AIM:

To find the minimum number of adjacent swaps required to make k ones appear consecutively in a given binary array.

ALGORITHM:

1 Read the array nums and integer k

2 Store the indices of all elements equal to one in a list

3 If k equals one then return zero because one element is already consecutive

4 Use a sliding window of size k on the list of one indices

5 For each window compute the median index position

6 Calculate the total number of swaps required to bring all ones in that window together around the median

7 Adjust the cost by subtracting the natural positions offset for consecutive placement

8 Keep track of the minimum cost over all windows

9 Output the minimum value

PROCEDURE:

1 Start the program

2 Input the binary array nums and integer k

3 Create a list to store the positions of all ones

4 Slide a window of size k across this list

5 For each group of k ones determine the middle position

6 Compute how many adjacent swaps are needed to move those ones into consecutive locations

7 Compare with the previously stored minimum and update if smaller

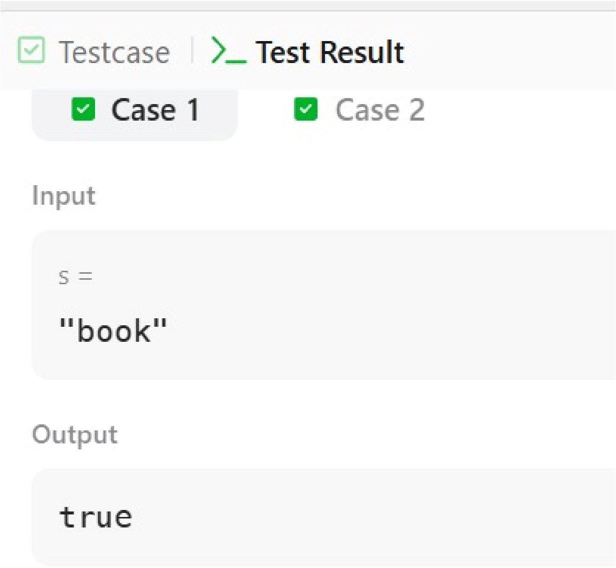
8 Display the minimum number of swaps

9 Stop the program

PROGRAM:

```
public class Solution {
   public boolean halvesAreAlike(String s) {
      Set<Character> vowels = new HashSet<>();
      vowels.add('a'); vowels.add('e'); vowels.add('i'); vowels.add('o'); vowels.add('u');
      vowels.add('A'); vowels.add('E'); vowels.add('I'); vowels.add('O'); vowels.add('U');

      int length = s.length();
      int midPoint = length / 2;

      String firstHalf = s.substring(0, midPoint);
      String secondHalf = s.substring(midPoint);

      return countVowels(firstHalf, vowels) == countVowels(secondHalf, vowels);
   }

   private int countVowels(String str, Set<Character> vowels) {
      int count = 0;
      for (char c : str.toCharArray()) {
         if (vowels.contains(c)) {
                count++;
            }
        }
        return count;
    }
}
```

OUTPUT:



☑ Testcase | >_ Test Result

☑ Case 1 ☑ Case 2

Input

```
s =
"book"
```

Output

```
true
```

RESULT:

The program successfully calculates the minimum number of adjacent swaps required to make k ones consecutive in the array and outputs the smallest possible value for the given input.

AIM:

To write a program that checks whether a given string is a Lapindrome that is a string whose two halves contain the same characters with the same frequency after ignoring the middle character when the length is odd.

ALGORITHM:

1 Read the integer T which represents the number of test cases

2 For each test case read the string S

3 Find the length N of the string

4 If N is even then take the first N by 2 characters as the left half and the remaining characters as the right half

5 If N is odd then take the first N by 2 characters as the left half ignore the middle character and take the remaining characters as the right half

6 Convert both halves into character arrays

7 Sort both arrays

8 Compare the two sorted arrays

9 If they are equal print YES otherwise print NO

PROCEDURE:

1 Start the program

2 Accept the input value T

3 For each test case input the string

4 Compute the length of the string

5 Divide the string into two halves based on whether the length is even or odd

6 Sort both halves

7 Compare the two halves

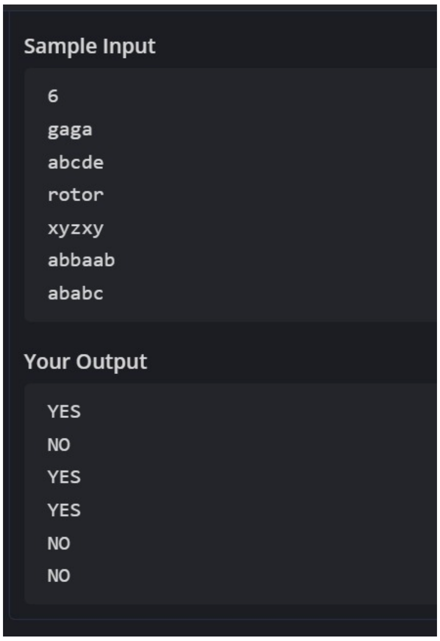
8 Display the result

9 Stop the program

PROGRAM:

```
import java.util.*;

class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int T = sc.nextInt();

        while (T-- > 0) {
            String s = sc.next();
            int n = s.length();

            String left, right;

            // Split into halves
            if (n % 2 == 0) {
                left = s.substring(0, n / 2);
                right = s.substring(n / 2);
            } else {
                left = s.substring(0, n / 2);
                right = s.substring(n / 2 + 1);
            }

            char[] a = left.toCharArray();
            char[] b = right.toCharArray();

            Arrays.sort(a);
            Arrays.sort(b);

            if (Arrays.equals(a, b)) {
                System.out.println("YES");
            } else {
                System.out.println("NO");
            }
        }

        sc.close();
    }
}
```

OUTPUT:



Sample Input

```
6
gaga
abcde
rotor
xyzxy
abbaab
ababc
```

Your Output

```
YES
NO
YES
YES
NO
NO
```

RESULT:

The program correctly identifies whether the given string is a Lapindrome. If both halves contain the same characters with the same frequency it prints YES otherwise it prints NO.