0; q \leq w.size(); q++$

$p_1 = BS_1(\text{List}, w(q), q, p_1, p_2);$

$p_2 = BS_2(\text{List}, w(q), q, p_1, p_2);$

$$\text{ans} = p_2 - p_1 + 1$$

Binary search for finding the first occurrence and last occurrence of the query in List of words.

③. HashMap

$O(NM^2 + Q \cdot M)$, $O(NM^2)$.

↓
Search the
Query in all
hashmaps
Strings of max
with m length.

stem

step

ste, k 2

ste, k 2

ste, k 2

stem - 1

step - 1

For each word $\rightarrow M^2$ hash keys.
N words $\Rightarrow NM^2$ space.

HashMap < string, int >
words, substrings
count

Optimise $O(N^2)$

Optimize a bit.

4. Get the hash values of substrings of given words by applying hash functions h_1 or h_2 and insert the calculated hash value, & count.

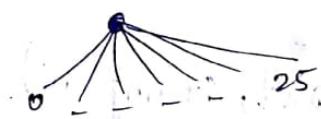
HashMap < Int, Int >.

$O(NM + QM)$, $O(NM)$.

④ TRIE

e.g. abc

a b c d
0 1 2 3



every node has 26 links
essentially pointing to
nulls, denoting all
the characters of alphabet



of leaf nodes = 3 best for draw
leaves = 12
but arr = 2

doesn't work.

for query dr X

Class Trie

```
{     node[c[26];
```

```
}
```

func createNode()

```
{     Trie *n = new Trie();
```

```
    for (int i=0; i<26; i++)
```

```
        n.c[i] = NULL;
```

```
    return n;
```

```
}
```

Main:

```
Trie *root = createNode();
```

```
for (int i=0; i<N;
```

```
    // read word
```

```
    insert(root, word);
```

word :: insert(Trie root, string w)

```
{     for (int i=0; i<w.size(); i++)
```

```
        int pdx = w[i] - 'a';
```

```
        if (root.c[pdx] == NULL)
```

```
            root.c[pdx] = createNode();
```

```
        root = root.c[pdx];
```

② Time Limit.

① But this gives wrong answer for
dray, drayel
so maintain flag values.
at every node
and initially worth all false.
Once we complete the word
mark flag as true.
At the end, cat or
tiger is true.



```
class Trie
{
    Trie c[26];
    bool flag;
}
```

```
Trie createNode()
```

```
{ n.flag = F;
void insert(---)
```

=====

```
after for loop:
root.flag = T;
```

Instead of flag maintain count

```
class Trie
```

```
{ Trie c[26];
int cnt;
```

```
Trie createNode()
```

```
{ Trie n = new Trie(); n.cnt = 0;
for (int i=0; i<26; i++)
    n.c[i] = NULL;
```

```
return n;
```

```
void insert(Trie root, string w)
```

```
{ for (int i=0; i<w.size(); i++)
    {
        int idx = w[i] - 'a';
        if (root.c[idx] == NULL)
            root.c[idx] = createNode();
        root.c[idx].cnt++;
    }
}
```

```
root = root.c[idx];
```

```
root.cnt++;
```

```
{
```

```

int query(TrieNode* root, string w)
{
    int qdix = w[0] - 'a';
    if (root->children[qdix] == NULL)
        return 0;
    root = root->children[qdix];
    return root->cnt;
}

```

Complexity: $O(\underbrace{N \cdot M}_{\text{N weeds}} + \underbrace{Q \cdot M}_{\text{for each query we search till N letters}})$, $O(N \cdot M)$ → Total space for N weeds of M letters.

LAB.

1. Doubts
2. HR - Practice → LL(5), Trees(5)
3. HR - SJ → Trees. (Page 8:5)
4. IB → LL(2) → Trees(2)
5. HR → SJ → Implement minHeap.
6. HR → Practice → DS → Trie(1).

~~Ex 14.18~~ ~~* Partition the given string such that each partition is present in the list of words given.~~
 Return True if yes, else False.

e.g.: List of words: $L = \{ "axe", "asym", "ax", "ear", "stop", "lap", "ear", "cp" \}$
String: $axe|asym|stop|lap|ear$ \Rightarrow True.

$axe|e| \rightarrow m o p |$ \Rightarrow HashSet<String>

bool isPartitioningPossible(List w, String str, int pdx).

{ if ($pdx == str.size()$)
 return true;

for (int i=pdx; i < str.size(); i++)

{ if ($L \in str[idx:i]$ && isPartitioningPossible(L, str, i))
 return true;

3 return false;

};

$str[idx:i] \Rightarrow$ substring(pdx, i): $axe|asym|$
 $str[idx:i] \Rightarrow$ substring from pdx to i belongs to List & next partitionings
 if impossible, \Rightarrow then return false.

$axe|ear|$ $axe|ax$
 $ear|X$ $axe|ear|X$

$Ear|ym|X$

Lot of computations are wasted as even after $axe|a$ is
 not present in list, we are also checking for $axe|a|$.

So use Tries:-

Task 1: [int 7726]

bool f(flag, root, L, stsi, idx, root)



axe

ear

axeay

bool f(L, stsi, idx, root)

{ if (idx == stsi.size())

return T;

Node stsi[idx] = root;

for (int i = idx; i < stsi.size(); i++)

{ int p = stsi[i] - 'a';

if (root->cnt[p] == null)

return F;

root = root->chil[p]; // check flag of the child

if (root->flag == T && f(L, stsi, i + 1, root))

return T;

}

return T;

}

* Find the maximum XOR-pair: $\max(\text{arr}(i) \wedge \text{arr}(j)) = ?$

arr :- 10 18 3 25 12 4
 (01010) (10010) (00011) (1100) (01100) (00100)

① Brute Force: Iteratively update max. ans:
 Comp $O(N^2), O(1)$.

$$\begin{array}{r} x = \\ \begin{array}{r} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{array} \\ \hline y = \\ \begin{array}{r} 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ \hline 0 \\ 4 \\ 3 \\ 2 \\ 1 \\ 0 \\ 2^5 - 1 \end{array} \end{array}$$

$$\text{ans} = \underline{\underline{1 \ 1 \ 1 \ 1 \ 1}}$$

let us fix 10 4th 3rd 2nd st 0th

$$a = 0 \ 1 \ 0 \ 1 \ 0$$

$$b = \underline{\underline{1 \ 0 \ 1 \ 0}}$$

$$\text{ans} = \underline{\underline{1 \ 1 \ 0 \ 0 \ 0}}$$

next fix 18. fix a and find b such that it gives max XOR.

Take ^{no. of} bits required for the max. no.

Here 25 $\Rightarrow \log_2 25 \Rightarrow 5$ bits. (0 to 4)

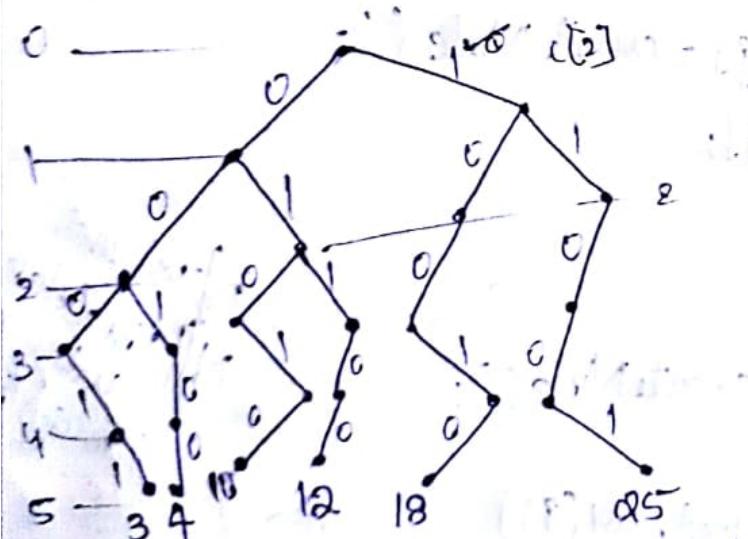
Tree: Left 0, Right 1.

repeat this for all no. of array & find max XOR.

$$\begin{array}{r} 4^{\text{th}} \ 3^{\text{rd}} \ 2^{\text{nd}} \ 1^{\text{st}} \ 0^{\text{th}} \\ \hline a: 10 \ 0 \ 1 \ 0 \ 1 \ 0 \\ b: \underline{\underline{1 \ 0 \ 0 \ 1 \ 0}} \\ \text{ans: } \underline{\underline{1 \ 1 \ 0 \ 0 \ 0}} \end{array}$$

for 4th bit, we have 0,
 to get 1, we need 1 at 0th and
 As we have that 0, put 1 instead
 & go to next bit 3rd.

\rightarrow for ans to be 1, we need 0 at 3rd bit.
 At 0th level 1, we have float 0 in ans
 \rightarrow Next we need 1 in 2nd as 0's there,
 but 1 is not present at level 2,
 but 0 in b & 0 in ans \rightarrow go on



// first know the no. of bits required to represent each no i.e. no. of bits req. for max no. of concat. bit bits

②

class Tsue:

{ Tsue c[3];

3

Tsue concatNode();

{

Tsue n = new Tsue();

for (int i=0; i<3; i++)

n.c[i] = null;

return n;

3

void insert(Node sroot, int x, int bits)

2

int j = bits - 1;

int j = checkBit(x, i) == T ? 1 : 0;

if (sroot.c[j] == null)

sroot.c[j] = concatNode();

sroot = sroot.c[j];

3.

Main:

loop(T)

Node sroot = concatNode();

loop(N)

insert(sroot, a[i])

ans = 0;

loop(N)

ans = max(ans, query(sroot, a[i], bits));

x is an ele. of concat
|| Insert all the ele
of ele of concat
In Tsue first

(ok)
Insert
Node \Rightarrow 32N

T | Test cases
N | No. of ele in
array
|| array

```

int query( Node root, int x, int bits ) // query func.
{
    int ans = 0;
    for( int i = bits - 1 ; i >= 0 ; i-- )
    {
        int b = checkBit( x, i ) == T ? 1 : 0;
        if( root.c[1-b] != null )
        {
            ans = ans + pow( 2, i );
            root = root.c[1-b];
        }
        else
            root = root.c[b];
    }
    return ans;
}

```

Time: $O(32N + N \times 32)$, $O(N \times 32)$.

\uparrow \uparrow
 Node insertion, Node query 32 Nodes for leaf.
 Leaf $\Rightarrow 32$.
 $\Rightarrow 32 \times N$ for Nodes.
 $\Rightarrow O(N), O(N)$.

④ Find the maximum subarray XOR :-

Ques: 10 18 3 25 12 4

① Brute force :- check all the subarrays & get max subarray XOR

Ans: $O(N^2 \times N^2)$ $\Rightarrow O(N^8), O(1)$

$$\text{No. of subarrays} = \frac{n(n+1)}{2}$$

② Carry forward :- $O(N^2), O(1)$

Approach:

max Subarray XOR (int arr[])

```

int ans=0;
for(int i=0; i<n; i++)
{
    int xor1=0;
    for(int j=i; j<n; j++)
        ans=max(ans, xor1^arr[j]);
}
return ans;

```

10: 0 1 0 1 0

18: 1 0 0 1 0

③. ①. $\text{XOR}[i, j] = \text{XOR}[0, j] \wedge \text{XOR}[0, i-1]$

②. XORs till index i :

Eg. $m_3 = \max \left\{ \begin{array}{l} \text{XOR}[0, 3], \\ \text{XOR}[1, 3], \\ \text{XOR}[2, 3], \\ \text{XOR}[3, 3] \end{array} \right\}$

m_3

$m_i = \max(\text{all XOR till } i \text{ ending index})$ / $\text{XOR}[i, j] = \text{XOR}[0, j] \wedge \text{XOR}[0, i-1]$

$m_3 = \max \left\{ \begin{array}{l} \text{XOR}[0, 3], \Rightarrow \text{XOR}[0, 3] \wedge \cancel{\text{XOR}[0, 0]} \\ \text{XOR}[1, 3], \Rightarrow \text{XOR}[0, 3] \wedge \text{XOR}[0, 1] \\ \text{XOR}[2, 3], \Rightarrow \text{XOR}[0, 3] \wedge \text{XOR}[0, 2] \\ \text{XOR}[3, 3] \Rightarrow \text{XOR}[0, 3] \wedge \cancel{\text{XOR}[0, 2]} \end{array} \right\}$

a \wedge b ↑ query

$$m_4 = \max \left\{ \begin{array}{l} \text{XOR}(0,4) \Rightarrow \text{XOR}(0,4) \wedge \text{XOR}(0,0) \\ \text{XOR}(1,4) \Rightarrow \text{XOR}(0,4) \wedge \text{XOR}(0,1) \\ \text{XOR}(2,4) \Rightarrow \text{XOR}(0,4) \wedge \text{XOR}(0,2) \\ \text{XOR}(3,4) \Rightarrow \text{XOR}(0,4) \wedge \text{XOR}(0,3) \end{array} \right. \quad \begin{array}{l} \text{pack} \\ \text{pair} \\ \text{value} \end{array}$$

$$m_0 = \max \{ \text{XOR}(0,0) \}$$

$$m_1 = \max \left\{ \begin{array}{l} \text{XOR}(0,1) \\ \text{XOR}(1,1) \end{array} \right\} \Rightarrow \begin{array}{l} \text{XOR}(0,1) \wedge \text{XOR}(0,0) \\ \text{XOR}(0,1) \wedge \text{XOR}(0,0) \end{array}$$

int maxSubArrayXOR (int arr[], int N, int bits)

```

1   int ans=0, a=0;
2   for( int i=0; i<N; i++ )
3     insert (sroot, a, bits)
4     a=a ^ arr[i]
5     ans=max( ans, query(sroot, a, bits));

```

return ans;

Complexity: $O(N)$.

④ Fibonacci: 1, 1, 2, 3, 5, 8, 13, 21, ...

Recursive

P.C.: int fibRecursive (int N) // Top-Down Memoziation

{ if (N==1 || N==2) return 1;

 return fibRecursive(N-1) + fibRecursive(N-2);

$$T(N) = T(N-1) + T(N-2),$$

⇒ Comp: $O(2^N)$, O(1)

Iterative

int fibIterative (int N) // Bottom Up; TABULATION

{ int arr[N+1]; arr[1] = arr[2] = 1;

 for (int i=3; i<=N; i++)

 arr[i] = arr[i-1] + arr[i-2];

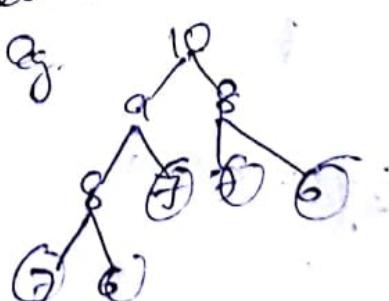
 return arr[N];

Comp: $O(N)$, O(N)

Recursive Code complexity is much higher $O(2^N)$ as the Computations are done again & again. Eg.

→ Overlapping Subproblems.

→ To compute the values and store those, so that when it comes next time, without recalculating it, it takes the precalculated value.



```

int arr[N+1] = {-1}; ← DPTable
int fibRec(int N)
{
    if (N == 1 || N == 2)
        return 1;
    if (arr[N] == -1)
        arr[N] = fibRec(N-1) + fibRec(N-2);
    return arr[N];
}

```

DYNAMIC PROGRAMMING.

1) Overlapping Subproblems.

DP State: $\{ dp(i) \rightarrow i^{\text{th}} \text{ Fibonacci number} \}$

DP Expression: $dp(i) = dp(i-1) + dp(i-2)$.

DP Table — size(DP Table)

e.g. arr[N+1] in fib.

for fib \Rightarrow No. of DP states = N

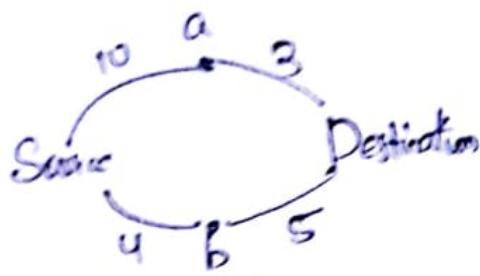
$arr[N+1] \Rightarrow$ 0 is not valid
N states valid.

Time Complexity: #DP states * T.C(1 state)

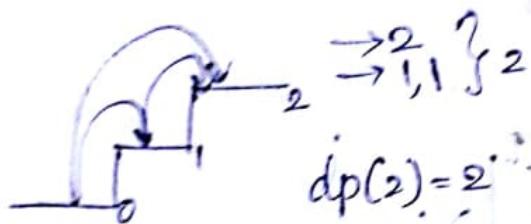
$$= N * 1 \\ = \underline{\underline{O(N)}}$$

Optimize the space in iterative fib as we just require the previous two values to get the current value.

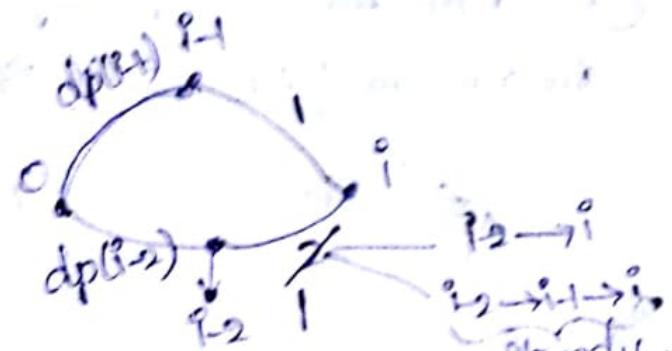
$$dp(i) = dp(i-1) + dp(i-2).$$



$$\begin{array}{r} \text{S.A.D.} \quad \text{S.b.D} \\ 10+3 + 4+5 \\ = 30+20 \\ = 50. \end{array}$$



$$dp(2) = 2$$



$$\Rightarrow dp(i) = dp(i-1) + dp(i-2)$$

$$\Rightarrow dp(i) = dp(i-1) + dp(i-2)$$

so there is also a step

$$dp(0)=1$$

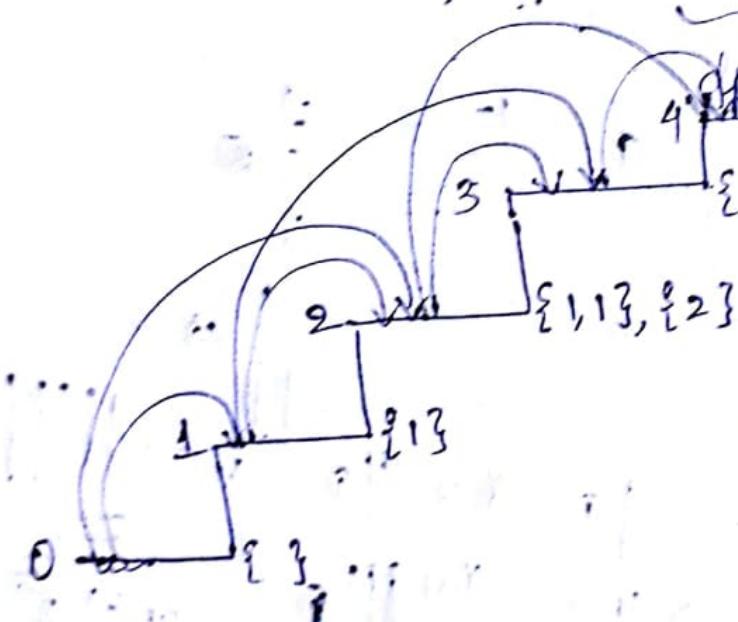
$$dp(1) = 1 - \frac{1}{2} = \frac{1}{2}$$

1310

dp exp

dp. basic cond.

$\{1, 1, 1\}, \{2, 1\}, \{1, 2\}$

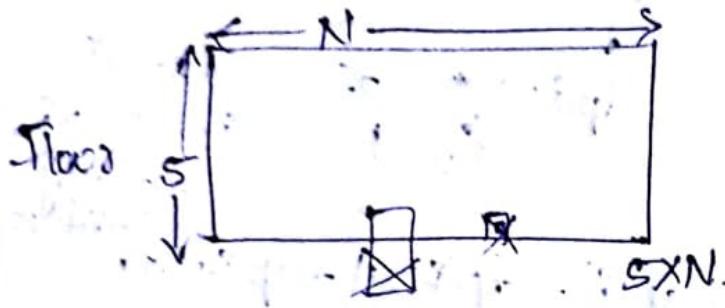


$$dp(0)=1$$

$$dp(1)=1$$

$$dp(i) = dp(i-1) + dp(i-2).$$

* find the no. of ways to place the tiles (5×1 size) in the floor ($5 \times N$ size).

