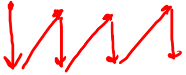
Merge Strings Alternatively

(2 pointers)

0 1 2 3

str1 = "GEEK"



str2 = "STER"

0 1 2 3

ans = "GSETEEKR"

i

↓

str1 = "Kunal"

0 1 2 3 4

str2 = "Banti"

0 1 2 3 4

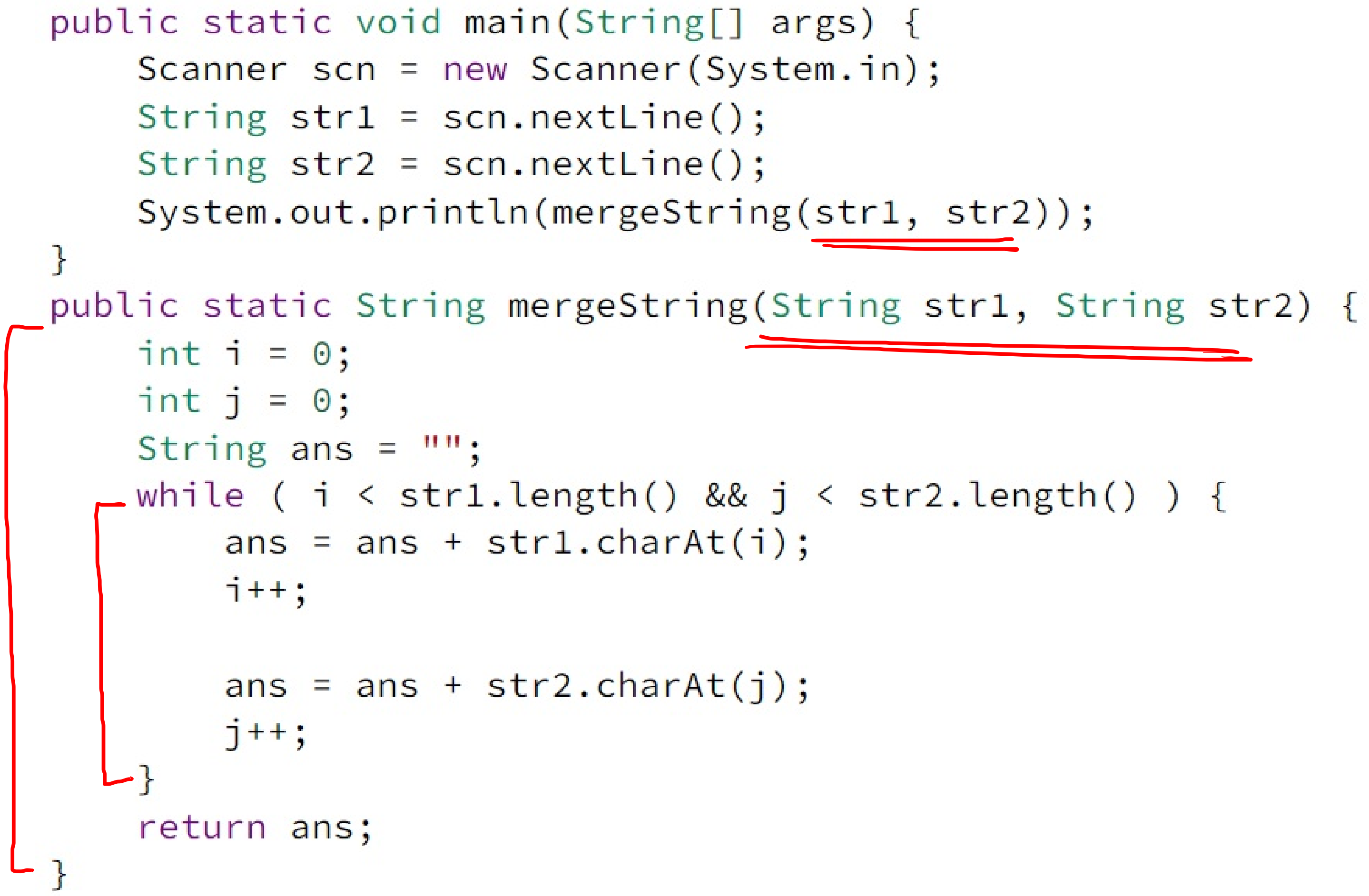
↑

j

ans = "KBuannatli"

Code

T.C = $O(N)$ → where N is str1.len + str2.len



```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    String str1 = scn.nextLine();
    String str2 = scn.nextLine();
    System.out.println(mergeString(str1, str2));
}
public static String mergeString(String str1, String str2) {
    int i = 0;
    int j = 0;
    String ans = "";
    while ( i < str1.length() && j < str2.length() ) {
        ans = ans + str1.charAt(i);
        i++;

        ans = ans + str2.charAt(j);
        j++;
    }
    return ans;
}
```

Long Pressed Name

0 1 2 3

str = "alex"

