# Problem 3398: Smallest Substring With Identical Characters I

## Problem Information

**Difficulty:** Hard
**Acceptance Rate:** 19.87%
**Paid Only:** No
**Tags:** Array, Binary Search, Enumeration

## Problem Description

You are given a binary string `s` of length `n` and an integer `numOps`.

You are allowed to perform the following operation on `s` **at most** `numOps` times:

* Select any index `i` (where `0 <= i < n`) and **flip** `s[i]`. If `s[i] == '1'`, change `s[i]` to `'0'` and vice versa.

You need to **minimize** the length of the **longest** substring of `s` such that all the characters in the substring are **identical**.

Return the **minimum** length after the operations.

**Example 1:**

**Input:** s = "000001", numOps = 1

**Output:** 2

**Explanation:**

By changing `s[2]` to `'1'`, `s` becomes `"001001"`. The longest substrings with identical characters are `s[0..1]` and `s[3..4]`.

**Example 2:**

**Input:** s = "0000", numOps = 2

**Output:** 1

**Explanation:**

By changing `s[0]` and `s[2]` to `'1'`, `s` becomes `"1010"`.

**Example 3:**

**Input:** s = "0101", numOps = 0

**Output:** 1

**Constraints:**

* `1 <= n == s.length <= 1000` * `s` consists only of `'0'` and `'1'`. * `0 <= numOps <= n`

## Code Snippets

**C++:**

```
class Solution {
public:
int minLength(string s, int numOps) {

}
};
```

**Java:**

```
class Solution {
public int minLength(String s, int numOps) {

}
}
```

**Python3:**

```
class Solution:
def minLength(self, s: str, numOps: int) -> int:
```