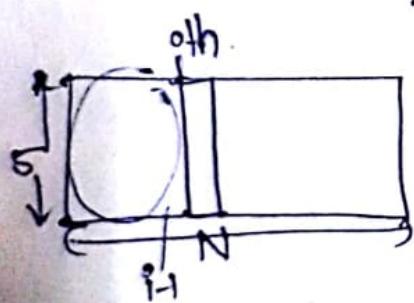


(oo) tiles \rightarrow #ways = ?

Placing: 6 { 5×1 } X

e.g.: 5×7
Floor
If $N=7$

for $N=7$
ans = 4.



$$\text{so } dp(0) = 1.$$

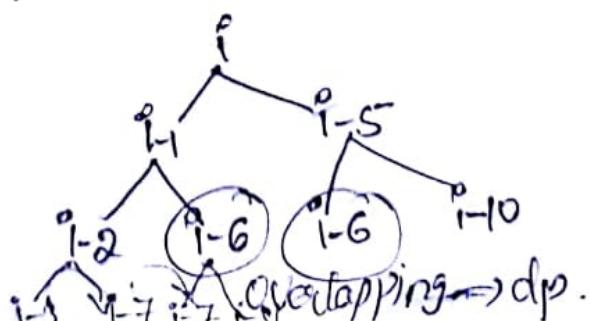
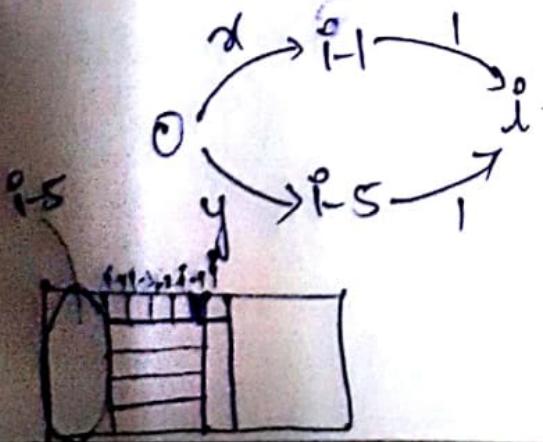
$$i=1 \Rightarrow | \textcircled{1} .$$

$$i=2 \Rightarrow || \textcircled{1} .$$

dp state

② $dp(i) \Rightarrow$ # ways to fill floor of size $5 \times i$

$$dp(i) = dp(i-1) + dp(i-5)$$



Base Conditions: $dp(0) = 1$

$dp(1) = 1$

$dp(2) = 1$

$dp(3) = 1$

$dp(4) = 1$

Dp. Expression: $\sum_{i=5}^N dp(i) = dp(i-1) + dp(i-5)$

Table Size: $\text{int } dp[N+1]$

Space Complexity: $O(N)$

Time Complexity: $O(N)$

Ans: $dp[N]$

Space Optimization: Take GELLE array as we req. -1 & -5 positions values

0	1	2	3	4	5
$dp(0)$	$dp(1)$	$dp(2)$	$dp(3)$	$dp(4)$	$dp(5)$

$$dp(5) = dp(0) + dp(4) \quad \text{shifting next}$$

$$(02) \sum_{i=5}^N dp(i \% 6) = dp((i-1)\%6) + dp((i-5)\%6)$$

i is getting override

$$dp(6) \Rightarrow dp(0) = dp(5) + dp(1)$$

$$dp(7) \Rightarrow dp(1) = dp(0) + dp(2)$$

0	1	2	3	4	5
0	X	2	3	4	5

Time for int overflow
(i.e. for max value)

Max value

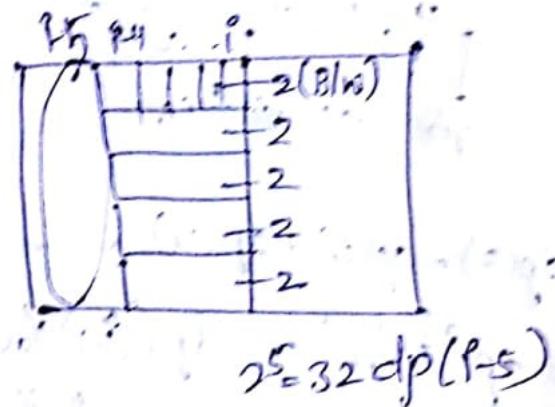
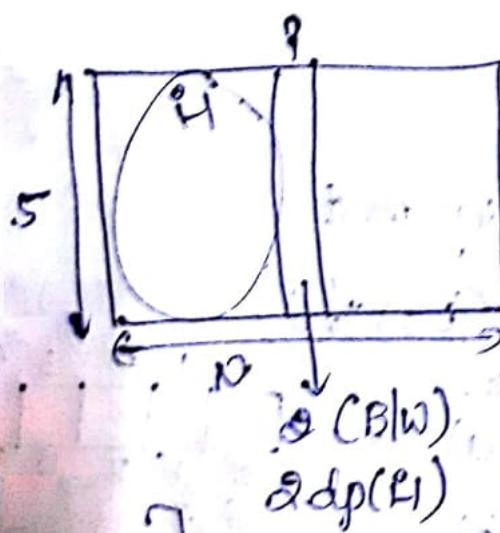
Overflow condition

Overflow condition

* Steps to Identify DP Problems

- a) Identify Overlapping Subproblems
 - b) Dp. State
 - c) Dp. expression
 - d) Base Conditions
 - e) Table Size
 - f) Time & Space Complexities
 - g) Where is the answer?
 - h) Space Optimisation

* Find the no. of ways to place the 2 coloured front- Black, back white tiles in floor ($5 \times N$).



$$\xrightarrow{\text{state}} dp(p) = 2 \times dp(p-1) + 32 \times dp(p-5)$$

$$dp(0) = 1$$

$$dp(1) = 2$$

$$dp(2) = 4$$

$$d_P(3) = 8$$

$$dp(u) = 16$$

Table 9.28 = first dp[N+1]

Space Comp.: $O(N)$, TC: $O(N)$

Ans :- $dp(N) \neq 2^N$

Space Optimization : yes ~~like~~

$$dp(9\%, 6) = 2 \cdot dp(8\%, 6) + 32 \cdot dp(7\%, 6)$$

* for given length N , find no. of strings with 0's or 1's such that there are no two 1's are adjacent.

$$N=1 \Rightarrow \frac{0}{1} \checkmark \quad \text{ans}=2$$

0000 ✓
0110 X
0011 X
0101 ✓

$$N=2 \Rightarrow \begin{array}{l} 00 \checkmark \\ 01 \checkmark \\ 10 \checkmark \\ 11 X \end{array} \quad \text{ans}=3$$

$N=3$. 000 ✓ 100 ✓
001 ✓ 101 ✓
010 ✓ 110 X
011 X 111 X

$\frac{N}{1}$ $\frac{\text{ans}}{2}$
 $\frac{2}{2}$ $\frac{\text{ans}}{3}$
 $\frac{3}{3}$ $\frac{\text{ans}}{5}$
 $\frac{4}{4}$ $\frac{\text{ans}}{8}$

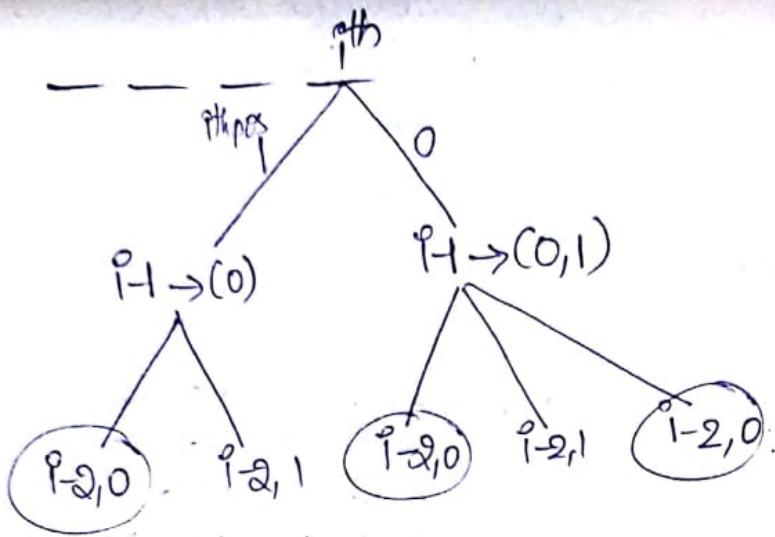
$$N=4 \Rightarrow \begin{array}{l} 0000 \checkmark \\ 0001 \checkmark \\ 0010 \checkmark \\ 0011 X \\ 0100 \checkmark \\ 0101 \checkmark \\ 0110 X \\ 0111 X \end{array}$$

<u>$\frac{N=1}{1}$</u>	<u>$\frac{N=2}{2}$</u>	<u>$\frac{N=3}{3}$</u>	<u>$\frac{N=4}{4}$</u>
0	00	000 010	0000 0100 1000 0010 1010 0011 0101 1001 0110 1011
1	01 10	010 100 011	

append 0's to $N=2$
append 01's to $N=1$
append 0's to $N=3$
append 01's to $N=2$

dp-state = no. of strings of length N without adjacent 1's.
dp-exp : $dp(i) = dp(i-1) + dp(i-2)$

Base Cond^t $dp(0)=1$
 $dp(1)=2$



We need no. of strings with length H ending with 0 & 1

— overlapping sub-problems.

a) dpo
 $dpo(i) \rightarrow \# \text{Binary strings of length } i, \text{ ending with } 0. (\text{ith=0})$
 $dpi(i) \rightarrow \# \text{Binary strings of length } i, \text{ ending with } 1. (\text{ith=1})$

$$\begin{aligned}
 \text{dp exp} \quad \sum_{i=1}^N / \text{dp0}(i) &= \text{dp0}(i-1) + \text{dp1}(i-1) \\
 \sum_{i=1}^N \text{dp1}(i) &= \text{dp0}(i-1) \\
 i=2 & \quad N=1 \quad \underline{0 \cdot \text{dp0}(1)} \\
 \hookrightarrow \text{dp0}(1) &= 1 \\
 \text{dp1}(1) &= 1
 \end{aligned}$$

Table Size: $dp[0] \Rightarrow N+1$ $dp[1] \Rightarrow N+1$

TC := $O(N)$, SC := $O(N)$

$$\text{Ans: } \text{dpo}(N) + \text{dp1}(N)$$

Depends only on previous 2 states

Spare Optimized : $O(1)$

HR:- Page 11 - Arranging Dominoes.

$$dp^c_i = dp^a_{i-1} + dp^b_{i-1}$$

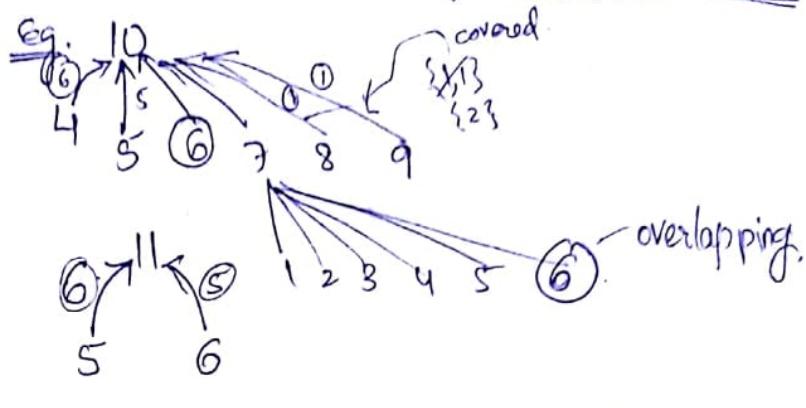
$$dp_1(i) = dp_0(p_1)$$

$$\begin{aligned}a &= c \\b &\in \emptyset\end{aligned}$$

21/4/19

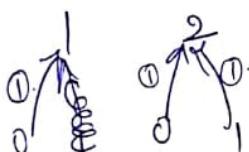
* Given a 6-sided dice, find the #ways to get a sum = N

$$\begin{aligned} \text{eg. } 5 &\rightarrow 8, 3 \\ &\rightarrow 1, 1, 1, 1, 1 \\ &\rightarrow 5 \end{aligned}$$



State: $dp(i)$ = #way to get sum = i

$$\underline{\text{dp exp:}} \quad dp(i) = \sum_{j=1}^6 dp(i-j) \quad i \geq 0$$



Base Condition : $dp(0) = 1$

Table Size :- $dp[N+1]$

S.C. = $O(N)$ = $O(N)$

T.C. = $O(N)$

Answer $\Rightarrow dp[N]$

Space Optimization: Array of size 7.

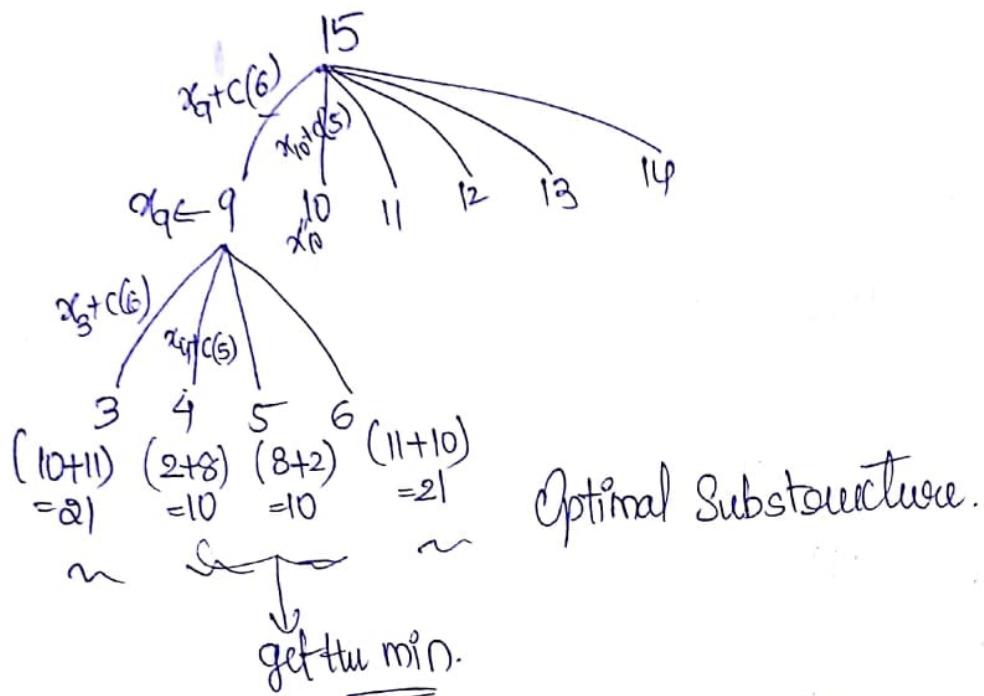
$$\begin{aligned} dp(i) &= \sum_{j=1}^6 dp((i-j) \% 7) \\ dp(i \% 7) &= \end{aligned}$$

$$dp(i \% 7) = \sum_{j=1}^6 dp((i-j) \% 7)$$

Base Conditions:

$$\begin{aligned} dp(0) &= 1 && \text{can be calc. from dp exp if we put } i, j \geq 0 \\ dp(1) &= 1 \\ dp(2) &= 2 \\ dp(3) &= dp(3-1) + dp(3-2) \\ &\quad + dp(3-3) \\ &= dp(2) + dp(1) + dp(0) \\ &= 2 + 1 + 1 \\ &= 4 \end{aligned}$$

- ④ Given a 6 sided dice, with costs, find the min cost to get a sum = N.
- $c(1) = 7$ Roll 3, 5 times $\Rightarrow 5 \times 10 = 50$ cost \rightarrow not min
- $c(2) = 3$ Roll 15, 3 times $\Rightarrow 3 \times 8 = 24$ not min
- $c(3) = 10$ Roll 4, 8 times and 3 once
- $c(4) = 2$ $(2 \times 3 + 10 = 16)$ ✓ min aus
- $c(5) = 8$
- $c(6) = 11$



for every dp state, we store the optimal substructure

dp state :- # ways to get sum = i

$$\text{dp exp: } \sum_{j=1}^N \min_{\substack{i-j \geq 0}} (\text{dp}(i-j) + c(j))$$

Base Condition: $\text{dp}(0) = 0$

Table Size: $\text{dp}[n+1]$

SC: $O(N)$, TC: $O(N)$

Answer: $\text{dp}[n]$

Space Optimization: Array of size 7 :-

$$\forall_{i=1}^N \text{dp}(i) = \min_{j=1}^6 (\text{dp}(i-j) + c(j)) \quad \text{if } i > 0$$

```
int dp[N+1];
```

$$dp[0] = 0$$

```
for i = 1 to N  
    dp[i] = ∞
```

for j = 1 to 6, $i-j \geq 0$

$$dp[i] = \min(dp[i], dp[i-j] + c[j])$$

}

Red Green Blue.

Q) Give N-houses. - Colouring each house in R, G, B costs are given

Paint all the N-houses using minimum cost such that two adjacent houses have

no same color.

N=5. 1 2 3 4 5

R: 5 10 3 12 1

G: 2 8 3 5 8

B: 7 16 12 7 8.

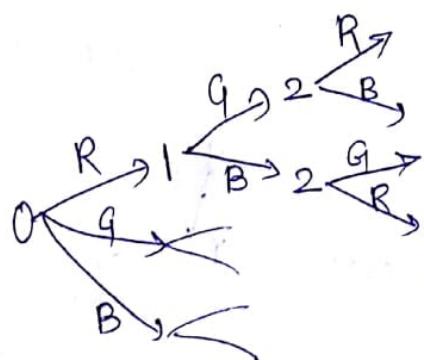
→ Greedy method cannot be applied all the times.

Q) Brute Force:- check All the possibilities $\Rightarrow \frac{1}{3 \text{ways}} \frac{2}{3} \frac{3}{3} \frac{4}{3} \frac{5}{3}$ houses

3^N .

Reduce some possibilities.

$\frac{1}{3} \frac{2}{2} \frac{3}{2} \frac{2}{2} \frac{2}{2}$



$f(i, c)$

if i th house is coloured with c .

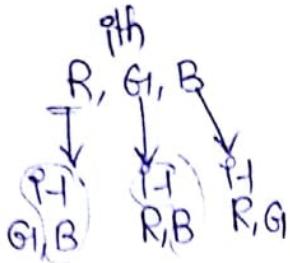
$\min\left(\frac{i+1, (c+1) \% 3}{i+1, (c+2) \% 3}\right)$ then $(i+1)^{\text{th}}$ house can be colored in $\min\left(\frac{(i+1, (c+1) \% 3)}{(i+1, (c+2) \% 3)}\right)$

$\Rightarrow 3^{42^{N-1}}$



Take $0R$ $1G$ $2B$ $C=0$

$\left(\begin{array}{l} (i+1, (c+1) \% 3) \\ (i+1, (c+2) \% 3) \end{array} \right)$



Overlapping Subproblems.

Optimal Substructure:

For painting N houses, we first check the min-cost till i th house.

$R(i)$ - costs for painting in red, $G(i)$ - Green, $B(i)$ - Blue.

dpState $dpR(i) \rightarrow$ min cost to paint i houses with i th house in Red.

$$\text{Exp} \quad dpR(i) = \min_{g1} (\underbrace{dpG(i-1)}_{g1}, \underbrace{dpB(i-1)}_b) + R(i) \quad g1 = \min(g1, b)$$

$dpG(i) \rightarrow$ min cost to paint i houses with i th house in Green

$$\text{Exp} \quad dpG(i) = \min_{g1} (\underbrace{dpR(i-1)}_{g1}, \underbrace{dpB(i-1)}_b) + G(i) \quad g1 = \min(g1, b) + G(i)$$

$dpB(i) \rightarrow$ min cost to paint i houses with i th house in Blue.

$$\text{Exp} \quad dpB(i) = \min_{b1} (\underbrace{dpG(i-1)}_{b1}, \underbrace{dpR(i-1)}_{g1}) + B(i) \quad b1 = \min(g1, b)$$

Indexed 1.

