Description

281. Zigzag Iterator

Given two 1d vectors, implement an iterator to return their elements alternately.

For example, given two 1d vectors:

```
v1 = [1, 2]
v2 = [3, 4, 5, 6]
```

By calling next repeatedly until hasNext returns false, the order of elements returned by next should be: [1, 3, 2, 4, 5, 6].

Follow up: What if you are given k 1d vectors? How well can your code be extended to such cases?

```
[1,2,3]
[4,5,6,7]
[8,9]
```

It should return [1,4,8,2,5,9,3,6,7].

Idea

This is a very straightforward implementation of the iterator. The only tricky part is to handle if two inputs are in different size.

Java

```
public class ZigzagIterator {
```

```
    private int i1;
    private int i2;
    private List<Integer> v1;
    private List<Integer> v2;

    public ZigzagIterator(List<Integer> v1, List<Integer> v2) {
        i1 = 0;
        i2 = 0;
        this.v1 = new ArrayList<>(v1);
        this.v2 = new ArrayList<>(v2);
    }

    public int next() {
        if (i1 < v1.size() && i1 <= i2 || i2 == v2.size()) {
            return v1.get(i1++);
        } else {
            return v2.get(i2++);
        }
    }

    public boolean hasNext() {
        return i1 < v1.size() || i2 < v2.size();
    }
}
```

C++

```
class ZigzagIterator {

private:
    vector<int> v1;
    vector<int> v2;
    int i1;
    int i2;

public:
    ZigzagIterator(vector<int>& v1, vector<int>& v2) {
        this->v1 = v1;
        this->v2 = v2;
        i1 = 0;
        i2 = 0;
    }

    int next() {
        return (i1 < v1.size() && i1 <= i2 || i2 == v2.size()) ?
                v1[i1++] : v2[i2++];
    }
```

```
    bool hasNext() {
        return i1 < v1.size() || i2 < v2.size();
    }
};
```

Summary

- Iterator, pure implementation, straightforward