Data Structure Design...

Video-13

Leetcode - 1845

~~Medium~~

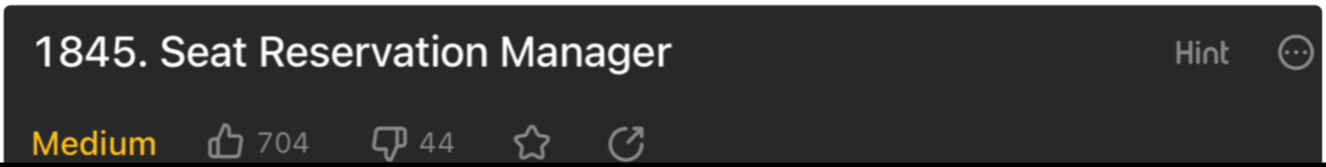
Facebook
Instagram → codestorywithMIK

(Twitter) → CSwithMIK

codestorywithMIK →

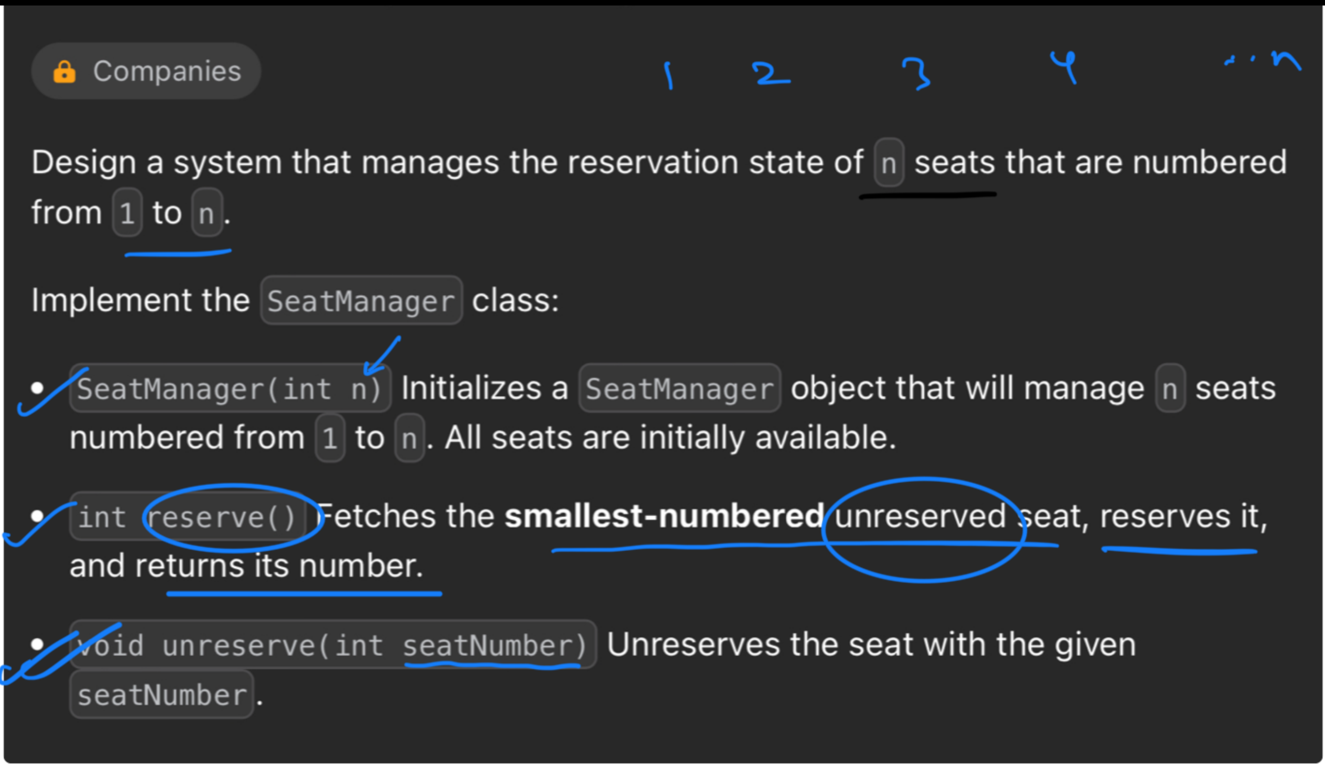
Seat Reservation Manager...

