COL106 - Data Structures and Algorithms

Stack ADT

- Stores Objects
- Insertions & deletions are strictly "Last in - First out"

Two operations on stacks

push(x) : inserts object x

pop() : removes the last inserted object from the stack and returns it.

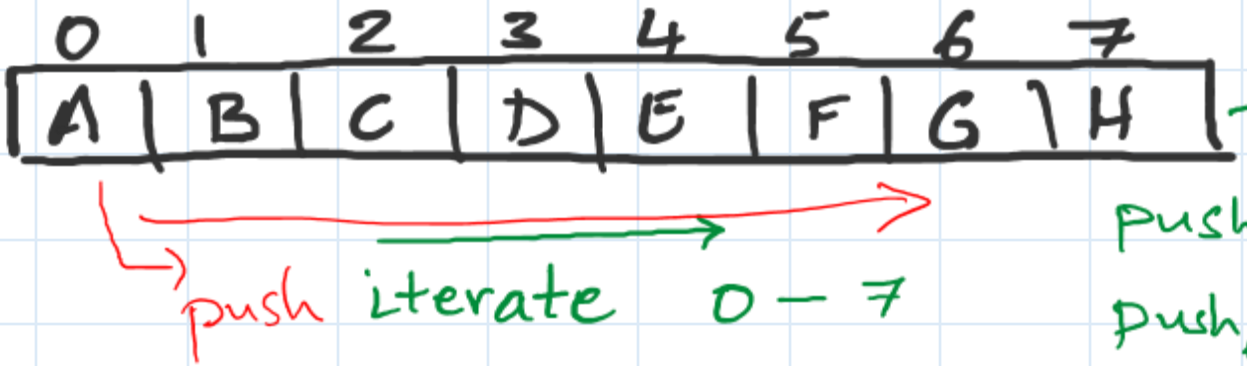
Additionally,

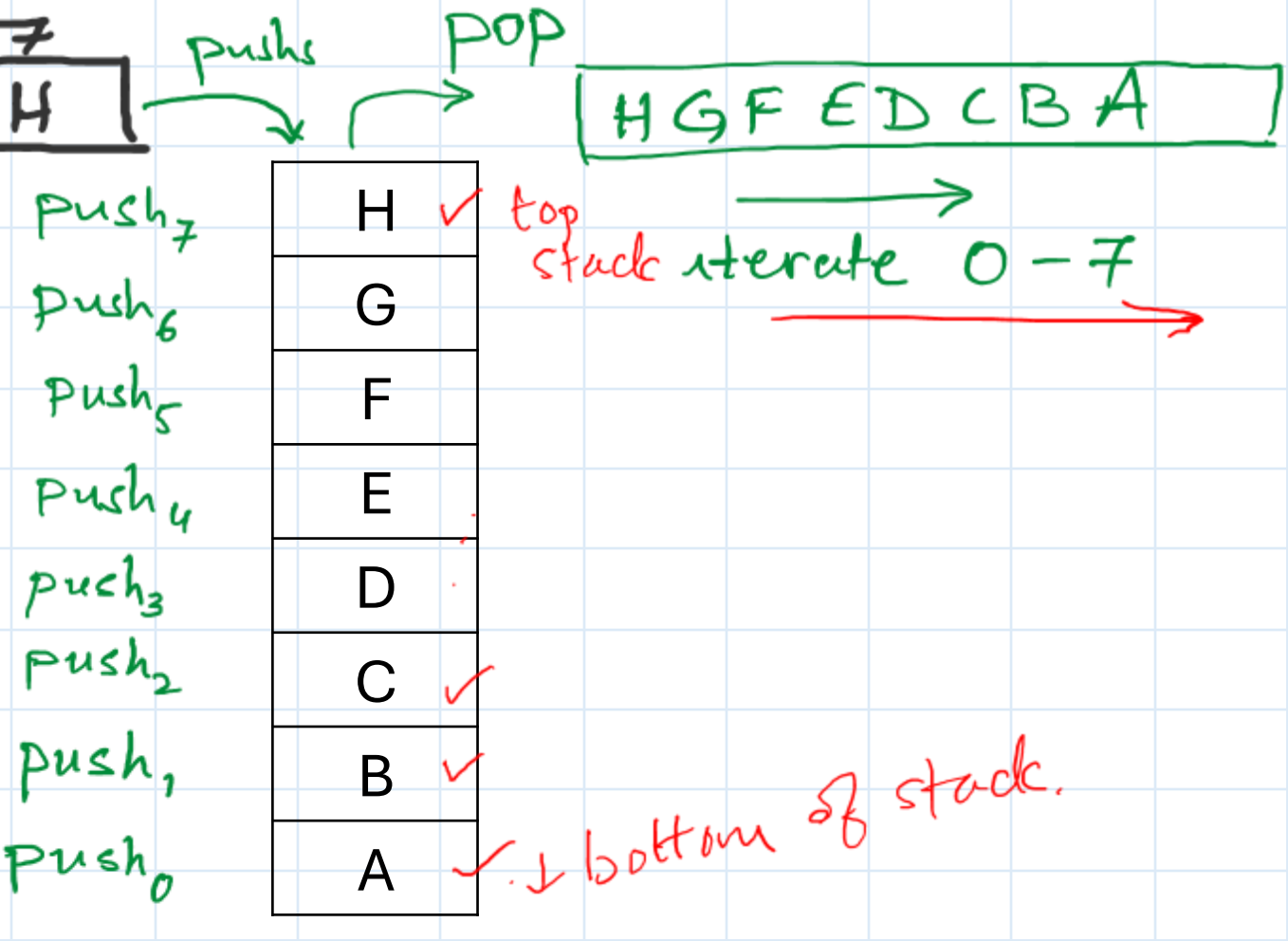
top() : returns the last inserted object

