Problem 710: Random Pick with Blacklist

Problem Information

Difficulty: Hard
Acceptance Rate: 0.00%
Paid Only: No

Problem 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 <= n <= 10

9

0 <= blacklist.length <= min(10

5

, n - 1)

0 <= blacklist[i] < n

All the values of

blacklist

are

unique

.

At most

2 * 10

4

calls will be made to

pick

.

Code Snippets

C++:

```
class Solution {
public:
Solution(int n, vector<int>& blacklist) {

}

int pick() {

}
};

/**
* Your Solution object will be instantiated and called as such:
* Solution* obj = new Solution(n, blacklist);
* int param_1 = obj->pick();
*/
```

Java:

```
class Solution {

public Solution(int n, int[] blacklist) {

}

public int pick() {

}
}

/**
* Your Solution object will be instantiated and called as such:
* Solution obj = new Solution(n, blacklist);
* int param_1 = obj.pick();
*/
```

Python3:

```
class Solution:

def __init__(self, n: int, blacklist: List[int]):



def pick(self) -> int:




# Your Solution object will be instantiated and called as such:
# obj = Solution(n, blacklist)
# param_1 = obj.pick()
```

Python:

```
class Solution(object):

def __init__(self, n, blacklist):
"""
:type n: int
:type blacklist: List[int]
"""
```

```
def pick(self):
"""
:rtype: int
"""



# Your Solution object will be instantiated and called as such:
# obj = Solution(n, blacklist)
# param_1 = obj.pick()
```

JavaScript:

```
/**
* @param {number} n
* @param {number[]} blacklist
*/
var Solution = function(n, blacklist) {

};

/**
* @return {number}
*/
Solution.prototype.pick = function() {

};

/**
* Your Solution object will be instantiated and called as such:
* var obj = new Solution(n, blacklist)
* var param_1 = obj.pick()
*/
```

TypeScript:

```
class Solution {
constructor(n: number, blacklist: number[]) {

}
pick(): number {

}
}

/**
* Your Solution object will be instantiated and called as such:
* var obj = new Solution(n, blacklist)
* var param_1 = obj.pick()
*/
```

C#:

```
public class Solution {

public Solution(int n, int[] blacklist) {

}

public int Pick() {

}
}

/**
* Your Solution object will be instantiated and called as such:
* Solution obj = new Solution(n, blacklist);
* int param_1 = obj.Pick();
*/
```

C:

```
typedef struct {

} Solution;


Solution* solutionCreate(int n, int* blacklist, int blacklistSize) {
}

int solutionPick(Solution* obj) {

}

void solutionFree(Solution* obj) {

}

/**
* Your Solution struct will be instantiated and called as such:
* Solution* obj = solutionCreate(n, blacklist, blacklistSize);
* int param_1 = solutionPick(obj);

* solutionFree(obj);
*/
```

Go:

```
type Solution struct {

}


func Constructor(n int, blacklist []int) Solution {

}


func (this *Solution) Pick() int {

}


/**
* Your Solution object will be instantiated and called as such:
* obj := Constructor(n, blacklist);
* param_1 := obj.Pick();
*/
```

Kotlin:

```
class Solution(n: Int, blacklist: IntArray) {

fun pick(): Int {

}

}

/**
* Your Solution object will be instantiated and called as such:
* var obj = Solution(n, blacklist)
* var param_1 = obj.pick()
*/
```

Swift:

```


class Solution {

init(_ n: Int, _ blacklist: [Int]) {

}

func pick() -> Int {

}
}

/**
* Your Solution object will be instantiated and called as such:
* let obj = Solution(n, blacklist)
* let ret_1: Int = obj.pick()
*/
```

Rust:

```
struct Solution {

}
```

```
/**
* `&self` means the method takes an immutable reference.
* If you need a mutable reference, change it to `&mut self` instead.
*/
impl Solution {

fn new(n: i32, blacklist: Vec<i32>) -> Self {

}

fn pick(&self) -> i32 {

}
}

/**
* Your Solution object will be instantiated and called as such:
* let obj = Solution::new(n, blacklist);
* let ret_1: i32 = obj.pick();
*/
```

Ruby:

```
class Solution

=begin
:type n: Integer
:type blacklist: Integer[]
=end
def initialize(n, blacklist)

end



=begin
:rtype: Integer
=end
def pick()

end
```

```
end

# Your Solution object will be instantiated and called as such:
# obj = Solution.new(n, blacklist)
# param_1 = obj.pick()
```