Base Conditions: $dpR(1) = R(1)$

$$dpG(1) = G(1)$$

$$dpB(1) = B(1)$$

$$dpR(0) = 0$$

$$dpG(0) = 0$$

$$dpB(0) = 0$$

$$g1 = g1$$

$$b1 = b1$$

$$g1 = g1$$

$$\text{ans} = \min(g1, b1)$$

Table Size: $dp[N+1]$

T.C., S.C. $O(3N), O(3N) \Rightarrow O(N), O(N)$

for $i=1$ to N

$$dpR(i) = ()$$

$$dpG(i) = ()$$

$$dpB(i) = ()$$

Space Optimization: Use Var,

Colours
 0 — Red
 1 — Green
 2 — Blue.
 mat $\begin{matrix} & i \\ \hline 0 & _ _ _ \\ 1 & _ _ _ \\ 2 & _ _ _ \end{matrix}$

int paint(int i, int mat[][], int c)

{

if ($i == 0$)

return 0;

return min (paint ($i-1$, mat, $(c+1) \% 3$) + mat [$(c+1) \% 3$] [i],
 paint ($i-1$, mat, $(c+2) \% 3$) + mat [$(c+2) \% 3$] [i]);

}

from Main:

ans = min (paint ($N-1$, mat, 0) + mat [0] [N],
 paint ($N-1$, mat, 1) + mat [1] [N],
 paint ($N-1$, mat, 2) + mat [2] [N]);

Memoize: the precalculated ones.

dp[i][c] \rightarrow house i , in colour c .

~~if (dp[i][(c+1) \% 3] == -1)~~

then $dp[i][(c+1) \% 3] = \text{paint } dp[i-1, m$

i denotes house number

c denotes the color in

which i^{th} pos house is painted

$\rightarrow (i-1)^{\text{th}}$ house should be

either $(c+1) \% 3$ or

$(c+2) \% 3 \rightarrow$ the min one.

houses indexed through 1

At 0, there is no house
return cost 0.

④ Given N jobs, 2 machines A, B and the costs associated with a job on the machines.

The switching cost from A-to-B or machine B-to-A is also given.
Find the minimum cost by which, we can complete all the N jobs.

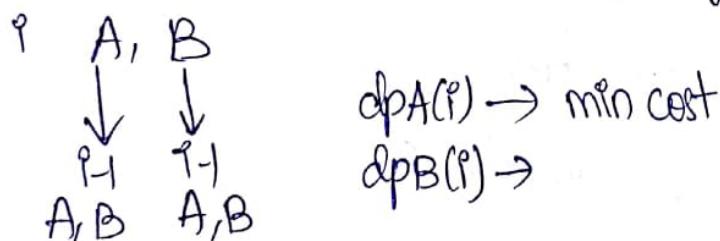
<u>Jobs:-</u>	1	2	3	4	5	6	7
Cost. A:	10	12	3	6	8	10	2
Cost. B:	2	5	10	7	1	5	17

— Greedy doesn't work always

eg.	A:	5	4	20	1	20
	B:	3	10	5	8	2

$$C.S. \quad 3+5+4+5+5 = 22 > 3+10+5 = 18.$$

Greedily: Taking min of switching \nearrow All on B,
without switching
Hence Greedy doesn't work.



* Max SubArray Sum :-

$\alpha[N] : 5 \ -1 \ 3 \ -10 \ 8 \ 3 \ 10 \ 6 \ -25 \ -3$

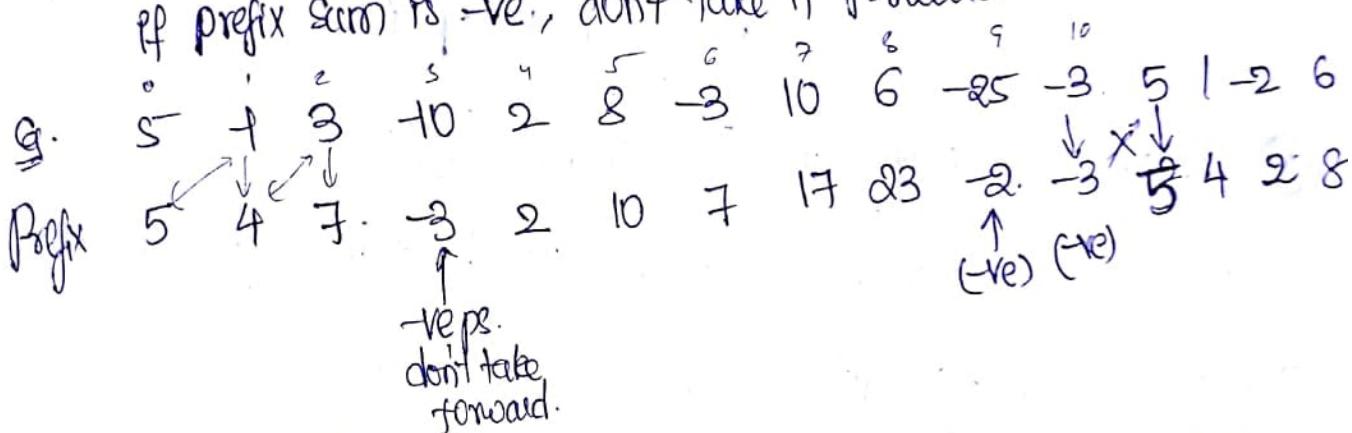
① Consider All Subarrays & find max sum

$$\Rightarrow O(N^2), O(1)$$

② Goay forward :- $O(N^2), O(1)$.

③ Calculate prefix sums.

If prefix sum is -ve, don't take it forward.



dp state :- $\forall i=1 \text{ to } N-1 \quad dp(i) \rightarrow \text{maximum subarray sum ending at } i \text{ including } i.$
 $= \max(dp(i-1), 0) + \alpha(i)$

$$dp(0) = \alpha(0).$$

Table Size :- $dp[N]$

Tc. Sc :- $O(N), O(N)$.

$$\underline{\text{Answer}} = \max_{i=0}^{N-1} (dp(i))$$

Spare Optimization :- $\underline{O(1)}$. $\text{ans} = \alpha(0), x = \alpha(0)$.

$$\forall i=1 \text{ to } N-1 \quad x = \max(x, 0) + \alpha(i)$$

$$\text{ans} = \max(\text{ans}, x)$$

* Max Subsequence Sum :-

① Brute force: sum of all +ve ele.s.

② Prefix Sum usage.

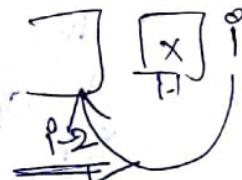
$$\text{arr: } 5 \ 1 \ 3 \ 10 \ 2 \ 8 \\ \text{dpst: } 5 \ 5 \ 8 \ 8 \ 10 \ 18$$

$\forall_{i=1}^{N-1} dp(i) = \text{maximum subsequence sum ending at } i$
 $= \max(\text{arr}(i), 0) + dp(i-1)$

* Max Non-Adjacent Subsequence Sum :-

- the chosen ele.s in the subsequence should not be adjacent.

$$\text{arr: } 5 \ 1 \ 3 \ 10 \ 2 \ 8 \ 3 \ 10 \ 6$$



$dp(i) \rightarrow \text{maximum non-adjacent subsequence sum till } i \text{ including } i$

$$dp[i]: \forall_{i=0}^{N-1} dp(i) = \max_{j=0}^{i-2} (dp(j)) + \text{arr}(i)$$

Base Conditions: No need. (works for all)

TC: $O(N^2)$

SC: $O(N)$

Ans: $\max(dp(i))$

Time Optimizations:

$$\forall_{i=0}^{N-1} m = \max(m, dp(i-2)) \\ dp(i) = m + \text{arr}(i)$$

Base Cond.

$$dp(0) = \text{arr}(0) \\ dp(1) = \text{arr}(1)$$

Spare Comp. Optimization: Use 3 variables a, b, c

$$\begin{aligned} a &= dp(0) = ar(0) \\ b &= dp(1) = ar(1) \end{aligned}$$

$$\begin{array}{l} m=0 \\ \forall i=2 \end{array} \begin{array}{l} m=\max(m, a) \\ c=m+ar(i) \\ a=b \\ b=c \end{array}$$

③ Change dp state.

$dp(i)$ = max nonadj subseq sum till i — may or
may not include i .

$$\forall i=2 \quad dp(i) = \max(ar(i) + dp(i-2), dp(i-1))$$

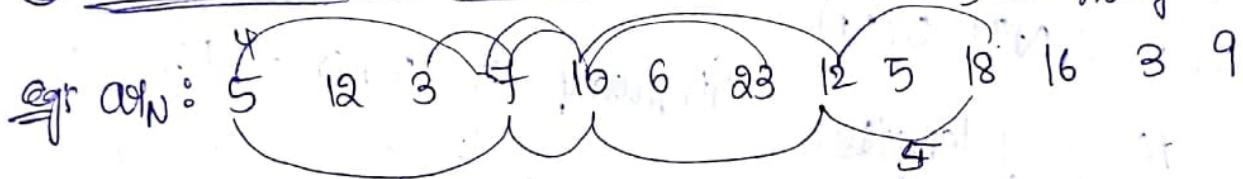
current taken
hence can't take
previous($i-1$),
should take ($i-2$)

If current not
taken, can take the
previous

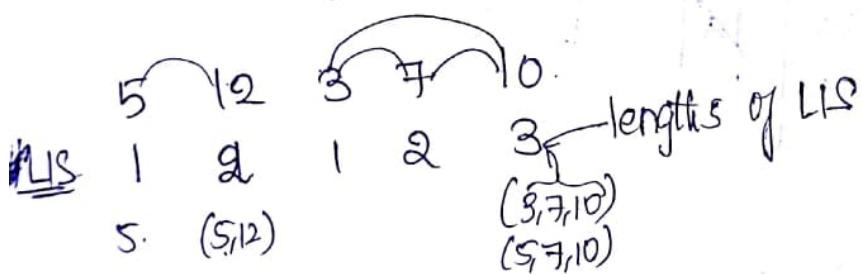
=

④ find the Length of Longest Increasing Subsequences (LIS).

5 — no. of LIS



Ans: 5. (length of LIS).



$\forall i=0 \quad dp(i) \Rightarrow$ length of longest LIS till i including i .

$$= \max_{j=0}^{i-1} (dp(j)) + 1$$

current ele
add length by 1

Table: $dp[i] = dp[i]$

$$TC: O(N^2), SC: O(N), Ans = \max_{i=0}^{N-1} (dp(i))$$

$T C = O(N^2)$
 N states
 for each state
 N time
 $\Rightarrow O(N^2)$

Space cannot be optimized as current state depends on
the previous state. and Time not opt

```

int maxLengthOfLIS ( int arr[] )
{
    int dp[N], ans=0;
    for( int i=0; i<N; i++ )
    {
        int m=0;
        for( int j=0; j<i; j++ )
        {
            if ( arr(j) < arr(i) )
                m= max(m, dp[j]);
        }
        dp[i]=m+1;
        ans= max(ans, dp[i]);
    }
}

```

3. return ans;

Compl:- $O(N^2)$, $O(N)$.

②. Array of lengths:-

Given array 5, 12, 3, 7, 10, 6, 23,
12, 5, 18, 16, 3, 9

Length of LIS	1	2	3	4	5	6	N
min ele of length	5	12	10	23	18	-	-
lengths array	3	7	9	12	16	-	-
arr	5	7	9	12	16	-	-
arr	5	7	9	12	16	-	-

for all the ele of array:-

First check for the cell of $arr(i)$ in lengths array.

If the cell exists, replace it with the current $arr(i)$.

If the cell is not present in lengths array, add $arr(i)$ to lengths array.

Compl:- $O(N \log N)$, $O(N)$

$\log N \rightarrow$ find ceil

To Do's :

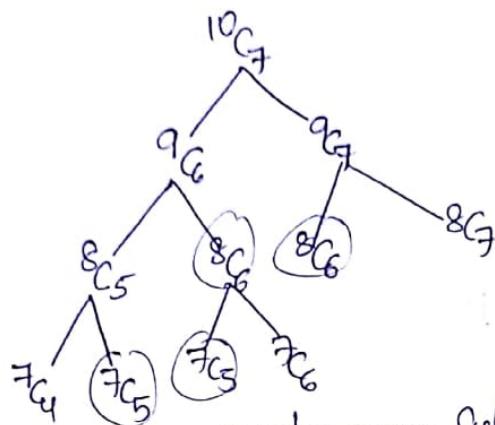
- 1) Memorization of Houses.
- 2) Printing LIS $\leftarrow N^2$ $N \log N$
- 3) #LIS (No. of LIS).
- 4) DP state of max non-dup subsequence.
- 5) N Jobs, Machines A, B.
- 6) No. of AP Subsequences.
- 7) No. of Sorted Subsequences.

~~$$N_{CR} = \frac{N!}{R!(N-R)!}$$~~

$S_3 = 10$

$$\Rightarrow N_{CR} = N-1 C_{R-1} + N-1 C_R$$

$$\begin{aligned}
 N-1 C_{R-1} + N-1 C_R &= \frac{(n-1)!}{(R-1)!(n-R)!} + \frac{(n-1)!}{S! (n-S)!} \\
 &= \frac{(n-1)!}{(R-1)!} \left\{ \frac{1}{(n-R)(n-R-1)!} + \frac{1}{S(n-S)!} \right\} \\
 &= \frac{(n-1)!}{(R-1)!(n-R)!} \left[\frac{1}{n-R} + \frac{1}{R} \right] \\
 &= \frac{n!}{n(n+1)}
 \end{aligned}$$



overlapping Subproblems

$dp(i,j) = \text{No. of ways we choose } j \text{ items from } i \text{ items} = {}^i C_j$

$$\forall i \in [N], \forall j \in [R] \quad dp(i, j) = \max_{k=1}^j \{dp(i-1, k) + \text{cost}(i, j)\}$$

Iterative

int dp[N+1][R+1];
 int f(dp);
 dp[0][0] = 1

1st row all zeroes except (0,0)
1st col all 1's.

```
for( i=1; i<=N; i++)
```

```
{   for(int j=1; j<=R; j++)
```

$$dp[i][j] = dp[i-1][j-1] + dp[i-1][j];$$

$$\text{Ans} = dp[N][R]$$

Space Opt.: $\sum_{i=1}^N \sum_{j=1}^N dp(i, j) = dp((P-1)\% 2, j-1) + dp((P-1)\% 2, j).$

—as we are using only the prev. slow.

Space of only 2 rows is req.

$$\Rightarrow SO(2N)$$

SC: O(NR)

$$TC = O(NR)$$

$$Ans = dp[N][R]$$

Recursive:

```
int f(int N, int R)
{
    if (R == 0)
        return 1;
    if (N < R)
        return 0;
    if (dp[N][R] == -1)
        dp[N][R] = f(N-1, R-1) + f(N-1, R);
    return dp[N][R];
}
```

Formula N_{CR} takes $O(N)$ time. $N_{CR} = \frac{N!}{R!(N-R)!} \% m$

But computing factorial goes out of range, so $\% m$.

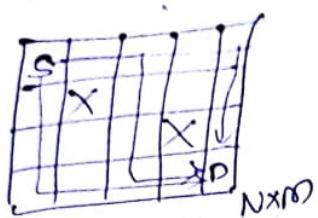
both in numerator & denominator, but again modulo
is not distributive over division. — ^{to do} inverse modulo.

So $N_{CR} = (N_{CR+} + N_{CR-}) \% m$. Is better

as we can reuse modulo and the values are also precomputed
so that the query requires only $O(1)$ time.

✓

Q) Find the no. of ways such that you go from source to Destination(D) without crossing any Blocked Cells(X)

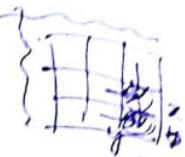


#ways = ?

S \rightarrow (0,0)

D \rightarrow (N, M)

X \rightarrow Blocked Cells.



Overlapping Subproblems.

$dp(i,j)$ \Rightarrow #ways to reach $(i,j)^{th}$ cell from source.

= 0, if (i,j) is blocked.

$$= dp(i-1, j) + dp(i, j-1)$$

Base Conditions:

$$\forall_{i=0}^N dp(i, 0) = 0$$

$$\forall_{j=0}^M dp(0, j) = 0$$

HR \rightarrow ST

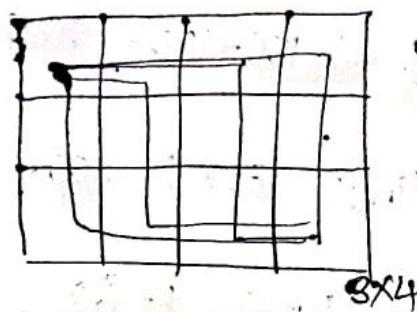
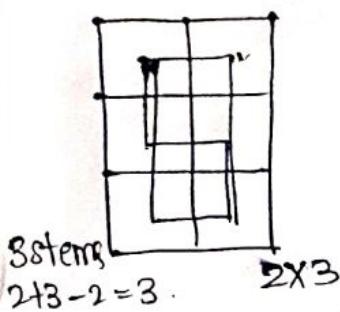
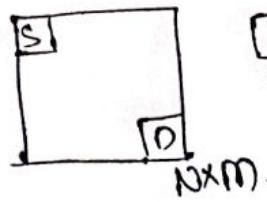
HR \rightarrow DP

DB \rightarrow DP

Total.

27/Jul/19

④ Always to reach from $s(0,0)$ to $D(N, M)$? (No. Blocked cells, formula?)

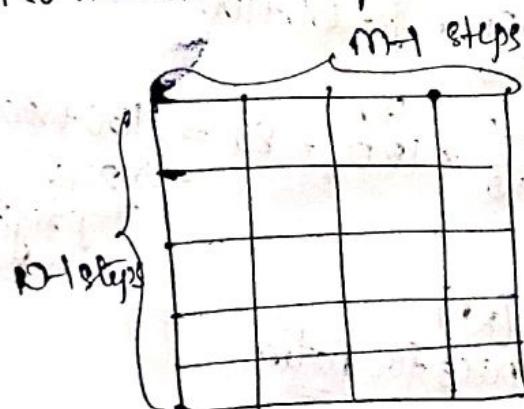


8 steps.

$$3+4-2 = 5.$$

$N \times M$ requires $N + M - 2$ steps.

No matter which path we take, it takes same no. of steps.



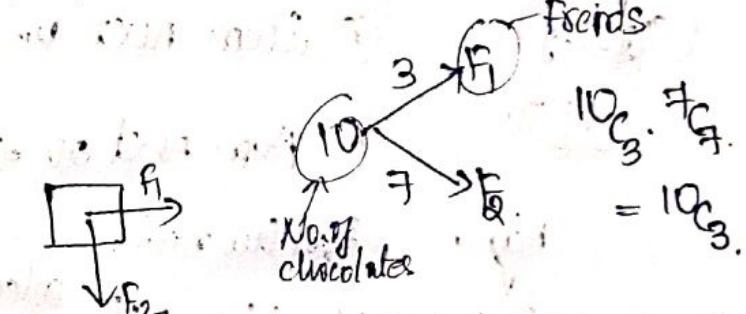
No. of chocolates

7 steps

$$\Rightarrow N + M - 2$$

$$\text{Ways} = \frac{N+M-2}{C_{N-1}} \quad \text{or} \quad \frac{N+M-2}{C_{M-1}}$$

~~$N+M-2 \in C_{N-1}$~~ ~~$N+M-2 \in C_{M-1}$~~



No. of chocolates

$$10 \in C_3 \quad 7 \in C_3$$

$$= 10 \in C_3$$

⑤ KnapSack Problem:-

Bag size e.g. $K = 50$

Maximize the value, such that the weight is $\leq k$.

Knapsack \rightarrow Fractional

Knapsack \rightarrow Integer \leftarrow 0-1 (Can take or not only once)

∞ (any no. of times)

Knapsack Problem:-

	①	②	③
N=3 weight $w(i)$:	20	10	30
Profit Values $V(i)$:	100	60	120.

Ratio $R(i) = \frac{V(i)}{w(i)} = 5 \quad \underline{\text{6}} \quad 4$

$$W = 20 + 30 = 50.$$

0-1 Knapsack Integer → 0-1 :- ① + ③ $\Rightarrow 100 + 120 = 220.$
Knapsack → 00 :- 10. W 5 times. $60 \times 5 = 300.$

Fractional :- ②. item first. $w=10, p=60.$

(Greedy) :- ① item next $w=10+20, p=60+100 = 160$
 $= 30.$

Ratio Sort the ratios $\frac{20}{30} \times \frac{40}{120} = 80 \Rightarrow 160 + 80 \Rightarrow 240$
Profit.

Complexity $O(N + N \log N + N^2)$ select

$O(N)$

~~Do it~~ Fractional Knapsack Code ~~Space for Ratio~~.

- Greedy approach cannot be applied for Integer as it will not always work.
- Maximizing & minimizing involves dp.

