31st Jan 2023

FIRST:-

Minimum times A has to be repeated such that B is a substring of it

Medium

Given two strings **A** and **B.** Find minimum number of times A has to be repeated such that B is a Substring of it. If **B** can never be a substring then return **-1**.

Example 1:

```
Input:
A = "abcd"
B = "cdabcdab"
Output:
3
Explanation:
Repeating A three times (abcdabcdabcd),
B is a substring of it. B is not a substring
of A when it is repeated less than 3 times.
```

Example 2:

```
Input:
A = "ab"
B = "cab"
Output :
-1
Explanation:
No matter how many times we repeat A, we can't
get a string such that B is a substring of it.
```

Your Task:
You don't need to read input or print anything. Your task is to complete the function **minRepeats()** which takes 2 strings A, and B respectively and returns the minimum number of times A has to be repeated such that B is a substring of it. Return -1 if it's not possible.

Expected Time Complexity: O(|A| * |B|)
Expected Auxiliary Space: O(|B|)

Constraints:
$1 \leq |A|, |B| \leq 10^3$
String A and B consists of lower case alphabets

CODE SECTION :-

```
int minRepeats(string A, string B)
  {

      string rep = A;
      int cnt = 1;
      while (A.size() < B.size())
      {
          A += rep;
          cnt++;
      }

      if (A.find(B) != -1)
          return cnt;
      A += rep;
      cnt++;
      if (A.find(B) != -1)
          return cnt;
      return -1;
  }
```

-: DONE FOR THE DAY :-