count of

Q 1 Permutation of string A as a substring in string B.

contiguous step

A = a b c

B = a b c b a c a b c

1 2 3 4 5

abc
acb
bac
bca
cab
cba

A: a b c d

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

1 2 3

abcd
bcda

B-F generate all possible permutations → n! T.C

permutations → frequency of all chars will be same

A: a b c d

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

freq[i]++

freqA

| a | b | c | d | . . . . |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | |

= 1

① a b c d . . freq[ ]

1 1 2 1

② a c d

1 2 1

a c c d

1 2 3

i-3 i-2 i-1 i

i-4

freq[i]++;

freq[i-n]--;

permut

lowercase

26

a b c d

1 1 1 1

a c c d

a b c d

1 1 2 1

0.

mode: 0

a b c d

B: b a c d d b a d c a c b d e f g h

a b c d

1 1 1 1

a b c d e f

1 1 2 0 0

0 1 1 2

1 0 1

26

$m \equiv len(B)$

```
freqA[26] = {0};
for( int i=0; i<n; i++)
    freq[A[i]-'a']++;
```

A

26

'a' − 97 − 'a' = 0

b − 98 = 1

c − 99 = 2

⋮

```
freq[26]={0};
for( int i=0; i<n; i++)
{
    freq[B[i]-'a']++;
}
compare freqA && freq
  if( freqA == freq) cont++;

for( int i=n; i<m; i++)
{
    freq[ B[i]-'a']++;
    freq[ B[i-n]-'a']--;
}
compare freq array
  if( freqA == freq)   // X
    cont++;
}
cout << cont;
```

T.C : $O(M \times 26)$

$O(M)$

$256 \times M$

(2)

n integers → (A)

minimise → max − min

at most (B) operations

A[i] + 1

A[i] − 1

| | | | | | | |
|---|---|---|---|---|---|---|
| | 2 | 6 | 3 | 8 | 9 | 7 |
| B = 1 | 3 | 6 | 3 | 8 | 9 | 7 |
| B = 2 | 3 | 6 | 3 | 8 | 8 | 7 |
| B = 3 | 4 | 6 | 4 | 8 | 8 | 7 |
| B = 4, B = 5, B = 6 | 4 | 6 | 4 | 7 | 7 | 7 |

9 − 2 = 7 diff

diff = 9 − 3 = 6

diff = 9 − 3 = 6

9 − 3 = 6

diff = 8 − 4 = 4

diff = 7 − 4 = 3

(1) To decrease the diffrence change only min or max elements

(2) To bring any change you need to perform operation on every occurence of min/max elent

2

2 2 3 4 5 6 6 7

1

5

diff = 7-2 = 5

B=1

2 2 3 4 5 6 6 6

~~B=2~~

B=3

3 3 3 4 5 6 6 6

B=6

3 3 3 4 5 5 5 5

B=9

4 4 4 4 5 5 5 5

B=13

d=0

5 5 5 5 5 5 5 5

2 2 3 4 5 6 6 7

x<y

min

max-1

max

max-1

2

x

y

max

y cost

-1

y=y cost

2 2 3 4 5 6 6 7

min = 2 max = 7 diff = 5

(2,0) (3,2) (6,2) (2,2) (3,1) (7,1) (4,1) (7,0) (5,1)

B=1

max = 6

B=2

freq(min) < freq(max)

↑ min

min = 3

3

max = 6

3

max+1

5

freq[3] += freq[2]

freq[2] = 0

B=3

freq[5] += freq[6]

freq[6] = 0

B += freq[6] = 6

```
Hashmap < int, int > m;
small = ∞        large = -∞
for ( int i=0, i<n; i++)
{
        m[a[i]]++;
        small = min(small, arr[i]);
        large = max( large, arr[i]);
}
B - int ans = large - small;
```

