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

Given two strings **A** and**B**. The task is to find the minimum number of times **A** has to be repeated such that **B** is a substring of it. If no such solution exsits print **-1**.

***Input :****A = “abcd”, B = “cdabcdab”****Output :****3  
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*

***Input :****A = “ab”, B = “cab”****Output :****-1*

// CPP program to find Minimum number of times A

// has to be repeated such that B is a substring of it

#include <bits/stdc++.h>

using namespace std;

// Function to check if a number

// is a substring of other or not

bool issubstring(string str2, string rep1)

{

int M = str2.length();

int N = rep1.length();

// Check for substring from starting

// from i'th index of main string

for (int i = 0; i <= N - M; i++) {

int j;

// For current index i,

// check for pattern match

for (j = 0; j < M; j++)

if (rep1[i + j] != str2[j])

break;

if (j == M) // pattern matched

return true;

}

return false; // not a substring

}

// Function to find Minimum number of times A

// has to be repeated such that B is a substring of it

int Min\_repetation(string A, string B)

{

// To store minimum number of repetations

int ans = 1;

// To store repeated string

string S = A;

// Untill size of S is less than B

while(S.size() < B.size())

{

S += A;

ans++;

}

// ans times repetation makes required answer

if (issubstring(B, S)) return ans;

// Add one more string of A

if (issubstring(B, S+A))

return ans + 1;

// If no such solution exits

return -1;

}

// Driver code

int main()

{

string A = "abcd", B = "cdabcdab";

// Function call

cout << Min\_repetation(A, B);

return 0;

}