Deck Revealed Increasing - Solution Notes

Problem Statement

You are given an integer array deck. Each card in deck has a unique integer. Initially, all cards are **face down** in one deck. The goal is to reorder the deck such that when you reveal cards following a specific rule, they appear in **increasing order**.

Revealing Process:

1. Take the top card of the deck, **reveal** it, and remove it from the deck.
2. If there are still cards left, move the next **top card** to the bottom of the deck.
3. Repeat the process until all cards are revealed.

Example 1:

Input: deck = [17,13,11,2,3,5,7]

Output: [2,13,3,11,5,17,7]

Explanation:

- After sorting: [2,3,5,7,11,13,17]
- Reordering the deck such that revealing follows the pattern described.

Thinking Process

1. **Sorting**: First, sort the deck in ascending order.
2. **Queue to simulate the revealing process**:
    - A queue stores the **indices** of positions where cards will be placed.
    - Place the smallest card at the front index, remove that index from the queue.
    - Move the next index to the bottom of the queue.
    - Repeat until all cards are placed correctly.

Code Implementation

```
class Solution {
public:
   vector<int> deckRevealedIncreasing(vector<int>& v) {
      sort(v.begin(), v.end());
      queue<int> q;
```

```
        for(int i = 0; i < v.size(); i++) {
            q.push(i);
        }

        vector<int> ans(v.size());
        int j = 0;

        while(q.size() > 0) {
            ans[q.front()] = v[j];
            q.pop();
            j++;

            if(q.size() == 0) break;

            q.push(q.front());
            q.pop();
        }

        return ans;
    }
};
```

Detailed Code Breakdown

1. **Sorting the deck**: The deck is sorted to ensure we place the smallest card first.
2. **Using a queue for index management**:
    - The queue initially stores indices [0,1,2,...,n-1].
    - The smallest card is placed at q.front(), and then removed from the queue.
    - The next available index is moved to the bottom of the queue.
3. **Placing elements in the correct order**:
    - The process ensures that when revealed, the cards appear in increasing order.

Conclusion

- **Time Complexity**: O(n log n), due to sorting and queue operations.
- **Space Complexity**: O(n), as we use an extra queue and answer array.
- This approach ensures that the deck is reordered efficiently to match the required reveal sequence.

Final Thought: The trick to solving this problem is **simulating the revealing process using a queue** rather than directly manipulating the deck. 🚀