Arrays, Strings & Linked Lists Lecture 3

Wednesday, 24 July 2024 6:02 AM

Strings

Sliding window algorithm

Problem:- Given a text T and pattern P, find whether P exists in T or not, if yes print all occurrences of pattern in text.

T :- a b a c a b a b a a c b a b a

0 4 6 12

0, 4, 6, 12

P :- a b a

|T| = n

|P| = m

a b a c a b a b a a c b a b a

i j

a b a

j

Match every substring of size m in T with P.

such substrings to match = $n - m + 1$

Time to match one pair of strings = m

Time Complexity = $O(nm - m^2 + m) \approx O(nm)$

https://leetcode.com/problems/find-the-index-of-the-first-occurrence-in-a-string/

0 1 2 3 4 5 6 7 8 9 10
a b s a d b u t s a d
i

$|T| = 11$

$|P| = 3$

$i \leftarrow 8$

sad

$i \leq |T| - |P|$

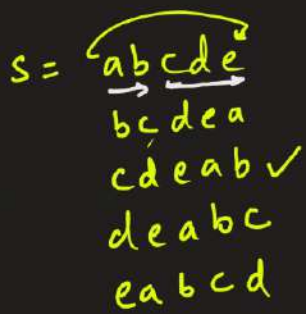
```
class Solution {
public:
    int strStr(string haystack, string needle) {
        if(needle.length() > haystack.length()) return -1;
        int j;
        for(int i=0; i<=haystack.length()-needle.length(); i++) {
            for(j=0; j<needle.length(); j++) {
                if(needle[j] != haystack[i+j])
                    break;
            }
            if(j==needle.length()) return i;
        }
        return -1;
    }
};
```

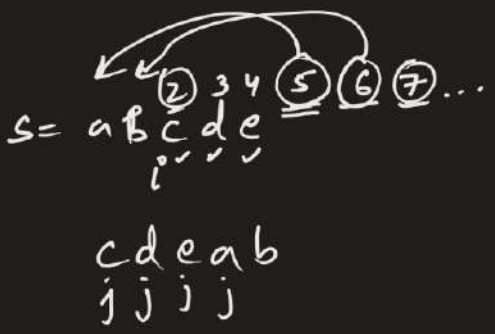
→ $(i+j)\%.n$

```
    }
};
```

→ (i+j)%n

https://leetcode.com/problems/rotate-string/



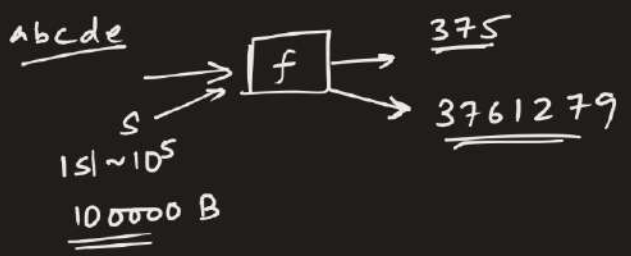



```python
class Solution:
    def rotateString(self, s: str, g: str) -> bool:
        if len(s)!=len(g):
            return False
        n = len(s)
        for i in range(n):
            flag = True
            for j in range(n):
                if s[(i+j)%n] != g[j]:
                    flag = False
                    break
            if flag:
                return True
        return False
```

```python
class Solution:
    def rotateString(self, s: str, g: str) -> bool:
        if len(s)!=len(g):
            return False
        return goal in s+s
```

Hashing of Strings

* Polynomial Hashing *



$O(\min(|s_1|, |s_2|))$

$\sim O(10^5)$

$s_1 == s_2$

$|s_1| \sim 10^5$ $|s_2| \sim 10^5$

int $h(s_1) = h(s_2)$ int

$O(1)$

$O(n \cdot m)$

$\downarrow$

$O(n)$

26 26 26 . . . . . $n-m+1$

[h_1 h_2 . . .]

$n \leftarrow |text| =$

$m \leftarrow |pattern| =$

h_p

Polynomial Hash :- $\text{hash}(s) = (s[0] + s[1] \cdot p + s[2] \cdot p^2 + s[3] \cdot p^3 + \dots) \bmod m$

$$= \left(\sum_{i=0}^{n-1} s[i] \cdot p^i \right) \bmod m$$

$p = 31$

$s[i] - 'a' + 1$

$10^9 + 7 + 1$

'a' - 1
'b' - 2
'c' = 3
⋮
'z' - 26

eg:

s	h(s)
a	1
b	2
c	3
d	4 ←
⋮	⋮
z	26
aa	32
ab	63
⋮	⋮
zz	832
aaa	993
⋮	
tjpabc ⋯	

$1 + 2 \times 31$

$1 + 1 \times 31 + 1 \times (31)^2$

Problem:-

Search duplicate strings in an array of strings.

Given a list of strings S_i, $1 \le i \le n$, each no longer then m characters, find all the unique strings.

$[\ S_0, S_1, S_2, S_3 \ldots, S_{n-1}\ ]$ $\quad |S_i| \le m$

$O(n^2)$ comparisions

$> \quad < \quad ==$

$O(m)$

$T \equiv O(n^2 \cdot m)$ ✓

Sorting. $O(n \log n)$ → $O(nm \log n)$ for strings ✓

$10^6 \times \log 10^4 \sim 10^8$

Using hashing:-

(2) ← (hashing.)

Using hashing:-

(2) ← [hashing.]

- Precomputed PP(m) ← O(m)
- hash value of all strings ← O(nm)
- Sort the hash values & compare adjacent values
  ↳ O(n log n)

$10^6 + 10^4 \times \log 10^4 \sim 10^6$

$O(nm + n\log n)$

$n \sim 10^4$

$m \sim 10^2$