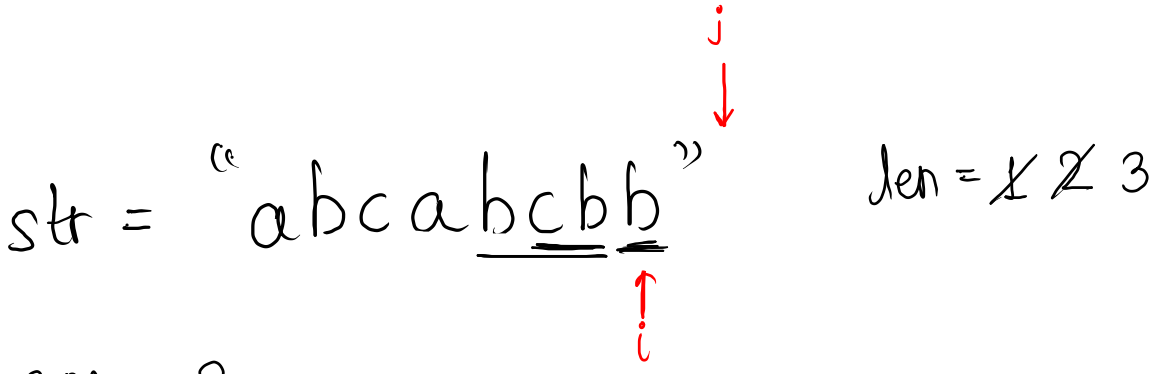
Longest Substring Without Repeating Characters 6



psudo code

- keep moving j pointer until there is no repetation
- keep moving i pointer until ther is repetation

str = " abcacbdcef "

0 1 2 3 4 5 6 7 8 9

i j

len = ~~1 2 3 4~~ 5

Substring

a
ab
abc
abca
bca
bcac
cac
ac

acb
acbd
acbdc
cbdc
bdc
bdce
bdcef

ope :- 2*N

T.C = O(2N)

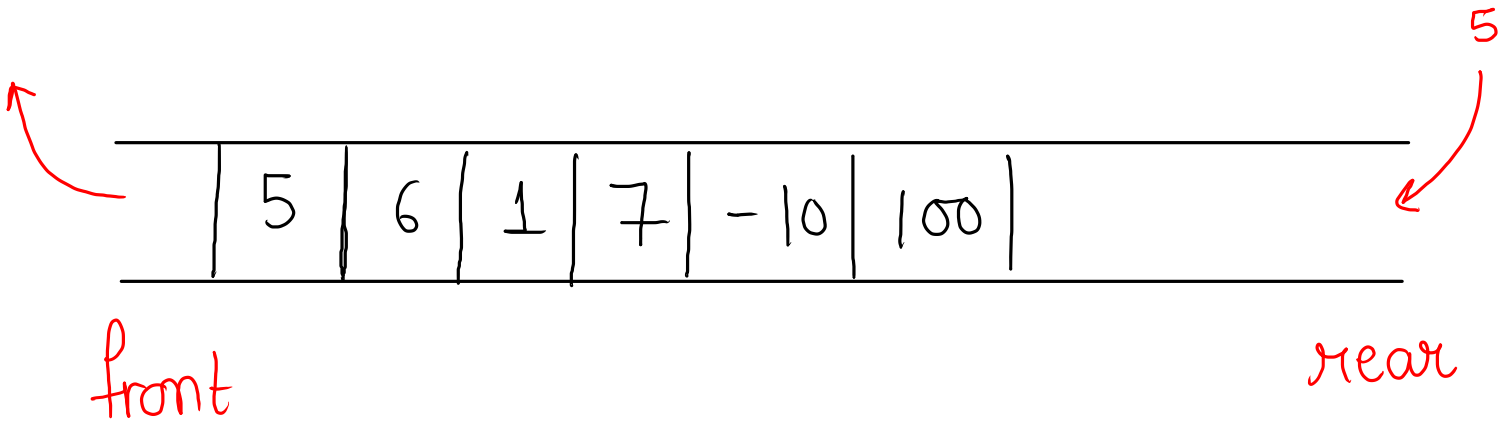
≅ O(N)

code

T.C = O(N) , S.C = O(N)

