Pascal's Triangle II

Given an index k, return the k^{th} row of the Pascal's triangle.

For example, given $k = 3$,
Return `[1,3,3,1]`.

Note:
Could you optimize your algorithm to use only $O(k)$ extra space?

Solution 1

The basic idea is to iteratively update the array from the end to the beginning.

```
class Solution {
public:
    vector<int> getRow(int rowIndex) {
        vector<int> A(rowIndex+1, 0);
        A[0] = 1;
        for(int i=1; i<rowIndex+1; i++)
            for(int j=i; j>=1; j--)
                A[j] += A[j-1];
        return A;
    }
};
```

written by LongsPeak original link here

Solution 2

```
  public List<Integer> getRow(int rowIndex) {
    List<Integer> list = new ArrayList<Integer>();
    if (rowIndex < 0)
        return list;

    for (int i = 0; i < rowIndex + 1; i++) {
        list.add(0, 1);
        for (int j = 1; j < list.size() - 1; j++) {
            list.set(j, list.get(j) + list.get(j + 1));
        }
    }
    return list;
}
```

written by micheal.zhou original link here

Solution 3

```
public List<Integer> getRow(int rowIndex) {
    List<Integer> res = new ArrayList<Integer>();
    for(int i = 0;i<rowIndex+1;i++) {
            res.add(1);
            for(int j=i-1;j>0;j--) {
                res.set(j, res.get(j-1)+res.get(j));
            }
    }
    return res;
}
```

written by ggcrosemond original link here