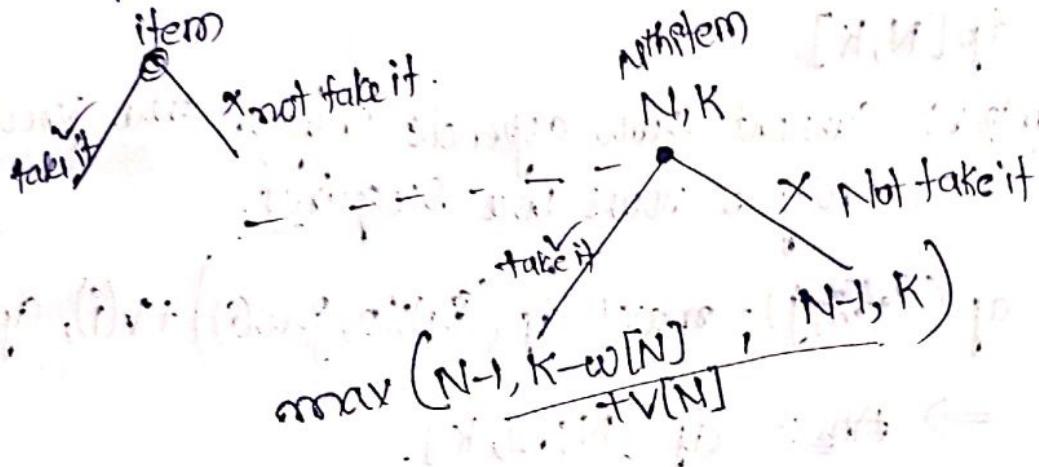
* Brute Force: Generate all the subsets & maximize the profit.

→ $O(2^N \times N)$

↑ Subsets ↑ checkbits.

In ~~Indigo~~ of
Greedy doesn't
work. dpv
ab.

- ① Overlapping Subproblems
 - ② Optimal Substructure (max. profit at each step).



(x, y)  knapsack.

$dp(i, j)$ = maximum profit to fill knapsack of size j with the first i items.

$\forall i \in [0, N], \forall j \in [0, K]$, $dp(i, j) = \max_{\substack{w(i) \leq w(j) \\ w(i) \neq w(j)}} \left(dp(i-1, j-w(i)) + v(i, j), dp(i, j) \right)$

i from l to N
j from o to M) benefit
 $\sum_{i=1}^N \sum_{j=1}^M K^{ij}$ $\forall dp(0,j) = 0$.

1 Based Index

1	2	3	4	5	6
0	0	0	0	0	0
2					

(N+1) rows

Space: $O(N \times k)$

$$(N+1)^*(K+1)$$

$$\text{dp}(i, j) = \max \left(\text{dp}(i-1, j-w(i)) + v(i), \text{dp}(i-1, j) \right)$$

$j \geq w(i)$ condition

S.C. $\Rightarrow O(N \times K)$

T.C. $\Rightarrow N \cdot K + O(1) \rightarrow O(N \cdot K)$ status.

Base Cond: 1 Based index

$$\sum_{j=0}^{K-1} \text{dp}(0, j) = 0.$$

Answer: $\text{dp}[N, K]$

Space Optimizes: Current row depends only on the previous row
 \rightarrow Only 2 rows are required.

$$\rightarrow \sum_{i=1}^N \sum_{j=0}^K \text{dp}(i \% 2, j) = \max \left((\text{dp}(i-1 \% 2, j-w(i)) + v(i)), \text{dp}(i-1 \% 2, j) \right)$$

\Rightarrow Ans: $\text{dp}[N \% 2, K]$.

Recursive: $\text{dp}[N, K] = \{ \}$; global N.Hemel
 int knapsack(int (N, K) inf w[], int v[], K - knapsack size.
 { if ($K < 0$) return -∞;
 if ($N == 0$) { for (int i=0; i<k; i++) if ($w[i] \neq v[i]$) return 0; }
 if ($\text{dp}[N, K] == -\infty$)
 {
 $\text{dp}[N][K] = \max \left(\text{knapack}(N-1, K-w[N]) + v[N], \text{knapack}(N-1, K) \right);$
 }
 return $\text{dp}[N][K];$

3

0 Based Index.

Basic Condition:

$$w(0)-1$$

$$\forall j \geq 0 \quad dp(0, j) = 0$$

$$\forall j \leq k \quad dp(0, j) = v(0)$$

$$(j=0)$$

0-Based

size ($N, k+1$)

	0	1	2	3	4	5	6	7
$w(i)$	3	6	5	2	4	7	9	11
$v(i)$	12	20	15	6	10			

$$N=5, k=7$$

0-Based

	0	1	2	3	4	5	6	7
w_i	0	0	0	12	12	12	12	12
x_1	0	0	0	12	12	12	20	20
x_2	0	0	0	12	12	15	20	20
x_3	0	0	6	12	12	18	20	22
x_4	0	0	6	12	12	18	20	22
x_5	0	0	6	12	12	18	20	22

$N \times (k+1)$

$$dp(2, 5) = \max(dp(1,$$

1-Based
Index.

	0	1	2	3	4	5	6	7
w_i	0	0	0	0	0	0	0	0
x_1	0	0	0	12	12	12	12	12
x_2	0	0	0	12	12	12	20	20
x_3	0	0	0	12	12	15	20	20
x_4	0	0	6	12	12	18	20	21
x_5	0	0	6	12	12	18	20	22

$N \times k+1$

6×8

$$w(i): \begin{matrix} ① & ② & ③ & ④ & ⑤ \\ 3 & 6 & 5 & 2 & 9 \end{matrix}$$

$$v(i): \begin{matrix} 6 & 20 & 15 & 6 & 10 \end{matrix}$$

$N=5$
 $k=7$

Infinity knapsack

K=80	w(i) 50	v(i)	40
	w(i) 200	v(i)	120
	w(i) 4	v(i)	3

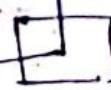
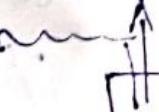
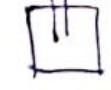
Not greedy. — If we take only 50. \rightarrow profit = 200.

But if we take 40 twice \rightarrow profit $120 \times 2 = 240$.

Greedy doesn't work, so dp.

Infinity knapsack can take the same thing multiple times.

So, it depends on previous value if current item is not taken, or else, if it is taken,

then, the current item can also ~~not~~  be taken again,  infinity knap 

so for the next item time,

choosing will be again from i items,

and not from $(i-1)$ items as the current item can be taken again. i.e. multiple times.

i items.

$$\Rightarrow \bigvee_{i=1}^N \bigvee_{j=0}^K dp(i, j) = \max \left(dp(i, j-w(i)) + v(i); dp(i-1, j) \right) \quad j \geq w(i).$$

$$\alpha dp(i, j - w(i)) + \alpha v(i)$$

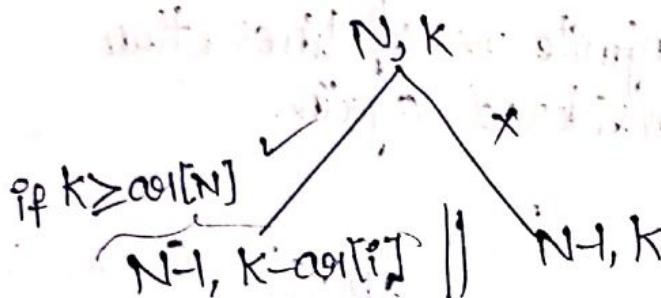
$$dp(i, j - w(i)) + \underline{\alpha v(i)}$$

- dp can be
- 1D array
- 2D Table
- Hash Map (key, value etc.)

④ COIN CHANGE

$\text{arr} = [10, 5, 13, 12, 6, 18, 25, 7]$
 $k=33$ then $(18+5+10)$.

N items, get the sum of k . 1 time each.



Boolean dp-matrix. [If either of two is true]

dp state: $dp(i, j) =$ if it is possible to get sum of j with first i coins.

dp expression: $\bigvee_{i=1}^N \bigvee_{j=0}^K dp(i, j) = dp(i-1, j - arr[i]) \parallel dp(i-1, j)$

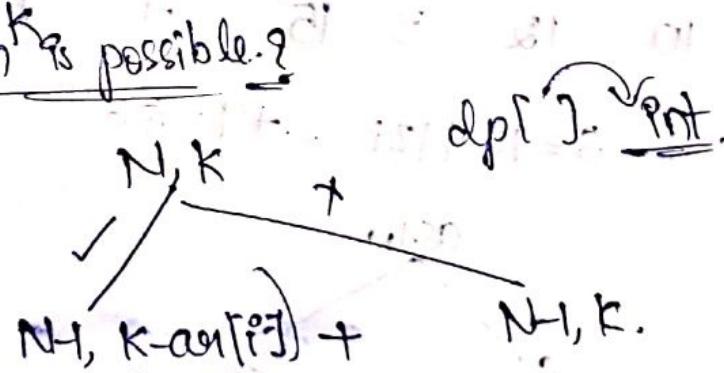
⑤ How many ways sum K is possible?

#ways:

$$1) 18+5+10=33$$

$$2) 12+6+5+10=33$$

$$3) \dots$$



⑥ Minimum no. of elements required to get sum K :

minimum no. of coins required to get sum K

$dp(i, j) = \min \left(\min \left(\begin{array}{l} N, K \\ N-1, K - arr[i] \end{array} \right) + 1, dp(i-1, j) \right)$

④ Min coins:

$dp(i, j) = \min$ no. of coins req. to get sum j from first i items.

$$\forall_{i=1}^N \forall_{j=0}^K dp(i, j) = \min \left\{ \begin{array}{l} dp(i-1, j - a[i]) + 1, \\ dp(i-1, j) \end{array} \right\}_{j \geq a[i]}.$$

To
carry
on

If every coin is present infinite no. of times, then current coin can be considered again.

a) K possible or not \Rightarrow ||

$$dp(i, j) = \min$$

b) #ways K possible \Rightarrow +

$$(dp(i-1, j - a[i]) + 1)$$

c) min coins. req. for $K \Rightarrow$ min()

$$(dp(i-1, j - a[i]) + 1)$$

* Given the array of integers, check if you can split the array into two parts such that $\text{sum}_1 = \text{sum}_2$.

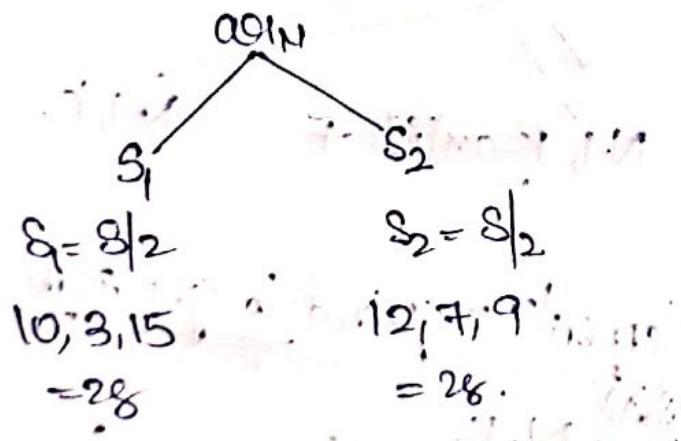
e.g. $a[N] = 10 \ 12 \ 3 \ 15 \ 7 \ 9$

$$\text{sum of all } S = 10 + 12 + \dots + 9 = 56.$$

2) sum diff \min
 $\min(|S_1 - S_2|)$

$$S = 56$$

$$S_1 = 28$$



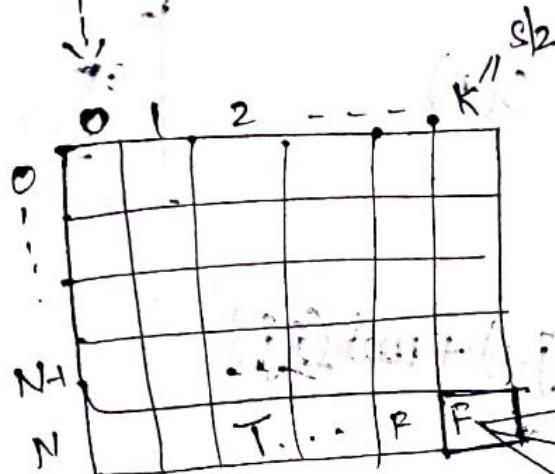
S2 Not possible for odd sums S_1 , so get the minimum difference of S_1 & S_2

$$K = \frac{S}{2} \text{ if even.}$$

$$\text{else } K = \frac{S}{2} - 1 \text{ if odd}$$

$$K = \frac{S}{2} - 2$$

$$K = \frac{S}{2} - 3$$



$$\min(S_1, S_2)$$

T/F:

$$\text{if True } S_1 = \frac{S}{2}, S_2 = S - S_1$$

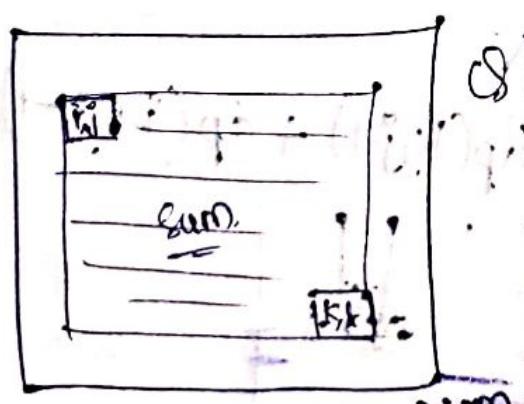
if false move to the left in the same row till you get true.

$$\text{here } S_1 = \frac{S}{2} - 1, S_2 = S - S_1$$

$$N \quad K = \frac{S}{2}$$

$$\forall i=1 \quad \forall j=0 \quad dp(i, j) = dp(i-1, j-\text{arr}(i)) \parallel dp(i-1, j).$$

- ④ Given top-left & right-bottom of matrix, get the sum of all elements present in the matrix:



i, j, k, l
 $0 \leq i \leq k \leq N$ → rows
 $0 \leq j \leq l \leq M$ → columns

①. Bottom force

$$s = 0;$$

$$\text{for } (x=i, x \leq k; x++)$$

$$\text{for } (y=j, y \leq l; y++)$$

$$s = s + m(x, y)$$

$$\text{sum}(\Omega \times N \times M), O(1)$$

② Prefix Sum on every row of matrix.

$s=0$.

$x = 0; x \leq k; x++$

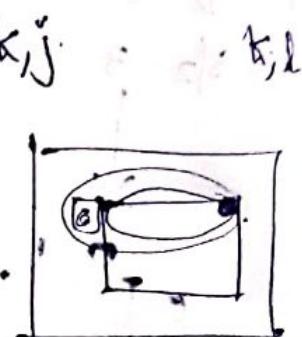
$$s = s + ps(x, l) - ps(x, j-1)$$

Complexity: $O((N \times m) + (O \times N))$, $O(1)$.

Prefix Row Sums

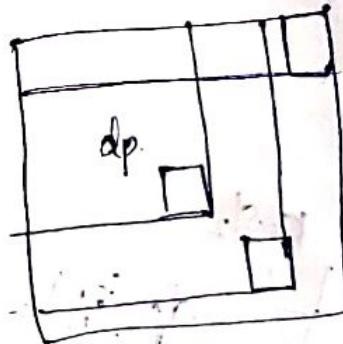
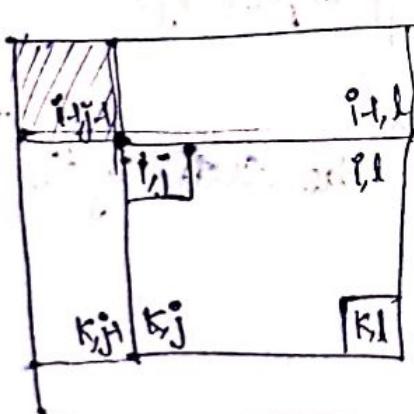
$i = 0 \text{ to } N$
 $j = 1 \text{ to } m$

$$ps(i, j) = ps(i, j-1) + mat(i, j).$$



③ Dynamic Programming

$$dp(i, j) = \sum_{x=0}^i \sum_{y=0}^j mat(x, y)$$

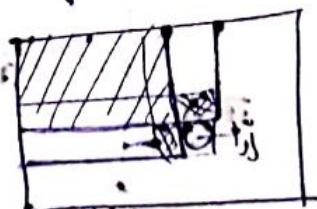


(Based Index) many base conditions req:

$$\text{ans} = dp(k, l) - dp(i-1, l) - dp(k, j-1) + dp(i-1, j-1)$$

1) Construct dp table optimally.

2) ans.



$$\text{DP Table: } \forall i, j, k, l \quad dp(i, j) = dp(i, j-1) + dp(i-1, j) - dp(i-1, j-1) + mat(i, j)$$

$$dp(i, j) = dp(i-1, j) + dp(i, j-1) + dp(i-1, j-1) + mat(i, j)$$

for Basic Conditions \Rightarrow Calc.
OBasic Index \Rightarrow

or Go with 1-Based Index.

dp[N+1][m+1]

Query $i, j, k, l \Rightarrow i+1, j+1, k+1, l+1$

$dp[N+1][m+1]$	0	1	2	\dots	m
0	0	0	0	0	0
1	0				
2	0				
3	0				
4	0				
5	0				
6	0				
7	0				
8	0				
9	0				
10	0				
11	0				
12	0				
13	0				
14	0				
15	0				
16	0				
17	0				
18	0				
19	0				
20	0				
21	0				
22	0				
23	0				
24	0				
25	0				
26	0				
27	0				
28	0				
29	0				
30	0				
31	0				
32	0				
33	0				
34	0				
35	0				
36	0				
37	0				
38	0				
39	0				
40	0				
41	0				
42	0				
43	0				
44	0				
45	0				
46	0				
47	0				
48	0				
49	0				
50	0				
51	0				
52	0				
53	0				
54	0				
55	0				
56	0				
57	0				
58	0				
59	0				
60	0				
61	0				
62	0				
63	0				
64	0				
65	0				
66	0				
67	0				
68	0				
69	0				
70	0				
71	0				
72	0				
73	0				
74	0				
75	0				
76	0				
77	0				
78	0				
79	0				
80	0				
81	0				
82	0				
83	0				
84	0				
85	0				
86	0				
87	0				
88	0				
89	0				
90	0				
91	0				
92	0				
93	0				
94	0				
95	0				
96	0				
97	0				
98	0				
99	0				
100	0				
101	0				
102	0				
103	0				
104	0				
105	0				
106	0				
107	0				
108	0				
109	0				
110	0				
111	0				
112	0				
113	0				
114	0				
115	0				
116	0				
117	0				
118	0				
119	0				
120	0				
121	0				
122	0				
123	0				
124	0				
125	0				
126	0				
127	0				
128	0				
129	0				
130	0				
131	0				
132	0				
133	0				
134	0				
135	0				
136	0				
137	0				
138	0				
139	0				
140	0				
141	0				
142	0				
143	0				
144	0				
145	0				
146	0				
147	0				
148	0				
149	0				
150	0				
151	0				
152	0				
153	0				
154	0				
155	0				
156	0				
157	0				
158	0				
159	0				
160	0				
161	0				
162	0				
163	0				
164	0				
165	0				
166	0				
167	0				
168	0				
169	0				
170	0				
171	0				
172	0				
173	0				
174	0				
175	0				
176	0				
177	0				
178	0				
179	0				
180	0				
181	0				
182	0				
183	0				
184	0				
185	0				
186	0				
187	0				
188	0				
189	0				
190	0				
191	0				
192	0				
193	0				
194	0				
195	0				
196	0				
197	0				
198	0				
199	0				
200	0				
201	0				
202	0				
203	0				
204	0				
205	0				
206	0				
207	0				
208	0				
209	0				
210	0				
211	0				
212	0				
213	0				
214	0				
215	0				
216	0				
217	0				
218	0				
219	0				
220	0				
221	0				
222	0				
223	0				
224	0				
225	0				
226	0				
227	0				
228	0				
229	0				
230	0				
231	0				
232	0				
233	0				
234	0				
235	0				
236	0				
237	0				
238	0				
239	0				
240	0				
241	0				
242	0				
243	0				
244	0				
245	0				
246	0				
247	0				
248	0				
249	0				
250	0				
251	0				
252	0				
253	0				
254	0				
255	0				
256	0				
257	0				
258	0				
259	0				
260	0				
261	0				
262	0				
263	0				
264	0				
265	0				
266	0				
267	0				
268	0				
269	0				
270	0				
271	0				
272	0				
273	0				
274	0				
275	0				
276	0				
277	0				
278	0				
279	0				
280	0				
281	0				
282	0				
283	0				
284	0				
285	0				
286	0				
287	0				
288	0				
289	0				
290	0				
291	0				
292	0				
293	0				
294	0				
295	0				
296	0				
297	0				
298	0				
299	0				
300	0				
301	0				
302	0				
303	0				
304	0				
305	0				
306	0				
307	0				
308	0				
309	0				
310	0				
311	0				
312	0				
313	0				
314	0				
315	0				
316	0				
317	0				
318	0				
319	0				
320	0				
321	0				
322	0				
323	0				
324	0				
325	0				
326	0				
327	0				
328	0				
329	0				
330	0				
331	0				
332	0				
333	0				
334	0				
335	0				
336	0				
337	0				
338	0				
339	0				
340	0				
341	0				
342	0				
343	0				
344	0				
345	0				
346	0				
347	0				
348	0				
349	0				
350	0				
351	0				
352	0				
353	0				
354	0				
35					

$$\underline{O(N \times m)} + \underline{O(S \times 1)} \neq O(N+m, S \times 1)$$

→ DP Table can also be constructed by row wise prefix sum and column wise suffix sum.

