```
class Solution {
    public int characterReplacement(String s, int k) {
        int[] cs = new int[26];
        int max = 0, res = 0, l = 0, r = 0;
        while (r < s.length()) {
            char c = s.charAt(r++);
            cs[c - 'A']++;
            if (cs[c - 'A'] > max) {
                max = cs[c - 'A'];
            }
            if (r - l - max <= k) {
                res = Math.max(res, r - l);
            } else {
                cs[s.charAt(l++) - 'A']--;
            }
        }
        return res;
    }
}
```

Given a string that consists of only uppercase English letters, you can replace any letter in the string with another letter at most *k* times. Find the length of a longest substring containing all repeating letters you can get after performing the above operations.

Note:
Both the string's length and *k* will not exceed 10^4.

Example 1:

```
Input:
s = "ABAB", k = 2

Output:
4

Explanation:
Replace the two 'A's with two 'B's or vice versa.
```