**LC#459:REPEATED SUBSTSRING PATTERN**

Given a string s, check if it can be constructed by taking a substring of it and appending multiple copies of the substring together.

**1)Input:** s = "abab" , **Output:** true

**Explanation:** It is the **substring "ab" twice**.

**2)Input:** s = "aba" ,**Output:** false

**3:Input:** s = "abcabcabcabc" , **Output:** true

**Explanation: It is the substring "abc" four times or the substring "abcabc" twice.**

**APPROACH 1: TC(O(N)) , SC:O(N)**

**->if it is a repeated means between the second &last character the string should appear again so**

**If s in (s+s)**

class Solution {

public boolean repeatedSubstringPattern(String s) {

String doubled = s + s;

return doubled.substring(1, doubled.length() - 1).contains(s);

}

}

**APPROACH 2: brute (o(n^2)) o(n)**

**Without string builder**

**->if it is divisor only go inside such that it can be repeated**

**-> loop from 1 to half , build a string from 0 ->i every time in the loop**

**->checks if the string repated until n /i=> equals the input string**

class Solution {

    public boolean repeatedSubstringPattern(String s) {

        int n = s.length();

**for (int i = 1; i <= n/2; i++) {**

            if (n % i == 0) {

                String temp = s.substring(0, i);

                if (temp.repeat(n/i).equals(s)) {

                    return true;

                }

            }

        }

        return false;

    }

}

**With string builder**

class Solution {

public boolean repeatedSubstringPattern(String s) {

int n = s.length();

for (int i = 1; i <= n / 2; i++) {

if (n % i == 0) {

**String temp = s.substring(0, i);**

**StringBuilder sb = new StringBuilder();**

**for (int j = 0; j < n / i; j++) {**

**sb.append(temp);**

**}**

**if (sb.toString().equals(s)) {**

**return true;**

}

}

}

return false; }}

**APPROACH 3: TC :O(N\*sqrt(N)) SC : O(N)**

**-> check the divisors only**

**-> if I is a factor then n/I is also a factor**

**-> so in a single loop we are checking for two factors I and i/n**

class Solution {

    public boolean repeatedSubstringPattern(String s) {

        int n = s.length();

        for (int i = 1; i \* i <= n; i++) {

            if (n % i == 0) {

                if (i < n) {

                    String sub = s.substring(0, i);

                    if (sub.repeat(n / i).equals(s)) {

                        return true;

                    }

                }

                // Check divisor n / i

                int other = n / i;

                if (other < n) {

                    String sub = s.substring(0, other);

                    if (sub.repeat(n / other).equals(s)) {

                        return true;

                    }

                }

            }

        }

        return false;

    }

}