Long pressed

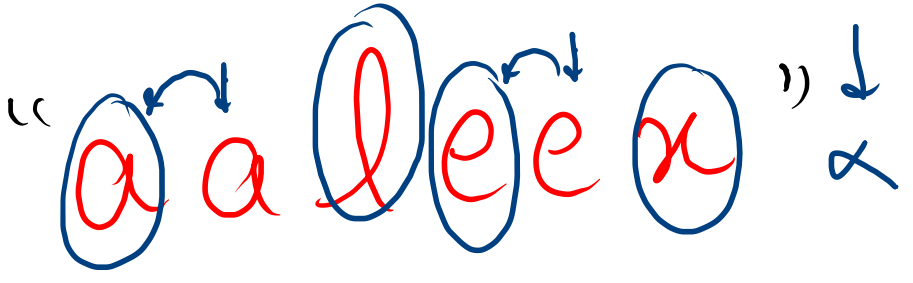
```java
public static boolean longPressed(String str, String tar) {
    int i = 0;
    int j = 0;
    while (j < tar.length()) {
        if ( i < str.length() && str.charAt(i) == tar.charAt(j) ) {
            i++;
        } else if ( j == 0 || tar.charAt(j) != tar.charAt(j - 1) ) {
            return false;
        }
        j++;
    }

    return i == str.length();
}
```

traverse

int

int

checking the sequence



True / false

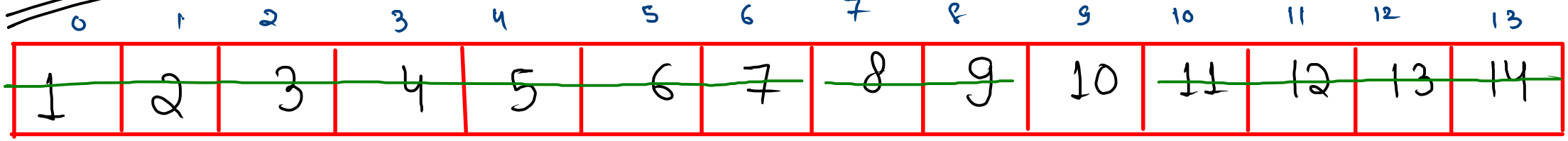
⇒ Binary Search T.C = O(logN)

↳ it is a searching algorithm.

↳ it can be applied only on sorted series

Note:- if in question, given series is sorted and it is expecting better T.C than O(N) means we binary search

series (sorted) len = n



si

si = 0 , ei = n-1 ;

int mid = (si + ei) / 2 ;

target = 10

find idx of target

ans = 10

mid = (9+9)/2 = 9

a) arr[mid] == target , return mid

b) arr[mid] > target , ei = mid - 1 ;

c) arr[mid] < target , si = mid + 1 ;

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }
    int target = scn.nextInt();

