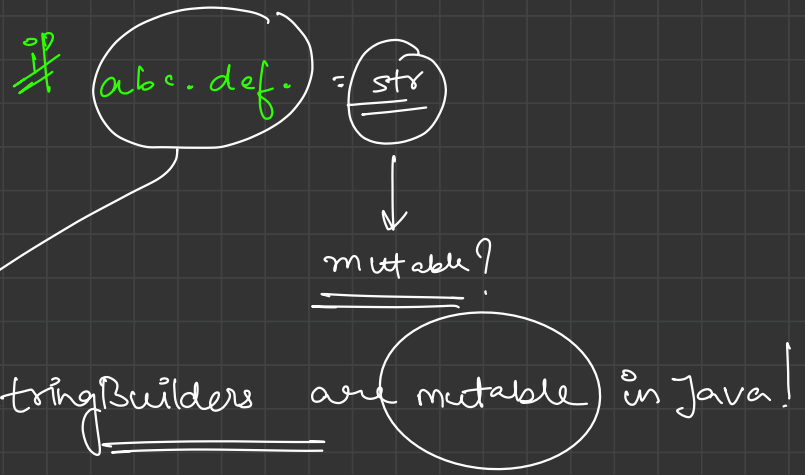
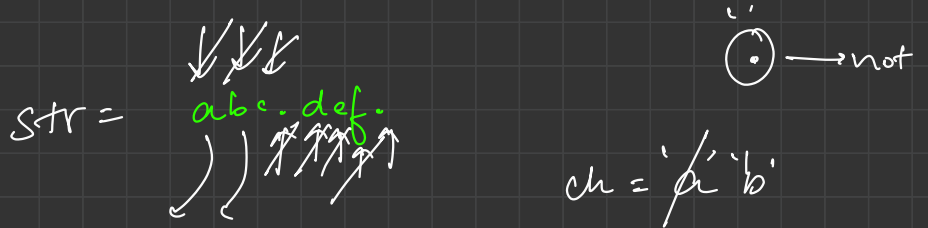
decrypt it 1


ip
abc.def.
= str
mutable?
StringBuilders are mutable in Java!
replace . to [.]




```
StringBuilder sb = new StringBuilder();
for(int i=0; i<str.length(); i++)
{
    char ch = str.charAt(i);
    if(ch == '.')
    {
        sb.append("[.]");
    }
    else
        sb.append(ch);
}
```

TC: O(N)

SC: O(N)

{ converts sb to a String }

```
return sb.toString();
```

str = . . . O(N) 3

o/p [.][.][.] O(3N) = O(N)

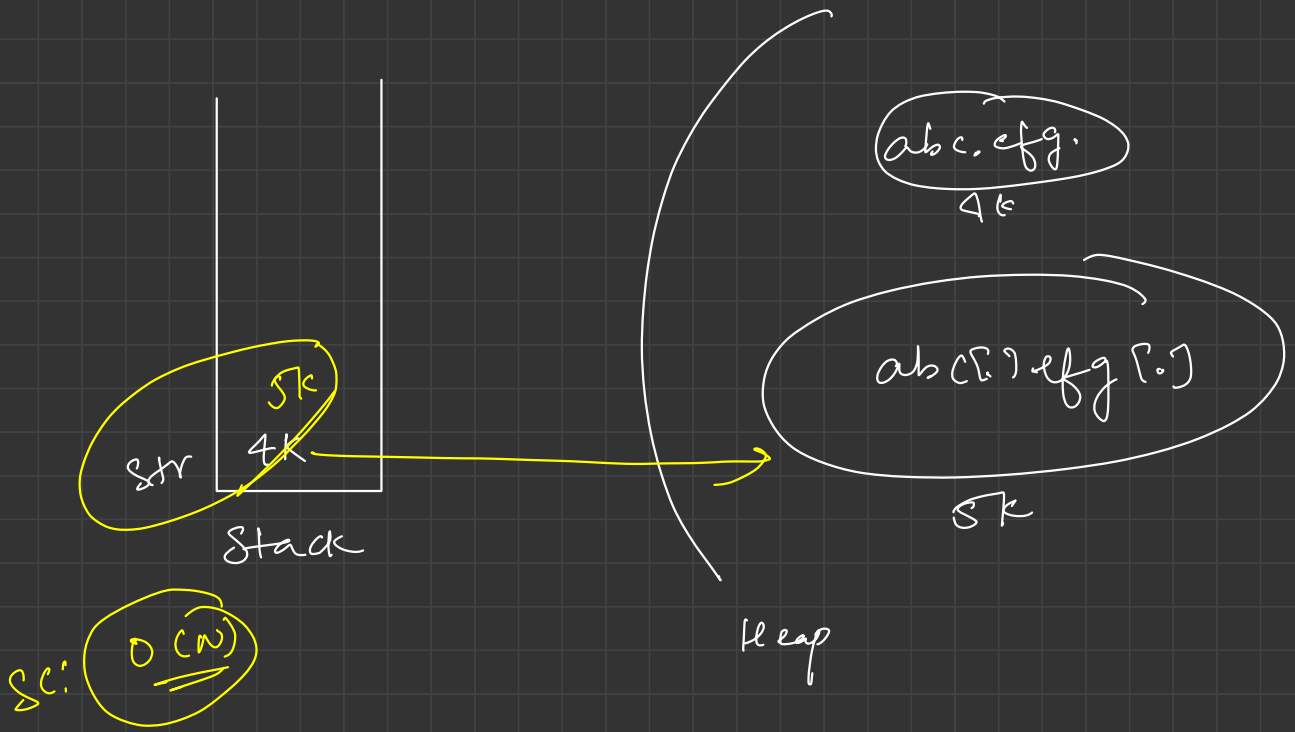
3xN O(3xN) = O(N)

