Date   Page

Day - 58

KMP Algo

* Longest Prefix suffix

⇒ We have to return longest equal prefix & suffix length.

Ex: A A C A A

```
A        -   1   -   A
A A      -   2   -   A A
A A C        ✗       C A A
A A C A      ✗     A C A A
A A C A A - 5 - A A C A A ⟶ we don't
                 consider the last case.
```

⇒ In the brute force approach, we will find all prefix & suffix then check.
T.C. → $O(N^2)$

⇒ In the second approach, we know that the ~~first & se~~ prefix & suffix first's letter will be equal.

Ex:
```
Prefix ↓   ↓   ↓   ↓   ↓   ↓   ↓   ↓   ✗
A      B   C   D   E   A   B   C   D
↑    Suffix
       ↑   ↑   ↑
```

⇒ This is also not a feasible approach.
T.C → $O(N^2)$

Date

Page

⇒ Now our next approach is most optimised approach.

⇒ Here we will create a LPS array.

Ex:

	A	B	C	D	E	A	B	C	D
LPS	0	0	0	0	0	1	2	3	4

⇒ Any no. in this array shows that the length of prefix & suffix at that char.

⇒ Ex: A → 0
AB → 0
ABC → 0
ABCD → 0
ABCDE → 0
ABCDEA → 1
ABCDE AB → 2
ABCDE ABC → 3
ABCDE ABCD → 4

⇒ 4 shows that the less than 4 index are equal to them.

⇒ For filling the table, we will use the first approach, when the prefix & suffix match, we will add the index of prefix with 1 & write at the place of suffix.

Date ____________ Page ________

⇒ And when the prefix does not equal to suffix then ~~switch the~~ switch the prefix to the ~~last~~ ~~v~~ index of the value of last element.

Ex:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	A	B	C	A	B	D	A	B	C	A	B	D	A	B	D	A	B
LPS	0	0	0	1	2	0	1	2	3	4	5	6	7	8	0	1	2

⇒ Here, when our prefix is at C & suffix at D then we will not ~~n~~ write 0.

⇒ Instead we will go the index of the value of previous of prefix.

⇒ That's means prefix shift to A.

⇒ Now, A is not equal to D, so we will again do this.

⇒ But, this time there is no element previous to prefix. So, we will write 0.

T.C → O(N)
S.C. → O(N)

Code

```
→ string s; // given
vector <int> lps (s.size(), 0);
int pre = 0, suf = 1;
while( suf < s.size() ){
```

Date

Page

```
if (s[pre] == s[suf]){
        lps[suf]= pre + 1;
        pre++, suf++;
}
else{
        if( pre == 0)
            lps[suf++] = 0;
        else{
            pre = lps[pre - 1];
        }
}
} return lps[s.size() - 1];
```