lowercase eng

Q. Find count of permutations of string A as a substring in string B.

A = abc

B = a b c b a c a b c

1 2 3 4 5

abc
acb
bca
bac
cab
cba

3!

A: a b c d

B: b a c c d b a d c a c b d

1 2 3

3
2

n

n

n!

B.F → { find all possible permutation of A first
└ every window of size N → ~~dead~~

Is there any common thing in permutation of A?

frequency of char — same

a b c

a b c b a c a b c

B → a b c
1 1 1

A
a b c d e..
1 1 1 0 0

A = a b c permt — c b a

b b c

a b c x d
1 1 1 0 0

A = a b c

B = x b c a b d a

x b c
1 1 1

b c a
1 1 1

b b c

b c
2 1

a b d
1 1 1

T.C:

abc

x a b c d

x a b c
~~1~~ 0 1 1 1

1) build freq array of A → n

2) build freq array for first window of B (0 – n-1) → m
 └ compare freqarrays → 26 ans++

i = n ; i < m ; → O(n)
freq i^{th} ++;
$i - n^{th}$ i--;

Counter
26

T.C: 26 * n
O(n)

compare freq arrays — 26

as++

O(26)

O(1)

i

freq[A[i]]++

'b' → 1

98 — 97

index - int

A[i] — 'a'

98 — 97 = 1

```
int freqA[26] = [0];
for( int i=0; i< A.length; i++)
    freqA[ A[i] — 'a' ]++;
```

```
freqB[ 26] = [0];
for( int i=0; i<n; i++)
    freqB[ B[i]— 'a']++;
```

0 - n-1   n^th

comp two arrays —   count++;   one more loop

SC: O(26)

O(1)

```
for i = n ; i < m ; i++
    freqB[ B[i] —'a'] ++;
    freqB[ B[i-n] —'a'] --;
    compare freq arrays — count++;
```

26 * n

26

Q2

N integers :- max min

minimise → max − min

3 4 1 5 6 → 4, 6

operation

a[i] + 1
or
a[i] − 1

Atmost B times

question

decrease max
increase min

max − min

2 6 3 8 9 7

max = 9, min = 2 } 7

B = 1

2 6 3 8 8 7

max = 8, min = 2 ⇒ 6

B = 2

3 6 3 8 8 7

freq of max & min

~~B = 3~~

B = 4

3 6 3 7 7 7

max = 7, min = 3 } 4

4 6 4 7 7 7

diff = 3

1) to get any change in the ans, we need to just focus on max & min

2) in order to make any change — freq of min & max

2 4 2 3 6 ~~7~~ 5 7

min = ~~2~~ ~~3~~ 3

max = 7

<2, 2> <6, 1>

<3, 3> <7, 2>

<4, 1>

<5, 1>

ans = max − min

ans = 5

B = 4

```
while (B >= 0 && ans != 0)
{
    if ( freq[min] <= freq[max])
    {
        freq[min+1] += freq[min]
        B -= freq[min]
        freq[min] = 0
        min += 1;
    }
    else
    {
        freq[max-1] += freq[max]
        B -= freq[max]
        freq[max] = 0
        max--;
    }
if ( B >= 0)
    ans--;
```

2 → 3

A) b > 0 && ans != 0

B) b > 0 || ans != 0

b = 0

F

T

T.C: n + min(B, max-min)

S.C: O(n)