Method 2

replace method of string

```
str = str.replace(".", "[.]");
return str;
```

TC : O(N)
SC : O(N)


abc.efg.
4k
5k
str
4k
Stack
ab[:].efg[:]
5k
Heap
SC:
O(N)

Increase It

arr[] = { 1, 2, 3, 3 }

↓

{ 1, 2, (4), 3 }

coins spend = 1

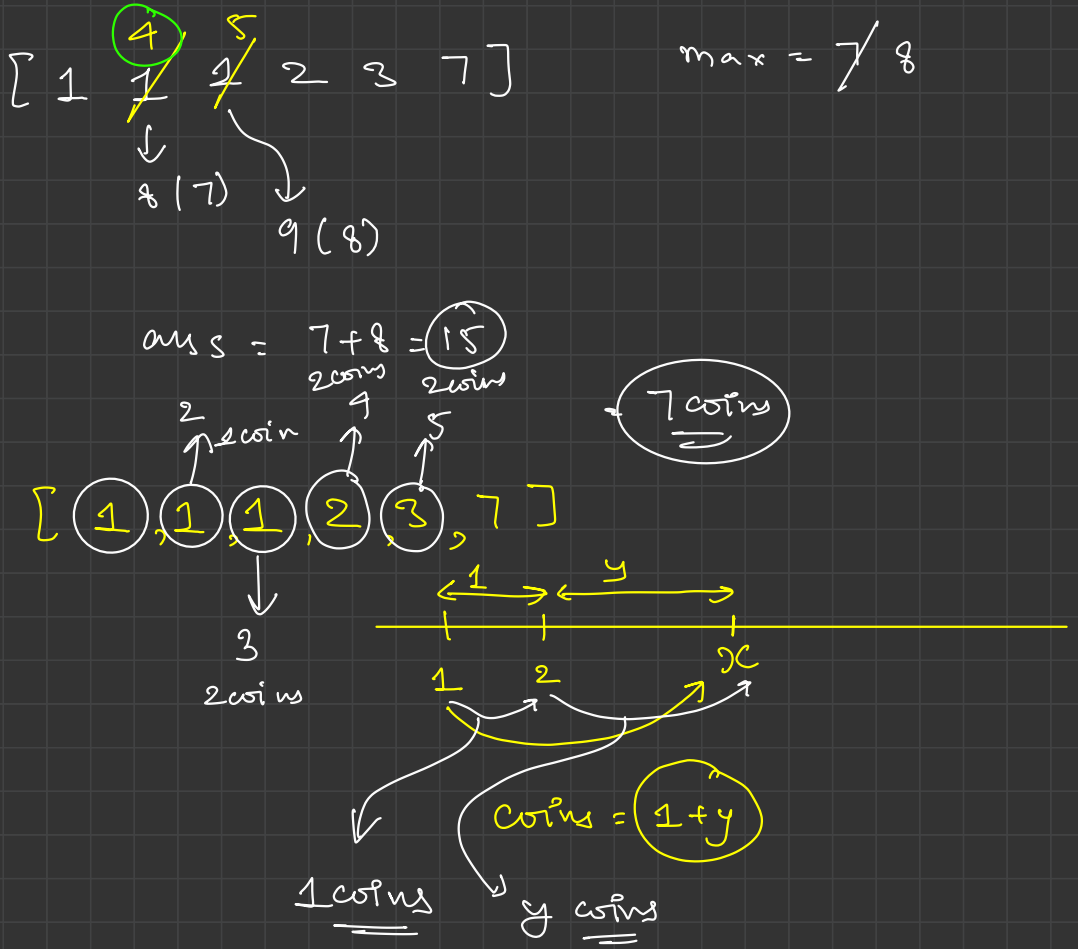
[1 1 1 2 3 4]
 +4 +5

X

→ 5 { work 5 − 1 = 4 coins }

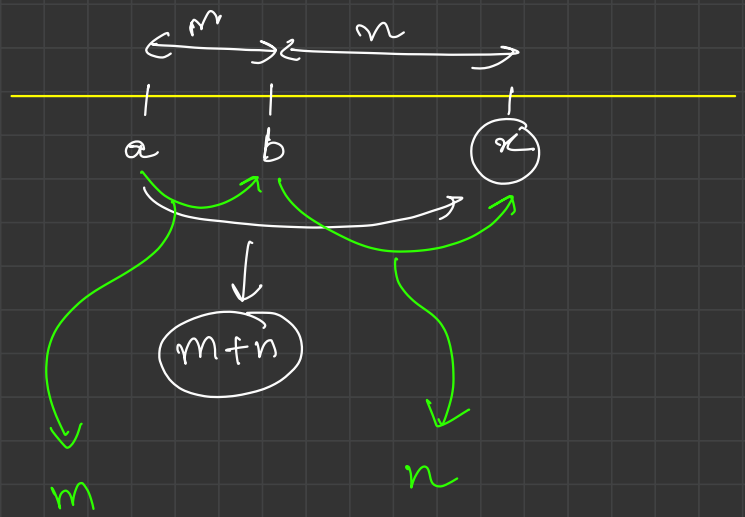
→ 6 { work 6 − 1 = 5 coins }

