214. Shortest Palindrome

You are given a string `s`. You can convert `s` to a

palindrome

by adding characters in front of it.

Return *the shortest palindrome you can find by performing this transformation*.

Example 1:

```
Input: s = "aacecaaa"
Output: "aaacecaaa"
```

Example 2:

```
Input: s = "abcd"
Output: "dcbabcd"
```

Constraints:

- `0 <= s.length <= 5 * 10^4`
- `s` consists of lowercase English letters only.

214. Shortest Palindrome

```
/*
   Brute force - TLE
   Time complexity: O(n*(n+n+n))= O(3n^2)=O(n^2)
   Space complexity: O(n)
*/
class Solution {
   public:
      bool is_palindrome(std::string& s, int left, int right){
         int i=left,j=right;
         while(left<=right && s[left]==s[right]){
            left++;
            right--;
         }
         return left>right;
      }
      std::string shortestPalindrome(std::string s) {
         int n=s.size();
         for(int i=n-1;i>=0;--i){
            if(is_palindrome(s,0,i)){
               std::string to_add=s.substr(i+1,n-1-i);
               std::reverse(to_add.begin(),to_add.end());
               return to_add+s;
            }
         }
         return "";
      }
};
```

214. Shortest Palindrome

```
/*
   Knuth–Morris–Pratt(KMP)
   Time complexity: O(n*(n+n+n))= O(3n^2)=O(n^2)
   Space complexity: O(n)
*/

class Solution {
   public:
      std::string shortestPalindrome(std::string s) {
         int n=s.size();

         std::string rev=std::string(s.rbegin(), s.rend());
         std::string needle=s+'@'+rev;
         int m=needle.size();

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

         int len=lps[needle.size()-1];

         return rev.substr(0,s.size()-len)+s;
      }
};
```

28. Find the Index of the First Occurrence in a String

Given two strings `needle` and `haystack`, return the index of the first occurrence of `needle` in `haystack`, or `-1` if `needle` is not part of `haystack`.

Example 1:

```
Input: haystack = "sadbutsad", needle = "sad"
Output: 0
Explanation: "sad" occurs at index 0 and 6.
The first occurrence is at index 0, so we return 0.
```

Example 2:

```
Input: haystack = "leetcode", needle = "leeto"
Output: -1
Explanation: "leeto" did not occur in "leetcode", so we return -1.
```

Constraints:

- `1 <= haystack.length, needle.length <= 10^4`
- `haystack` and `needle` consist of only lowercase English characters.

28. Find the Index of the First Occurrence in a String

```
/*
   Brute force
   Time complexity: O(nm)
   Space complexity: O(1)
*/
class Solution {
public:
   int strStr(std::string haystack, std::string needle) {
      if(needle=="") return 0;
      int n=haystack.size();
      int m=needle.size();
      for(int i=0;i<=n-m;++i){
         if(haystack.substr(i,m)==needle) return i;
      }
      return -1;
   }
};
```

28. Find the Index of the First Occurrence in a String

```
/*
   KMP
   Time complexity: O(n+m)
   Space complexity: O(1)
*/
class Solution {
    public:
        int strStr(string haystack, string needle) {
            if(needle=="") return 0;
            // Longest prefix suffix
            int n=haystack.size();
            int m=needle.size();
            std::vector<int> lps(m,0);
            int prev_lps_index=0,i=1;
            while(i<m){
                if(needle[prev_lps_index]==needle[i]){
                    lps[i]=prev_lps_index+1;
                    prev_lps_index++;
                    i++;
                }
                else if(prev_lps_index==0){
                    lps[i]=0;
                    i++;
                }
                else prev_lps_index=lps[prev_lps_index-1];
            }
            i=0; // Pointer for haystack
            int j=0; // Pointer for needle
            while(i<n){
                if(haystack[i]==needle[j]){
                    i++;
                    j++;
                }
                else if(j==0) i++;
                else j=lps[j-1];
                if(j==m) return i-m;
            }
            return -1;
        }
};
```