④ Find the maximum SubMatrix Sum

- ① Brute matrix: for each sub-matrix find the sum, get max sum

N^4 submatrices.

find sum in O(1)

$$j=0 \text{ to } m$$

$$k = i \text{ to } N$$

$i=j$ to m

Complexity: $O(N^4)$, $O(1)$ space

\rightarrow $\delta(\text{eff})$.

- ② Kadane's Algo.

If you're, don't forward.

else reward it

max subarray sum

and is max.

$$2 \ 3 \ -5 \ -10 \ 3 \ -2$$

2 5 0 . 40 3 . 1

1

Kadane's
Use Kadane's algorithm in 2D considering column sum.
To get the column sum in $O(1)$ → maintain column wise prefix sum.

	j					
i	3	-1	8	-4	3	1
	10	-25	31	15	-1	2
	-8	9	10	-3	5	3
k	7	2	-5	8	2	4
	1	4	-7	3	1	5

NxN.

$$\begin{matrix} 9+9+2 \\ \downarrow \\ 25 \end{matrix} \quad \begin{matrix} 0(1) \\ q \quad s \quad 8 \quad 28 \quad 34 \end{matrix}$$

don't forward.

int maxSubMatrixSum (int ps[], int mat[][]) {

for i = 0 to N.

K = i to N

s = 0

j = 0 to M

$$s = s + ps(k, j) - (i! = 0) ? ps(i-1, j) : 0;$$

$$ans = \max(ans, s)$$

$$\text{if } (s < 0) \Rightarrow s = 0;$$

$$(O) s = \max(s, 0).$$

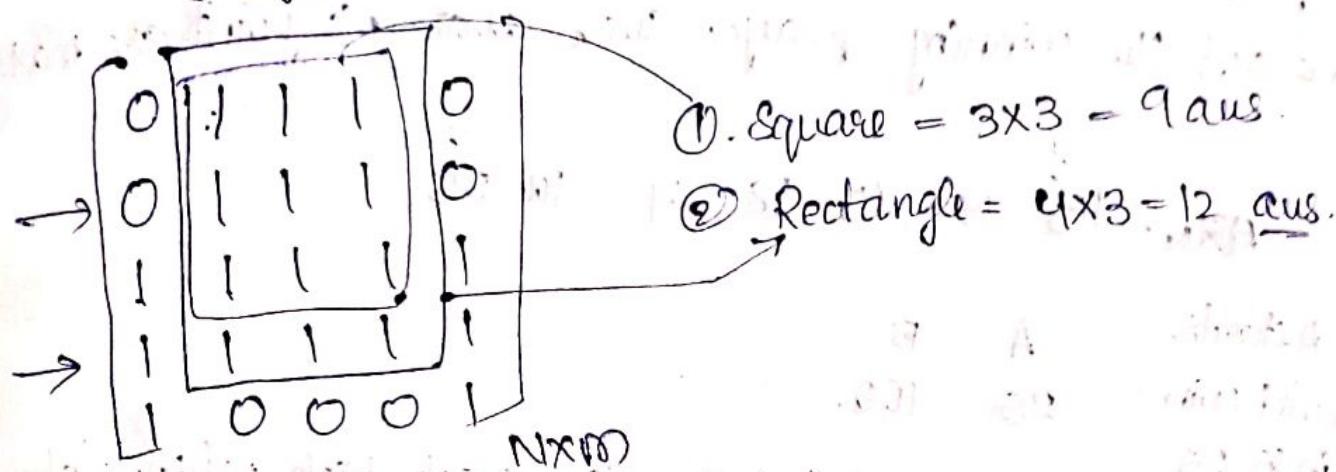
3 return ans;

(Exp:- $O(N^2 \times m), O(1)$)

Ps → column

Prefix sum

- * ① Find the largest square submatrix which contains only 1's.
- ② Largest Rectangle sub matrix, which contains only 1's.



① Fix 2 rows to use Kadane's Algorithm -

$O(N^2m)$, $O(1)$.

② Optimal: Do it.

asylia

- Q: Two players A and B, a player should pickup elements only from edges one at a time. Find out the winning player such that his points are max.

Ex: $A_N = 15 \ 3 \ 10 \ 18 \ 7 \ 100 \ 25$

Answers.

A B.

A should win

25 100.

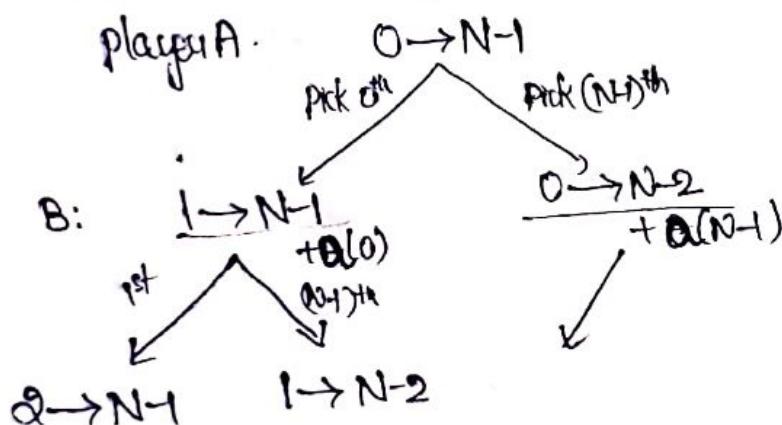
but in 15, 25

If A greedily picks 25, B is left with 100 to pick which makes A loose the game.

Hence the greedy doesn't work.

Go with Dynamic Prog.

A array



$dp(i, j)$ = max sum a player can get from i to j index.

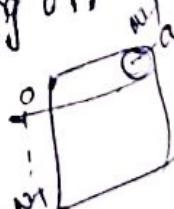
= maximum sum for 1st player, playing on subarray $[i, j]$

for one player

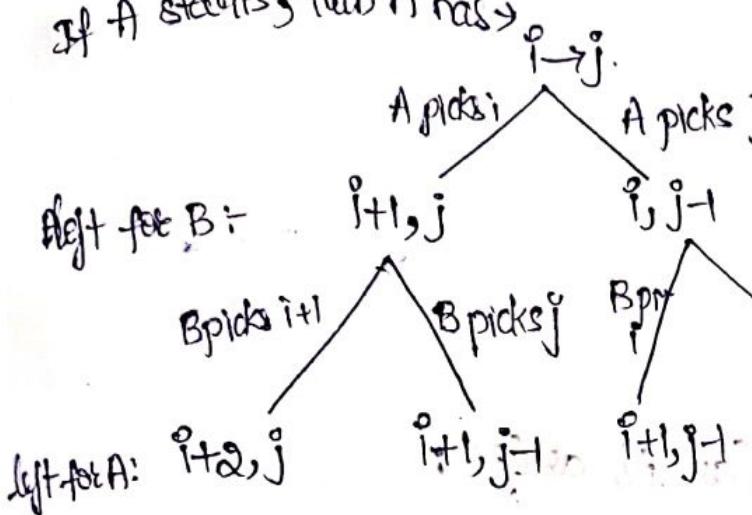
$$S_A = dp(0, N-1)$$

$$S_B = \text{TotalSum} - S_A$$

$\max(S_A, S_B)$ will be the winning player.

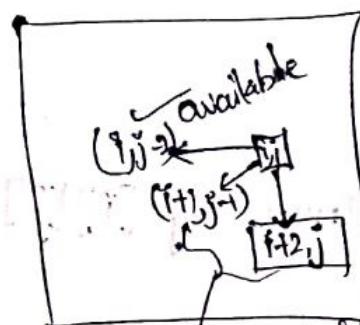
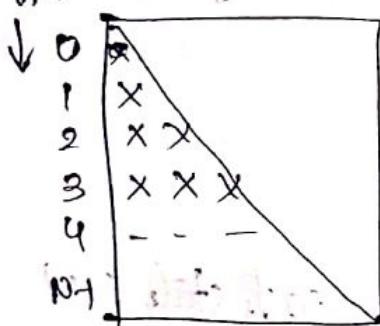


If A starts, then A has \overrightarrow{ij}



For A to be max,
B should be min.

$$dp[i, j] = \max \left\{ \begin{array}{l} \alpha(i) + \min(dp[i+2, j], dp[i+1, j-1]) \\ \alpha(j) + \min(dp[i+1, j-1], dp[i, j-2]) \end{array} \right\}$$



not available if we go top to bottom
hence go bottom to top is
filling up array.

Base Conditions: i) fill all 1 array ele. with the ele. itself.

length & & length are not valid

as we need $i+2, i+1, i-1, i-2$

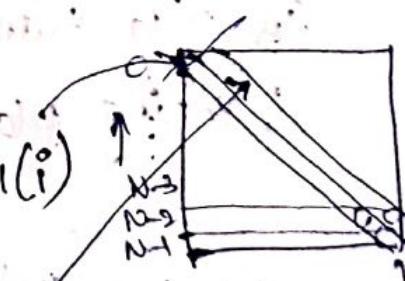
2) fill all α length subarray

ele. with the max of two. \Rightarrow 2)

j79. Pterative

$$\forall i \in N \quad dp(i, i) = \alpha(i)$$

$$\forall_{i=0}^{N-2} dp(i, i+1) = \max(\alpha_i(i), \alpha_i(i+1))$$



$$dp(i, j) = \max \begin{cases} \text{ari}(j) + \min(dp(i+2, j), dp(i+1, j-1)) \\ \text{ari}(j) + \min(dp(i+1, j-1), dp(i, j-2)) \end{cases}$$

$\text{dp}(P, j) =$

Recursive:

int f(int i, int j)

{ if ($i == j$)

return arr[i];

if ($j == i+1$)

return max(arr[i], arr[j]);

if ($dp[i][j] == -1$)

$dp[i][j] = \max \begin{cases} arr[i] + \min(f(i+2, j), f(i+1, j-1)), \\ arr[j] + \min(f(i+1, j-1), f(i, j-2)) \end{cases}$

}

return dp[i][j];

f

N^2 states, each state $O(1)$

$\Rightarrow O(N^2)$.

$$S_A = f(0, N-1)$$

$$S_B = \text{Total Sum} - S_A$$

Winning player \Rightarrow has $\max(S_A, S_B)$

Matrix Multiplication

$$A_{5 \times 3} * B_{3 \times 8} = R_{5 \times 8}$$

1	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0
1	0	0	0	1	0	0	0
0	0	1	0	0	1	0	0
0	0	0	0	0	0	1	0

1	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0
1	0	0	0	1	0	0	0
0	0	1	0	0	1	0	0
0	0	0	0	0	0	1	0

5×8 , 40 ele.
each ele. takes 3 bytes
 $\Rightarrow 40 \times 3 = 120$ total bytes

$$i) \underbrace{(A_{5 \times 3} \cdot B_{3 \times 8})}_{5 \times 3 \times 8 \text{ opr}} \cdot C_{8 \times 7}$$

$$+ \underbrace{(AB_{5 \times 8}) \cdot C_{8 \times 7}}_{5 \times 8 \times 7 \text{ opr}}$$

$$\rightarrow 5 \times 3 \times 8 + 5 \times 8 \times 7 = 120 + 280 = \underline{\underline{400 \text{ opr}}}$$

$$ii) A_{5 \times 3} \cdot \underbrace{(B_{3 \times 8} \cdot C_{8 \times 7})}_{(5 \times 3 \times 7) + (3 \times 8 \times 7) \text{ opr}}$$

$$\begin{array}{r} \frac{1}{168} \\ \frac{1}{168} \\ \hline \frac{1}{273} \end{array}$$

$$105 + 168$$

minimum iterations = 273 opr

Matrix Chain Multiplication:

- Given an array of size N with (N-1) matrix orders.
Find out the minimum no. of iterations required to multiply all the matrices.

e.g. $A_{N=7} : \underbrace{5}_{①} \underbrace{3}_{②} \underbrace{8}_{③} \underbrace{7}_{④} \underbrace{10}_{⑤} \underbrace{3}_{⑥} \underbrace{8}_{⑦} \underbrace{6}_{⑧}$ Matrix Orders:

$A: a_{ij} : \underbrace{5}_{①} \underbrace{3}_{②} \underbrace{8}_{③} \underbrace{7}_{④} \underbrace{10}_{⑤} \underbrace{3}_{⑥} \underbrace{8}_{⑦} \underbrace{6}_{⑧}$ ① 5x3
 ② 3x8
 ③ 8x7
 ④ 7x10
 ⑤ 10x3
 ⑥ 3x8

$\min \left\{ \begin{array}{l} [0,5] + [5,6] + A(0) * A(1) * A(6) \\ \quad \quad \quad \text{cost of multi. from 0 to 5.} \\ \quad \quad \quad K \text{ goes from } i \text{ to } j-1 \\ [0,6] + [6,7] + A(0) * A(2) * A(6) \\ \quad \quad \quad 5x3 \quad 3x8 \\ [0,4] + [4,5] + A(0) * A(5) * A(6) \\ \quad \quad \quad 5x3 \quad 3x8 \end{array} \right\}$

$dp(i,j)$ = minimum iterations required to multiply matrices $A_{i,j}$. (cas)

$$\min_{0 \leq i \leq j < N} dp(i,j) = \min_{K=1}^{j-1} [dp(i,K) + dp(K+1,j) + A(i) * A(K+1) * A(j)]$$

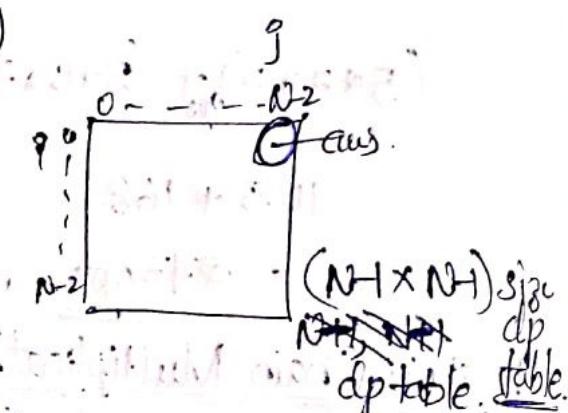
Iterative: Base Condition $\Rightarrow dp(i,i) = 0$

Recursive: Main func. call $f(i,j) \approx f(0, N-2)$

N^2 -states, each state takes $O(N)$

$\Rightarrow O(N^3)$ Complexity.

Ans. is $dp[0, N-2]$



More DP Problems:

1) Building Bridges.

2) Stacking Boxes

3) Bitonic Subsequence

4) LCS (Longest Common Subsequence)

5) LCSt (Longest Common Substring)

6) EDIT DISTANCE (EDIST) \leftarrow convert A to B \min no. of steps.

7) Rod Cutting

8) Palindrome Partitioning (min. no. of partitions.)

LIS

(Longest

Increasing

Subsequence)

$O(N^2)$.

FF-{i,j} in $O(1)$

$\frac{1}{2}(det + dp) + 1$

MCPO
(Matrix
Chain
Mul)

$$5 * 3 - 2 * 8 - 5$$

$$15 - 16 - 5 = \underline{-6}.$$

$$5 * (3 - 2) * (8 - 5)$$

$$5 * 1 * 3 = 15$$

q) Parenthesise exp such that ans. of arithmetic exp is maximum.

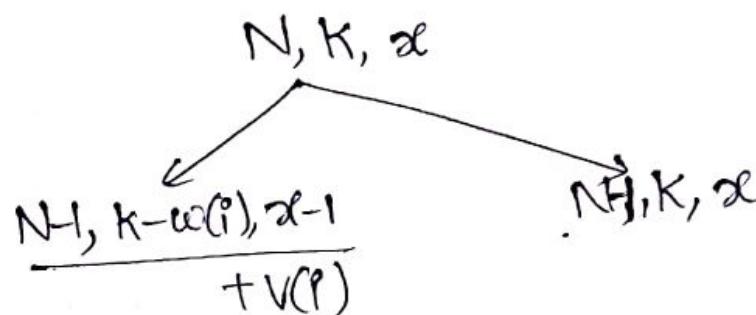
10) No. of ways to fill Blanks \square such that the logical exp. results True:

$$\boxed{?} \& \boxed{?} \parallel \boxed{?} \& \boxed{?} == T.$$

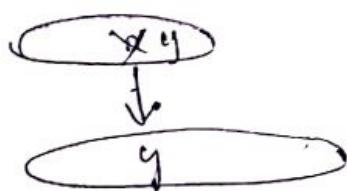
* Additional Constraint in knapsack:-

Not to take more than x -items.

then dp state is like.



* Edit Distance Problem:-



Replacing Cost - C_R

Inserting Cost - C_I

Deleting Cost - C_D

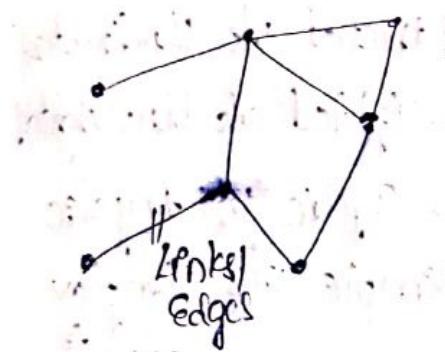
minimizing the ^{total} cost.

& given that first K operations can be done for free.

11) Word WRAP.

GRAPH THEORY.

①
Nodes / Vertices



- ①
 → Undirected
 → Cyclic
 → Disconnected
 → Unweighted
 → Simple
 (no self loops,
 no multi-edges.)

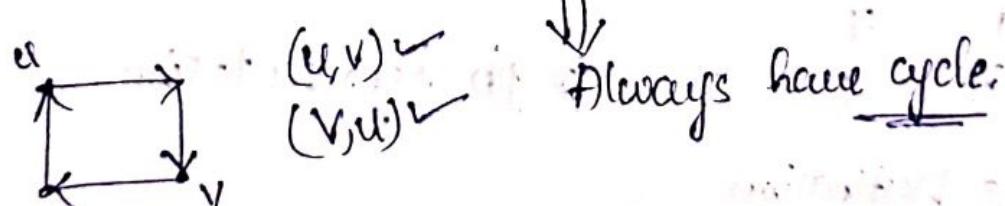
- VS. Directed
 VS. Acyclic
 VS. Connected
 VS. Weighted
 VS. Complex

②

Directed
 Weakly Connected G.
 Strongly Connected G.

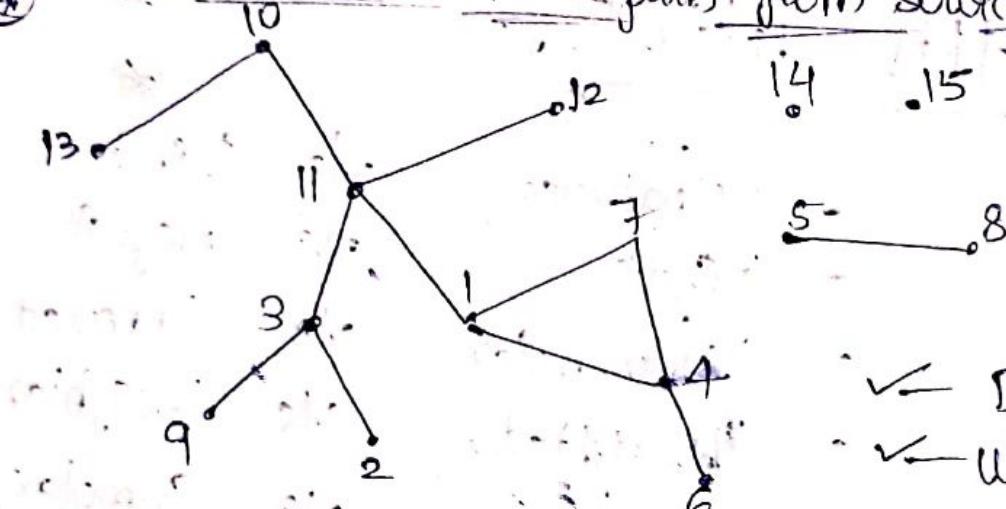
connected graph: vertices
 Every pair (u, v) should have path.

- Directed graph having undirected paths b/w every pair of vertices is a Weakly Connected Graph.
- Directed Graphs having directed paths b/w every pair of vertices is a Strongly Connected Graph.