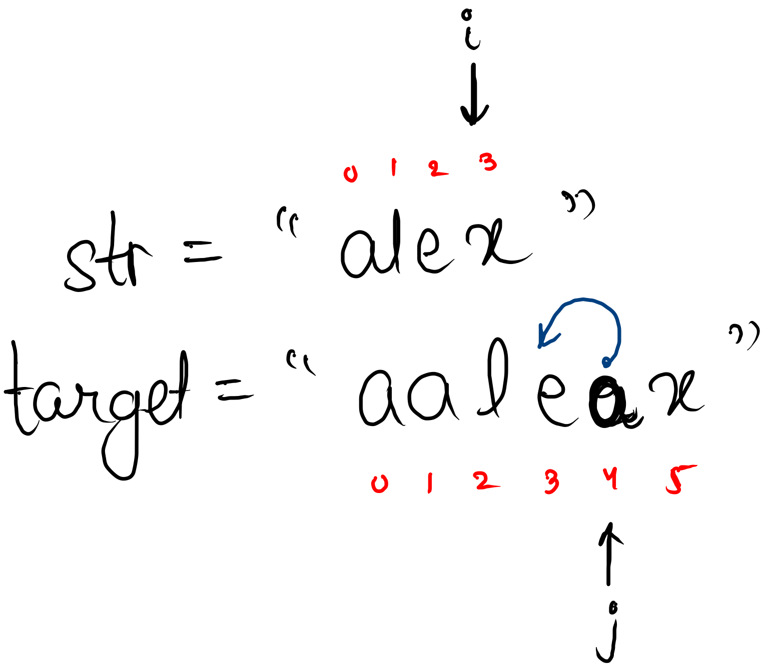
target = "aaleex"

0 1 2 3 4 5

Observations

↳ each char of str should be there in target

↳ and also should be in same order



str = "alex"

tar = "aaleex"

true

```
if char at i == char at j
        i++, j++
else
        j++
```

psudo code

```
1) initialise i pointer at 0 and j pointer at 0
2) loop until j < tar.length
    2.1) if char at i == char at j
             i++
    2.2) else if char at j != char at(j-1)
             return false
    j++
```

i

str = "alex"

target = "aaleex$\underline{abc}$"

j

i

str = " "

tar = "a"

j

false

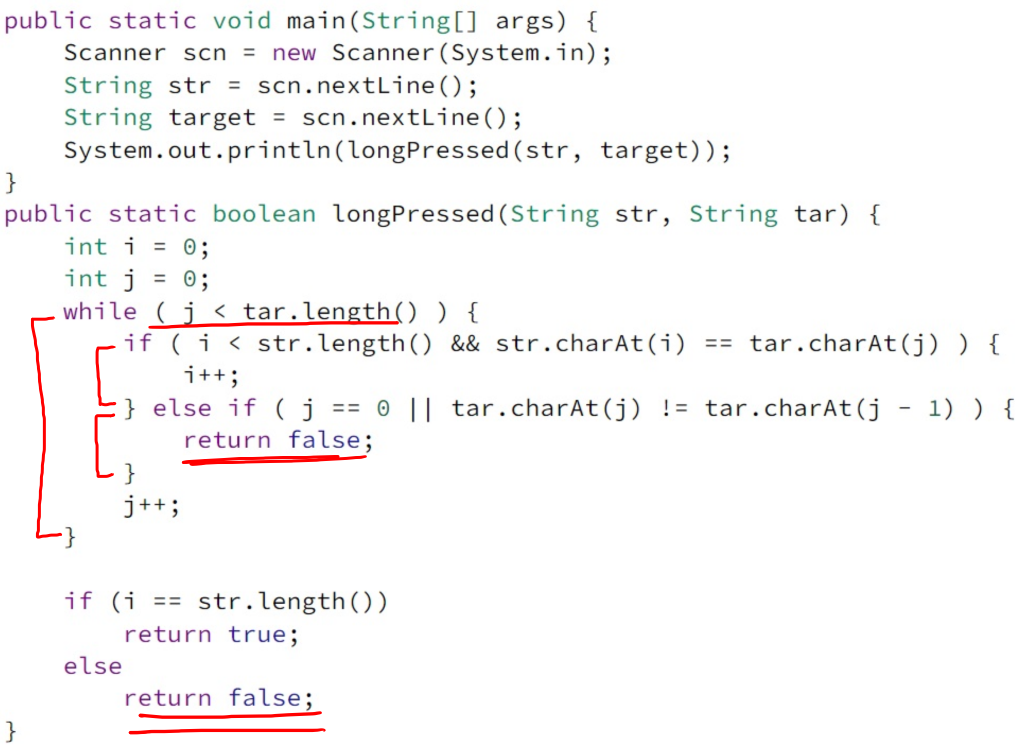
i

str = "kunal"

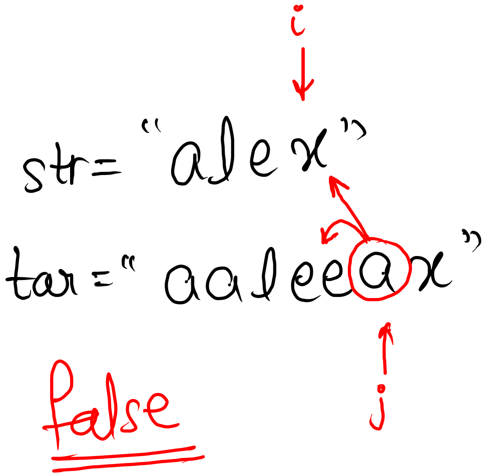
tar = "uunaall"