Companies :- Meta



1845. Seat Reservation Manager

Medium 704 44

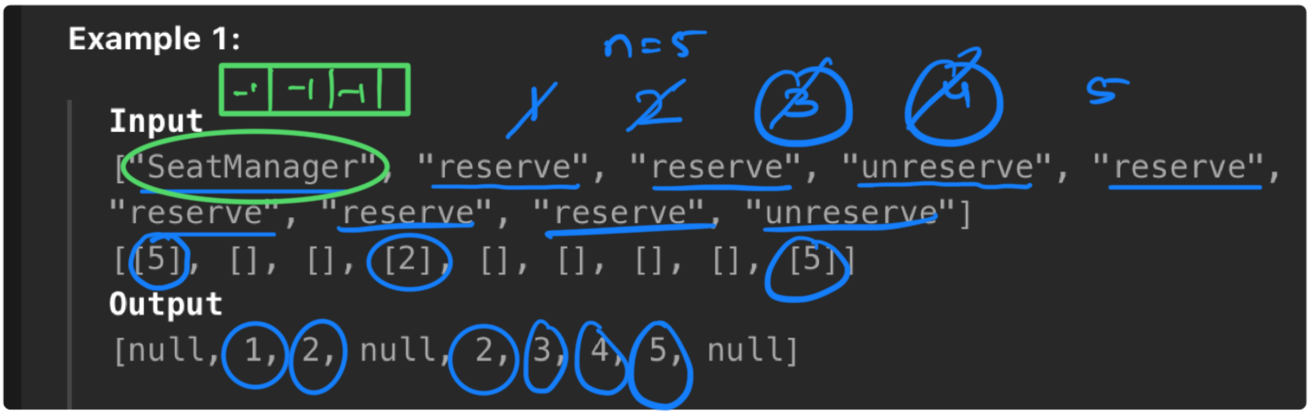


🔒 Companies

Design a system that manages the reservation state of `n` seats that are numbered from `1` to `n`.

Implement the `SeatManager` class:

- `SeatManager(int n)` Initializes a `SeatManager` object that will manage `n` seats numbered from `1` to `n`. All seats are initially available.
- `int reserve()` Fetches the **smallest-numbered** unreserved seat, reserves it, and returns its number.
- `void unreserve(int seatNumber)` Unreserves the seat with the given `seatNumber`.



Example 1:

```
Input
["SeatManager", "reserve", "reserve", "unreserve", "reserve",
"reserve", "reserve", "reserve", "unreserve"]
[[5], [], [], [2], [], [], [], [], [5]]
Output
[null, 1, 2, null, 2, 3, 4, 5, null]
```

Brute Force :-

★ We just have to keep track of seats. → Unreserved = -1 ×

↳ Reserved = 1 ✓

☆ Take an array to denote seats."

(1 2 . . . n)

n=5

0	1	2	3	4	5
	1	-1	-1	-1	-1

↑

(n+1)

O(n)

m times

```
reserve() {
    // array traversal // O(n) ←
    // pick smallest seat with -1 .
}
```

} O(m*n) ← TLE.

```
unreserve(seat) {
        arr[seat] = -1 ;   O(1).
}
```

Optimal Approach

Intuition → Why → where was the Problem ?