size() :

Applications of Stacks

Reversing an array.

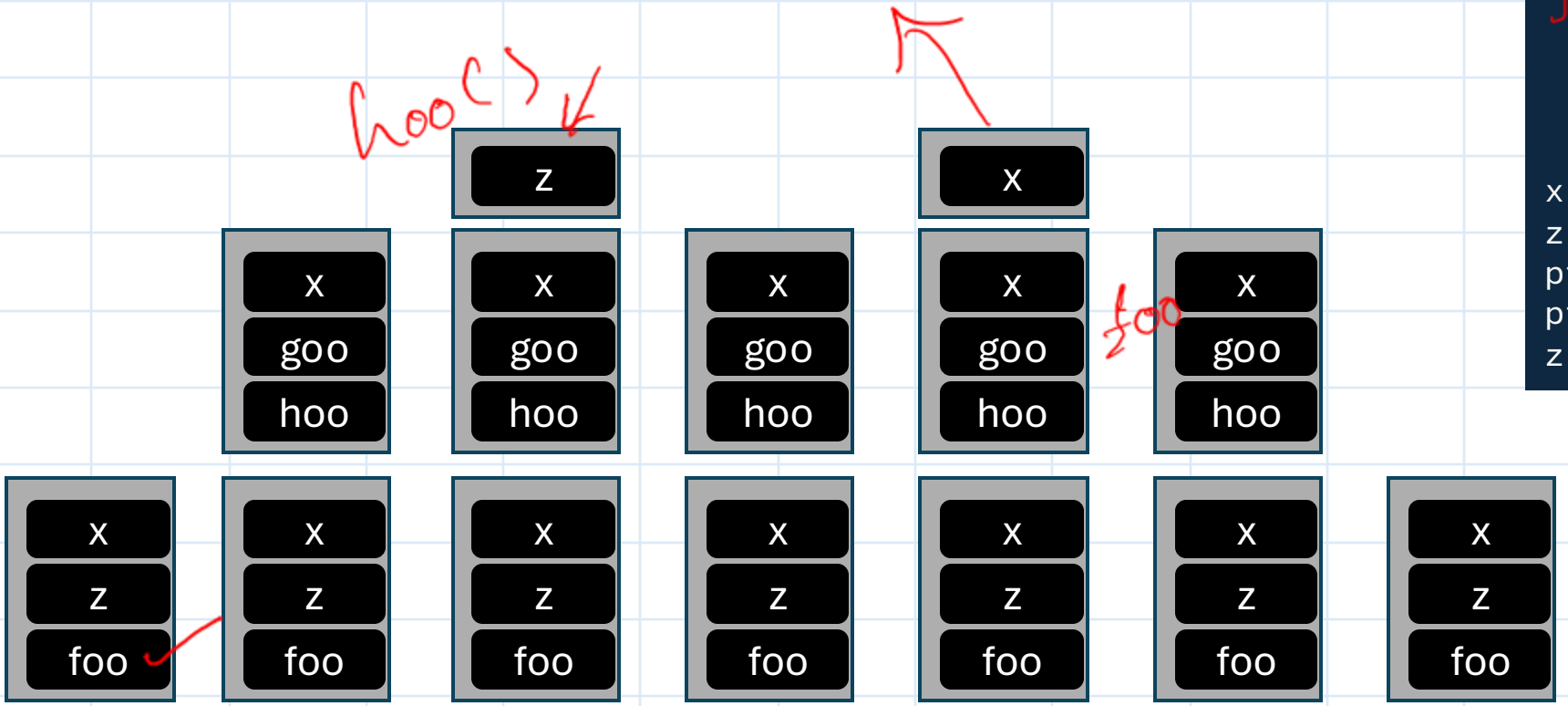




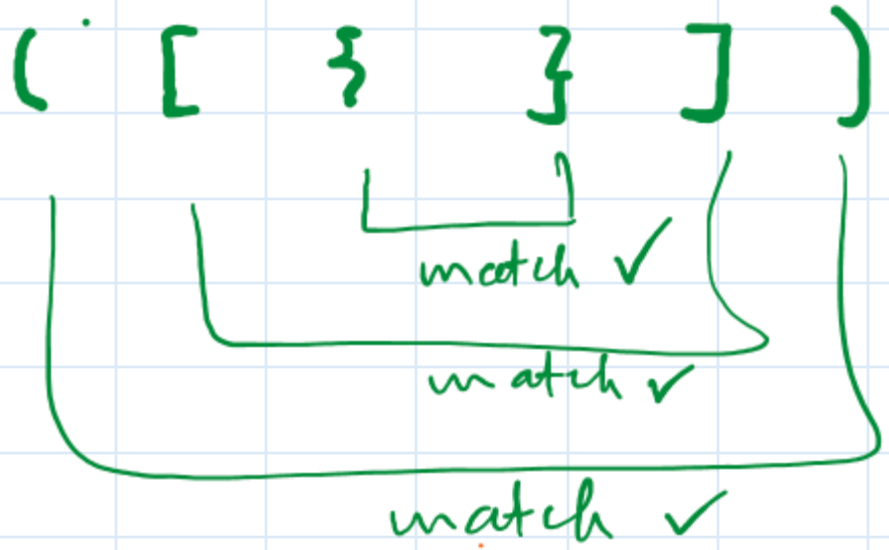
Stacks for function calls and return values

```
def foo (x):
    def goo ():
        x = "abc"
        print (”foo-goo x = ", x)
    def hoo ():
        z = x
        print (”foo-hoo z = ", z)
    x = x + 1
    print (”foo x = ", x)
    hoo ()
    goo ()
    print (”foo2 x = ", x)
    return goo


x = 3
z = foo(x)
print ("x = ", x)
print ("z = ", z)
z()
```



Parantheses matching



WHICH OF THE FOLLOWING IS VALID ?

