Arrays: Leetcode

(Part 3)

Q118. Pascal's Triangle

```
vector<vector<int>> generate(int numRows) {
    vector<vector<int>> res(numRows);
    for (int i = 0; i < numRows; ++i) {
        res[i].resize(i+1, 1);
        for (int j = 1; j < i; ++j)
            res[i][j] = res[i-1][j-1] + res[i-1][j];
    }
    return res;
}
```

Q119. Pascal's Triangle II

```
vector<int> getRow(int rowIndex) {
    vector<int> row(rowIndex + 1, 1);
    for (int i = 1; i < rowIndex; ++i)
        for (int j = i; j >= 1; --j)
            row[j] += row[j - 1];
    return row;
}
```

Q134. Gas Station

```
int canCompleteCircuit(vector<int>& gas, vector<int>& cost)
{
    int total = 0, curr = 0, start = 0;
    for (int i = 0; i < gas.size(); ++i) {
        total += gas[i] - cost[i];
        curr += gas[i] - cost[i];
        if (curr < 0) {
            curr = 0;
            start = i + 1;
        }
    }
    return total < 0 ? -1 : start;
}
```

Q135. Candy

```
int candy(vector<int>& ratings) {
    int n = ratings.size(), res = 0;
    vector<int> candies(n, 1);
    for (int i = 1; i < n; ++i)
        if (ratings[i] > ratings[i-1]) candies[i] =
candies[i-1] + 1;
    for (int i = n - 2; i >= 0; --i)
        if (ratings[i] > ratings[i+1]) candies[i] =
max(candies[i], candies[i+1] + 1);
    return accumulate(candies.begin(), candies.end(), 0);
}
```

Q153. Find Minimum in Rotated Sorted Array

```
int findMin(vector<int>& nums) {
    int l = 0, r = nums.size() - 1;
    while (l < r) {
        int m = (l + r) / 2;
        if (nums[m] > nums[r]) l = m + 1;
        else r = m;
    }
    return nums[l];
}
```

Q162. Find Peak Element

```
int findPeakElement(vector<int>& nums) {
    int l = 0, r = nums.size() - 1;
    while (l < r) {
        int m = (l + r) / 2;
        if (nums[m] > nums[m + 1]) r = m;
        else l = m + 1;
    }
    return l;
}
```

Q189. Rotate Array

```
void rotate(vector<int>& nums, int k) {
    k %= nums.size();
    reverse(nums.begin(), nums.end());
    reverse(nums.begin(), nums.begin() + k);
    reverse(nums.begin() + k, nums.end());
}
```

Q198. House Robber

```
int rob(vector<int>& nums) {
    int prev1 = 0, prev2 = 0;
    for (int num : nums) {
        int tmp = prev1;
        prev1 = max(prev2 + num, prev1);
        prev2 = tmp;
    }
    return prev1;
}
```

Q200. Number of Islands

```
void dfs(vector<vector<char>>& grid, int i, int j) {
    if (i < 0 || j < 0 || i >= grid.size() || j >=
grid[0].size() || grid[i][j] == '0')
        return;
    grid[i][j] = '0';
    dfs(grid, i+1, j); dfs(grid, i-1, j);
    dfs(grid, i, j+1); dfs(grid, i, j-1);
}

int numIslands(vector<vector<char>>& grid) {
    int count = 0;
    for (int i = 0; i < grid.size(); ++i)
        for (int j = 0; j < grid[0].size(); ++j)
            if (grid[i][j] == '1') {
                dfs(grid, i, j);
                ++count;
            }
    return count;
}
```

Q219. Contains Duplicate II

```
bool containsNearbyDuplicate(vector<int>& nums, int k) {
    unordered_map<int, int> m;
    for (int i = 0; i < nums.size(); ++i) {
        if (m.count(nums[i]) && i - m[nums[i]] <= k)
            return true;
        m[nums[i]] = i;
    }
    return false;
}
```