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

    if( B >= 0)
    {
        ans = min(ans, large-small);
    }
}
```

yes

B < 0

ans - negative

negative pos

max - min

-1 - -5

= 4

③ count total set bits in all no's from 0 → n

unset set

③

| | | |
|---|---|---|
| 1: | 001 | 1 |
| 2: | 010 | 1 |
| 3: | 011 | 2 |

④

⑤

| | | |
|---|---|---|
| 1: | 001 | 1 |
| 2: | 010 | 1 |
| 3: | 011 | 2 |
| 4: | 100 | 1 |
| 5: | 101 | 2 |

⑦

B.F go from 1 → n

count set bits for every no

T.C: n log n

log n

n

count += (n%2)

n = n/2

| | | |
|---|---|---|
| 2 | 31 | 1 |
| 2 | 15 | 1 |
| 2 | 7 | 1 |
| 2 | 3 | 1 |
| | 1 | |

x

y = x/2

x = 13, y = 6: 3, 2, 1

x = 18, y = 9: 2, 2, 0

NOS(x) − NOS(y) <= 1

case I: x is odd

....0100 1

y = x/2

1100101

x/2

14 → 1110 (8 4 2 1)

14 = 8 + 4 + 2

$y = x >> 1$

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

$28 = 2*8 + 2*4 + 2*2$

11 00 1 0 1

0 1 1 0 0 1 0

Only 1 set bit is decreasing

case 2: $x$ is even

0 unset

$y = x/2$

$y = x >> 1$

No change in no of set bits

$x$

$y = x/2$

odd

$NOS(x) - NOS(y) = 1$

$NOS(x) = NOS(y)$

13 (3) → +1 → 6 (2) → 3 (2) → +1 → 1 (1)

setbits[i] = setbits[i/2] + i%2

+1

O(1)

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 2 | 1 | 2 | 2 | 3 |

+1 +1

$10^9$

T.C : O(N)

S.C : O(N)

$N = 2^x \rightarrow \log_2 N = x$

$0 - (2^x - 1)$

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

$x = \lfloor \log_2 n \rfloor$

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

$2^{x-1}$ $\quad 2^{x-1} * x$

N

230 | 127 | 0

$+ \frac{N - (2^x - 1)}{2}$ MSB

| | | | | |
|---|---|---|---|---|
| 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 | | | | |
| 14 | | | | |
| 15 | | | | |

$2^x/2$ $\quad 2^x/2$ $\quad 2^x/2$

$2^x$ $\quad 2^{x-1}$

$2^x$ $\quad 2^x$

x+1 $\quad$ xbit

N

0 1 2 3 4 5 6 7

3 2 1 0

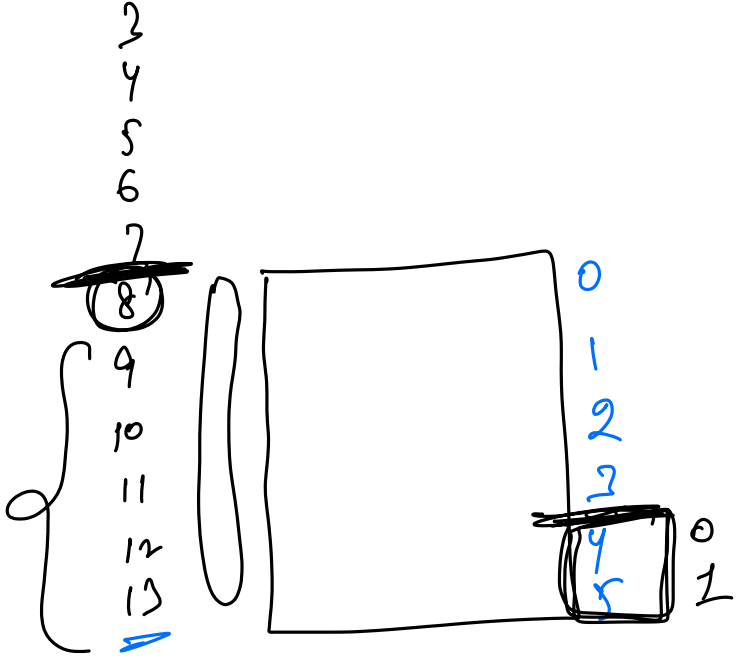
8 → 1 0 0 0

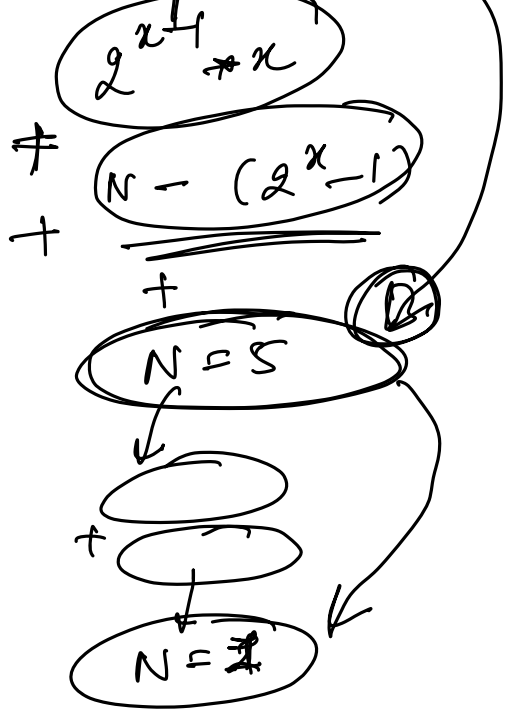
4 bit $\quad$ x bit

x+c

0 1 2

N = 13






