796. Rotate String

Given two strings `s` and `goal`, return `true` *if and only if* `s` *can become* `goal` *after some number of* ***shifts*** *on* `s`.

A **shift** on `s` consists of moving the leftmost character of `s` to the rightmost position.

- For example, if `s = "abcde"`, then it will be `"bcdea"` after one shift.

Example 1:

```
Input: s = "abcde", goal = "cdeab"
Output: true
```

Example 2:

```
Input: s = "abcde", goal = "abced"
Output: false
```

Constraints:

- `1 <= s.length, goal.length <= 100`
- `s` and `goal` consist of lowercase English letters.

796. Rotate String

```
/*
   Straight simulation v1: without STL
   Time complexity: O(n^2)
   Space complexity: O(1)
*/
class Solution {
   public:
      bool rotateString(std::string s,std::string goal) {
         int n=s.size();
         if(n!=goal.size()) return false;
         while(n--){
            s.push_back(s[0]);
            s.erase(s.begin());
            if(s==goal) return true;
         }
         return false;
      }
};
```

796. Rotate String

```
/*
   Straight simulation v2: using STL rotate()
   Time complexity: O(n^2)
   Space complexity: O(1)
*/
class Solution {
   public:
      bool rotateString(std::string s,std::string goal) {
         int n=s.size();
         if(n!=goal.size()) return false;
         for(int i=1;i<=n;++i){
            std::rotate(s.begin(),s.begin()+1,s.end());
            if(s==goal) return true;
         }
         return false;
      }
};
```

796. Rotate String

```
/*
   Concatenation check
   Time complexity: O(n)
   Space complexity: O(2n)
*/
class Solution {
   public:
      bool rotateString(std::string s,std::string goal) {
         int n=s.size();
         if(n!=goal.size()) return false;

         // if `goal` can be obtained by rotating `s`, `goal` must be
         // a subtring of  s+s
         std::string ss=s+s;
         return ss.find(goal)!=std::string::npos;
      }
};
```

796. Rotate String

```
/*
   KMP (see LC's problem 28 )
   Time complexity: O(n)
   Space complexity: O(2n+m)
*/
class Solution {
   public:
      bool rotateString(std::string s,std::string goal){
         if(s.size()!=goal.size()) return false;

         std::string ss=s+s;

         int n=ss.size();
         int m=goal.size();

         // Build Longest prefix suffix array
         std::vector<int> lps(m,0);
         int prev=0,i=1;
         while(i<m){
            if(goal[prev]==goal[i]){
               lps[i]=prev+1;
               prev++;
               i++;
            }
            else if(prev==0){
               lps[i]=0;
               i++;
            }
            else prev=lps[prev-1];
         }
```

```
        // Check the 1st occurence of `goal` in `s+s`
        i=0;
        int j=0;
        while(i<n){
            if(ss[i]==goal[j]){
                i++;
                j++;
            }
            else if(j==0) i++;
            else j=lps[j-1];

            if(j==m) return true;
        }

        return false;
    }

};
```