PHP:

```
class Solution {
/**
* @param Integer $n
* @param Integer[] $blacklist
*/
function __construct($n, $blacklist) {

}

/**
* @return Integer
*/
function pick() {

}
}

/**
* Your Solution object will be instantiated and called as such:
* $obj = Solution($n, $blacklist);
* $ret_1 = $obj->pick();
*/
```

Dart:

```
class Solution {

Solution(int n, List<int> blacklist) {

}

int pick() {

}
}

/**
* Your Solution object will be instantiated and called as such:
* Solution obj = Solution(n, blacklist);
* int param1 = obj.pick();
*/
```

Scala:

```
class Solution(_n: Int, _blacklist: Array[Int]) {

def pick(): Int = {

}

}

/**
* Your Solution object will be instantiated and called as such:
* val obj = new Solution(n, blacklist)
* val param_1 = obj.pick()
*/
```

Elixir:

```
defmodule Solution do
@spec init_(n :: integer, blacklist :: [integer]) :: any
def init_(n, blacklist) do

end

@spec pick() :: integer
def pick() do

end
end

# Your functions will be called as such:
# Solution.init_(n, blacklist)
# param_1 = Solution.pick()
# Solution.init_ will be called before every test case, in which you can do some necessary initializations.
```

Erlang:

```
-spec solution_init_(N :: integer(), Blacklist :: [integer()]) -> any().
solution_init_(N, Blacklist) ->
.

-spec solution_pick() -> integer().
solution_pick() ->
.



%% Your functions will be called as such:
%% solution_init_(N, Blacklist),
%% Param_1 = solution_pick(),

%% solution_init_ will be called before every test case, in which you can do some necessary initializations.
```

Racket:

```
(define solution%
(class object%
(super-new)

; n : exact-integer?
; blacklist : (listof exact-integer?)
(init-field
n
blacklist)

; pick : -> exact-integer?
(define/public (pick)
)))

;; Your solution% object will be instantiated and called as such:
;; (define obj (new solution% [n n] [blacklist blacklist]))
;; (define param_1 (send obj pick))
```

Solutions

C++ Solution:

```
/*
* Problem: Random Pick with Blacklist
* Difficulty: Hard
* Tags: array, math, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
Solution(int n, vector<int>& blacklist) {

}

int pick() {

}
};

/**
* Your Solution object will be instantiated and called as such:
* Solution* obj = new Solution(n, blacklist);
* int param_1 = obj->pick();
*/
```

Java Solution:

```
/**
* Problem: Random Pick with Blacklist
* Difficulty: Hard
* Tags: array, math, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/
```

```
class Solution {

public Solution(int n, int[] blacklist) {

}

public int pick() {

}
}

/**
* Your Solution object will be instantiated and called as such:
* Solution obj = new Solution(n, blacklist);
* int param_1 = obj.pick();
*/
```

Python3 Solution:

```
"""
Problem: Random Pick with Blacklist
Difficulty: Hard
Tags: array, math, hash, sort, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:

def __init__(self, n: int, blacklist: List[int]):



def pick(self) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):

def __init__(self, n, blacklist):
"""
:type n: int
:type blacklist: List[int]
"""



def pick(self):
"""
:rtype: int
"""




# Your Solution object will be instantiated and called as such:
# obj = Solution(n, blacklist)
# param_1 = obj.pick()
```

JavaScript Solution:

```
/**
* Problem: Random Pick with Blacklist
* Difficulty: Hard
* Tags: array, math, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {number} n
* @param {number[]} blacklist
*/
var Solution = function(n, blacklist) {

};

/**
* @return {number}
```

```
*/
Solution.prototype.pick = function() {

};

/**
* Your Solution object will be instantiated and called as such:
* var obj = new Solution(n, blacklist)
* var param_1 = obj.pick()
*/
```

TypeScript Solution:

```
/**
* Problem: Random Pick with Blacklist
* Difficulty: Hard
* Tags: array, math, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
constructor(n: number, blacklist: number[]) {

}

pick(): number {

}
}

/**
* Your Solution object will be instantiated and called as such:
* var obj = new Solution(n, blacklist)
* var param_1 = obj.pick()
*/
```

C# Solution:

```
/*
* Problem: Random Pick with Blacklist
* Difficulty: Hard
* Tags: array, math, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {

public Solution(int n, int[] blacklist) {

}

public int Pick() {

}
}

/**
* Your Solution object will be instantiated and called as such:
* Solution obj = new Solution(n, blacklist);
* int param_1 = obj.Pick();
*/
```

