

6113. Smallest Number in Infinite Set

My Submissions (/contest/weekly-contest-301/problems/smallest-number-in-infinite-set/submissions/)

Back to Contest (/contest/weekly-contest-301/)

You have a set which contains all positive integers [1, 2, 3, 4, 5, ...].

Implement the `SmallestInfiniteSet` class:

- `SmallestInfiniteSet()` Initializes the **`SmallestInfiniteSet`** object to contain **all** positive integers.
- `int popSmallest()` **Removes** and returns the smallest integer contained in the infinite set.
- `void addBack(int num)` **Adds** a positive integer `num` back into the infinite set, if it is **not** already in the infinite set.

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Medium

Example 1:

**Input**  
["SmallestInfiniteSet", "addBack", "popSmallest", "popSmallest", "popSmallest", "addBack", "popSmallest", "popSmallest", "popSmallest"]  
[[], [2], [], [], [], [1], [], [], []]

**Output**  
[null, null, 1, 2, 3, null, 1, 4, 5]

**Explanation**  
SmallestInfiniteSet smallestInfiniteSet = new SmallestInfiniteSet();  
smallestInfiniteSet.addBack(2); // 2 is already in the set, so no change is made.  
smallestInfiniteSet.popSmallest(); // return 1, since 1 is the smallest number, and remove it from the set.  
smallestInfiniteSet.popSmallest(); // return 2, and remove it from the set.  
smallestInfiniteSet.popSmallest(); // return 3, and remove it from the set.  
smallestInfiniteSet.addBack(1); // 1 is added back to the set.  
smallestInfiniteSet.popSmallest(); // return 1, since 1 was added back to the set and  
// is the smallest number, and remove it from the set.  
smallestInfiniteSet.popSmallest(); // return 4, and remove it from the set.  
smallestInfiniteSet.popSmallest(); // return 5, and remove it from the set.

Constraints:

- 1 <= num <= 1000
- At most 1000 calls will be made in total to `popSmallest` and `addBack`.

JavaScript

1 function Bisect() {  
2 return { insert\_right, insert\_left, bisect\_left, bisect\_right }  
3 function insert\_right(a, x, lo = 0, hi = null) {  
4 lo = bisect\_right(a, x, lo, hi);  
5 a.splice(lo, 0, x);  
6 }  
7 function bisect\_right(a, x, lo = 0, hi = null) { // > upper\_bound  
8 if (lo < 0) throw new Error('lo must be non-negative');  
9 if (hi == null) hi = a.length;  
10 while (lo < hi) {  
11 let mid = parseInt((lo + hi) / 2);  
12 a[mid] > x ? hi = mid : lo = mid + 1;  
13 }  
14 return lo;  
15 }  
16 function insert\_left(a, x, lo = 0, hi = null) {  
17 lo = bisect\_left(a, x, lo, hi);  
18 a.splice(lo, 0, x);  
19 }  
20 function bisect\_left(a, x, lo = 0, hi = null) { // >= lower\_bound  
21 if (lo < 0) throw new Error('lo must be non-negative');  
22 if (hi == null) hi = a.length;  
23 while (lo < hi) {  
24 let mid = parseInt((lo + hi) / 2);  
25 a[mid] < x ? lo = mid + 1 : hi = mid;  
26 }  
27 return lo;  
28 }

https://leetcode.com/contest/weekly-contest-301/problems/smallest-number-in-infinite-set/

1/3

```

28     }
29 }
30
31 function TreeSet(elements) {
32     let ts = [], se = new Set(), bisect = new Bisect();
33     initialize();
34     return { add, first, last, poll, pollLast, floor, ceiling, lower, higher, remove, contains, size, clear, show };
35     function initialize() {
36         if (elements) {
37             for (const e of elements) {
38                 if (!se.has(e)) {
39                     bisect.insort_right(ts, e);
40                     se.add(e);
41                 }
42             }
43         }
44     }
45     function add(e) {
46         if (!se.has(e)) {
47             bisect.insort_right(ts, e);
48             se.add(e);
49         }
50     }
51     function first() {
52         return ts[0];
53     }
54     function last() {
55         return ts[ts.length - 1];
56     }
57     function poll() {
58         let res = ts[0];
59         ts.splice(0, 1);
60         se.delete(res);
61         return res;
62     }
63     function pollLast() {
64         let res = ts.pop();
65         se.delete(res);
66         return res;
67     }
68     function ceiling(e) { // >= lower_bound
69         let idx = bisect.bisect_right(ts, e);
70         let res = ts[idx - 1] == e ? e : ts[bisect.bisect_right(ts, e)];
71         return res == undefined ? null : res;
72     }
73     function higher(e) { // > upper_bound
74         let idx = bisect.bisect_right(ts, e);
75         let res = ts[idx] > e ? ts[idx] : ts[bisect.bisect_right(ts, e) + 1];
76         return res == undefined ? null : res;
77     }
78     function floor(e) { // <=
79         let idx = bisect.bisect_left(ts, e);
80         let res = ts[idx] == e ? e : ts[bisect.bisect_left(ts, e) - 1];
81         return res == undefined ? null : res;
82     }
83     function lower(e) { // <
84         let idx = bisect.bisect_left(ts, e);
85         let res = ts[idx] < e ? ts[idx] : ts[bisect.bisect_left(ts, e) - 1];
86         return res == undefined ? null : res;
87     }
88     function remove(e) {
89         let idx = bisect.bisect_left(ts, e);
90         if (ts[idx] == e) ts.splice(idx, 1);
91         se.delete(e);
92     }
93     function contains(e) {
94         return se.has(e);
95     }
96     function size() {
97         return ts.length;
98     }
99     function clear() {
100         ts = [];
101         se.clear();
102     }
103     function show() {
104         return ts;


```

```
105     }
106   }
107
108   function SmallestInfiniteSet() {
109     let ts = new TreeSet();
110     for (let i = 1; i <= 1000; i++) ts.add(i);
111     return { popSmallest, addBack };
112   }
113   function popSmallest() {
114     return ts.poll();
115   }
116   function addBack(x) {
117     ts.add(x);
118   }
119 }
```

☐ Custom Testcase[Use Example Testcases](#)[Run](#)[Submit](#)**Submission Result: Accepted** (</submissions/detail/743038744/>) ?[More Details >](/submissions/detail/743038744/)

Share your acceptance!

Copyright © 2022 LeetCode

[Help Center \(/support/\)](/support/) | [Jobs \(/jobs/\)](/jobs/) | [Bug Bounty \(/bugbounty/\)](/bugbounty/) | [Online Interview \(/interview/\)](/interview/) | [Students \(/student/\)](/student/) | [Terms \(/terms/\)](/terms/) | [Privacy Policy \(/privacy/\)](/privacy/) [United States \(/region/\)](/region/)