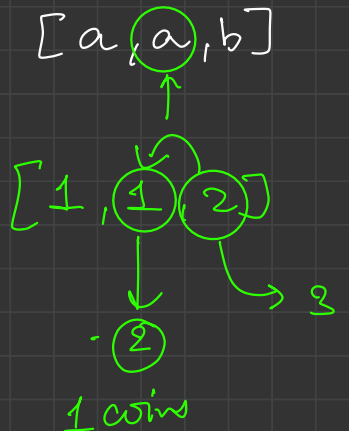
= 9 coins

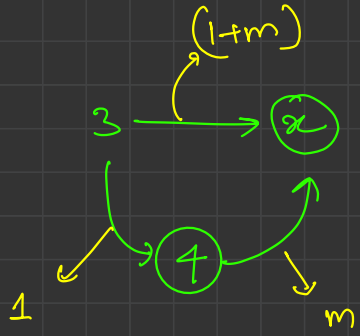

[1 1 1 2 3 7]
4
5
8 (7)
9 (8)
max = 7/8
ans = 7+8 = 15
2 coins
2 coins
2
2 coin
4
5
7 coins
[1,1,1,2,3,7]
3
2 coins
1
y
x
1
2
Coins = 1+y
1 coins
y coins

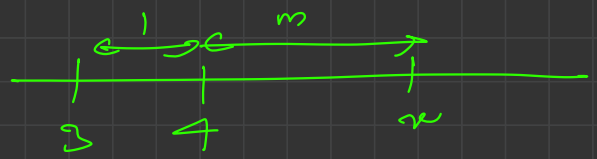
Sorting

- ascending order
- duplicates comes together





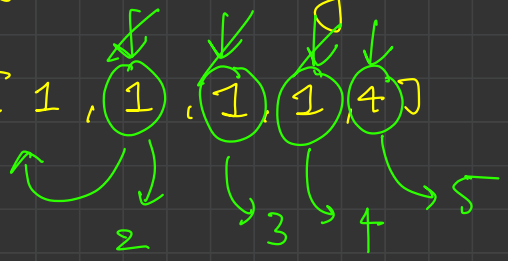




arr: [1, 1, 1, 1, 4]