Always have cycle

* Check if there exists a path from Source to Destination



SIP N M
13 Nodes, 12 Edges

- Given in Problem
- ✓ → Directed v/s Undirected
 - ✓ → Weighted v/s Unweighted
 - ✗ → Algo -acyclic v/s cyclic
 - ✗ → Simple v/s Complex
- (u, v) if
- 1) $u = v$
 - 2) (u_1, v_1) $u_1 = u_2$
 (u_2, v_2) & $v_2 = v_1$

<u>u</u>	<u>v</u>	w (weight)	→ may (may not)
1	7	5	
6	4	10	$G = (V, E)$
11	12		$N = V $
3	11		$M = E $
5	8		
10	13		
11	10		
1	11		
9	3		
3	2		
1	4		
7	4		

Q. How do we represent graphs in Memory?

1) Adjacency Matrix

2) Adjacency List

Graph Representation.

Source Destination

S	D	<u>Ans.</u>
11	6	true
14	15	false.

①. Adjacency Matrix:

int $G1[N+1][N+1] = \{0\};$

(or)

Boolean Matrix for undirected graph.

$G1[u][v] = 1;$ } undirected graph.

$G1[v][u] = 1;$ } will have path from u to v, & v to u.

If weighted undirected graph:

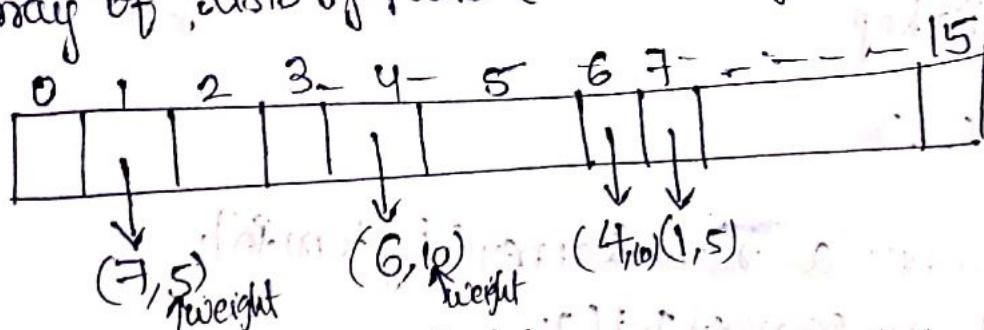
$G1[u][v] = w;$

$G1[v][u] = w;$

②. Adjacency List:

Array of lists of pairs (vertex, weight).

Array.



$G1[u].push(v);$

$G1[u].push(u);$

③. Undirected Unweighted - Graph : Adjacency Matrix

class GraphsAdjMatrix

{

psvm.c)

{ Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int m = sc.nextInt();

int[][] mat = new int[n+1][n+1];

```

for (int i=0; i<m; i++)
{
    int u = sc.nextInt();
    int v = sc.nextInt();
    mat[u][v] = 1;
    mat[v][u] = 1;
}

```

⑤. Undirected ~~Weighted~~ Graph : Adjacency List.

class. AdjListRep (N*2M) space

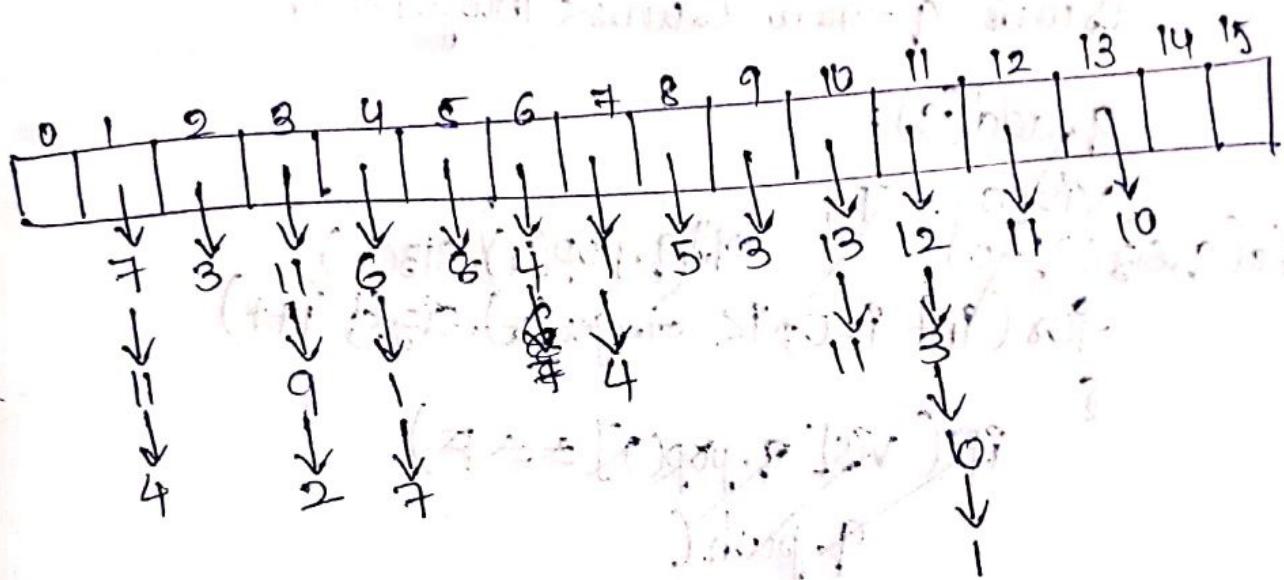
perm(...)

```

Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
int m = sc.nextInt();

ArrayList<ArrayList<Pair<Integer, Integer>> G
    = new ArrayList<>();
for (int i=0; i<n; i++)
{
    ArrayList<Pair<Integer, Integer> a
        = new ArrayList<>();
    for (int j=0; j<m; j++)
    {
        int u = sc.nextInt();
        int v = sc.nextInt();
        int w = sc.nextInt();
        a.add(new Pair<Integer, Integer>(v, w));
        a.add(new Pair<Integer, Integer>(u, w));
    }
    G.add(a);
}

```

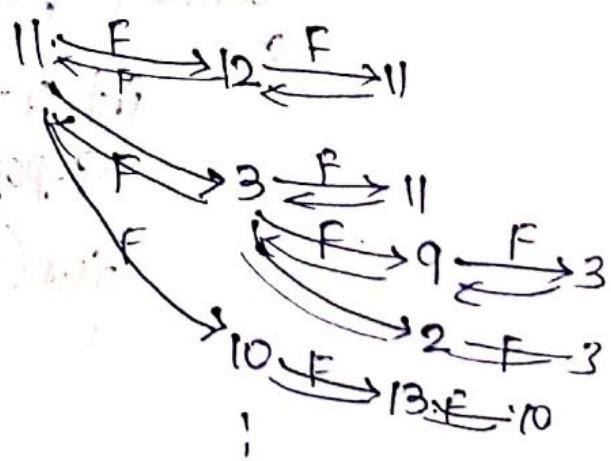


- Start from source look for Destination.

Graph Traversals:

- 1) Bfs (Breadth first Search)
 - 2) Dfs (Depth first Search)

BFS: $\text{bfs}(\text{vis}[N+1] = \{\})$;



Quest: ✓ 12, ✓ 10, 1, 13.

while pushing it, mark it as Vis, = T., remove it, push its neighbor
mark Vis.

bool BFS (ArrayList < ArrayList < Pair < Integer, Integer > > g,
int s, int D)

{

int N = G1.size();

bool[] vis = new bool[N];

bool vis[N] = { F };

Queue<Integer> q = new Queue<Integer>();

q.add(s);

vis[s] = T;

while (q.size() != 0):
for (int i=0; i < G1.get(u).size(); i++)

{

if (vis[q.pop()] == F):

q.push(

Queue < int > q;

q.push(s);

vis[s] = T;

while (q.size() > 0):

int u = q.front();

q.pop();

if (u == D) return T;

for (int v : G1[u])

{

if (vis[v] == F)

{ q.push(v);

vis[v] = T;

return vis[D];

BFS

$$N = |V| \\ M = |E|$$



BFS: $O(N+m)$, $O(N+m)$
queue visar

DFS



BFS: $O(|V|+|E|)$, $O(|V| \cdot N)$

DFS: $O(|V|+|E|)$, $O(|V|)$

Complexity: $O(N+m)$, $O(N)$

Visiting
Nested
dig

bool DFS(G1, int s, int D, bool vis[]);

{ if (vis[s] == T)

return F;

if (s == D)

return T;

vis[s] = T;

for (int v : G1[s])

{ if (DFS(G, v, D, vis) == T)

return T;

}

return F;

};

};

};

};

};

};

};

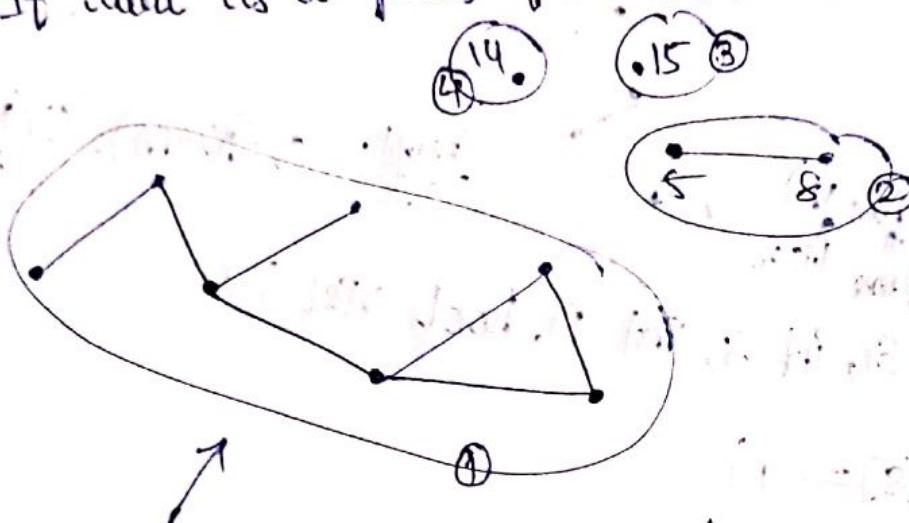
};

};

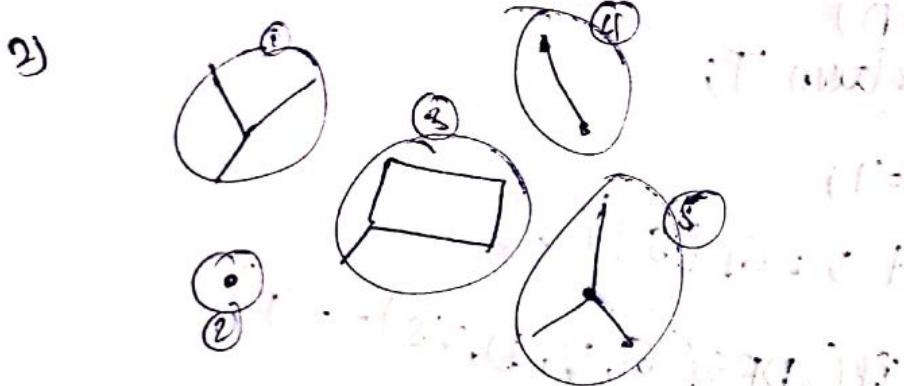
};

④ Count the no. of connected Components:

— If there is a path from (u, v) to $u \rightarrow v$, then $u, v \in$ same component.



→ Ans:- 4 Connected Components!



Ans:- 5 Connected Components.

Use DFS | BFS only to mark all the connected nodes as visited in the visited array.

```

bool vis[N+1] = {F};

int countComponents ( - . G)
{
    int count=0;
    for( int i=1; i<=N; i++)
    {
        if( vis[i] == F )
        {
            count++;
            DFS( G, i, vis);
        }
    }
    return count;
}

```

Modify DFS accordingly to just mark the visited nodes of same component as TRUE.

We don't call DFS for all vertices.

⇒ Complexity: $O(|V| + (|V| + |E|))$

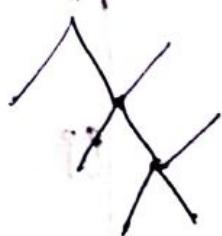
* Check if the given undirected & unweighted graph is a tree:

Graph is a tree —

when i) It is connected. [It shouldn't be disconnected]

& ii) No cycles in the graph.

Eg: 1)



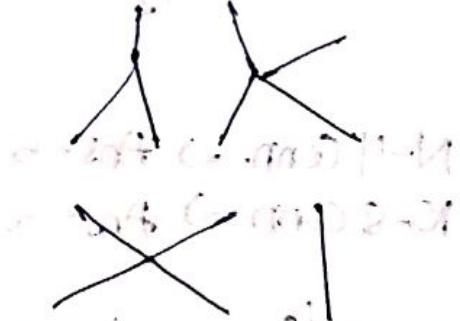
Tree ✓

2)



Tree X as Cycle is present.

3)



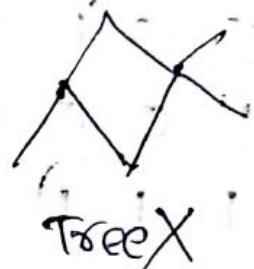
Tree X

Not connected.

(Count of conn. compn's > 1)

Forest Tree ✓ (Each comp. is tree)

4)



Tree X

i) Connected : # connected components = 1

& ii) Acyclic : $|E| = |V| - 1$

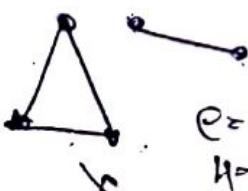
$$m = N - 1$$

(i.e., No. of edges = No. of vertices - 1)



$$V = 5, E = 4$$

2) ✓ 1) X



$$e = n - 1$$

$$4 - 5 - 1$$

$n = 4 \neq 5$ But not connected.

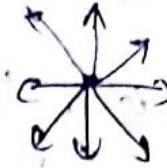
* for forest tree
#CC > 1

Add 1 edge more makes it cyclic.

④ Count the no. of islands:

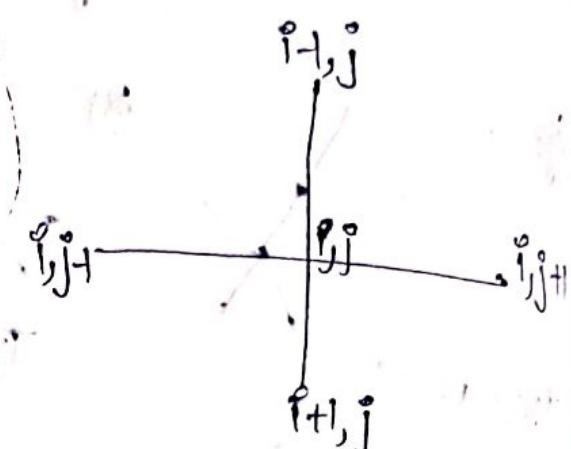
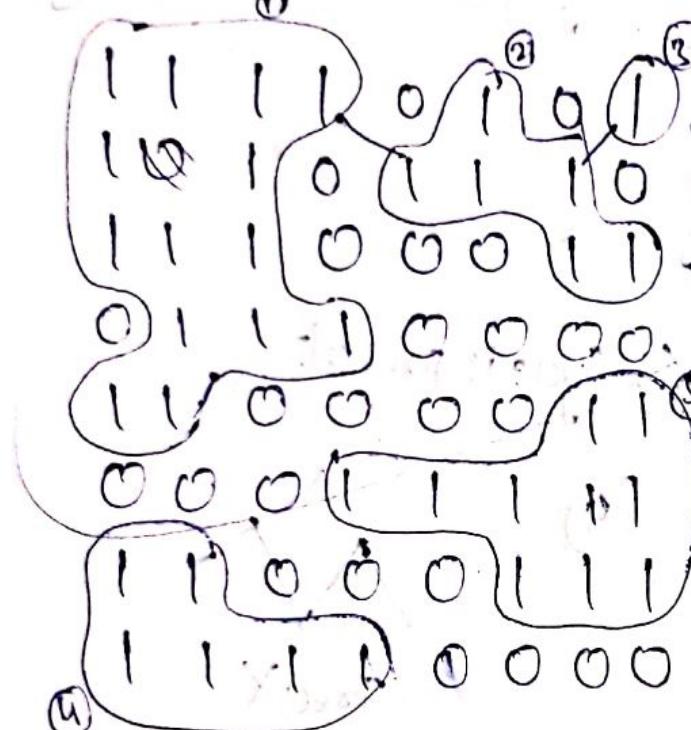
O - water
I - land.

N-4 Connectivity



N-8

• Connectivity



N-4 Conn. \Rightarrow fns = 5

$$N=8 \text{ conn} \Rightarrow \# \text{ways} = 3.$$

count the no. of connected components.

for masking
visited.
 \Rightarrow make
1 as 0.

int noOfIslands (int mat[][])

```
{ int count=0;
```

for ($i=0$; $i < N$; $i++$)

{ for(j=0; j < m; j++)

if (mat(i)[j] == -1)

{ count++;

DFS(i, j, mat, N, m)

3

return count

void DFS(int i, int j, int mat[][], int N, int m) first dir), int dj[]).

{ if ($i < 0 \text{ || } i > N \text{ || } j < 0 \text{ || } j > m \text{ || } mat[i][j] == 0$)

return;

$mat[i][j] = 0$

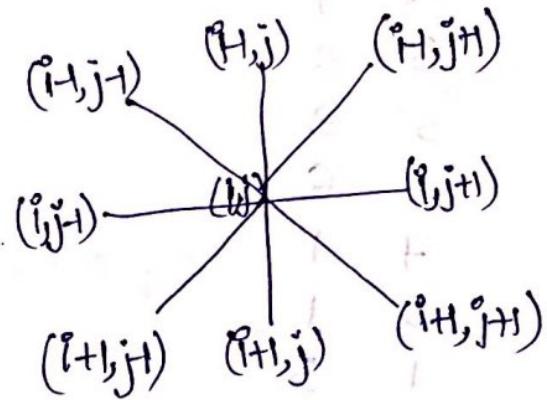
for (int k=0; k<8; k++)

DFS (mat, i+di[k], j+dj[k], N, m)

}

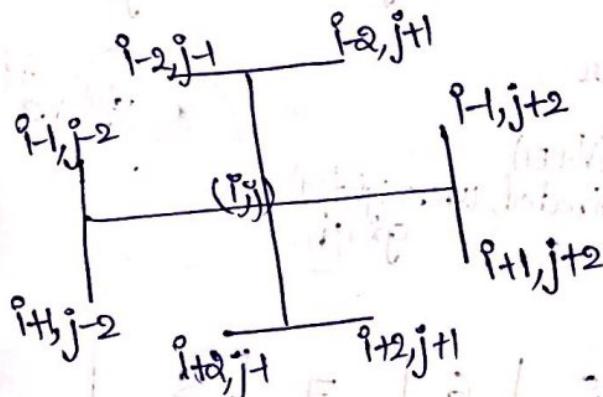
global
di[] = { -1, 1, -1, 0, 1, 1, 1, 0 }

dj[] = { -1, 0, +1, 1, 1, 0, -1, -1 }



$\Rightarrow O(N \times M)$

④ check if knight can move to all the cells except the blocked chess (horse) After DFS, check if the whole mat[] becomes



1	1	1	1	0	1	0
0	0	1	1	1	1	1
1	1	0	0	1	1	1
0	1	0	1	0	1	0
0	1	0	1	1	0	1
0	1	1	1	1	1	1

If yes ✓
If No X

Nxm

0 \Rightarrow Blocked

S \Rightarrow Knight's position

Space Complexity = O(1)

di[8] = { -2, -2, -1, 1, 2, 2, 1, -1 }

dj[8] = { -1, 1, 2, 2, 1, -1, -2, -2 }

void DFS(int i, int j, int mat[][], int N, int m, int di[], int dj[])

{ if ($i < 0 \text{ || } i > N \text{ || } j < 0 \text{ || } j > m \text{ || } mat[i][j] == 0$)

return;

$mat[i][j] = 0$

for (int k=0; k<8; k++)

DFS (mat, i+di[k], j+dj[k], N, m, di, dj)

}

② Shortest Path between single source & single destination.

Nodes edges:

8 10

Edges:

5 4

8 6

7 8

2 7

3 2

5 3

1 6

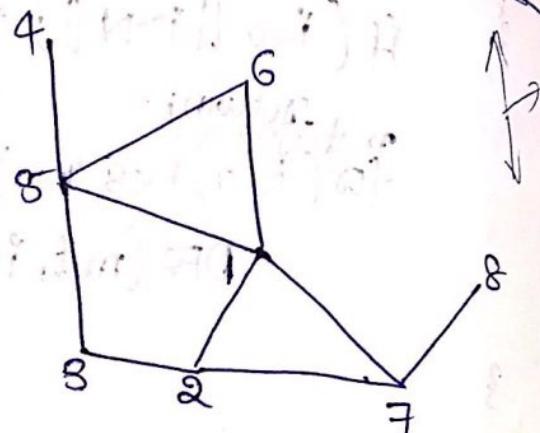
7 1

2 1

1 5

Source:- 5

Destination:- 8



5 to 8.