    binarySearch(arr, n, target);
}

public static void binarySearch(int[] arr, int n, int target) {
    int si = 0;
    int ei = n - 1;
    while ( si <= ei ) {
        int mid = (si + ei) / 2;
        if ( arr[mid] == target ) {
            System.out.println(mid);
            return;
        } else if ( arr[mid] > target ) {
            ei = mid - 1;
        } else {
            si = mid + 1;
        }
    }
    System.out.println(-1);
}
```

Note:-

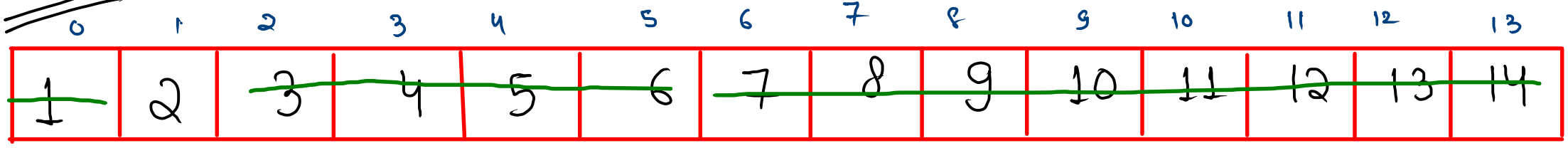
This scenario is only for unique values and not repeated ones.

Note:-

with each operation, we are discarding half of the array

means we are decreasing our range to half

series (sorted) len = n



ei

si

mid

target = 2

ans = 1

mid = (0 + 13)/2 = 6

= (0 + 5)/2 = 2

= (0 + 1)/2 = 0

= (1 + 1)/2 = 1

length = n

$$n + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \frac{n}{16} + \cdots\cdots\cdots + 1 = \log(n)$$

(Tylor series)

Proof of T.C = log(N)

replace n with n/2

$$T(n) = 1 + T(n/2) \quad \left[\frac{n}{2^1}\right]$$

$$T\left(\frac{n}{2}\right) = 1 + T\left(\frac{n}{4}\right) \quad \left[\frac{n}{2^2}\right]$$

$$T\left(\frac{n}{4}\right) = 1 + T\left(\frac{n}{8}\right) \quad \left[\frac{n}{2^3}\right]$$

$$T\left(\frac{n}{2^{k-1}}\right) = 1 + T\left(\frac{n}{2^k}\right) \quad \left[\frac{n}{2^k}\right]$$

K operation

$$T(n) = \underbrace{(1+1+1\ldots\ldots+1)}_{K \text{ times}} + T\left(\frac{n}{2^k}\right)$$

$T\left(\frac{n}{2^k}\right) \to 1$

$$\Rightarrow \frac{n}{2^k} = 1$$

$$\Rightarrow n = 2^k$$

$$\Rightarrow \text{take } \log_2 \text{ both side}$$

$$\Rightarrow \log_2(n) = \log_2(2^k)$$

$$\log_2(n) = k \log_2(2)$$

$$\log(n) = k * 1 = k$$

binary search complexity is O(k)

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

Find Square Root

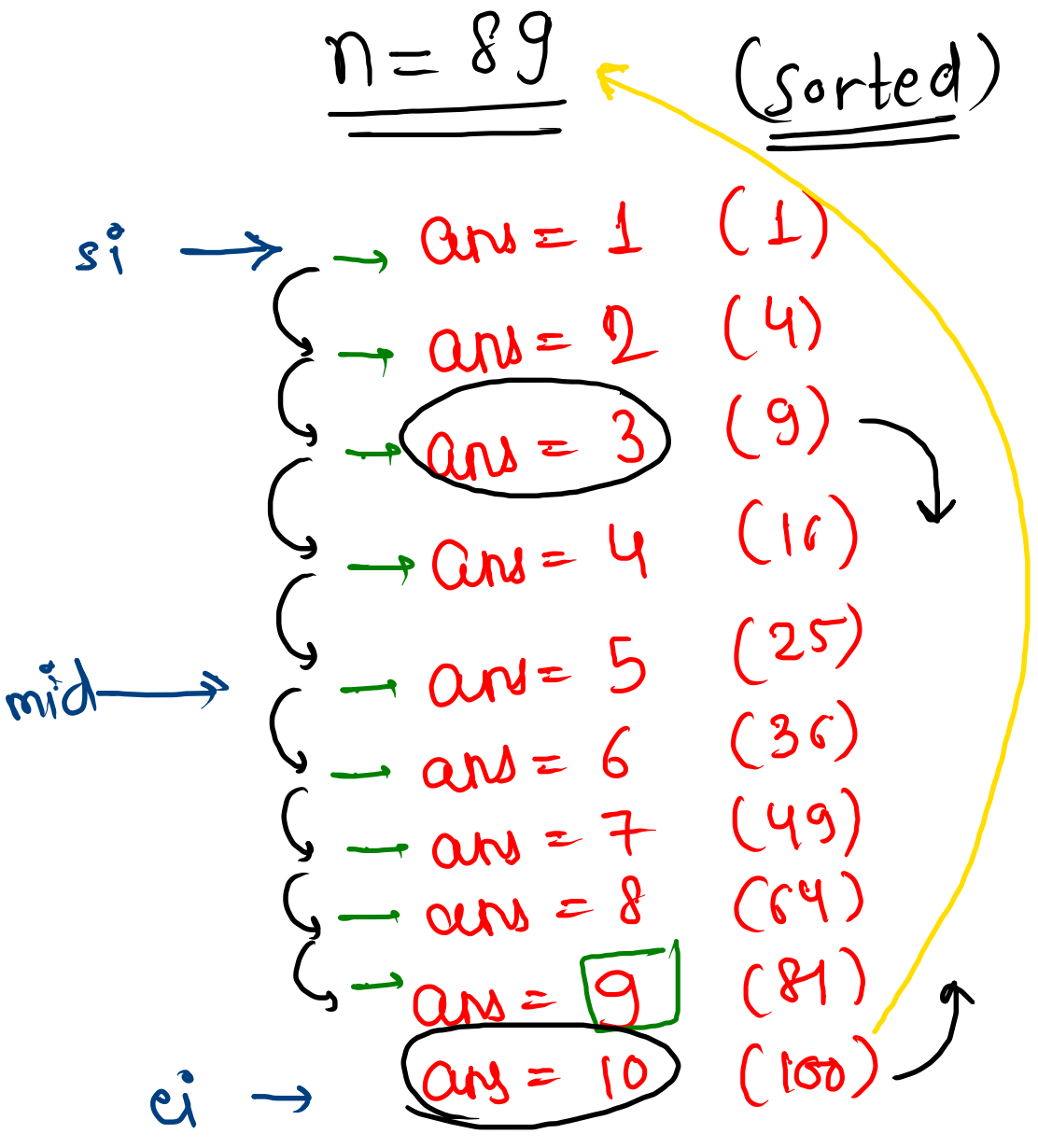
$n = 25$, $ans = 5$

$n = 37$, $ans = 6$

example

$n = 12$,

	Square
$ans = 1$,	$1 \times 1 = 1$
$= 2$,	$2 \times 2 = 4$
$= 3$,	$3 \times 3 = 9$
$= 4$,	$4 \times 4 = 16$



assuming $n = 89$

(magic of B.S is you can traverse in imaginary range as well

$si = 1$, $ei = n$

```
int mid = (si + ei) / 2;
if (mid * mid > n) {
    ei = mid - 1;
} else {
    si = mid + 1;
}
```

val = arr[mid] ✗

val = mid ✓

S.C = O(1)

T.C = O(logN)

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

    binarySearch(n);
}

public static void binarySearch(int n) {
    int si = 1;
    int ei = n;
    while ( si <= ei ) {
        int mid = (si + ei) / 2;
        if ( mid * mid <= n ) {
            si = mid + 1;
        } else {
            ei = mid - 1;
        }
    }
    System.out.println(ei);
}
```

a

b

n = 225

si = 1, ei = 225, mid = 113

si = 1, ei = 112, mid = 56

si = 1, ei = 55, mid = 28

si = 1, ei = 27, mid = 14

si = 15, ei = 27, mid = 21

si = 15, ei = 20, mid = 17

si = 15, ei = 16, mid = 15

si = 16, ei = 16, mid = 16

si = 16, ei = 15, mid = 15