10:25

③ Find number of set bits from $0 \to N$

0 – 3

0 : 000 0
1 : 001 1
2 : 010 1
3 : 011 2

4

0 – 6

$0 \to 0$
$1 \to 1$
$2 \to 1$
$3 \to 2$
$4 \to 1$
$5 \to 2$
$6 \to 2$

9

BF → traverse $0 \to n$
for every no – count set bits ← $\log_2 n$

TC: $n \log n$

x $y = x/2$

$x = 13$ $y = 6$
$x = 18$ $y = 9$

$Nos(x) - Nos(x/2)$

x is odd

case I

7 – odd
$7/2 = 3$ – odd

right shift
$x/2$ – left shift
right shift

1 1 1 0

$14 = 8 + 4 + 2$
$2 * 14 = 2 * (8 + 4 + 2)$
$= 16 + 8 + 4$

one set bit is reduced

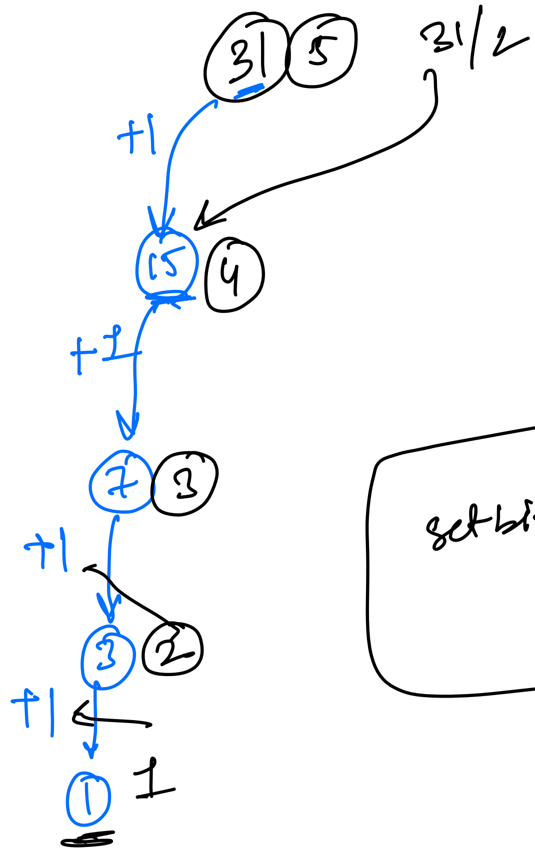
$$\boxed{Nos(x) - Nos(x/2) = 1}$$ if x is odd

$x/2$ – right shift

case II x is even

no loss of set bit

$NOS(x) - NOS(x/2) = 0$ if x is even



$$\boxed{setbits[i] = setbits[i/2] + i \% 2}$$

2/2

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

+1

O(1)

O(n)

$2^x/2$ $2^x/2$ $2^x/2$ x x

8/2 8/2 8/2 ← no of set bits

x posti

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

8-8=0, 9-0, 10-2 → 0, 1, 2, 3, 4, 5

0–5

perfect pow of 2

8

2^x

2^x

N

2^{x+1}

0

$2^x - 1$

x

$2^x \to x+1$

x bits

$2^x \to x+1$

$8 \to 1000$

2^3

7 111

8 1000

3+1 4

3

$$TSB = \frac{2^x}{2} * x \quad x = \log_2 N$$

nearest pow of 2

$N = 2^x + t$

N ... $2^x - 1$ ... 0

$N \leftarrow 2^x$

All the MSBs will be set

$\frac{2^x}{2} * x$ 1

$+ N - (2^x - 1)$ 2

$x = \log_2 N$ $2^x \to 0 - 2^x - 1$ $2^x \to N$

$$\boxed{NOS(N) = \frac{2^x}{2} * x + N - (2^x - 1) + NOS(N - 2^x)}$$

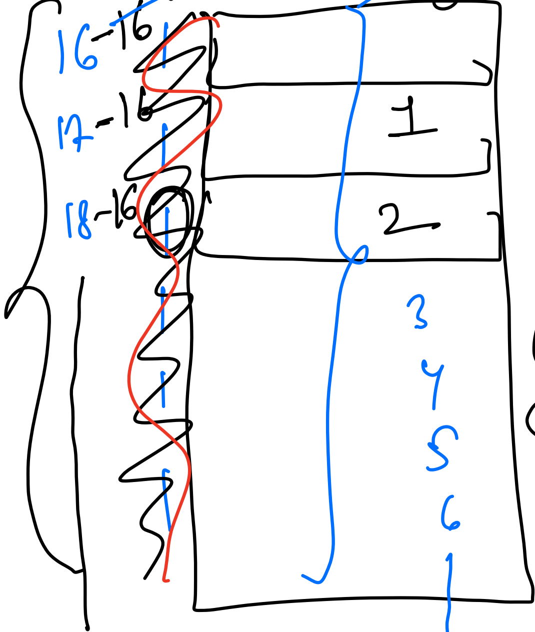
MSB

recursion

0 → 0

10^{0-30} 0–30

$x = \log_2 30$



ans=0

30

$+0-15 \rightarrow \frac{2^4}{2} * 4$

$+30-16+1$
$= 30-(16-1) \quad - \quad N-(2^x-1)$

$\frac{2^x}{2} * x$

14

0-14

$30-16 = 0 \rightarrow 14$

$30 \rightarrow 14$

$N=0 = 0$
$N=1 = 1$

$N \longrightarrow 2^x \quad 2^x-1 \quad 0$

Number of set bits (N)

{

if(N==0 || N==1) return N;

int x = $\log_2 N$;

// 0 — (2^x-1) $\equiv$ $2^x/2 * x$

ans = 0;

ans += $\frac{2^x}{2} * x$] 1

ans += $N - 2^x + 1$ ← MSB

ans += number of set bits($N - 2^x$)

return ans;

}

1

T. C: $\log_2 N$

S.C. $\log_2 N$

$O(N) = 10^9$ ($N=10^9$)

$O(\log N) = 29 \approx 30$ ($N=10^9$, $\log 10^9$)

30-16