Shortest path:

$5 \rightarrow 1 \rightarrow 7 \rightarrow 8$.

But if we follow DFS order,

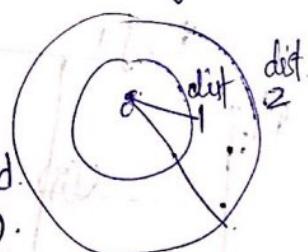
$5 \rightarrow 4 \rightarrow 6$. from 6 to 1 going down

this will not give shortest path, hence go with BFS or.

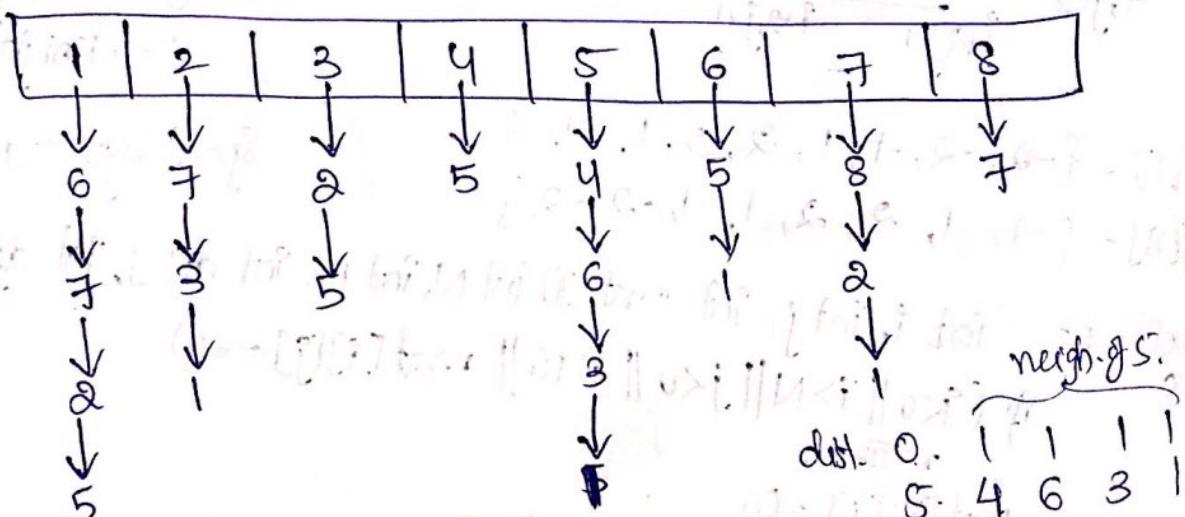
BFS with level order will give the minimum distance:

BFS: $O(V+E)$

undirected, unweighted graph.



Adjacency List:



maintain dist array. $dist[s] = 0$.

update it as the levels change |

BFS works for both directed & undirected unweighted graph.

- Q Given an undirected, unweighted acyclic graph (tree), find the length of longest path.

- ① For every pair of nodes, run BFS and find length of paths; find max of it.

Co_p: $|V|^2 * (|V| + |E|)$, $|V|$.

- ② If we run BFS on any node, we can get the distance of all the ^{remaining} nodes from that node,

so the run BFS on all nodes & not on pairs.

& find the maximum dist.

Co_p: $\frac{|V| * (|V| + |E|)}{\text{time}} \approx \frac{|V|}{\text{space}}$

int BFS longestPath ()

{ int dist[N+1] = {-1}; dist[u] = 0;

Queue <q> q;

q.push(u);

while (q.size() > 0)

{ u = q.pop();

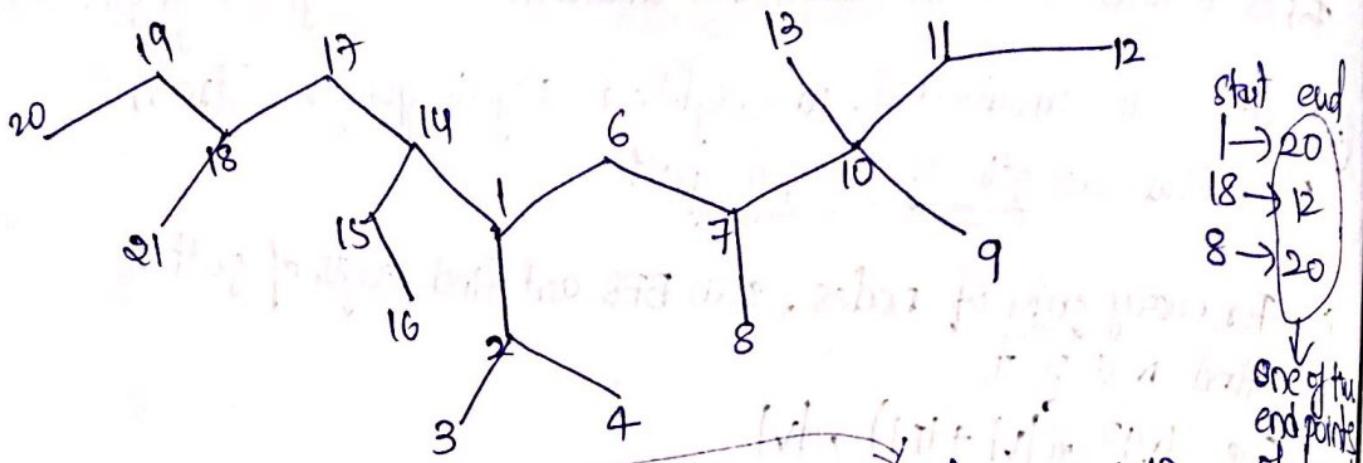
for (v : G(u))

{ if (dist[v] == -1)

{ dist[v] = dist[u] + 1;

q.push(v);

} return max(dist);



start from

20. end is 12
14 end is 12 or 20
2 end is 12 or 14.

dist.	0	1	2	3	3	4	5	5	6	6	6	7	7	7
queue.	20	19	18	17	21	14	15	13	10	11	12	13	14	15
dist:	8	8	9	9	9	9	10	10	11	11	12	12	12	12

(B)

From wherever you start BFS, we will get one of the longest length path.

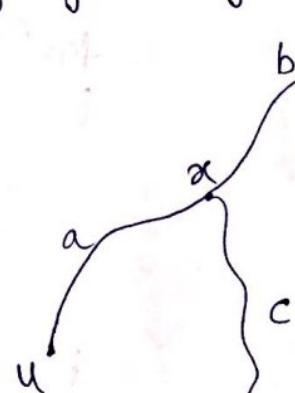
So 1st BFS gives one of the end points of longest length path, then 2nd BFS finds the other end point of the longest length path. Hence we do 2 times BFS to get longest path length.

Comp: $\Theta(2 * (|V| + |E|))$, $|V|$.

BFS ends with one of the nodes which belong the longest path.

MAIN:

$u = \text{BFS}_1(G, 1)$ // last popped node. (Simple BFS)
ans = $\text{BFS}_2(G, u)$ // get max dist value.
BFS with dist. arrays.



If uv is longest path then from a , it reaches to either u or v . but if there is some z nodes whose dist is greater than uv , then uv is not longest.

or. can do in 1 BFS. Only.

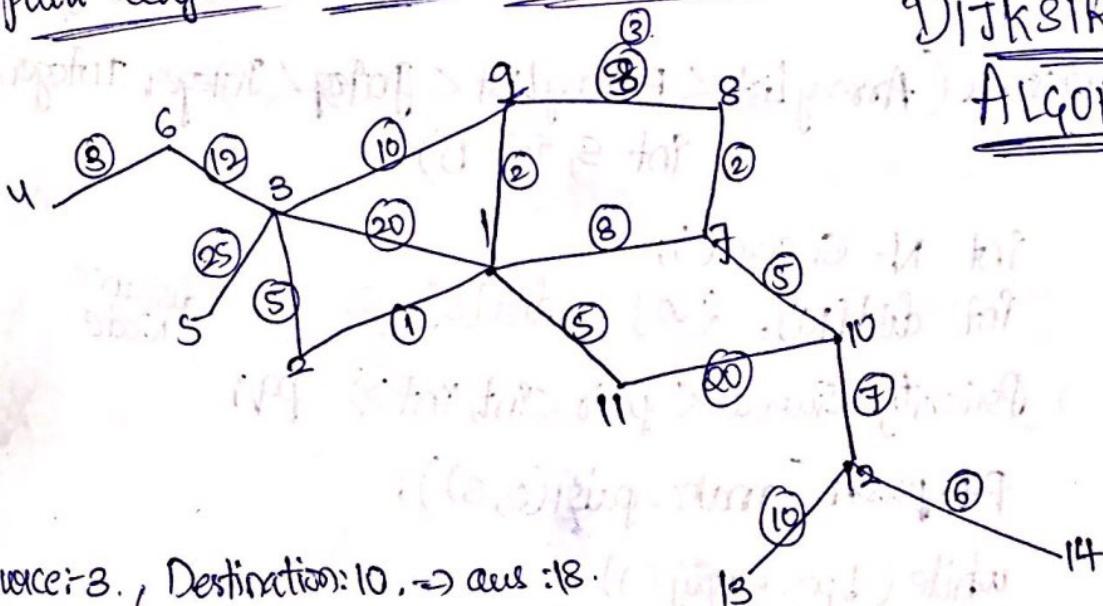
```

int BFS1(global)
{
    int u;
    while (!q.size())
    {
        if (u = q.pop());
        u = BFS1(G, l, F);
        if (flag == F)
            return u; // BFS1 return node
        ans = BFS(G, u, T);
        if (flag == T)
            return dist[u]; // BFS2 return max dist.
    }
}

```

* Given an undirected, weighted graph, find the shortest path length between source & distance.

DJIKSTRA's ALGORITHM.



Source: 3., Destination: 10. \rightarrow ans: 18.

i:	0	3	0	1	6	7	8	9	10	5	12	13	14
dist[i]	∞	6	∞	0	∞	∞	∞	∞	∞	∞	0	0	0
	28	5		15	25	12	14	18	10	31	11		
	6				13	16	18	10	12				

continue like this
until the popped ele. is destination node

dist array of size $N+1$: $\text{dist}[N+1] = \{\infty\}$

$\text{dist}[\text{source}] = 0$.

$U \xrightarrow{w} V$

Weighted graph:- edge U, V with weight W . $U \rightarrow (V, w)$

Maintain Min heap of Nodes (distance, Node) $V \rightarrow (U, w)$

as we need to remove the node whose distance is minimum.

Update the $\text{dist}[V] = \text{dist}[U] + \text{cost}(U, V)$ only if the distance in Node (Distance, Node) $\not\leq \text{dist}[\text{Node}]$.

First insert the source node $(\text{dist}, \text{node})$ into min heap,
then for neighbours, if the distance in the dist array gets updated,
then insert that neighbour in the min heap.

Pop the node from min

int Dijkstra (ArrayList<ArrayList<Pair<Integer, Integer>>> G,
 int S, int D)

 int N = G.size();
 int dist[N] = { ∞ }; $\text{dist}[S] = 0$;
 C++ Heap \rightarrow Priority Queue < pair<int, int>> pq;

 pq.push (make-pair(0, S));

 while (!pq.empty ())

 pair<int, int> p = pq.pop ();

 if int d = p.first (), U = p.second ();

 if (U == D) return dist[U] or dist[D] or d

 for (pair<int, int> q : G[U])

{

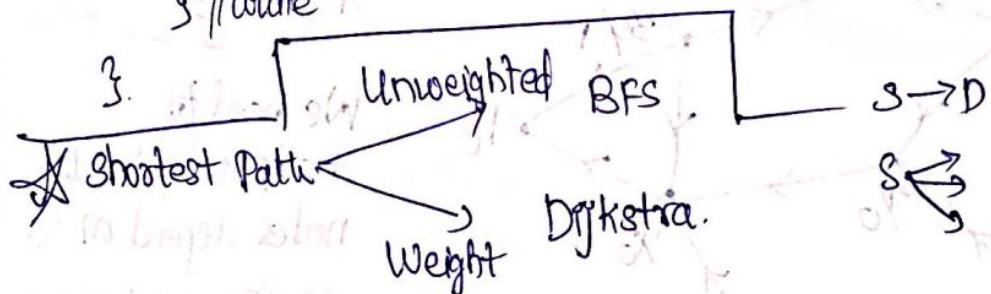
 if (dist[q] <= d) optimize.

```

v=q.first; // Node
w=q.second; // Distance.
nd=d+w.

if( dist[v] > nd)
{
    dist[v]=nd;
    pq.push( make-pair(nd,v));
}

```



- * All pairs shortest path : (u, v) call BFS | Dijkstra for all Nodes.
- * Floyd Warshall. (Better than Adj. Matrix).

- * for negative weighted graph \Rightarrow Bellman Ford Algorithm.
(Dijkstra doesn't work as updates go till $-\infty$.)
- * Cycle Detection

Cycle Detection

```

graph TD
    A["Cycle Detection"] --> B["Undirected"]
    A --> C["Directed"]
  
```

No Cycle $\Leftrightarrow |E| = |V| - 1$

~~cycle~~: $|E| \geq |V| - 1$

for $V \Rightarrow \text{inc } V$.
for E .

vis[3] = T.

$$e = e + \theta(3j) \cdot size$$

edge is counted twice

Hence check $e_{1/2} = \nu - 1$

so should be considered for every component of graph is disconnected without component.

ed for component graph is disconnected.

Undirected



$C=2$ // No. of components.

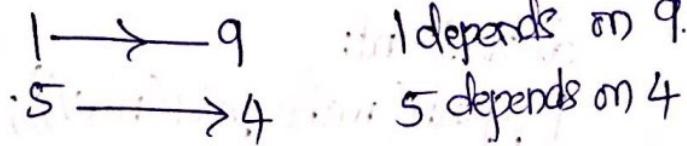
$$|E|=|V|-c$$

$$S = S - 2$$

$$3 = 3$$

Diseased:

If there is
cycle, dep.
exists always
in circular way.



Geologic Map

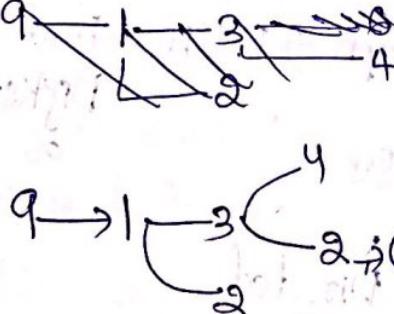
Geologic Map

Queue with Outdegree 0.

$g \times 23$

We need to check which all nodes depend on q.
so we need to traverse all the nodes adj. list ~~adj.~~ bcc
list. — bit back's couple mes!

So change the
directions;
check indegree.



Start with the nodes which depends on none. (i.e no outgoing edges)
these. q.

Then 1 depends on 9, as 9 is done, can go with 1 next and so on
starts on all nodes

outdegree :- no. of outgoing edges $\rightarrow O(|V| + (M+|E|))$

Indegree :- no. of incoming edges $\rightarrow O(|V| + |E|)$

→ iterate only on its neighbours

Change the given directions:

Edge. $u, v. u \rightarrow v \Rightarrow$ make as $v \rightarrow u \Rightarrow G[v].push(u)$
& update indegree. of $u. \Rightarrow in[u]++;$

Start with indegree 0 node, then update indegree of its next nodes $\Rightarrow in[adj\ list\ nodes]--;$

Considering indegrees, we need only to iterate only on the edges present in its adjacency list.

TOPOLOGICAL SORT.

Kahn's Algorithm

Code:

```
int in[N+1] = {0}
```

```
for i=0 to M
```

```
{ Read u, v
```

```
 G[v].push(u);
```

```
 in[u]++;
```

```
 while (!L.empty())
```

```
{ u = L.pop();
```

```
 for (v : G[u])
```

```
 in[v]--; // Dec. indegrees of neighbours.
```

```
 if (in[v] == 0)
```

```
 L.push(v);
```

```
 }
```

```
 for (int i=1; i<N+1; i++)
```

```
{ if (in[i] != 0)
```

```
 return false;
```

// Cycle Detection.

If all indegrees become 0 \Rightarrow then we have traversed all the nodes \Rightarrow No cycle.

else cycle. as some mode ^{not} travelled

If there exists two different nodes, of which one is already visited, \Rightarrow then there is a cycle.

If topological sort returns true \Rightarrow cycle.

Directed Graph: - edge classification.

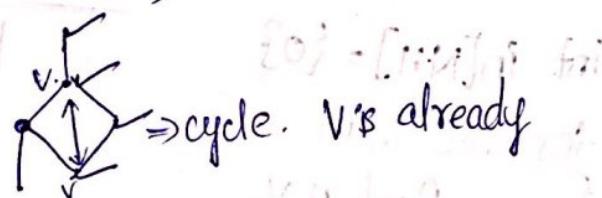
Forward edge,

Backward edge

Cross edges.

Same edge.

Undirected Graph:

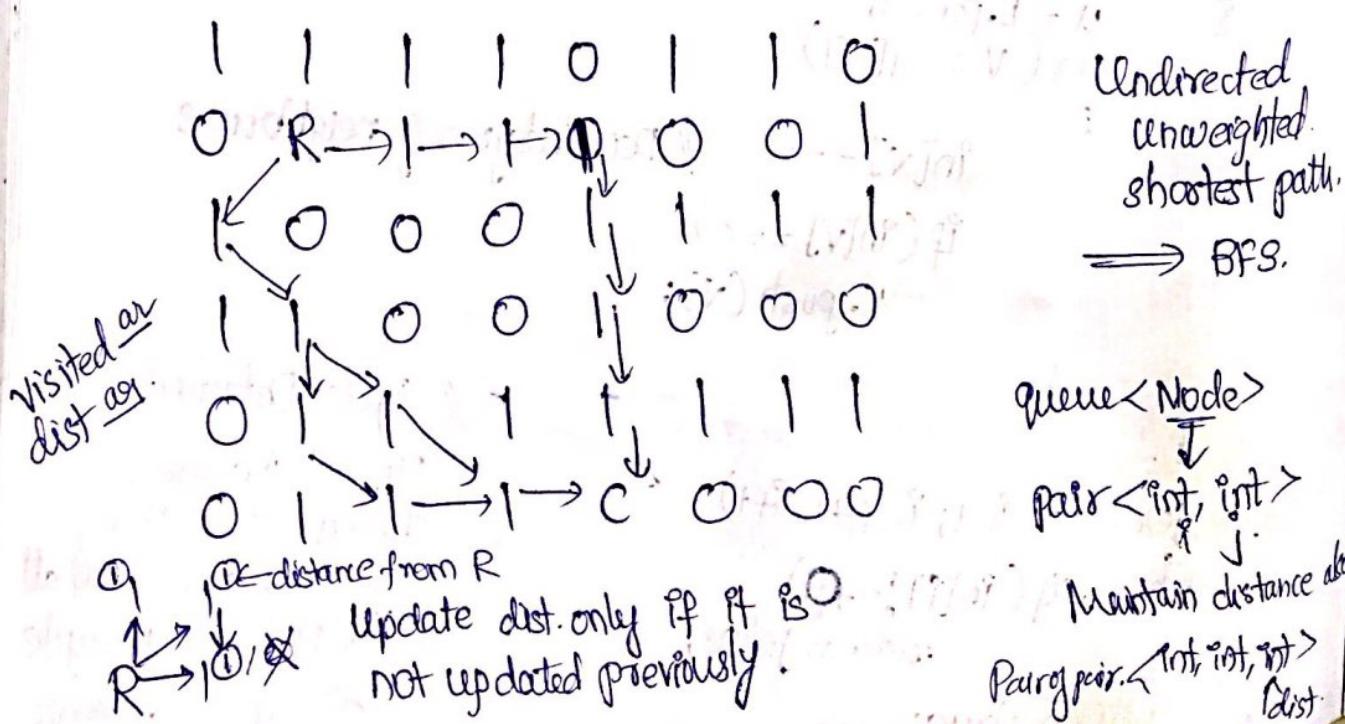


② Rat and Cheese, find Lexicographical shortest order

Path from (g_i, g_j) to (g, g) .

\rightarrow Land.

$\circlearrowleft \rightarrow$ Water.



for lexicographic order

Maintain d_i and d_j in that order.

$$d_i[] = \{$$

$$\} \quad \begin{matrix} ① & ② & ③ \\ ④ & ij & ⑤ \end{matrix}$$

$$d_j[] = \{$$

$$\} \quad \begin{matrix} ⑥ & ⑦ & ⑧ \end{matrix}$$

For finding the path, we need to store the parents of the nodes.

