🔢 LeetCode 1903: Largest Odd Number in String

Problem in simple words

You are given a **string of digits** (like "35427").

👉 You must return the **largest odd-number substring** that:

- starts at index 0
- ends at **some position**
- represents an **odd number**

If no such substring exists, return an **empty string** "".

🔍 Key Observation (MOST IMPORTANT)

A number is **odd** if its **last digit** is odd.

👉 Odd digits are:
1, 3, 5, 7, 9

So the problem reduces to:

Find the **rightmost odd digit** in the string
and return everything **from index 0 to that digit**

✅ Approach 1: Forward Traversal (Left → Right)

💡 Idea

- Traverse the string **from left to right**
- Whenever you see an **odd digit**, update the answer
- Keep extending the substring until the **last odd digit**

Why this works

- Any prefix ending at an odd digit forms an odd number
- The **largest** odd number is the one that ends at the **last odd digit**

🧠 Step-by-Step Example

Input:

num = "35427"

Traversal:

3 → odd → candidate = "3"

5 → odd → candidate = "35"

4 → even → ignore

2 → even → ignore

7 → odd → candidate = "35427"

Final answer:

"35427"

⏱ Complexity

- **Time:** O(n)
- **Space:** O(1) extra space

✅ Java (Forward)

```
class Solution {
    public String largestOddNumber(String num) {
        String result = "";

        for (int i = 0; i < num.length(); i++) {
            int digit = num.charAt(i) - '0';
            if (digit % 2 == 1) {
                result = num.substring(0, i + 1);
            }
        }
        return result;
    }
}
```

✅ C++ (Forward)

```
class Solution {
public:
    string largestOddNumber(string num) {
        string result = "";
```

```
        for (int i = 0; i < num.size(); i++) {
            int digit = num[i] - '0';
            if (digit % 2 == 1) {
                result = num.substr(0, i + 1);
            }
        }
        return result;
    }
};
```

✅ **Python (Forward)**

```
class Solution:
    def largestOddNumber(self, num: str) -> str:
        result = ""
        for i in range(len(num)):
            if int(num[i]) % 2 == 1:
                result = num[:i+1]
        return result
```

🟤 **Approach 2: Backward Traversal (Right → Left) ☆ BEST & CLEAN**

💡 **Idea**

- Traverse the string **from right to left**
- The **first odd digit** you encounter is the best answer
- Immediately return the prefix ending there

Why this is better

- No need to keep updating answers
- Stops early (faster in practice)
- Very clean logic

□ Step-by-Step Example

Input:

num = "35427"

Backward traversal:

7 → odd → STOP

Return num[0 : index+1] → "35427"

⏱ Complexity

- **Time:** O(n)
- **Space:** O(1)
- **More optimal in practice**

✅ Java (Backward)

```
class Solution {
    public String largestOddNumber(String num) {
        for (int i = num.length() - 1; i >= 0; i--) {
            int digit = num.charAt(i) - '0';
            if (digit % 2 == 1) {
                return num.substring(0, i + 1);
            }
        }
        return "";
    }
}
```

✅ C++ (Backward)

```
class Solution {
public:
    string largestOddNumber(string num) {
        for (int i = num.size() - 1; i >= 0; i--) {
            int digit = num[i] - '0';
```

```
            if (digit % 2 == 1) {
                return num.substr(0, i + 1);
            }
        }
        return "";
    }
};
```

✅ **Python (Backward)**

```
class Solution:
    def largestOddNumber(self, num: str) -> str:
        for i in range(len(num) - 1, -1, -1):
            if int(num[i]) % 2 == 1:
                return num[:i+1]
        return ""
```

🏁 **Final Comparison (Interview Ready)**

Approach	Traversal	Time	Space	Notes
Forward	Left → Right	O(n)	O(1)	Works but updates result
Backward	Right → Left	O(n)	O(1)	☆ Cleaner & preferred