Leetcode 860 柠檬水找零  
思路：模拟+贪心，由于顾客只可能给你三个面值的钞票，而且我们一开始没有任何钞票，只能接受顾客的钞票，分情况讨论如下：

* 如果最开始收到的是5块，直接手下
* 如果最开始收到的是10块，如果没有5块，则无法正确找零，
* 如果最开始收到的是20块，我们可以有两种找零的方式，10块+5块和5块+5块+5块这两种方式，第一种方式是我们比较推荐的，因为5块的使用场合较多。在第一种和第二种场合中均可以使用。

class Solution {

public boolean lemonadeChange(int[] bills) {

int five=0,ten=0;

for(int bill:bills){

if(bill==5){

five++;

}else if(bill==10){

if(five==0){

return false;

}

five--;

ten++;

}else{

if(five>0 && ten>0){

five--;

ten--;

}else if(five>3){

five-=3;

}else{

return false;

}

}

}

return true;

}

}

LeetCode 122 买卖股票的最佳时机II

思路：题目没有限制交易次数，所以，只要当前的股价比昨天的高就交易，这个代码可以计算最大的利润，但并不是真正的交易过程。

class Solution {

public int maxProfit(int[] prices) {

int ans=0;

int n=prices.length;

for(int i=1;i<n;i++){

ans+=Math.max(0,prices[i]-prices[i-1]);

}

return ans;

}

}

LeetCode 814 模拟机器人行走

思路：直接模拟机器人行走的过程，在行走的过程中，计算每一个坐标点与原点之间的欧式距离，得到距离的最大值。

机器人如何调整方向：机器人旋转的时候，如何变幻坐标没有搞懂这个问题。

LeetCode 455 分发饼干

思路：排序+贪心  
class Solution {

public boolean lemonadeChange(int[] bills) {

int five=0,ten=0;

for(int bill:bills){

if(bill==5){

five++;

}else if(bill==10){

if(five==0){

return false;

}

five--;

ten++;

}else{

if(five>0 && ten>0){

five--;

ten--;

}else if(five>3){

five-=3;

}else{

return false;

}

}

}

return true;

}

}

LeetCode 55 跳跃游戏  
思路：如果以某一个作为起跳点的格子可以跳跃的距离是3，那么表示后面的3个格子都是可以作为起跳点的；可以对每一个能作为起跳点的格子都尝试跳一次；

class Solution {

public boolean canJump(int[] nums) {

int k=0;

for(int i=0;i<nums.length;i++){

if(i>k) return false;

k=Math.max(k,i+nums[i]);

}

return true;

}

}

Leetcode 45 跳跃游戏  
思路：从数组的第0个位置开始跳，跳的距离小于等于数组上对应的数，求出跳到最后一个位置需要的最短步数。使用贪婪算法，在可跳的范围内选择可以使得跳的最远的位置。  
class Solution {

public int jump(int[] nums) {

int end=0;

int maxPosition=0;

int steps=0;

for(int i=0;i<nums.length-1;i++){

maxPosition=Math.max(maxPosition,nums[i]+i);

if(i==end){

end=maxPosition;

steps++;

}

}

return steps;

}

}

LeetCode 200岛屿数量

思路：利用深度优先搜索，如果有一个位置为1的话，则以这个点为起始节点开始进行深度优先搜索，在深度优先搜索的过程中，搜索到的1都会被重新标记为0。

class Solution {

void dfs(char[][] grid,int r,int c){

int nr=grid.length;

int nc=grid[0].length;

if(r<0 || c<0 || r>=nr || c>=nc|| grid[r][c]=='0'){

return;

}

grid[r][c]='0';

dfs(grid,r-1,c);

dfs(grid,r+1,c);

dfs(grid,r,c-1);

dfs(grid,r,c+1);

}

public int numIslands(char[][] grid) {

if(grid==null || grid.length==0){

return 0;

}

int nr=grid.length;

int nc=grid[0].length;

int count=0;

for(int r=0;r<nr;r++){

for(int c=0;c<nc;c++){

if(grid[r][c]=='1'){

count++;

dfs(grid,r,c);

}

}

}

return count;

}

}