Date 22/11/2023 Page

Day - 59

# KMP Algo - 2

* **String Matching:**

⇒ We have to check if the second string is present in the first string as substring.

Ex: a b c a b d e f g → haystack
d e f → needle

⇒ In brute force approach, we will check one by one that def is present or not.

Code

```
n = haystack.size(), m = needle.size();
for (i=0; i<n-m; i++){
  int first = i, second = 0;
  while ( second < m ){
        if ( haystack[first] != needle[second])
              break;
        else
              first++, second++;
  }
  if (second == m)
        return first-second;
}
return -1;
```

T.C → O(nm)
S.C → O(1)

Date _____ Page _____

=1 Now our task is to solve this problem in O(n) time.

=1 So, for this, we will check when the char not matched that any part of the needle is matched with the haystack.

=1 So, we will take the decision so that we don't have to move our first pointer.

=1 So, for this, we will use KMP, or LPS array.

Ex: onionionions on
first ↓
second ↓
onions

| 0 | 0 | 0 | 1 | 2 | 0 |
|---|---|---|---|---|---|

=1 When any mismatch occur, we will shift the second pointer to the ~~in~~ index value of previous of lps.

Code

```
vector<int> lps(needle.size(), 0);
lps find(lps, needle);
int first=0, second=0;
while(second < needle.size() && first < haystack.size()){
    if(needle[second] == haystack[first])
        second++, first++;
    else {
```

Date _____ Page _____

```
            if (second == 0)
                first++;
            else{
                second = lps[second - 1];
            }
        }
    }
    if(second == needle.size()
        return first - second;
    return -1;
```