**Week 9: String Algorithms**

1. **naive\_string\_matching**:

* **Guide**:
  + The naive approach checks for a match of the pattern in the text at every position.
* **Pseudocode**:
* FUNCTION naive\_string\_matching(text, pattern):  
   INITIALIZE an empty list for starting indices of occurrences  
   FOR i from 0 to length of text - length of pattern:  
   IF text[i to i + length of pattern] matches pattern:  
   ADD i to the list  
   RETURN the list

1. **rabin\_karp**:

* **Guide**:
  + Rabin-Karp utilizes a rolling hash function to check if the pattern hash matches any of the text’s window hashes. This helps in avoiding unnecessary character comparisons.
* **Pseudocode**:
* FUNCTION rabin\_karp(text, pattern, d, q):

n = LENGTH(text)  
m = LENGTH(pattern)  
h = d^(m-1) MOD q  
p = 0  
t = 0  
  
FOR i from 0 to m-1:  
 p = (d\*p + ASCII(pattern[i])) MOD q  
 t = (d\*t + ASCII(text[i])) MOD q  
  
occurrences = []  
  
FOR s from 0 to n-m:  
 IF p == t:  
 is\_match = TRUE  
 FOR i from 0 to m-1:  
 IF pattern[i] != text[s+i]:  
 is\_match = FALSE  
 BREAK  
 IF is\_match:  
 ADD s to occurrences  
  
 IF s < n - m:  
 t = (t - h\*ASCII(text[s])) MOD q  
 t = (t\*d + ASCII(text[s+m])) MOD q  
 t = (t + q) MOD q  
  
RETURN occurrences

1. **kmp\_pattern\_preprocessing**:

* **Guide**:
  + The preprocessing step of KMP constructs the lps array that represents the longest prefix which is also a suffix. It helps in avoiding unnecessary character comparisons.
* **Pseudocode**:
* FUNCTION kmp\_pattern\_preprocessing(pattern):  
   INITIALIZE the lps array of size of pattern to 0s  
   COMPUTE the lps values for the pattern  
   RETURN the lps array

1. **trie\_insert**:

* **Guide**:
  + A trie is a tree-like data structure that stores a dynamic set of strings. Nodes are associated with a character, and traversing down the tree gives a string.
* **Pseudocode**:
* FUNCTION trie\_insert(root, key):  
   INITIALIZE current node as root  
   FOR each character in key:  
   IF character is not in current node's children:  
   ADD a new node for the character  
   MOVE to the child node of the character  
   MARK current node as end of word