**Q.18. String Matching**

In this exercise, you implement a variant of the Knuth-Morris-Pratt algorithm.

* Suppose that we want to find all matches of *T* in *S*. Write a function to do the following. First, compute (and store in an array) the failure function of the concatenated string *TS*. In the second stage, use these failure function values to find out whether *T* is a border (not necessarily proper or longest) of *TS*[0...*i*] for all relevant *i*. Finally, prepare (and return) the list of all matches of *T* in *S*, using the information available from the second stage. Your function should run in linear time. Do not implement the standard KMP loop in the second or the third stage.
* A particular pattern (regular expression) is of the form *U\*V*, where *U* and *V* are strings and \* is a symbol not present in the string alphabet Σ. A match of *U\*V* in *S* means an occurrence in *S* of *U* followed by zero or more symbols and then by *V*. Write a function that calls the above function to locate all possible matches of the pattern *U\*V* in *S*.

For simplicity, assume that the string alphabet is Σ = {0,1}.

### Sample output

n = |S| = 100

m = |T| = 5

\*\*\* String matching

S = 0101000101110011000001001001101110010010001000101101001001111100101100100000110010110010111001011100

T = 01001

5 matches found at indices 20 23 34 50 53

\*\*\* Pattern matching

S = 1100101011010101001000100101001111110111011001010110000000001110001010010100001010111101001110110100

T = 1011\*11011

Pattern matches at index pairs (6,34) (6,38) (6,91) (35,91) (39,91) (47,91) (80,91)

**Q.19. Pattern matching**

Given an n-by-n array of black(1) and white(0) pixels, design a linear algorithm that finds the largest square subarray that consists of entirely black pixels. As an example, the following 8-by-8 array contains a 3-by-3 sub array entirely of black pixels:

**10111000**

**00010100**

**00111000**

**00111010**

**00111111**

**01011110**

**01011010**

**00011110**

Implement your algorithm and confirm that the order of growth of its running time is linear in the number of pixels. Extra credit: Design an algorithm to find the largest rectangular black subarray.

**Q. 20. Hash Table**

Write a program to implement a hash table using *python list*, use a class name **SymblTable** for the implementation.

**=========================OOO=========================**