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    String str = scn.nextLine();
    System.out.println(longestSubstringWithoutRepeatingCharacters(str));
}
public static int longestSubstringWithoutRepeatingCharacters(String str) {
    int i = 0;
    int j = 0;
    HashSet<Character> set = new HashSet<>();
    int ans = 0;
    while ( j < str.length() ) {
        if ( set.contains( str.charAt(j) ) ) {
            set.remove( str.charAt(i) );
            i++;
        } else {
            ans = Math.max( ans, j - i + 1 );
            set.add( str.charAt(j) );
            j++;
        }
    }
    return ans;
}
```

⇒ Queue [FIFO :- first in first out]



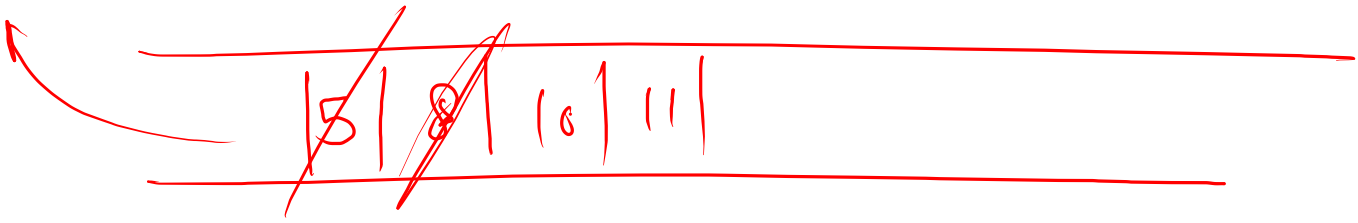
Note :- always add elements from rear
and always remove elements from front

Syntax

```
Queue<Integer> que = new LinkedList<>();
```

inbuilt fn

- `que.add(x);` // add element from rear end
- `que.remove();` } // used to remove element from front end
- `que.poll();` }
- `que.peek();` // return front element
- `que.size();`
- `que.isEmpty();`




```
que.add(5);
que.add(8);
que.size(); //2
que.peek(); //5
que.poll(); //5
que.peek(); //8
```

```
que.isEmpty(); // false
que.add(10);
Que.add(11);
que.remove(); //8
que.peek(); //10
```

Queue Syntax Learning


```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    Queue<Integer> que = new LinkedList<>();
    int t = scn.nextInt();
    for (int i = 0; i < t; i++) {
        int c = scn.nextInt();
        if ( c == 1 ) {
            System.out.println(que.size());
        } else if ( c == 2 ) {

            if ( que.isEmpty() ) {
                System.out.println("-1");
            } else {
                int ans = que.poll();
                // System.out.println(ans);
            }

        } else if ( c == 3 ) {

            int x = scn.nextInt();
            que.add(x);

        } else if ( c == 4 ) {

            if ( que.isEmpty() ) {
                System.out.println("-1");
            } else {
                int ans = que.peek();
                System.out.println(ans);
            }