→ parent of a node is the one who updated it.

→ so maintain a separate matrix for parent with pair<int, int>

→ For pointing path, can use stack or recursion.

For marking visited, we can change 1's as 0's.

Printing
after
function call.

→ for distance

— same matrix inc value, ans is value-1 at C

— In queue node, maintain distance even.

— Or. maintain separate matrix for distance from R.

④ (MST) : Minimal Spanning Tree:

— Connected Acyclic minimal weighted tree.

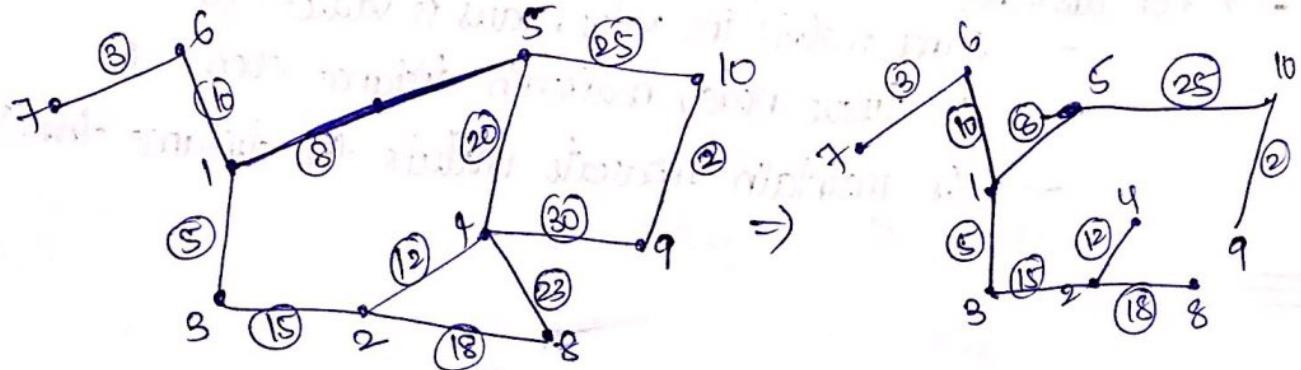
↳ PRIM's Algorithm.

↳ KRUSKAL's Algorithm

Kruskal's Algorithm:

→ Sort the edges first.

→ Go on taking the edges with minimum weights if taking some edge doesn't form a cycle.



List < Edge >

Edge
 {
 int u
 int v
 Pnt w.
 }

N m.
 m {
 u ~ w.

No need of forming Adjacency list. just store first form graph at steps next one by one edge consideration.

→ Sort the edge based on weights $\Rightarrow |E| \log_2 |E|$

Graph G, null initially
 Considering each edge, add it to the graph, check if the addition of the edge forms a cycle,
 if no ok, if yes, then remove that edge from graph & check with next.

for $|E|$ edges cycle detection takes $|V| + |E|$.

$$\Rightarrow (|E|)(|V| + |E|)$$

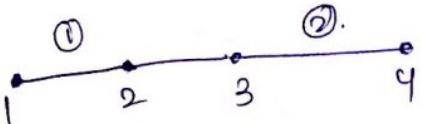
Cycle Detection in disconnected graph $\Rightarrow |E| = |V| - C$

$C \rightarrow$ connected component

DFS/BFS.

Comp:- $|E| \log_2 |E| + (|E|) (\underbrace{|V| + |E|}_{|V| \text{ edges don't cross } |V|})$.

Decrease the time taken to detect cycle:

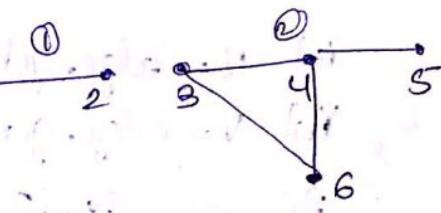


new edge (2,3)

2 belongs to ① comp.

3 belongs to ② comp.

(2,3) edge ~~4~~



new edge (3,6)

3 \in ② comp

6 \in ② comp.

(3,6) \times cycle

If parents $g(U, V)$ are different \Rightarrow no cycle.

If parents are same then cycle.

Sort the edges first, then

1	0	1	2	3	4	5	6	7	8	9	10
PI[i]	0	1	2	3	4	5	6	7	8	9	10
parent.	1	1	2	1	1	1	6	1	1	1	9

$5 \rightarrow 1$
 $4 \rightarrow 2 \rightarrow 1$

Update s's parent with parent of 2.

component of 9-10
is now connected
to component with parent 1.

1's parent 1
2's parent 2 \Rightarrow OK.

8 \rightarrow 1
 $4 \rightarrow 2 \rightarrow 1$

already q's parent is 1
found as 1
so update it
while returning
so that don't need to iterate many times.

given $q \Rightarrow$ parent of q
will not be $PI[q]$ directly,
until & unless $q \leftarrow PI[q]$,
we call $PI[q]$ these
many times.

Kruskal's Algo:-

$$|E|=|V|-cc.$$

Edge

{
int u;
int v;
int w;

constructor.

getters

setters.

// Sort the edges based on the weights.

Edge edge=new Edge(); form parent array p[0:N];
for (int i=0; i<m; i++) for (int i=0; i<N; i++)
{
p[i]=i;

int u=edge.getU();

int v=edge.getV();

int pu=findParent(u,p);

int pv=findParent(v,p);

if (pu!=pv)

{
if (pu<pv)

p[pv]=pu;

-else

p[pu]=pv;

DISJOINT SETS.

Union, find Data Struc.

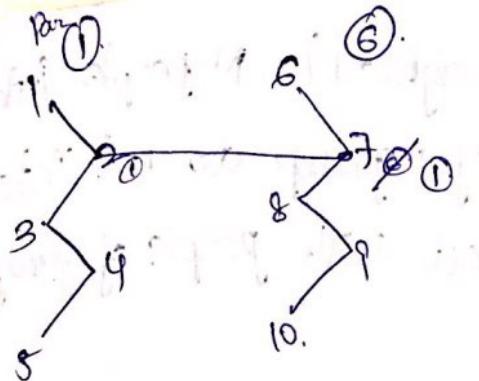
*. int findParent(int x, int p[])

{
if (x==p[x])

return x;

p[x]=findParent(p[x], p)

return p[x];



To Do:

Find why we update higher nodes parent with lower nodes parent.

~~Yours is the best~~

Disjoint Sets can be used in Cycle Detection if $p_u = p_v \Rightarrow$ cycle.

⑧ SNAKE & LADDERS

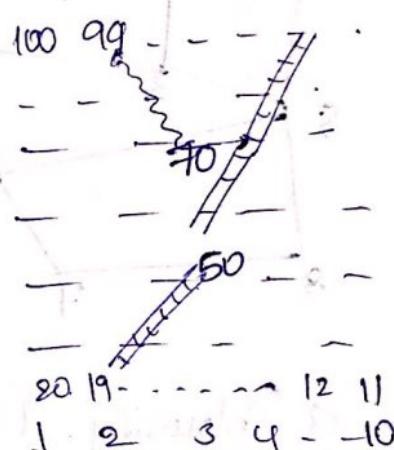
Snakes - bring you down
ladder - takes you ~~up~~ up.

find the shortest path

— in Single Source & Single Destination

Directed Unweighted Graph

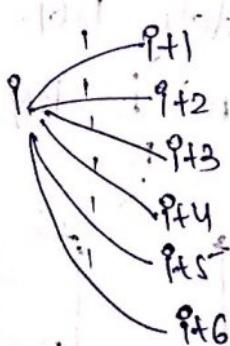
— BFS shortest Path from 1 to 100.



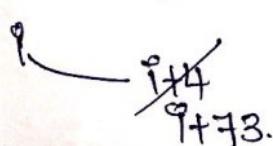
Dice Nearest:

\Rightarrow Unweighted.

(BFS)



If ladder at 4 to 73.
then



Dice Cost:

Weighted.

(Dijkstra's)

dice cost

1 \rightarrow 10

2 \rightarrow 15

3 \rightarrow 7

4 \rightarrow 20

5 \rightarrow 3

6 \rightarrow 18

Create graph
first, & then

do BFS/Dijkstra

Snakes: $[(s_i, e_i), \dots]$

Ladders: $[(s_i, e_i), \dots]$

Starting point s_i, j

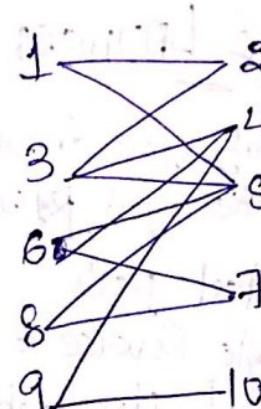
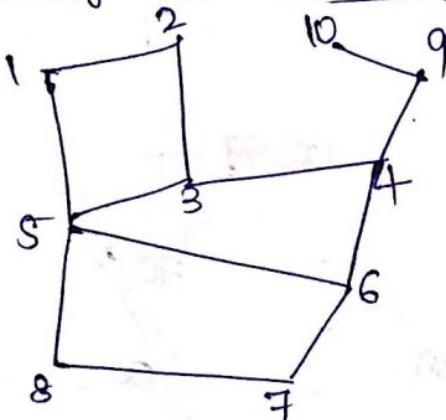
Ending point e_i, j

Given a graph (undirected unweighted). N-people problem.
 Partition the graph such people of 1 group do not interact
 with people of group¹, but interact with people of group².

Intra-Interaction X

Inter-Interaction ✓

2-Colouring: B.GI (Bipartite Graph):



2-Colouring. (B.W).

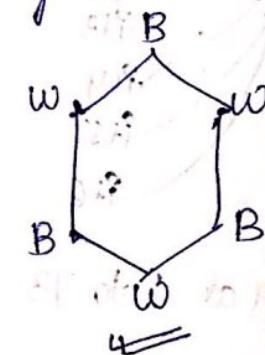
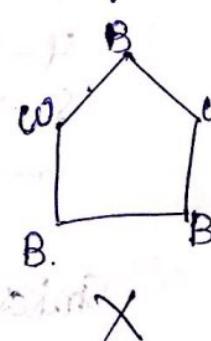
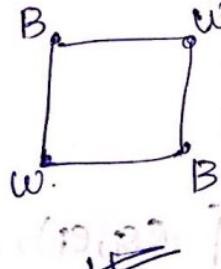
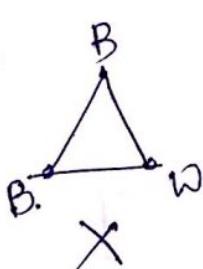
(B6) Bipartite Graph: If a graph is 2-colourable then it is B.GI.

~~If a graph has cycle \Rightarrow It is not B.GI.~~

~~If a graph has no cycle \Rightarrow It is definitely B.GI.~~

~~If a graph has cycle \Rightarrow May be may not be B.GI.~~

Eg.

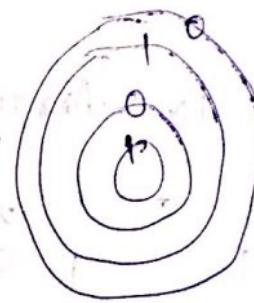


Cycle length Odd \Rightarrow Not B.GI

Cycle length Even \Rightarrow B.GI.

BFS & DFS.

First all Nodes colors let 2. default.



def. colors :- 2 2 2 2 2 2 2 2

1 → 0. 1 1 1 0 0 0 0

neigh → neighbours

neighbours

In BFS if the nodes colours
are uninitialized i.e 2 update it $0 \rightarrow 1$.

main:

BFS/
if ($\text{DFS}(G, 1)$)

but if it is already initialized

$i \rightarrow 0$

$i \rightarrow 1$

$0 \rightarrow 1$

$0 \rightarrow 0$

point YES

else

point NO.

Graphs Problem:

1) Max-Flow Algorithm.

2) Strongly Connected Graph Components.

3) Vertex Cut / Articulation Points.

4) Hamiltonian Path.

(Unfinished) visiting each node once

and also visit every edge once

so that no node is visited twice

Hamiltonian

path

revisiting nodes

so that total sum of edges

is equal to number of edges

so that path is closed

* Given Array, find Maximum value in subarray i to j :

$\underline{arr}[] =$	0	1	2	3	4	5	6	7	8	9
	5	1	8	3	10	2	4	5	1	7

$Q \downarrow$	$i, j: \max_{k=i}^j (arr[k])$	$\left\{ \begin{array}{l} \text{Range} \\ \text{Queries} \end{array} \right.$
	$0 \leq i \leq j \leq N$	

①. Brute force:- Iterate & find MAX.

Cmplx: $O(Q * N)$, $O(1)$.

For Range Queries:- 3 solutions.

→ 1) BIT (Binary Index Tree)

2) Square Root Decomposition

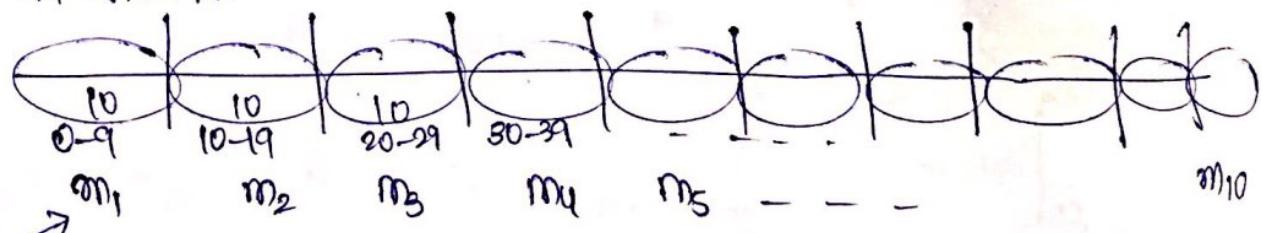
3) Segment Trees

②. Square Root Decomposition:- Cmplx: $O(N + Q + \sqrt{N})$

Eg. $N=100$ \sqrt{N} parts with \sqrt{N} ele. each.

Precompute max in \sqrt{N} eles. of all \sqrt{N} parts.

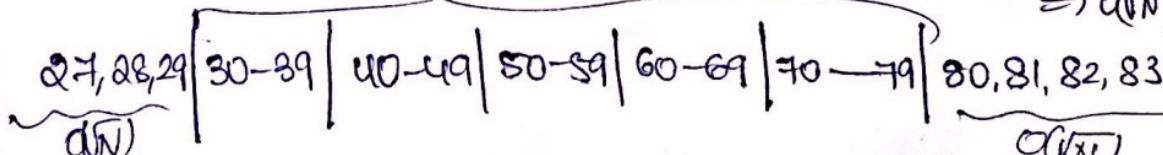
$$\sqrt{N} \Rightarrow \sqrt{N} = N.$$



m_1 stores max ele in first 10 ele. (0 to 9 idx), m_2, \dots, m_{10} do.

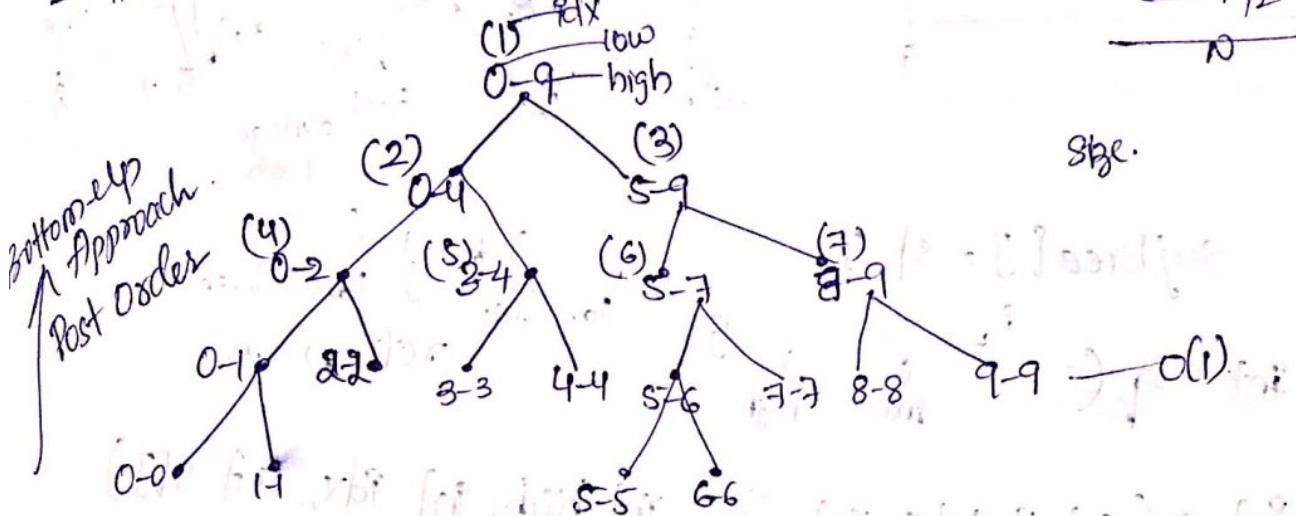
Eg. Query 27, 83.

At max worst case \sqrt{N} positions. \Rightarrow Each has $m_i - O(1)$.
 $\Rightarrow O(\sqrt{N})$



③ Segment Trees.

- Root Node stores max from 0-9.



Store ^{max} values in Array =

Create Seg. Tree:-

int * f(int arr[], int segTree[], int low, int high, int idx)

{

 if (low == high)

 {

 segTree[idx] = arr[low];

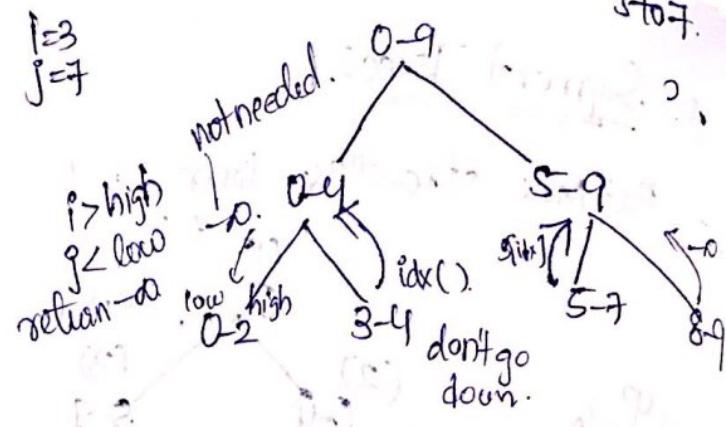
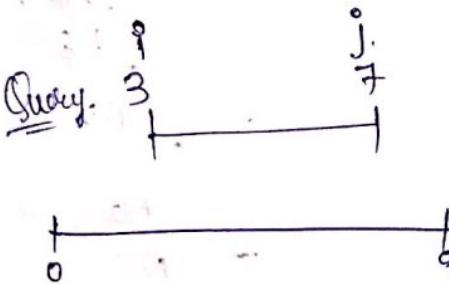
 return segTree[idx];

 int mid = low + (high - low) / 2;

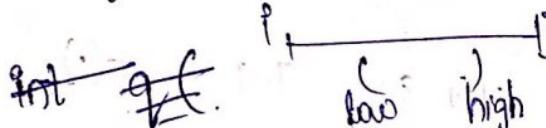
 segTree[idx] = max(f(arr, segTree, low, mid, 2 * idx),
 f(arr, segTree, mid + 1, high, 2 * (idx + 1)),

 return segTree[idx];

}



$\text{segTree}[] = ST[]$



$\text{int } q(\text{int } i, \text{int } j, \text{int } low, \text{int } high, \text{int } idx, \text{int } s[])$

{
 if ($i > high \text{ || } j < low$)

 return $-\infty$;

 if ($low \geq i \text{ && } high \leq j$)

 return $s[idx]$;

 mid = $low + (high - low) / 2$;

 return max.($q(i, j, low, mid, 2*idx, s)$,

$q(i, j, mid+1, high, 2*idx+1, s)$);

Main $\text{int segTree}[2N];$

CreateAll $f(0, segTree, 0, N-1, 1);$

for($k=0$ to Q)

 Read i, j

$q(i, j, 0, N-1, 1, segTree)$

③. Complexity: $O(N \log N)$.

$O(N + Q \log N)$, $O(N)$

Updatons: $\frac{Q \cdot S}{O}$ update with.

Max Node \rightarrow each ele update takes $\log N$ time with Q fun
 $\Rightarrow O(QN \log N)$.

\hookrightarrow Don't go with Segment Trees then

Go with

LAZY PROPAGATION:

INTERVIEW PREPARATION.

— Aptitude Quants.

— OS

— DBMS

— CN

— Puzzles.

— System Design.

— OOPS

— DP.