

710. Random Pick with Blacklist

Description

You are given an integer `n` and an array of **unique** integers `blacklist`. Design an algorithm to pick a random integer in the range `[0, n - 1]` that is **not** in `blacklist`. Any integer that is in the mentioned range and not in `blacklist` should be **equally likely** to be returned.

Optimize your algorithm such that it minimizes the number of calls to the **built-in** random function of your language.

Implement the `Solution` class:

- `Solution(int n, int[] blacklist)` Initializes the object with the integer `n` and the blacklisted integers `blacklist`.
- `int pick()` Returns a random integer in the range `[0, n - 1]` and not in `blacklist`.

Example 1:

```
Input
["Solution", "pick", "pick", "pick", "pick", "pick", "pick", "pick"]
[[7, [2, 3, 5]], [], [], [], [], [], [], []]
Output
[null, 0, 4, 1, 6, 1, 0, 4]

Explanation
Solution solution = new Solution(7, [2, 3, 5]);
solution.pick(); // return 0, any integer from [0,1,4,6] should be ok. Note that for every call of pick,
                  // 0, 1, 4, and 6 must be equally likely to be returned (i.e., with probability 1/4).
solution.pick(); // return 4
solution.pick(); // return 1
solution.pick(); // return 6
solution.pick(); // return 1
solution.pick(); // return 0
solution.pick(); // return 4
```

Constraints:

- $1 \leq n \leq 10^9$
- $0 \leq \text{blacklist.length} \leq \min(10^5, n - 1)$
- $0 \leq \text{blacklist}[i] < n$
- All the values of `blacklist` are **unique**.
- At most $2 * 10^4$ calls will be made to `pick`.

