1062. Longest Repeating Substring

Medium 445 27 Add to List Share

Given a string `s`, find out the length of the longest repeating substring(s). Return `0` if no repeating substring exists.

Example 1:

```
Input: s = "abcd"
Output: 0
Explanation: There is no repeating substring.
```

Example 2:

```
Input: s = "abbaba"
Output: 2
Explanation: The longest repeating substrings are "ab" and "ba", each of which occurs twice.
```

Example 3:

```
Input: s = "aabcaabdaab"
Output: 3
Explanation: The longest repeating substring is "aab", which occurs 3 times.
```

Example 4:

```
Input: s = "aaaaa"
Output: 4
Explanation: The longest repeating substring is "aaaa", which occurs twice.
```

Constraints:

- The string `s` consists of only lowercase English letters from `'a'` - `'z'`.
- `1 <= s.length <= 1500`

Accepted 22,253 | Submissions 37,580

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Google | 5 Facebook | 2 Amazon

Related Topics

String Binary Search Dynamic Programming Rolling Hash Suffix Array Hash Function

Hide Hint 1

Generate all substrings in O(N^2) time with hashing.

Hide Hint 2

Choose those hashing of strings with the largest length.

Java Autocomplete

```java
class Solution {
  /*
    Search a substring of given length
    that occurs at least 2 times.
    Return start position if the substring exits and -1 otherwise.
    */
  public int search(int L, int n, String S) {
    HashSet<Integer> seen = new HashSet();
    String tmp;
    int h;
    for(int start = 0; start < n - L + 1; ++start) {
      tmp = S.substring(start, start + L);
      h = tmp.hashCode();
      if (seen.contains(h)) return start;
      seen.add(h);
    }
    return -1;
  }

  public int longestRepeatingSubstring(String S) {
    int n = S.length();
    // binary search, L = repeating string length
    int left = 1, right = n;
    int L;
    while (left <= right) {
      L = left + (right - left) / 2;
      if (search(L, n, S) != -1) left = L + 1;
      else right = L - 1;
    }

    return left - 1;
  }
}
```