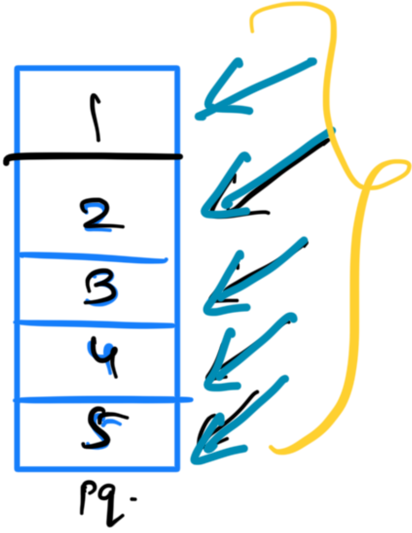
```
            m times
reserve() {                                  O(m*n)
    // array traversal  ←  O(n)
    // pick smallest seat with -1 .
}
```

"We need a Data Structure which can find me the smallest seat in better time complexity". ⇒ " Min heap"

⇒ ① $\boxed{\text{pq} \rightarrow \text{min\_heap}}$

m log n.

```
O(n log n) → for(i=1; i<=n; i++) {  ←
                 pq.push(i)  ← log(n)
```



m ↓

② reser

reserve → smallest

$O(m*\log(n))$

seat = Pq.top(); //1 → O(1)

Pq.pop(); ← $O(\log(n))$

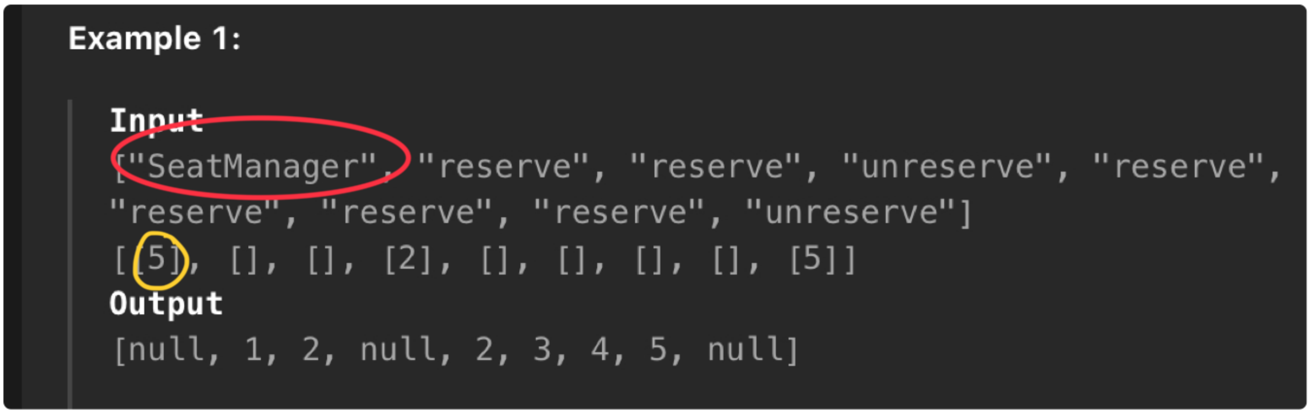
return seat; //1

③ unreserve(seat) m

Pq.push(seat); $O(\log(n))$

Improvement :-

without Pre-filling the Pq.



Example 1:

```
Input
["SeatManager", "reserve", "reserve", "unreserve", "reserve",
"reserve", "reserve", "reserve", "unreserve"]
[[5], [], [], [2], [], [], [], [], [5]]
Output
[null, 1, 2, null, 2, 3, 4, 5, null]
```

~~1~~ ~~2~~ ~~3~~ 4 5

Seat-marker = ~~1~~ ~~2~~ 3

```
SeatManager (int n) {

    Seat-marker = 1 ;   //   O(1).

}


int reserve ( )     {

    if (!Pq. empty()) {
        seat = Pq. top();   pq. pop(); ← log(n).
        return seat;
    }
    seat = seat-marker; //3
    seat-marker ++; // 4   ←  } O(1).
    ret seat;
}


void unreserve (int seatNumber) {

        Pq.Push (seatNumber);   // log(n).
```

y