📓 Problem: Roman to Integer

🔷 Question Statement

Given a Roman numeral string s, convert it into an integer.

🔷 Roman Numeral Rules

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

🔷 Special Subtractive Cases

Roman	Value
IV	4
IX	9
XL	40
XC	90
CD	400
CM	900

✨ Example

Input: "MCMXCIV"

Output: 1994

✅ APPROACH 1: Explicit Subtractive Pair Checking

💡 Idea

- Store **single symbols + subtractive pairs** in HashMap

- Traverse string:
  - First check **2-character substring**
  - If found → add its value & skip one index
  - Else → add single character value

▯ Dry Run (Approach 1)

Input: "MCMXCIV"

Index	Substring	Found?	Value Added	Result
0	"MC"	✗	M = 1000	1000
1	"CM"	✅	900	1900
3	"XC"	✅	90	1990
5	"IV"	✅	4	1994

✓ Final Answer = **1994**

▯ Another Example

Input: "LVIII"

Index	Substring	Found?	Value	Result
0	"LV"	✗	L = 50	50
1	"VI"	✗	V = 5	55
2	"II"	✗	I = 1	56
3	"I"	✗	I = 1	58

✓ Output = **58**

💻 Code – Approach 1

✅ Java

```
class Solution {
    public int romanToInt(String s) {
        HashMap<String, Integer> hm = new HashMap<>();
        hm.put("I",1); hm.put("V",5); hm.put("X",10);
        hm.put("L",50); hm.put("C",100);
        hm.put("D",500); hm.put("M",1000);
        hm.put("IV",4); hm.put("IX",9);
        hm.put("XL",40); hm.put("XC",90);
        hm.put("CD",400); hm.put("CM",900);

        int res = 0;
        int i = 0;

        while (i < s.length()) {
            if (i + 1 < s.length() && hm.containsKey(s.substring(i, i+2))) {
                res += hm.get(s.substring(i, i+2));
                i += 2;
            } else {
                res += hm.get(s.substring(i, i+1));
                i++;
            }
        }
        return res;
    }
}
```

✓ **C++**

```
class Solution {
public:
    int romanToInt(string s) {
        unordered_map<string,int> mp = {
            {"I",1},{"V",5},{"X",10},{"L",50},
            {"C",100},{"D",500},{"M",1000},
            {"IV",4},{"IX",9},{"XL",40},
            {"XC",90},{"CD",400},{"CM",900}
        };

        int res = 0;
        for(int i = 0; i < s.size(); ) {
            if(i + 1 < s.size() && mp.count(s.substr(i,2))) {
                res += mp[s.substr(i,2)];
                i += 2;
            } else {
                res += mp[s.substr(i,1)];
                i++;
            }
        }
        return res;
    }
};
```

✓ **Python**

```
class Solution:
    def romanToInt(self, s: str) -> int:
        mp = {
            "I":1,"V":5,"X":10,"L":50,
            "C":100,"D":500,"M":1000,
            "IV":4,"IX":9,"XL":40,
            "XC":90,"CD":400,"CM":900
        }

        res = 0
        i = 0
        while i < len(s):
            if i+1 < len(s) and s[i:i+2] in mp:
                res += mp[s[i:i+2]]
                i += 2
            else:
                res += mp[s[i]]
                i += 1
        return res
```

✅ APPROACH 2: Compare Current & Next (BEST / INTERVIEW)

💡 Idea

Roman rule:

If a **smaller value comes before a larger one**, subtract it.

So:

- If curr < next → subtract
- Else → add

▯ Dry Run (Approach 2)

Input: "MCMXCIV"

i	Char	curr	next	Operation	Result
0	M	1000	100	+1000	1000
1	C	100	1000	-100	900
2	M	1000	10	+1000	1900
3	X	10	100	-10	1890
4	C	100	5	+100	1990
5	I	1	5	-1	1989
6	V	5	0	+5	1994

✔ Final Answer = **1994**

▯ Example: "LVIII"

Char	curr	next	Action	Result
L	50	5	+50	50

Char	**curr**	**next**	**Action**	**Result**
V	5	1	+5	55
I	1	1	+1	56
I	1	1	+1	57
I	1	0	+1	58

✓ Output = **58**

💻 **Code – Approach 2**

✅ **Java**

```
class Solution {
    public int romanToInt(String s) {
        HashMap<Character, Integer> hm = new HashMap<>();
        hm.put('I',1); hm.put('V',5);
        hm.put('X',10); hm.put('L',50);
        hm.put('C',100); hm.put('D',500);
        hm.put('M',1000);

        int res = 0;
        for (int i = 0; i < s.length(); i++) {
            int curr = hm.get(s.charAt(i));
            int next = (i+1 < s.length()) ? hm.get(s.charAt(i+1)) : 0;

            if (curr < next) res -= curr;
            else res += curr;
        }
        return res;
    }
}
```

✅ **C++**

```
class Solution {
public:
  int romanToInt(string s) {
    unordered_map<char,int> mp = {
      {'I',1},{'V',5},{'X',10},
      {'L',50},{'C',100},
      {'D',500},{'M',1000}
    };

    int res = 0;
    for(int i = 0; i < s.size(); i++) {
      int curr = mp[s[i]];
      int next = (i+1 < s.size()) ? mp[s[i+1]] : 0;

      if(curr < next) res -= curr;
      else res += curr;
    }
    return res;
  }
};
```

✅ **Python**

```
class Solution:
  def romanToInt(self, s: str) -> int:
    mp = {
      'I':1,'V':5,'X':10,
      'L':50,'C':100,
      'D':500,'M':1000
    }

    res = 0
```

```
for i in range(len(s)):
    curr = mp[s[i]]
    next_val = mp[s[i+1]] if i+1 < len(s) else 0

    if curr < next_val:
        res -= curr
    else:
        res += curr
return res
```

🏆 Final Interview Conclusion

Approach	**Verdict**
Pair checking	Easy to understand
Compare logic	**BEST & INTERVIEW FAVORITE** ✅