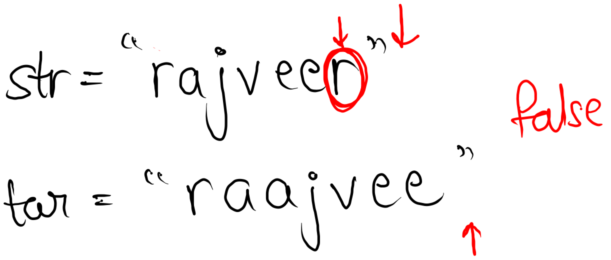
j

Code



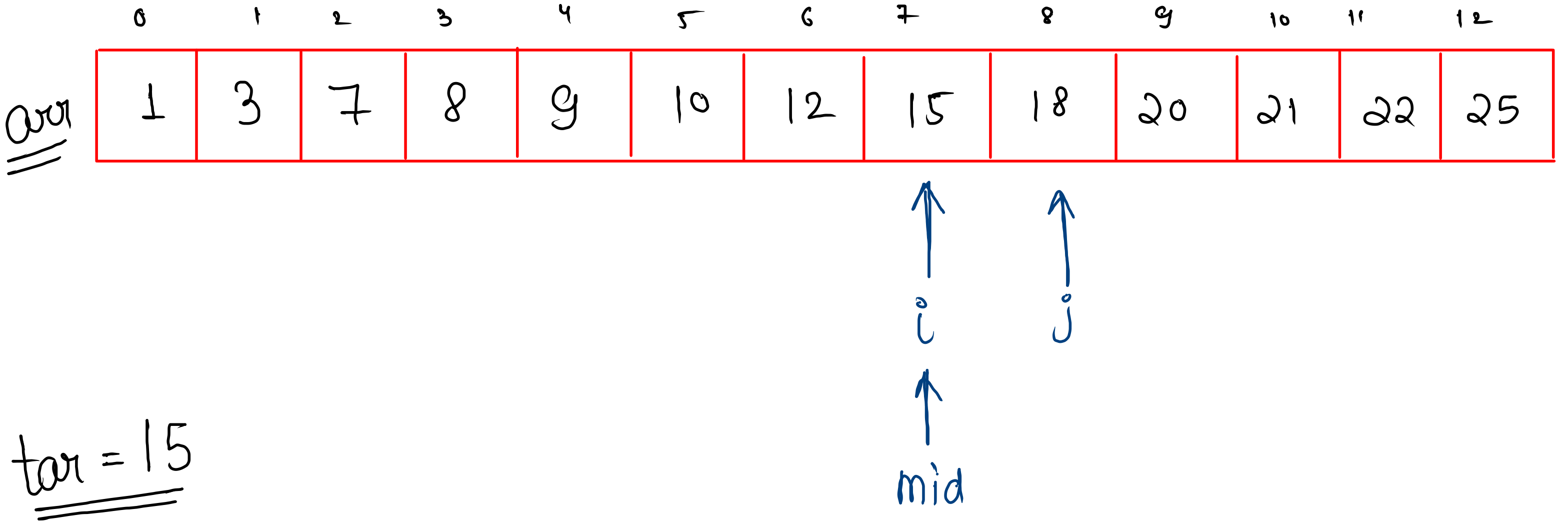
```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    String str = scn.nextLine();
    String target = scn.nextLine();
    System.out.println(longPressed(str, target));
}
public static boolean longPressed(String str, String tar) {
    int i = 0;
    int j = 0;
    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++;
    }

    if (i == str.length())
        return true;
    else
        return false;
}
```





⇒ Binary Search (searching algorithm) in O(log N) time

↳ pre-requisite condition :- array must be sorted

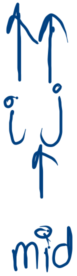


	0	1	2	3	4	5	6	7	8	9	10	11	12
arr	1	3	7	8	9	10	12	15	18	20	21	22	25

tar = 15

```
int i=0, j=n-1;
while( i<=j) {
    mid= (i+j)/2;
    if ( tar == arr[mid]) {
        return true;
    } else if( tar < arr[mid]) {
        j= mid-1;
    } else {
        i = mid+1
    }
}
```

	0	1	2	3	4	5	6	7	8	9	10	11	12
arr	1	3	7	8	9	10	12	15	18	20	21	22	25



tar = 5

mid = ~~6~~ ~~2~~ 1

```
int i=0, j=n-1;
while( i <= j) {
    mid = (i+j)/2;
a [ if ( tar == arr[mid]) {
        return true;
b [ } else if( tar < arr[mid]) {
        j= mid-1;
c [ } else {
        i = mid+1
    }
}
return false;
```

Why T.C of B.S is log(N)

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

Taylor's eq.