# 424. Longest Repeating Character Replacement

Medium | Topics | Companies

You are given a string `s` and an integer `k`. You can choose any character of the string and change it to any other uppercase English character. You can perform this operation at most `k` times.

Return *the length of the longest substring containing the same letter you can get after performing the above operations*.

**Example 1:**

```
Input: s = "ABAB", k = 2
Output: 4
Explanation: Replace the two 'A's with two 'B's or vice versa.
```

**Example 2:**

```
Input: s = "AABABBA", k = 1
Output: 4
Explanation: Replace the one 'A' in the middle with 'B' and form "AABBBBA".
The substring "BBBB" has the longest repeating letters, which is 4.
There may exists other ways to achieve this answer too.
```

**Constraints:**

- `1 <= s.length <= 10^5`
- `s` consists of only uppercase English letters.
- `0 <= k <= s.length`

Accepted 566.8K | Submissions 1.1M | Acceptance Rate 52.7%

Approach 1: Brute force

checking for every substring

$T(n): O(n^2)$

$S(n): O(1)$

Approach 2: Sliding window

Q: How do we decide if a substring will have

Q: How do we decide if a substring will have repeating characters after characters replacement?

window : A B A B A (0 1 2 3 4)  K = 2

map : <A, 3>
<B, 2>

Total length of window = 5
highest frequency = 3
Remaining = 5-3
= 2
and it is ≤ K
means this window will have repeating characters.

if window length - highest frequency > K then it means that window will not have repeating characters even after K replacements. So it doesn't make sense to increase window size further. Hence we slide it.

```
class Solution {
public:
    int characterReplacement(string s, int k) {
        int ans=0;
        int l=0,r=0;
        unordered_map<char,int> m;
        int maxi=0;  To keep track of highest frequency in a window

        while(r<s.length()){
            m[s[r]]++;
            maxi=max(maxi,m[s[r]]);
            while((r-l+1)-maxi > k){
                m[s[l]]--;
                if(m[s[l]] == 0) m.erase(m[s[l]]);
                l++;
            }
            ans=max(ans,r-l+1);
            r++;
        }

        return ans;

    }
};
```

$T(n) : O(n+n)$

$S(n) : O(26)$