1. ( ) (( )) {([( )])} ✓ ✓
2. (((( ) (( )) {([( )])} X
3. ({[ ])} X

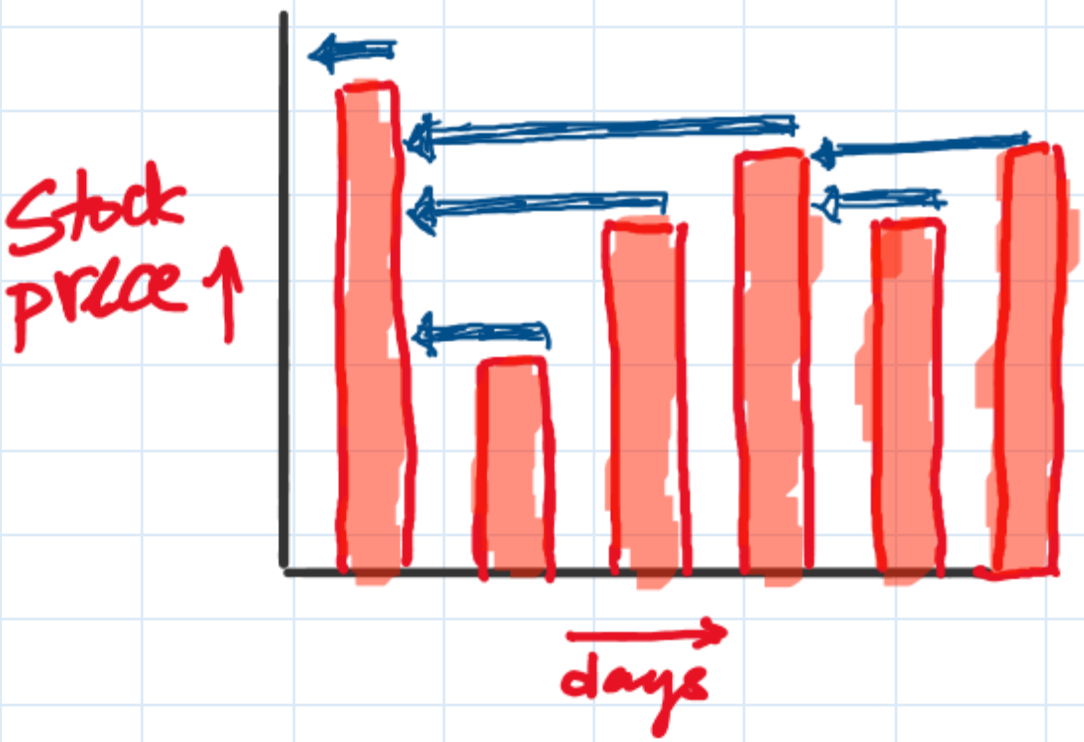
```
public static boolean matching (String expression) {
    final String opening = "({[";
    final String closing = ")}]";
    Stack <Character> buffer = new LinkedStack<>();

    for (char c: expression.toCharArray()) {
        if (opening.indexOf(c) != -1)
            buffer.push(c);
        else if (closing.indexOf(c) != -1) {
            if (buffer.isEmpty()) return false;
            if (closing.indexOf(c) !=
opening.indexOf(buffer.pop())) return false;
        }
    }
    return buffer.isEmpty();
}
```

0, 1, 2

invalid expression

Computing the Span of Each Day.



For each day i, its span S[i] is the maximum number of consecutive elements X[j] immediately preceding X[i], such that X[j] ≤ X[i]

We are on i-th day

→ Pop elements from the stack if top() ≤ stockPrice[i].