① Sort the array

② [1, 1, 1, 1, 4]



coins = 1 + 2 + 3 + 1

= 7 coins

```
fun ( int[] arr , int N)
{
    Arrays.sort(arr);
    int coins = 0;
    for( int i = 1 ; i < N ; i++)
    {
        if( arr[i] <= arr[i-1])
        {
            coins+= (arr[i-1]+1) - arr[i];
            arr[i] = arr[i-1]+1;
        }
    }
} return coins;
```

```java
public static long increaseIt(int[] arr, int N) {
    // Code here
    Arrays.sort (arr);
    int coins = 0;
    for (int i = 1; i < N; i++) {
        if (arr[i - 1] >= arr[i]) {
            coins += (arr[i - 1] + 1) - arr[i];
            arr[i] = arr[i - 1] + 1;
        }
    }
    return coins;
}
```

(N log N)

N

2 - 1 = 1 coin

3 - 1 = 2

4 - 1 = 3

5 - 4 = 1

Coins = 1 + 2 + 3 + 1 = 7

arr[] = { 1, 1, 4, 1, 1 }

{ 1, 1, 1, 2, 4 }

2 3 4 5

[1, 2, 3, 4, 5]

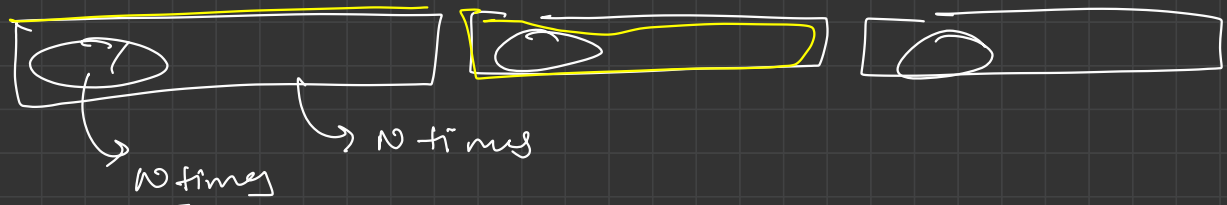
① duplicates together

② asc. order

{ TC : O(N log N)
SC : O(1) }

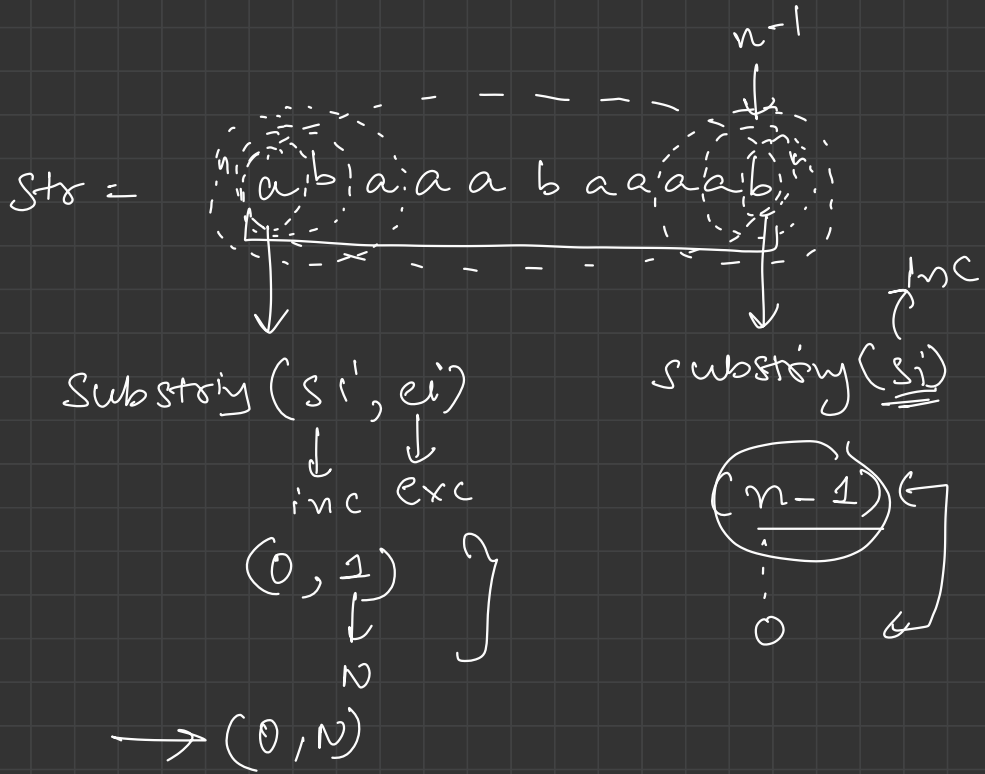
Str = "abc abc abc xyz mn ab abc no abc"