```
count of setbits ( int N)
{   if( N==0 || N==1) return n;
    int x = floor( log2n)
    int ans =0;
    ans += 2^(x-1) * x
    // remain = N-(2^x-1)
    // for all remain elements MSB is set
    ans += N-(2^x-1)
    ans += countsetbits( N-(2^x-1));
    return ans;
}
```

$\lfloor \log_2 n \rfloor$

$O(\log_2 n)$

4.2
4.7

$O(1)$

$O(1)$

$O(1)$ 1

$// \; 0-2^x-1$

31 → 16

$\log_2 n \; \lfloor 4.9 \rfloor$

$2^1$ $O(1)$

31

0 – 15

32 $O(1)$

perfect power of 2 – 1

15 → 0-7 → 7 → 0-1 → 3 → 1

$\log_2 n \equiv$

$n = 10^9$

$\log_2 10^9$

$= 9 \times 3$

$= 27$

$\approx O(1)$

$O(N)$

$10^9$

$\left\{ \begin{array}{l} 0-7 \\ + \; 13-8 \\ \quad 0-5 \end{array} \right.$ 13 MSB

Q1 optimised

need

A : a b c d

B : b a c c d b a

d c a c b d

n = 4

need = 4

need[ 26 ] :

a b c d

1 1 1 1

0 0 0 0

1 1 1 -1

0 0 0 1

a c c d b

need = 3

need = 2

need = 1

need = 0

need = 1

need = 0

need = 1

X

X

c d b a

c d b a d

need[]+1 need[]--;

need = 0

need = 1

```
freq[26] = {0};

for( int i=0; i<n; i++)
        freq[ A[i]-'a' ]++;

int need=n;
for( int i=0; i<n; i++)
{
        freq[ B[i]-'a' ]--;
        if( freq[ B[i]-'a' ] >=0)
                need--;
}

if( need ==0)
        ans++;

for( int i=n; i<m; i++)
{
        if( B[i] != B[i-n])
        {
                freq[ B[i] -'a' ]--;
                freq[ B[i-n] -'a' ]++;
                if(freq[ B[i]- 'a' ] >=0)
                        need--;
                if( freq[ B[i-n]-'a' ] >0)
}
```

// for 1st window

abcd a

97 — 97

A[i] = 2

'c'

for [99]

H.M. array

=0

need++;

}

if(need == 0)

cnt++;

}

$O(N)$

$O(M * 26)$