C Solution:

```
/*
* Problem: Random Pick with Blacklist
* Difficulty: Hard
* Tags: array, math, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/
```

```
typedef struct {

} Solution;


Solution* solutionCreate(int n, int* blacklist, int blacklistSize) {

}

int solutionPick(Solution* obj) {

}

void solutionFree(Solution* obj) {

}

/**
* Your Solution struct will be instantiated and called as such:
* Solution* obj = solutionCreate(n, blacklist, blacklistSize);
* int param_1 = solutionPick(obj);

* solutionFree(obj);
*/
```

Go Solution:

```
// Problem: Random Pick with Blacklist
// Difficulty: Hard
// Tags: array, math, hash, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

type Solution struct {

}


func Constructor(n int, blacklist []int) Solution {
}


func (this *Solution) Pick() int {

}


/**
* Your Solution object will be instantiated and called as such:
* obj := Constructor(n, blacklist);
* param_1 := obj.Pick();
*/
```

Kotlin Solution:

```
class Solution(n: Int, blacklist: IntArray) {

fun pick(): Int {

}

}

/**
* Your Solution object will be instantiated and called as such:
* var obj = Solution(n, blacklist)
* var param_1 = obj.pick()
*/
```

Swift Solution:

```

class Solution {

init(_ n: Int, _ blacklist: [Int]) {

}

func pick() -> Int {


}
}

/**
* Your Solution object will be instantiated and called as such:
* let obj = Solution(n, blacklist)
* let ret_1: Int = obj.pick()
*/
```

Rust Solution:

```
// Problem: Random Pick with Blacklist
// Difficulty: Hard
// Tags: array, math, hash, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

struct Solution {

}



/**
* `&self` means the method takes an immutable reference.
* If you need a mutable reference, change it to `&mut self` instead.
*/
impl Solution {

fn new(n: i32, blacklist: Vec<i32>) -> Self {

}

fn pick(&self) -> i32 {

}
}

/**
```

```
* Your Solution object will be instantiated and called as such:
* let obj = Solution::new(n, blacklist);
* let ret_1: i32 = obj.pick();
*/
```

Ruby Solution:

```
class Solution

=begin
:type n: Integer
:type blacklist: Integer[]
=end
def initialize(n, blacklist)

end



=begin
:rtype: Integer
=end
def pick()

end



end

# Your Solution object will be instantiated and called as such:
# obj = Solution.new(n, blacklist)
# param_1 = obj.pick()
```

PHP Solution:

```
class Solution {
/**
* @param Integer $n
* @param Integer[] $blacklist
*/
function __construct($n, $blacklist) {
}

/**
* @return Integer
*/
function pick() {

}
}

/**
* Your Solution object will be instantiated and called as such:
* $obj = Solution($n, $blacklist);
* $ret_1 = $obj->pick();
*/
```

Dart Solution:

```
class Solution {

Solution(int n, List<int> blacklist) {

}

int pick() {

}
}

/**
* Your Solution object will be instantiated and called as such:
* Solution obj = Solution(n, blacklist);
* int param1 = obj.pick();
*/
```

Scala Solution:

```
class Solution(_n: Int, _blacklist: Array[Int]) {

def pick(): Int = {
}

}

/**
* Your Solution object will be instantiated and called as such:
* val obj = new Solution(n, blacklist)
* val param_1 = obj.pick()
*/
```

Elixir Solution:

```
defmodule Solution do
@spec init_(n :: integer, blacklist :: [integer]) :: any
def init_(n, blacklist) do

end

@spec pick() :: integer
def pick() do

end
end

# Your functions will be called as such:
# Solution.init_(n, blacklist)
# param_1 = Solution.pick()

# Solution.init_ will be called before every test case, in which you can do
some necessary initializations.
```

Erlang Solution:

```
-spec solution_init_(N :: integer(), Blacklist :: [integer()]) -> any().
solution_init_(N, Blacklist) ->
.

-spec solution_pick() -> integer().
solution_pick() ->
.
```

```
%% Your functions will be called as such:
%% solution_init_(N, Blacklist),
%% Param_1 = solution_pick(),

%% solution_init_ will be called before every test case, in which you can do
some necessary initializations.
```

Racket Solution:

```
(define solution%
(class object%
(super-new)

; n : exact-integer?
; blacklist : (listof exact-integer?)
(init-field
n
blacklist)

; pick : -> exact-integer?
(define/public (pick)
)))

;; Your solution% object will be instantiated and called as such:
;; (define obj (new solution% [n n] [blacklist blacklist]))
;; (define param_1 (send obj pick))
```