Flatten 2D Vector

Implement an iterator to flatten a 2d vector.

For example,
Given 2d vector =

```
[
  [1,2],
  [3],
  [4,5,6]
]
```

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

1. How many variables do you need to keep track?
2. Two variables is all you need. Try with `x` and `y`.
3. Beware of empty rows. It could be the first few rows.
4. To write correct code, think about the invariant to maintain. What is it?
5. The invariant is `x` and `y` must always point to a valid point in the 2d vector. Should you maintain your invariant *ahead of time* or *right when you need it*?
6. Not sure? Think about how you would implement `hasNext()`. Which is more complex?
7. Common logic in two different places should be refactored into a common method.

Follow up:
As an added challenge, try to code it using only iterators in C++ or iterators in Java.

Solution 1

Since the OJ does `while (i.hasNext()) cout << i.next();`, i.e., always calls `hasNext` before `next`, I don't really have to call it myself so I could save that line in `next`. But I think that would be bad, we shouldn't rely on that.

C++

```
class Vector2D {
    vector<vector<int>>::iterator i, iEnd;
    int j = 0;
public:
    Vector2D(vector<vector<int>>& vec2d) {
        i = vec2d.begin();
        iEnd = vec2d.end();
    }

    int next() {
        hasNext();
        return (*i)[j++];
    }

    bool hasNext() {
        while (i != iEnd && j == (*i).size())
            i++, j = 0;
        return i != iEnd;
    }
};
```

Java

```
public class Vector2D {

    private Iterator<List<Integer>> i;
    private Iterator<Integer> j;

    public Vector2D(List<List<Integer>> vec2d) {
        i = vec2d.iterator();
    }

    public int next() {
        hasNext();
        return j.next();
    }

    public boolean hasNext() {
        while ((j == null || !j.hasNext()) && i.hasNext())
            j = i.next().iterator();
        return j != null && j.hasNext();
    }
}
```

written by StefanPochmann original link here

Solution 2

//1. with positions of vectors

```
class Vector2D {
   int row;
   int col;
   vector<vector<int>> data;

public:
   Vector2D(vector<vector<int>>& vec2d) {
       data = vec2d;
       row = 0;
       col = 0;
   }

   int next() {
       return data[row][col++];
   }

   bool hasNext() {
       while(row < data.size() && data[row].size() == col)
           row++, col = 0;
       return row < data.size();
   }
};
```

//2. with Iterator

```
class Vector2D {
   vector<vector<int>> data;
   vector<vector<int>>::iterator rowIter;
   vector<int>::iterator colIter;

public:
   Vector2D(vector<vector<int>>& vec2d) {
       data = vec2d;
       rowIter = data.begin();
       if(rowIter != data.end())
           colIter = rowIter->begin();
   }

   int next() {
       int r = *colIter;
       colIter++;
       return r;
   }

   bool hasNext() {
       while(rowIter != data.end() && colIter == rowIter->end()) {
           rowIter++;
           if(rowIter != data.end())
                colIter = rowIter->begin();
       }

       return rowIter != data.end();
   }
 };
```

written by jaewoo original link here

Solution 3

- Put all iterator in a queue
- Keep track of the current iterator
- Check hasNext() and next() of current

public class Vector2D {

```
Queue<Iterator<Integer>> queue;
Iterator<Integer> current = null;

public Vector2D(List<List<Integer>> vec2d) {
    queue = new LinkedList<Iterator<Integer>>();
    for (int i = 0; i < vec2d.size(); i++){
        queue.add(vec2d.get(i).iterator());
    }
    current = queue.poll(); // first
}

public int next() {
    if (!current.hasNext()) return -1;

    return current.next();
}

public boolean hasNext() {
    if (current == null) return false;

    while (!current.hasNext()) {
        if (!queue.isEmpty()) {
            current = queue.poll();
        } else return false;
    }

    return true;
}
```

}

written by angelvivienne original link here