Problem

> Given two strings A and B. Find the minimum number of steps required to transform string A into string B. The only allowed operation for the transformation is selecting a character from string A and inserting it in the beginning of string A.

Example

Input:

```
A = "abd"
B = "bad"
```

Output:

```
1
```

Explanation:

```
The conversion can take place in
1 operation: Pick 'b' and place it at the front.
```

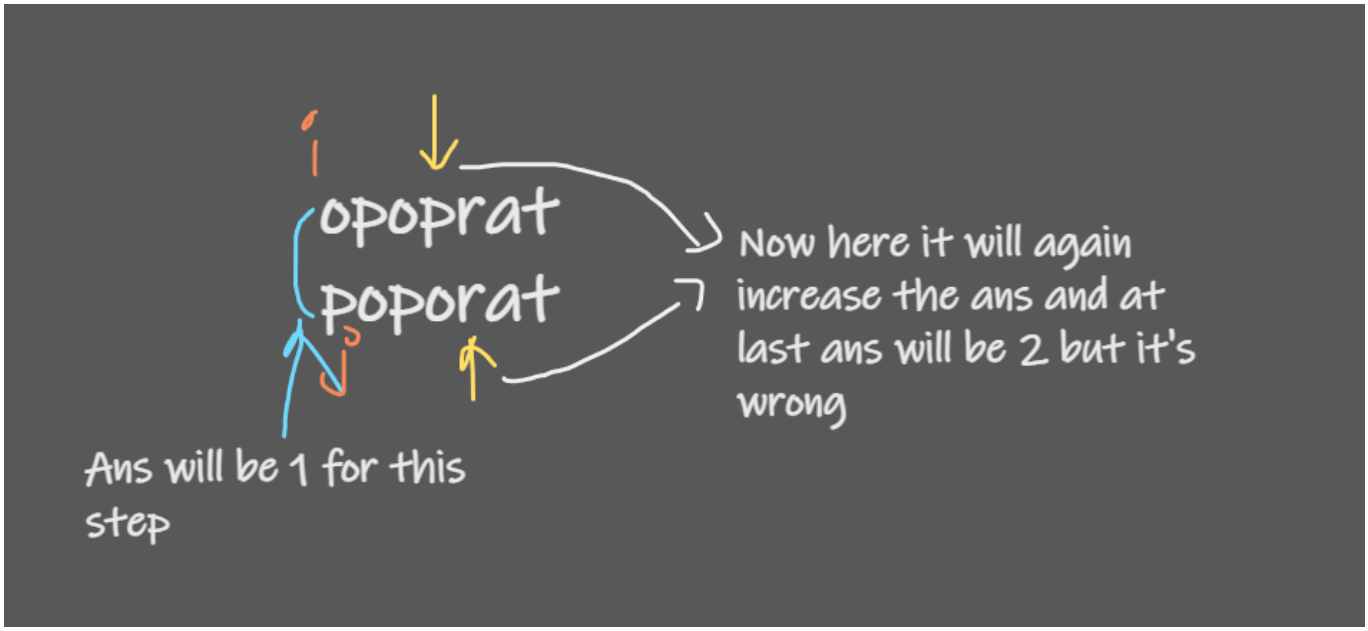
Approach:

⟶ Here we will use 2 pointer approach and we will start from end of the both strings.

For example we have 2 pointers `i` and `j` and now we have 2 conditions here:

1. If `a[i]==b[j]` then we will decrease both i and j
2. If `a[i]!=b[j]` then we will Only decrease i and increase the ans

we could start from `0` then we have to increase `j` if both elements are not equal and if both are equal then increase both i and j. But this logic gives wrong answer which we can see in below image.



So that's why we have to start i and j from `n-1`

Final code will look like this:

```
int transform (string A, string B)
    {
        unordered_map<char,int>freq;
        for(int i=0;i<A.size();i++){
            freq[A[i]]++;
        }
        for(int i=0;i<B.size();i++){
            freq[B[i]]--;
        }
        for(auto it:freq){
                        // If any variable is different in both
strings then we can't get desired string so we will return -1.
                        // Because both strings have equal and
same number of characters.
            if(it.second != 0){
```

```
                return -1;
            }
        }
        int n = A.size();
        int i = n-1,j=n-1;
        int ans = 0;
        while(i >= 0 && j >= 0){
            if(A[i] == B[j]){
                i--;
                j--;
            }
            else if(A[i] != B[j]){
                i--;
                ans++;
            }
        }
        return ans;
    }
```