Interviewer Packet A - TIPS 4

# Interviewer:

## Behavioral:

Tell me about a time when you had to rely on written communication to get your ideas across to your team.

## Question:

<https://leetcode.com/problems/asteroid-collision/>

We are given an array asteroids of integers representing asteroids in a row.

For each asteroid, the absolute value represents its size, and the sign represents its direction (positive meaning right, negative meaning left). Each asteroid moves at the same speed.

Find out the state of the asteroids after all collisions. If two asteroids meet, the smaller one will explode. If both are the same size, both will explode. Two asteroids moving in the same direction will never meet.

## Examples:

Example 1:

Input: asteroids = [5,10,-5]

Output: [5,10]

Explanation: The 10 and -5 collide resulting in 10. The 5 and 10 never collide.

Example 2:

Input: asteroids = [8,-8]

Output: []

Explanation: The 8 and -8 collide exploding each other.

Example 3:

Input: asteroids = [10,2,-5]

Output: [10]

Explanation: The 2 and -5 collide resulting in -5. The 10 and -5 collide resulting in 10.

Example 4:

Input: asteroids = [-2,-1,1,2]

Output: [-2,-1,1,2]

Explanation: The -2 and -1 are moving left, while the 1 and 2 are moving right. Asteroids moving the same direction never meet, so no asteroids will meet each other.

## Follow up Q&A:

Constraints:

* 2 <= asteroids.length <= 104
* -1000 <= asteroids[i] <= 1000
* asteroids[i] != 0

## Hint(s):

* Say a row of asteroids is stable. What happens when a new asteroid is added on the right?
* What data structures would probably be good for this?

## Solution(s): (General concept and time/space complexity)

### Name of Solution

Description

Time complexity: *O*(*N*), where

* N
* *N* is the number of asteroids. Our stack pushes and pops each asteroid at most once.

Space complexity: *O*(*N*). We use a stack to keep track of the intermediate results. In the worst case, the states do not evolve at the end, *i.e.* we need

O(N)

*O*(*N*) space where

N

*N* is the number of input asteroids.

#### Approach #1: Stack [Accepted]

Intuition

A row of asteroids is stable if no further collisions will occur. After adding a new asteroid to the right, some more collisions may happen before it becomes stable again, and all of those collisions (if they happen) must occur right to left. This is the perfect situation for using a *stack*.

Algorithm

Say we have our answer as a stack with the rightmost asteroid top, and a new asteroid comes in. If new is moving right (new > 0), or if top is moving left (top < 0), no collision occurs.

Otherwise, if abs(new) < abs(top), then the new asteroid will blow up; if abs(new) == abs(top) then both asteroids will blow up; and if abs(new) > abs(top), then the top asteroid will blow up (and possibly more asteroids will, so we should continue checking.)

class Solution {

public:

vector<int> asteroidCollision(vector<int>& a) {

vector<int> s; // use vector to simulate stack.

for (int i = 0; i < a.size(); i++) {

if (a[i] > 0 || s.empty() || s.back() < 0) // a[i] is positive star or a[i] is negative star and there is no positive on stack

s.push\_back(a[i]);

else if (s.back() <= -a[i]) { // a[i] is negative star and stack top is positive star

if(s.back() < -a[i]) i--; // only positive star on stack top get destroyed, stay on i to check more on stack.

s.pop\_back(); // destroy positive star on the frontier;

} // else : positive on stack bigger, negative star destroyed.

}

return s;

}

};

### Other questions follow up

*Ask if there is more than 5 minutes remaining when they finish their code and testing.*

* What are the time and space complexities of the algorithm?

# Interviewee:

## Question:

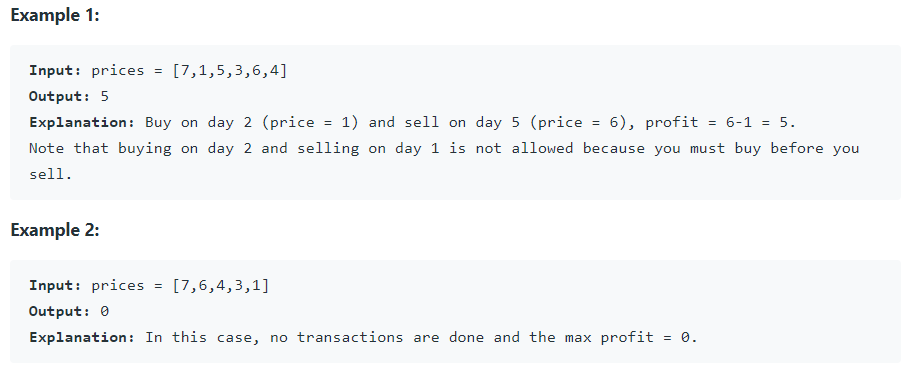
<https://leetcode.com/problems/best-time-to-buy-and-sell-stock/>

You are given an array prices where prices[i] is the price of a given stock on the ith day.

You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock.

Return *the maximum profit you can achieve from this transaction*. If you cannot achieve any profit, return 0.

## Examples:



## Code below or on leetcode