        }
    }
}
```

Print Binary

decimal

00	10	20		91	100	110	120		91		
01	11	21		92	101	111	121		92		
02	12	22		93	102	112	122		93		
03	13	23		94	103	113	123		94		
04	14	24		95	104	114	124		95		
05	15	25		96	105	115	125		96		
06	16	26		97	106	116	126		97		
07	17	27		98	107	117	127		98		
08	18	28		99	108	118	128		99		
09	19	29			109	119	129				

binary no.

0 ✓

1 ✓

10 ✓

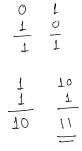
11 ✓

100 ✓

101 ✓

110 ✓

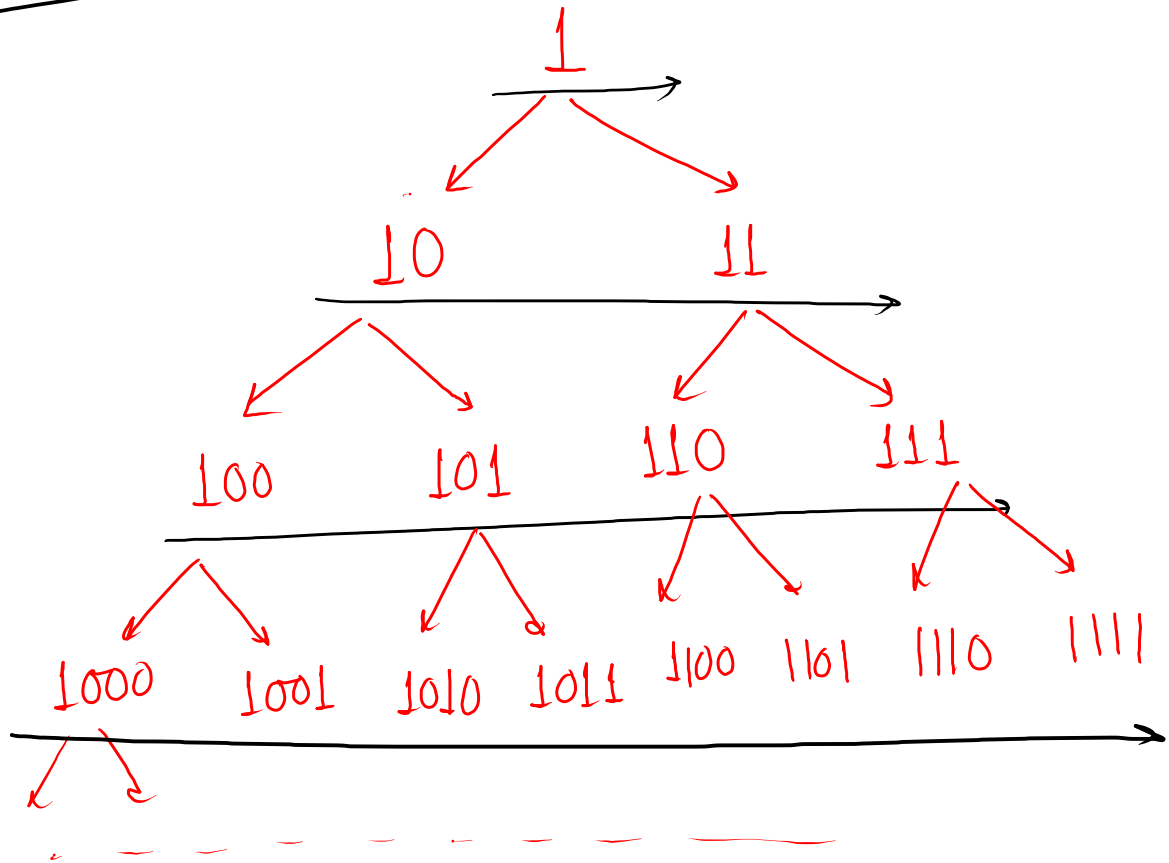
