KMP Algorithm

The fundamental string searching (matching) problem is defined as follows: given two strings – a text and a pattern, determine whether the pattern appears in the text.

First let's look at a naive solution.
suppose the text is in an array: char T[n]
and the pattern is in another array: char P[m].
One simple method is just to try each possible position the pattern could appear in the text.
Naive string matching:

```
 for (i=0; T[i] != '\0'; i++)
  {
  for (j=0; T[i+j] != '\0' && P[j] != '\0' && T[i+j]==P[j]; j++) ;
  if (P[j] == '\0') found a match
  }
```

There are two nested loops; the inner one takes O(m) iterations and the outer one takes O(n) iterations so the total time is the product, O(mn). This is slow; we'd like to speed it up.

Now, The KMP algorithm sort of takes care of the inner loop and thus reduces the time complexity by a great deal.
The Knuth-Morris-Pratt idea is, in this sort of situation, after you've invested a lot of work making comparisons in the inner loop of the code, you know a lot about what's in the text. Specifically, if you've found a partial match of j characters starting at position i, you know what's in positions T[i]...T[i+j-1].(Here T[] refers to the text).
You can use this knowledge to save work in two ways. First, you can skip some iterations for which no match is possible. Try overlapping the partial match you've found with the new match you want to find
Now the most important question is that - How to decide which characters to skip. This is decided by the failure function which shows us the position to start from in case of a mismatch.
This is just a brief idea about the algorithm.

For further details please refer to any of the following links :

1. http://www.geeksforgeeks.org/searching-for-patterns-set-2-kmp-algorithm/
2. https://www.topcoder.com/community/data-science/data-science-tutorials/introduction-to-string-searching-algorithms/
3. https://www.ics.uci.edu/~eppstein/161/960227.html

YouTube Video:
https://youtu.be/GTJr8OvyEVQ (actual algo)
https://youtu.be/KG44VoDtsAA (computing failure function)

Questions:

1. http://www.spoj.com/problems/PERIOD/
2. http://codeforces.com/problemset/problem/126/B