In this article, we will find the minimum deletions required to reduce a string with at most two unique characters. Suppose the string given is “statement”. Then, the minimum number of deletions required in the given string is 4 by deleting (s,a,m,n), we get “tteet”, which is our desired result with two unique characters. Here, we are taking a string s, which contains lowercase English alphabets. The challenge is determining how many minimum characters must be deleted from the given string to be reduced to a string with at most two unique characters.

## **Explanation**

The problem statement instructs to determine how many character deletions must be made to the supplied string to reduce it to a string with no more than two unique characters.

Let’s understand it with a few examples:

**Example 1:**

string s = “syllabus”

Here, take two unique characters from this string so that after selecting these two characters, the remaining characters which will be deleted should be minimized.

Let’s understand with this table:

|  |  |
| --- | --- |
| **Selected characters** | **Minimum deletion required** |
| s, y (sys) | 5 (llabu) |
| s, a (sas) | 5 (yllbu) |
| a, b (ab) | 6 (syllus) |
| s, l (slls) | 4 (yabu) |

So if we select ‘s’ and ‘l’ and delete the other characters, our goal is achieved. The minimum deletion is 4.

**Example 2:**

string s = “helloworld”

|  |  |
| --- | --- |
| **Selected characters** | **Minimum deletion required** |
| h, e (he) | 8 (lloworld) |
| h, l (hlll) | 6 (eoword) |
| h, o (hoo) | 7 (ellwrld) |
| l, o (llool) | 5 (hewrd) |

So, if we select ‘l’ and ‘o’ and delete the other characters, then our goal is achieved. The minimum deletion is 5.

## **Approach 1: Using Integer Type Array**

Let’s discuss the algorithm, explanation and then code the program.

### **Algorithm**

**Step 1:** Create a function minDeletions which takes the string as input and returns the output.

**Step 2:** Store the frequency of each character in an array of size 26.

**Step 3:** Save the length of the input string.

**Step 4:** Using for loop, go through each character in the string s.

**Step 5:** Update the frequency of each character in the array.

**Step 6:** Create an integer variable minimum and initialize with INT\_MAX.

**Step 7:** Use nested loop to find the desired result and store in minimum variable.

**Step 8:** Return minimum as output.

Let’s see the explanation of this approach.

### **Explanation**

We create an integer array of size 26 to hold the frequency of each alphabet in the input string. Then, iterate over all pairs of distinct characters in the array. For each pair, calculate the sum of their frequencies. After that find the minimum of INT\_MAX and the difference in length of the input string with the sum of frequencies we get earlier.

### **Program**

|  |
| --- |
| #include <iostream>  #include <climits>  using namespace std;  int minDeletions(string s)  {  int feq[26] = {0};  int n = s.size();  for(int i = 0; i < n; i++)  {  char ch = s[i];  feq[ch - 'a'] += 1;  }  int minimum = INT\_MAX;  for(int i = 0 ; i < 26; i++)  {  for(int j = i + 1;j < 26; j++)  {  int k = feq[i] + feq[j];  minimum = min(minimum, n - k);  }  }    return minimum ;  }    int main()  {  string s;  cout << "Enter the string: ";  cin >> s;  cout << "Minimum deletions required: " << minDeletions(s);  } |

**Output**:

|  |
| --- |
| Enter the string: syllabus  Minimum deletions required: 4 |

Now let’s see another approach in which we are using unordered map to solve the given problem.

## **Approach 2: Using Unordered Map**

Here is the algorithm to solve the problem statement with unordered map.

### **Algorithm**

**Step 1:** We create an unordered\_map called "freq" which stores the frequencies of the characters given in the input string.

**Step 2:** Find the input string size and store it into an integer variable ‘n’.

**Step 3:** Using for loop increases the frequency count for each character in the freq map.

**Step 4:** Create a variable called minimum and initialize it with INT MAX.

**Step 5:** Now, with the help of two stacked loops iterate through all character pairs that are unique in the freq map.

**Step 6:** Calculate the total frequency of each pair of characters, then put it into a new variable k.

**Step 7:** Calculate the difference between the length of the input string and k.

**Step 8:** Find the minimum of the current value and the difference in the last step.

**Step 9:** After processing all the calculations, Return the minimum value.

### **Explanation**

Let's discuss explanation of this approach which provides a simple way to determine the frequency of each character in O(1) time by storing the frequency of each character in the input string in an unordered map. The nested loops iterate over the input text, computing the frequencies of all conceivable combinations of distinct characters and accumulating them in a variable called "k". The minimal variable is then updated with the least value of minimum and n - k, where n is the length of the input text. The minimum value is then returned as the outcome.

### **Program**

|  |
| --- |
| #include <iostream>  #include <unordered\_map>  #include <climits>  using namespace std;  int minDeletions(string s)  {  unordered\_map<char, int> freq;  int n = s.size();  for (int i = 0; i < n; i++) {  freq[s[i]]++;  }  int minimum = INT\_MAX;  for (auto it1 = freq.begin(); it1 != freq.end(); it1++) {  for (auto it2 = next(it1); it2 != freq.end(); it2++) {  int k = it1->second + it2->second;  minimum = min(minimum, n - k);  }  }  return minimum;  }  int main()  {  string s;  cout << "Enter the string: ";  cin >> s;  cout << "Minimum deletions required: " << minDeletions(s);  return 0;  } |

**Output**:

|  |
| --- |
| Enter the string: syllabus  Minimum deletions required: 4 |

## **Conclusion**

In this article, we understand how to find minimum deletions from a string to reduce it to a string with at most two unique characters. We also learn to code this problem in C++ programming language. Hope this article help you in solving this question in a better way.