Random Pick Index

Given an array of integers with possible duplicates, randomly output the index of a given target number. You can assume that the given target number must exist in the array.

Note:
The array size can be very large. Solution that uses too much extra space will not pass the judge.

Example:

```
int[] nums = new int[] {1,2,3,3,3};
Solution solution = new Solution(nums);

// pick(3) should return either index 2, 3, or 4 randomly. Each index should have e
qual probability of returning.
solution.pick(3);

// pick(1) should return 0. Since in the array only nums[0] is equal to 1.
solution.pick(1);
```

Subscribe to see which companies asked this question

Solution 1

```
public class Solution {

    int[] nums;
    Random rnd;

    public Solution(int[] nums) {
        this.nums = nums;
        this.rnd = new Random();
    }

    public int pick(int target) {
        int result = -1;
        int count = 0;
        for (int i = 0; i < nums.length; i++) {
            if (nums[i] != target)
                continue;
            if (rnd.nextInt(++count) == 0)
                result = i;
        }

        return result;
    }
}```
```

written by dettier original link here

Solution 2

Because I've made a rather naive map-of-index-lists Java solution and it was happily accepted by the OJ. So far I see three types of solutions:

1. Like mine, O(N) memory, O(N) init, O(1) pick.
2. Like @dettier's Reservoir Sampling. O(1) init, O(1) memory, but O(N) to pick.
3. Like @chin-heng's binary search: O(N) memory, O(N lg N) init, O(lg N) pick.

Are all three kinds acceptable?

written by SergeyTachenov original link here

Solution 3

```
class Solution {
public:
    vector<int> n;
    Solution(vector<int> nums)
    {
        n = nums;
    }

    int pick(int target)
    {
        int count = 0, res = -1;
        for (int i = 0; i < n.size(); ++i)
        {
            if(n[i] != target) continue;
            if(++count == 1) res = i;
            else
                if(!(rand()%count)) res = i;
        }
        return res;
    }
};
```

written by ycf303 original link here