111 ✓

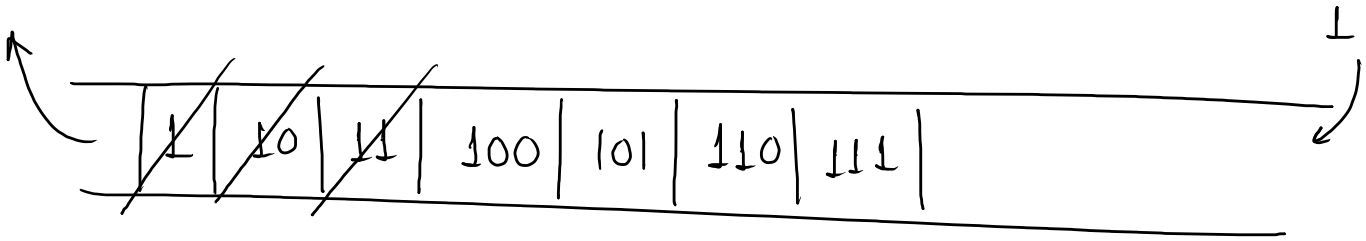


Tree structure

left branch = add 0

right branch = add 1





ans = 1 10 11 100 . . . . . .

code

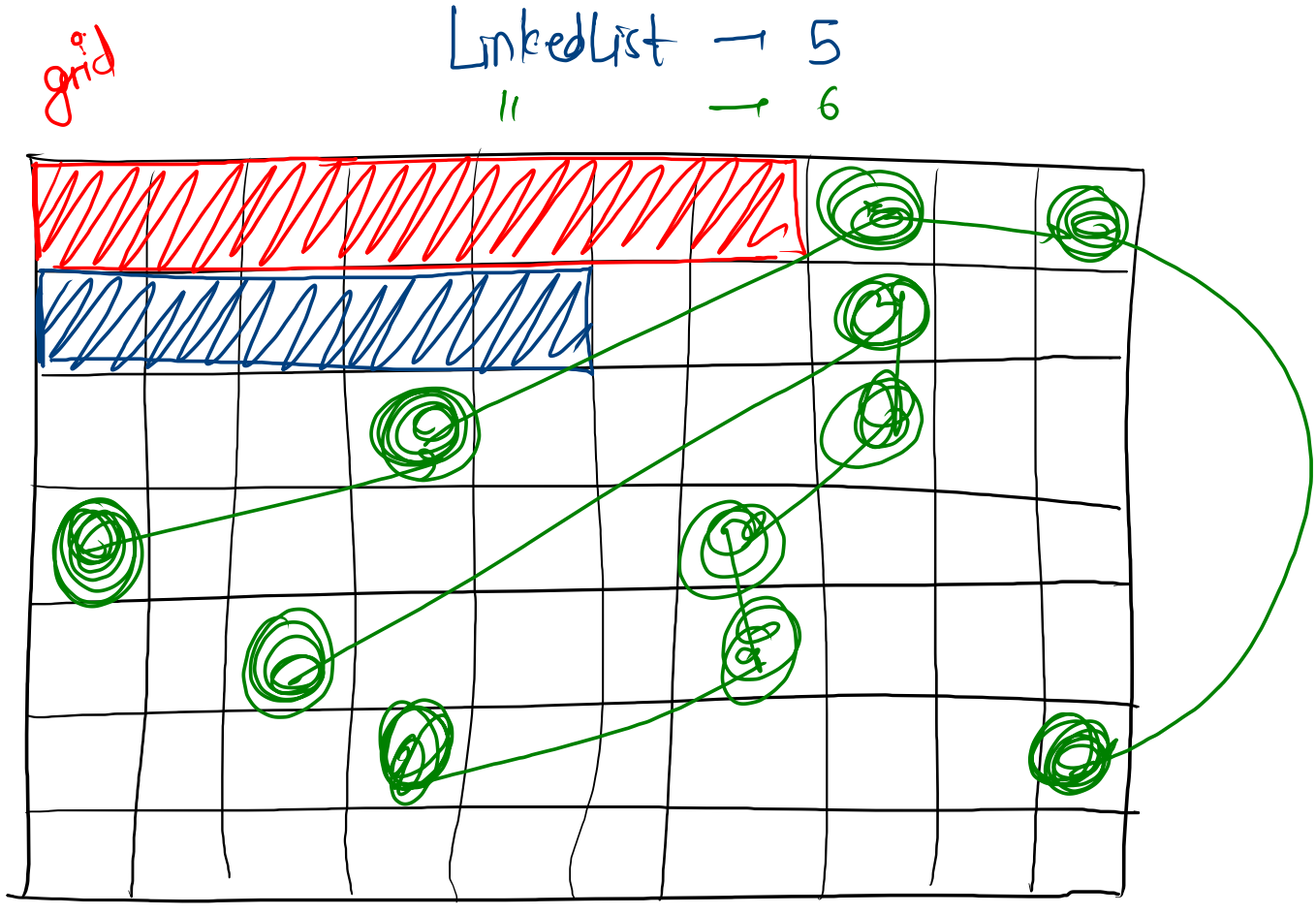
```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    printBinary(n);
}
public static void printBinary(int n) {
    Queue<String> que = new LinkedList<>();
    que.add("1");
    for ( int i = 1; i <= n; i++ ) {
        String ele = que.poll();
        System.out.print(ele + " ");

