Data Structures

Stack Homework #2

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Problem #1: 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.

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.
```

Problem #1: Asteroid Collision

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.
```

Problem #2: Insert at the bottom

- Add a new function to the stack: void insert_at_bottom(int x)
- It simply makes x the bottom element (last one)
- Assume stack is [4, 3, 2, 1] and x = 10 [1 is top, 4 is last]
- Now is: [10, 4, 3, 2, 1]
- What is the time & memory order?
- Constraints
  - Implement it **recursively**
  - Don’t create new stack or new arrays

Problem #3: Reverse Stack

- Develop member function inside stack class: void **reverse**()
- The function reverses the stack content
- What is the time & memory order?
- Constraints
  - Implement it recursively
  - Don’t create new stack or new arrays
  - Tip: use the Insert at the bottom function
- Stack {1, 2, 3, 4} ⇒ {4, 3, 2, 1}

Problem #4: Manual System Stack

- **Background**: We know a recursive function will have its calls stored in a stack. Sometimes, we can’t use the default system stack as function is so deep (e.g. facing stack overflow).
- In this problem, we would like to simulate this OS process of a stack of function calls. We will apply that on factorial function
  - That is, with every supposed recursive call, we add an entry to the stack
  - Later when its result is computed, we give it back to the caller
- See the following slide as a tip if you couldn’t imagine how to start!

Problem #4

- This is part of my code
- You don’t have to follow

```
struct StackElement {
    int n;
    int result { -1 };

    StackElement(int n = 1, int result = -1) :
            n(n), result(result) {
    }
};

int factorial_stack(int n) {
    if (n <= 1)
        return 1;
    Stack st(n);
    st.push(StackElement(n));
    StackElement cur(1);

    while (!st.isEmpty()) {
        // TODO
    }
    return cur.result;
}
```

Problem #5: Score of Parentheses

Given a balanced parentheses string `S`, compute the score of the string based on the following rule:

- `()` has score 1
- `AB` has score `A + B`, where A and B are balanced parentheses strings.
- `(A)` has score `2 * A`, where A is a balanced parentheses string.

- Use stack
- Inputs
  - () ⇒ 1
  - (()) ⇒ 2
  - ()() ⇒ 2
  - (()()) ⇒ 4
  - (() (())) ⇒ 6
  - ()((()) ()) ⇒ 7

Problem #6: Next Greater Element

- Given an array, for every number, find the first number after it of a higher value
- E.g. 5 10 5 7 15 11 ⇒ 15 7 15 -1 -1
  - First number greater than 10 ⇒ 15
  - First number greater than 5 ⇒ 7
  - First number greater than 7 ⇒ 15
  - First number greater than 15 and 11 ⇒ None, so use -1
- Input:
  - 8 73 74 75 71 69 72 76 73 ⇒ 74 75 76 72 72 76 -1 -1
- Find O(n) solution

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."