Search Pattern (KMP-Algorithm)

Difficulty: **Medium** Accuracy: **45.04%** Submissions: **98K+** Points: **4**

Given two strings, one is a text string **txt** and the other is a pattern string **pat**. The task is to print the indexes of **all the occurrences** of the pattern string in the text string. Use **0-based** indexing while returning the indices.
Note: Return an empty list in case of no occurrences of pattern.

Examples :

Input: txt = "abcab", pat = "ab"
Output: [0, 3]
Explanation: The string "ab" occurs twice in txt, one starts at index 0 and the other at index 3.

Input: txt = "abesdu", pat = "edu"
Output: []
Explanation: There's no substring "edu" present in txt.

Input: txt = "aabaacaadaabaaba", pat = "aaba"
Output: [0, 9, 12]

Java (1.8) Start Timer

```java
// } Driver Code Ends


class Solution {
ArrayList<Integer> search(String pat, String txt) {
        ArrayList<Integer> result = new ArrayList<>();
        int m = pat.length();
        int n = txt.length();
        int[] lps = computeLPSArray(pat);
        int i = 0;
        int j = 0;
        while (i < n) {
            if (pat.charAt(j) == txt.charAt(i)) {
                i++;
                j++;
            }
            if (j == m) {
                result.add(i - j);
                j = lps[j - 1];
            } else if (i < n && pat.charAt(j) != txt.charAt(i)) {
                if (j != 0) {
                    j = lps[j - 1];
                } else {
                    i++;
                }
            }
        }
        return result;
    }
```

```java
    private int[] computeLPSArray(String pat) {
        int m = pat.length();
        int[] lps = new int[m];
        int len = 0;
        int i = 1;
        lps[0] = 0;
        while (i < m) {
            if (pat.charAt(i) == pat.charAt(len)) {
                len++;
                lps[i] = len;
                i++;
            } else {
                if (len != 0) {
                    len = lps[len - 1];
                } else {
                    lps[i] = 0;
                    i++;
                }
            }
        }
        return lps;
    }
}
```