24 May 2024 21:34

Strings

→ 2 algo

→ KMP algorithm

Roman to integer

I, V, X, L, C, D, M

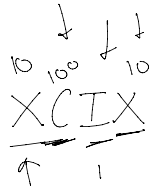
I → 1 D → 500

V → 5 M → 1000

X → 10

L → 50

C → 100



-10 + 100 - 1 + 10

⇒ 99

1) Create a map and a variable ans = 0.

2) Insert all the character with corresponding values into the map.

3) Iterate the input string 's'. (i):

a) If i < s.length() - 1:

i) If (m[s[i]] < m[s[i+1]]:

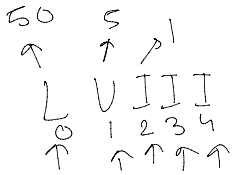
ans -= m[s[i]]

ii) else

ans += m[s[i]]

b) else

ans += m[s[i]]



+ 50 + 5 + 1 + 1 + 1

⇒ 58

4) return ans.

Intiger to Roman

223

400

⇒ CCXXIII

892

⇒ DCCCXCII

I
II
III
IV
V
VI
VII
VIII
IX
X

→ I
V 5
→ X 10
L 50
→ C 100
→ D 500
M 1000

469

⇒ CDLXIX

1) M[ ] = {"", "M", "MM", "MMM"}

2) C[ ] = {"", "C", "CC", "CCC", "CD", "D", "DC", "DCC", "DCCC", "CM"}

3) X[ ] = {"", "X", "XX", "XXX", "XL", "L", "LX", "LXX", "LXXX", "XC"}

4) I[ ] = {"", "I", "II", "III", "IV", "V", "VI", "VII", "VIII", "IX"}

5) ans = "";

6) ans += M[num/1000]; num = num % 1000

7) ans += C[num/100]; num = num % 100;

3467 "MMM"

467 MMMCD

67 MMMCDLX

8) ans += X[num/10]; num = num % 10;

9) ans += I[num]; return ans.

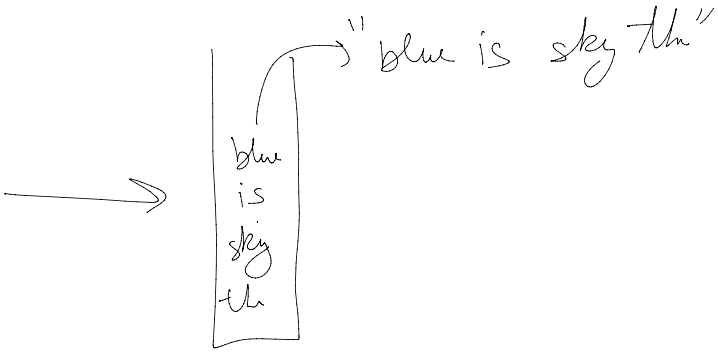
T: O(1), S: O(1)

7 "MMMCDLXVII"

Reverse the string

"the sky is blue" → "eulb si yks eht"

"blue is sky the"




```
stack<string> s;
for(int i=0; i<str.length(); i++) {
    string temp="";
    while(i<str.length && str[i]!=" ") {   // " " = space
        temp += str[i]; i++;
    }
    s.push(temp);
}
string ans="";
while(!s.isEmpty()){
```

```
while( !s.isEmpty()){
    ans+=s.top();
    if(s.size()>1) ans+=" ";
    s.pop();
}
return ans;
```

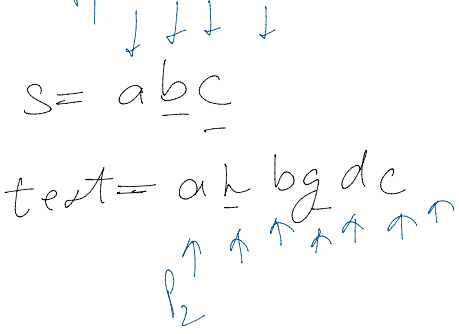
Is subsequence

→ abc a a a b c d e ✓

→ aab a b a b a b ✓

→ acd a x y z c x y z d d ✓

→ aab a b a d d ✗




```
if( s[p1]==text[p2]){
    p1++; p2++
}
else{
    p2++;
}
if( p1== s.length()){
    return true.
```

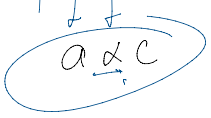
if (P_1 == s.length())
return true;

}

1) Take two pointers $P_1 = 0$ & $P_2 = 0$

2) Iterate through : (P_2 < test.size())

a) if (s[P_1] == test[P_2]) {
P_1++; P_2++;



a b c d
P_2

}

b) else
P_2++

c) If (P_1 == s.length()) return true.

3) return false

xy

Longest common Prefix ⇒ ["xyz", "xy", "xyza"]

a b c d

["flower", "flow", "flight"]

"" , f , fl , flo

a
ab
abc
abcd

["xy", "xyz", "xyab"]

string s = "";

```
string s="";
for(int i=0; i< strs[0].length(); i++) {
    for(int j=0; j< strs.length(); j++) {
        if( strs[j].length() < i || strs[j][i] != strs[0][i]){
            return s;
        }
    }
    s += strs[0][i];
}
return s;
```

n → no. of strings in strs

m → length of shortest string

T: $O(nm)$ S: $O(m)$

Length of the last word

```
string ans="";
int i= s.length()-1;
while(s[i]==" ") i--;
```

"Fly me to the moon" → 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

```
while (s[i] == " ") i--;
while ( i >= 0 && s[i] != " ") {
    ans = s[i] + ans;  i--;
}
return ans.length();
```

moon

Add Binary

```
  1 1
  1 0 1 1
  0 1 0 1
  -------
  1 0 0 0
  -------
```

		O/P	C
0 + 1	→	1	0
1 + 1	→	1	1
1 + 1 + 1	→	1	1

```
  1 1
  1 1 1
  -------
  1 1 1 0
  -------
```

```
n = a.length()
m = b.length()
int carry = 0
String ans = "";
while( n >= 0 && m >= 0) {
    int x = a[n] - '0';
    int y = b[m] - '0';
    int num = x + y + carry;
    if (num == 0) { num = 0; carry = 0; }
else if (num == 1) { num = 1, carry = 0; }
    else { num = 1; carry = 1; }
    ans = char(num) + '0' + ans;
```

```
while( n >= 0) {
    int num = a[n] - '0';
    num += carry;
    if (num == 0) { num = 0; carry = 0; }
else if (num == 1) { num = 1, carry = 0; }
    else { num = 1; carry = 1; }
    ans = char(num) + '0' + ans;
    n--;
}
while( m >= 0) {
    int num = b[m] - '0';
    num += carry;
    if (num == 0) { num = 0; carry = 0; }
else if (num == 1) { num = 1, carry = 0; }
```

```
        ans = char(num) + '0' + ans;
        n--; m--;
    }
    if(num == 0) { num = 0; carry = 0; }
    else if(num == 1) { num = 1; carry = 0; }
    else { num = 1; carry = 1; }
        ans = char(num) + '0' + ans;
    m--;
}
if(carry == 1)
    ans = '1' + ans;
return ans;
```