Problem 2156: Find Substring With Given Hash Value

Problem Information

Difficulty: Hard
Acceptance Rate: 25.57%
Paid Only: No
Tags: String, Sliding Window, Rolling Hash, Hash Function

Problem Description

The hash of a **0-indexed** string `s` of length `k`, given integers `p` and `m`, is computed using the following function:

* `hash(s, p, m) = (val(s[0]) * p0 + val(s[1]) * p1 + ... + val(s[k-1]) * pk-1) mod m`.

Where `val(s[i])` represents the index of `s[i]` in the alphabet from `val('a') = 1` to `val('z') = 26`.

You are given a string `s` and the integers `power`, `modulo`, `k`, and `hashValue.` Return `sub`,_the**first** **substring** of _`s` _of length_`k` _such that_`hash(sub, power, modulo) == hashValue`.

The test cases will be generated such that an answer always **exists**.

A **substring** is a contiguous non-empty sequence of characters within a string.

Example 1:

Input: s = "leetcode", power = 7, modulo = 20, k = 2, hashValue = 0 **Output:** "ee" **Explanation:** The hash of "ee" can be computed to be hash("ee", 7, 20) = (5 * 1 + 5 * 7) mod 20 = 40 mod 20 = 0. "ee" is the first substring of length 2 with hashValue 0. Hence, we return "ee".

Example 2:

Input: s = "fbxzaad", power = 31, modulo = 100, k = 3, hashValue = 32 **Output:** "fbx" **Explanation:** The hash of "fbx" can be computed to be hash("fbx", 31, 100) = (6 * 1 + 2 * 31 + 24 * 312) mod 100 = 23132 mod 100 = 32. The hash of "bxz" can be computed to be hash("bxz", 31, 100) = (2 * 1 + 24 * 31 + 26 * 312) mod 100 = 25732 mod 100 = 32. "fbx" is the first substring of length 3 with hashValue 32. Hence, we return "fbx". Note that "bxz" also has a hash of 32 but it appears later than "fbx".

Constraints:

* `1 <= k <= s.length <= 2 * 104` * `1 <= power, modulo <= 109` * `0 <= hashValue < modulo` * `s` consists of lowercase English letters only. * The test cases are generated such that an answer always **exists**.

Code Snippets

C++:

```
class Solution {
public:
string subStrHash(string s, int power, int modulo, int k, int hashValue) {

}
};
```

Java:

```
class Solution {
public String subStrHash(String s, int power, int modulo, int k, int
hashValue) {

}
}
```

Python3:

```
class Solution:
def subStrHash(self, s: str, power: int, modulo: int, k: int, hashValue: int)
-> str:
```