[**Asteroid Collision**](https://leetcode.com/problems/asteroid-collision/)

**import** java.util.Arrays;

**import** java.util.Stack;

**public** **class** AsteroidCollision {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int**[] asteroids = {5, 10, -5};

**int**[] result = *asteroidCollisionWithStack*(asteroids);

**for**(**int** i : result) {

System.***out***.print(i + " ");

}

System.***out***.println();

result = *asteroidCollisionWithoutStack*(asteroids);

**for**(**int** i : result) {

System.***out***.print(i + " ");

}

}

//Less Efficient solution as it using extra space for storing the state in stack

**public** **static** **int**[] asteroidCollisionWithStack(**int**[] asteroids) {

**if**(asteroids == **null** || asteroids.length == 0) {

**return** **new** **int**[] {};

}

Stack<Integer> stack = **new** Stack<>();

**for**(**int** asteroid : asteroids) {

**if**(asteroid > 0) {

stack.push(asteroid);

}

**else** {

**while**(!stack.isEmpty() && stack.peek() > 0 && stack.peek() < -asteroid) { //negative asteroid cancels positive asteroid

stack.pop();

}

**if**(stack.isEmpty() || stack.peek() < 0) { //the asteroids are moving left

stack.push(asteroid);

}

**else** **if**(stack.peek() == -asteroid) { //negative cancels positive asteroid

stack.pop();

}

}

}

**return** stack.stream().mapToInt(i -> i).toArray();

}

//More efficient as no extra space is required

**public** **static** **int**[] asteroidCollisionWithoutStack(**int**[] asteroids) {

**if**(asteroids == **null** || asteroids.length == 0) {

**return** **new** **int**[] {};

}

**int** index = 0;

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

**int** value = asteroids[i];

**if**(asteroids[i] > 0) {

asteroids[index++] = value;

}

**else** {

value = -value;

**while**(index >= 0) {

**if**(index == 0 || asteroids[index - 1] < 0) { //the asteroids are moving left

asteroids[index++] = -value;

**break**;

}

**else** **if**(asteroids[index - 1] < value) { //negative asteroid cancels positive asteroid

index--;

}

**else** **if**(asteroids[index - 1] > value) { //positive asteroid cancels negative asteroid

**break**;

}

**else** {

index--; //negative cancels positive asteroid

**break**;

}

}

}

}

**return** Arrays.*copyOfRange*(asteroids, 0, index);

}

}

Time complexity for stack solution: O(n). Number of elements in asteroids array

Space Complexity for stack solution: O(n). Number of elements in asteroids array

Time complexity for efficient solution: O(n). Number of elements in asteroids array

Space Complexity for efficient solution: O(1), constant space