        String str1 = ele + "0";
        que.add(str1);

        String str2 = ele + "1";
        que.add(str2);
    }
}
```

+0

int "100" + '0'

1000

0

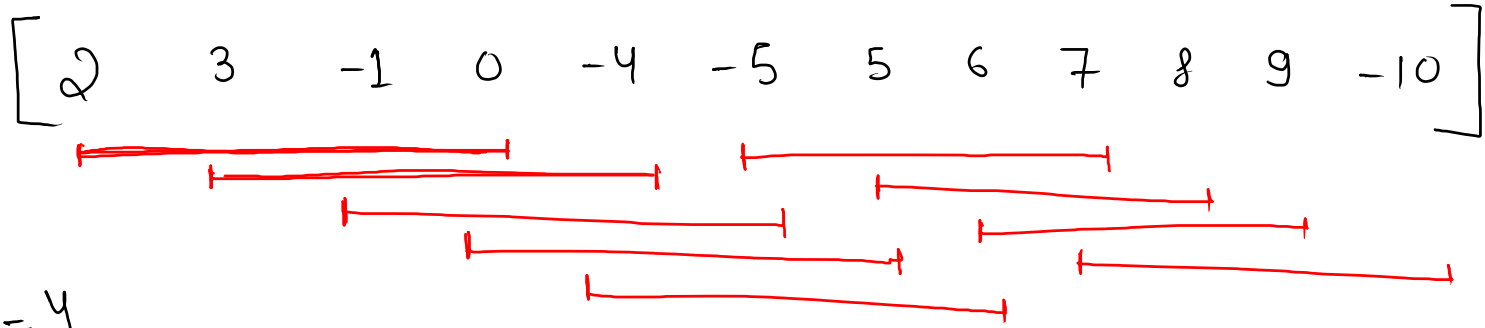

grid
LinkedList → 5
|| → 6

First Negative Integer 2

arr = -8 2 3 -6 10

k = 2

ans = -8 0 -6 -6



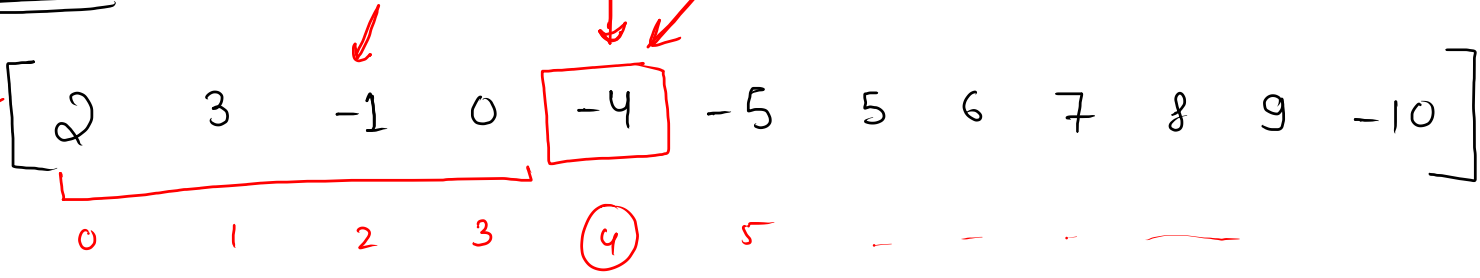
K = 4

ans :- -1 -1 -1 -4 -4 -5 0 0 -10

Brute force :- $(n-K) * K$

$k = 4$

arr



↳ que will contain only -ve no.'s

↳ instead of no., add index in que