

#### 424. Longest Repeating Character Replacement

This I found actually quite tricky. Given in the problem, we need to find a maximum same character repeating substring and maximum of k character is allowed.

In the question it is given there will be a string s and number of operations k allowed. We have to find what is the maximum length of same character substring that can form out of it.

The solution is as follows:

1. We take left and right pointers.
2. We keep count of frequency of the characters in a hashmap.
3. We increment the right pointer upto the end of the string and keep updating the count frequency and want to find which character is the most occurring in a window with a maxCnt.
4. If the number of non majority character is greater than k, we update left pointer++ and reduce the count frequency of character at left pointer.
5. In the loop, we keep track of largest sliding window we can obtain within k operations.
6. Return the answer.

Handwritten notes illustrating the sliding window process for the string "ABBBAAAB" with k=2:

String: A B B B A A A B, k=2

Step 1: A → [A] (1-1 ≤ 2) ans=1

Step 2: A B → [A, B] (2-1 ≤ 2) ans=2

Step 3: A B B → [A, B, B] (3-1 ≤ 2) ans=3

Step 4: A B B A → [A, B, B, A] (4-1 = 3 > 2) → update left pointer to 2, [B, B, A] (3-2 = 1 ≤ 2) ans=3

Step 5: B B A A → [B, B, A, A] (4-2 = 2 ≤ 2) ans=4

Step 6: B B A A A → [B, B, A, A, A] (5-2 = 3 > 2) → update left pointer to 3, [B, A, A, A] (4-3 = 1 ≤ 2) ans=4

Step 7: B B A A A B → [B, A, A, A, B] (5-3 = 2 ≤ 2) ans=5

Step 8: B B A A A B B → [B, A, A, A, B, B] (6-3 = 3 > 2) → update left pointer to 4, [A, A, A, B, B] (5-4 = 1 ≤ 2) ans=5

Final answer: 5

```
class Solution {
public:
    int characterReplacement(string s, int k) {
        int ans = 0;
        int maxCnt = 0;
        int i = 0;
        unordered_map<char, int> mp;
        for(int j=0; j < s.size(); j++){
            mp[s[j]]++;
            maxCnt = max(maxCnt, mp[s[j]]);
            if(j - i + 1 - maxCnt > k) {
                mp[s[i]]--;
                i++;
            }
            ans = max(ans, j-i+1);
        }
        return ans;
    }
};
```