

In this article, we are going to see another important problem with the string. We have to find the total number of minimum swaps which are required to group similar characters in a given string side by side. Let’s understand this problem statement with some examples to understand the problem statement better.

This question is also one of the imprtant exmaple of string interpretation. In this example we have to count the minimum number fo swaps which are required to group similar characters side by side. Let'’s understand with few examples.

**To show you some examples:**

**Example 1:**

1. The given input string is “abcb”.
2. Now in this string first see how many characters are the same. As in this case we have 3 unique characters with ‘b’ two times.
3. Now, our task is to pace those two b’s together in a minimum number of swaps.

In this case, if we swap ‘c’ and ‘b’ then it becomes “acbb” which is our desired outcome. So our output is 1 as it requires only 1 swap. 

**Example 2:**

Suppose the input string is “abbaacb”.

In this case, we have 3 unique characters.

Now we have to swap the characters in such a way that all the same characters are put aside.

For this, if we swap first ‘a’ with ‘c’ and third ‘b’  with second ‘a’ then we will get the string as “cbbbaaa”.

As we can see this is the string that we want because in this string similar characters are side by side. So in this case the output is 2 as we swap the characters two times.

With the help of the above examples, we understand the problem statement. Now it's time to solve this.

Approach 1: Recursive Approach

Let’s first solve this question using a recursive approach. We are going to see first the algorithm followed by its explanation and then the C++ program.

**Algorithm**

1. Create a boolean function to check if any two characters next to one another in a string match then it will return true.
2. Create a recursive function minSwaps which returns the minimum swaps required to get the desired output. This function takes the input string, left index, right index, current count, and minimum as input.
3. Initialize the current count and minimum in the main function.
4. In the recursive function, set a base case as If there are no adjacent characters in the string that are the same, return the count if the left index matches the right index.
5. Go through a for loop and swap and then call minSwaps repeatedly with the modified values.
6. At last, find the minimum value of swaps and return it.

Explanation

In this approach, we are using a recursive approach to solve the problem statement. We are using two functions: the first one is a boolean function to check adjacent characters and a recursive function that returns the minimum swaps which is our desired output. The goal is to take into account every permutation that results from a two-element swap as well as those that don't involve doing so.

**Program Code:**

#include <iostream>

using namespace std;

bool sameCharAdj(string str){

int n = str.length(), i;

set<char> st;

st.insert(str[0]);

for (i = 1; i < n; i++) {

if (str[i] == str[i - 1])

continue;

if (st.find(str[i]) != st.end())

return false;

st.insert(str[i]);

}

return true;

}

int minSwaps(string str, int l, int r, int ct, int min){

    if (l == r){

        if (sameCharAdj(str))

            return ct;

        else

            return INT\_MAX;

    }

    for (int i = l + 1; i <= r; i++){

        swap(str[i], str[l]);

        cnt++;

        int x = minSwaps(str, l + 1, r, ct, min);

        swap(str[i], str[l]);

        cnt--;

        int y = minSwaps(str, l + 1, r, ct, min);

        minm = min(minm, min(x, y));

    }

    return min;

}

int main(){

string str = "abbaacb";

int n = str.length(), ct = 0, min = INT\_MAX;

cout << "Minimum swaps = " << minSwaps(str, 0, n - 1, ct, min) << endl;

return 0;

}

**Output:**

Minimum swaps = 2

We learn a recursive approach to solve the given problem. Let’s see another approach to solve this.

**Approach 2: Non-recursive Approach**

In this approach, we are simply using an unordered map to solve the given problem statement. It will help us to find out the minimum swaps to group similar characters side by side.

**Algorithm**

1. Create a function that returns the minswap required, it takes input as a string.
2. Create an unordered map to record the positions of each character in the string.
3. Traverse through the input string and fill in the indexes of characters in the map.
4. Swap the characters to get the desired output and count the number of swaps.
5. Return the number of swaps as output.

**Program Code:**

#include <iostream>

#include <vector>

#include <unordered\_map>

using namespace std;

int countMinSwaps(string str) {

    unordered\_map<char, vector<int>> charIndices;

    for(int i=0; i<str.length(); i++) {

        charIndices[str[i]].push\_back(i);

    }

    int swaps = 0;

    for(auto it=charIndices.begin(); it!=charIndices.end(); it++) {

        vector<int> indices = it->second;

        for(int i=0; i<indices.size()-1; i++) {

            if(indices[i+1] - indices[i] > 1) {

                swap(str[indices[i]], str[indices[i+1]]);

                swaps++;

            }

        }

        for(int i=0; i<indices.size(); i++) {

            charIndices[str[indices[i]]][i] = indices[i];

        }

    }

    return swaps;

}

int main() {

    string str = "abbaacb";

    int result = countMinSwaps(str);

    cout << "Minimum number of swaps needed: " << result << endl;

    return 0;

}

**Output:**

Minimum number of swaps needed: 2

**Conclusion**

In this article, we learn about another problem with string. We understand how to find the minimum swaps to group similar characters side by side. We solve this problem using two different approaches and code in C++ language.