Str = "aaaaaa baaaaa"



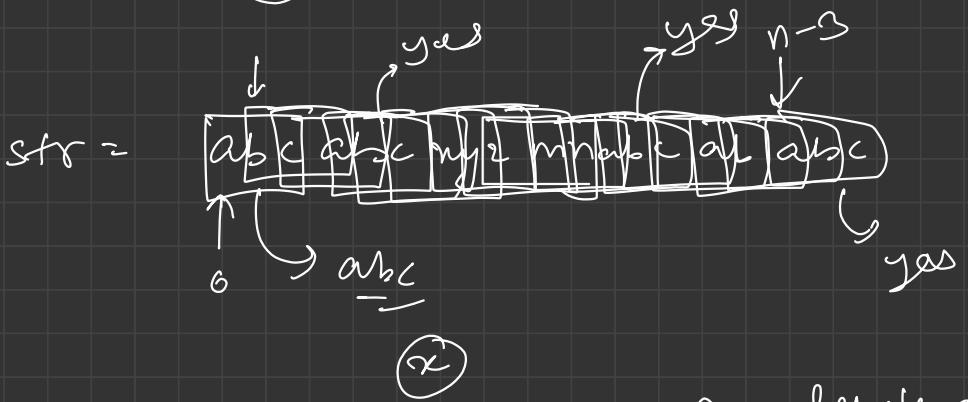
① choose a smaller substring
→ with starts the string
→ ends the string

② gets its frequency



str = "b"

b → 1



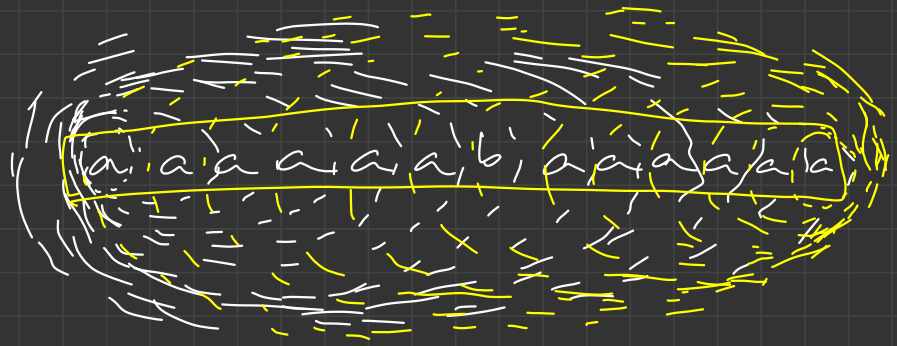
x

n - length of substring

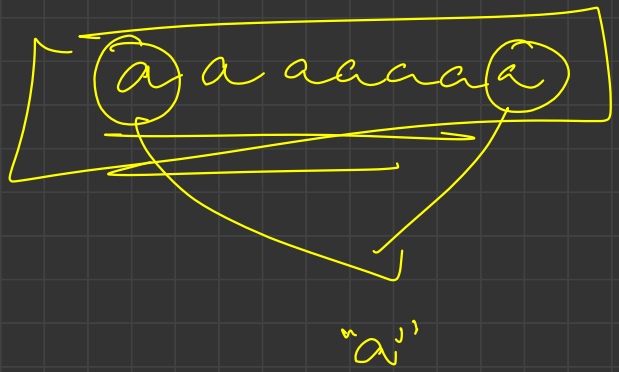
K 4

1, K+n

1, 4 → 3



Story =


"a"