

281. Zigzag Iterator

Medium 512 27 Add to List Share

Given two vectors of integers `v1` and `v2`, implement an iterator to return their elements alternately.

Implement the `ZigzagIterator` class:

- `ZigzagIterator(List<int> v1, List<int> v2)` initializes the object with the two vectors `v1` and `v2`.
- `boolean hasNext()` returns `true` if the iterator still has elements, and `false` otherwise.
- `int next()` returns the current element of the iterator and moves the iterator to the next element.

Example 1:

Input: `v1 = [1,2], v2 = [3,4,5,6]`
Output: `[1,3,2,4,5,6]`
Explanation: By calling `next` repeatedly until `hasNext` returns `false`, the order of elements returned by `next` should be: `[1,3,2,4,5,6]`.

Example 2:

Input: `v1 = [1], v2 = []`
Output: `[1]`

Example 3:

Input: `v1 = [], v2 = [1]`
Output: `[1]`

Constraints:

- $0 \leq v1.length, v2.length \leq 1000$
- $1 \leq v1.length + v2.length \leq 2000$
- $-2^{31} \leq v1[i], v2[i] \leq 2^{31} - 1$

Follow up: What if you are given `k` vectors? How well can your code be extended to such cases?

Clarification for the follow-up question:


The "Zigzag" order is not clearly defined and is ambiguous for $k > 2$ cases. If "Zigzag" does not look right to you, replace "Zigzag" with "Cyclic".

Example:

Input: `v1 = [1,2,3], v2 = [4,5,6,7], v3 = [8,9]`
Output: `[1,4,8,2,5,9,3,6,7]`

Accepted 75,416 Submissions 124,406

Seen this question in a real interview before?

Companies  

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Yandex 3 Amazon 2

Related Topics

Array Design Queue Iterator

Similar Questions

Binary Search Tree Iterator	Medium
Flatten 2D Vector	Medium
Peeking Iterator	Medium
Flatten Nested List Iterator	Medium
Merge Strings Alternately	Easy

```
1 public class ZigzagIterator {
2     private List<List<Integer>> vectors = new ArrayList<>();
3     private LinkedList<Pair<Integer, Integer>> queue = new LinkedList<>();
4
5     public ZigzagIterator(List<Integer> v1, List<Integer> v2) {
6         this.vectors.add(v1);
7         this.vectors.add(v2);
8         int index = 0;
9         for (List<Integer> vec : this.vectors) {
10             if (vec.size() > 0)
11                 // <index to vec, index to element within vec>
12                 this.queue.add(new Pair<Integer, Integer>(index, 0));
13             index++;
14         }
15     }
16
17     public int next() {
18         // if (this.queue.size() == 0)
19         // throw new Exception("Out of elements!");
20
21         // <index to vec, index to element within vec>
22         Pair<Integer, Integer> pointer = this.queue.removeFirst();
23         Integer vecIndex = pointer.getKey();
24         Integer elemIndex = pointer.getValue();
25         Integer nextElemIndex = elemIndex + 1;
26         // append the pointer for the next round
27         // if there are some elements left.
28         if (nextElemIndex < this.vectors.get(vecIndex).size())
29             this.queue.addLast(new Pair<>(vecIndex, nextElemIndex));
30
31         return this.vectors.get(vecIndex).get(elemIndex);
32     }
33
34     public boolean hasNext() {
35         return this.queue.size() > 0;
36     }
37 }
```