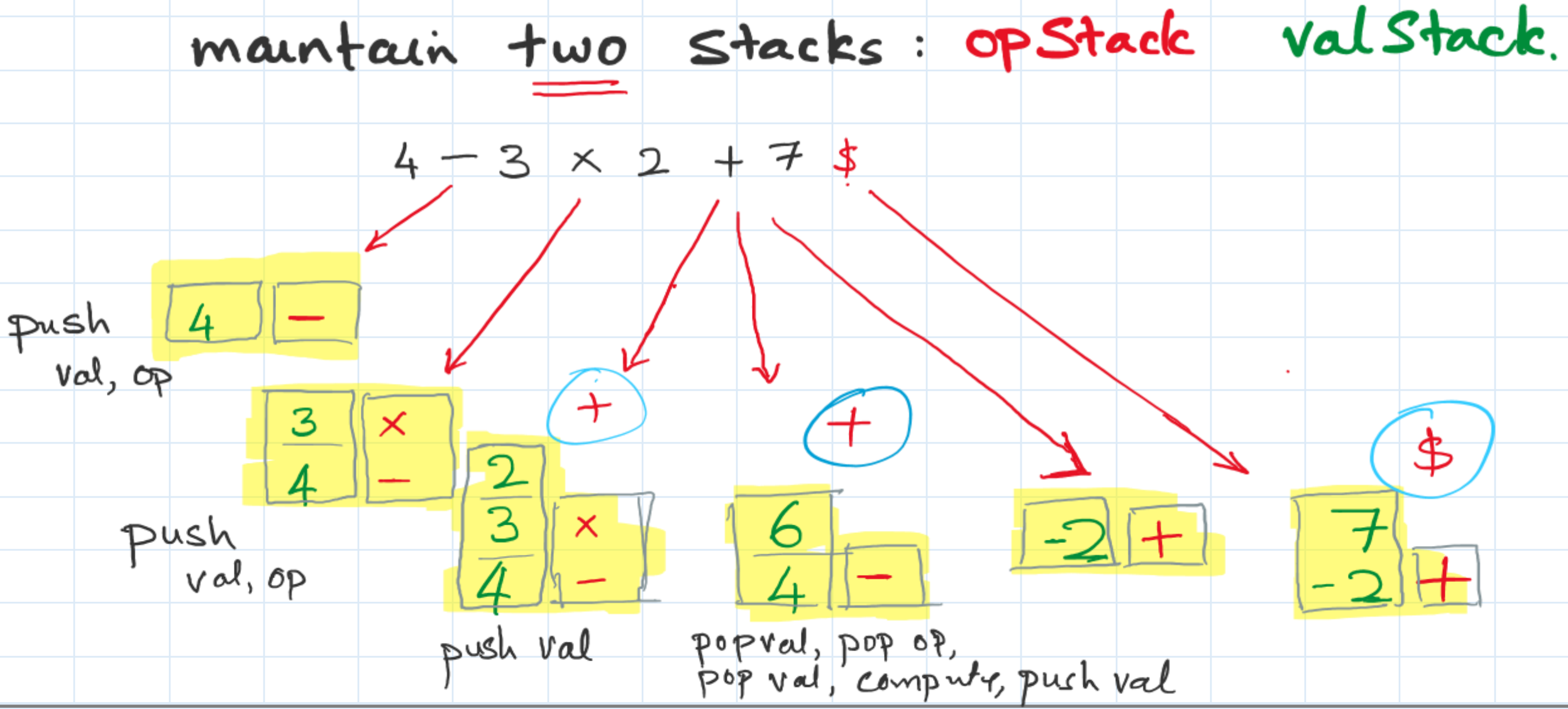
→ Span[i] = # elements popped.

→ push(stockPrice[i])

```
Stack_Span (X, n) {
  S ← empty array of n elements                          1
  A ← empty stack                                        1
  for i ← 0 to n-1 {                                     n
    while (!A.isEmpty() && X[A.top()] ≤ X[i]) {          n
       A.pop()                                           n
    }
    if (A.isEmpty()) then                                n
       S[i] ← S[i]+1                                     n
    else
       S[i] ← i - A.top()                                n
    A.push(i)                                            n
  }
  return S;                                              1
}
```

From

Expression Evaluation



Other Applications

- Browser history

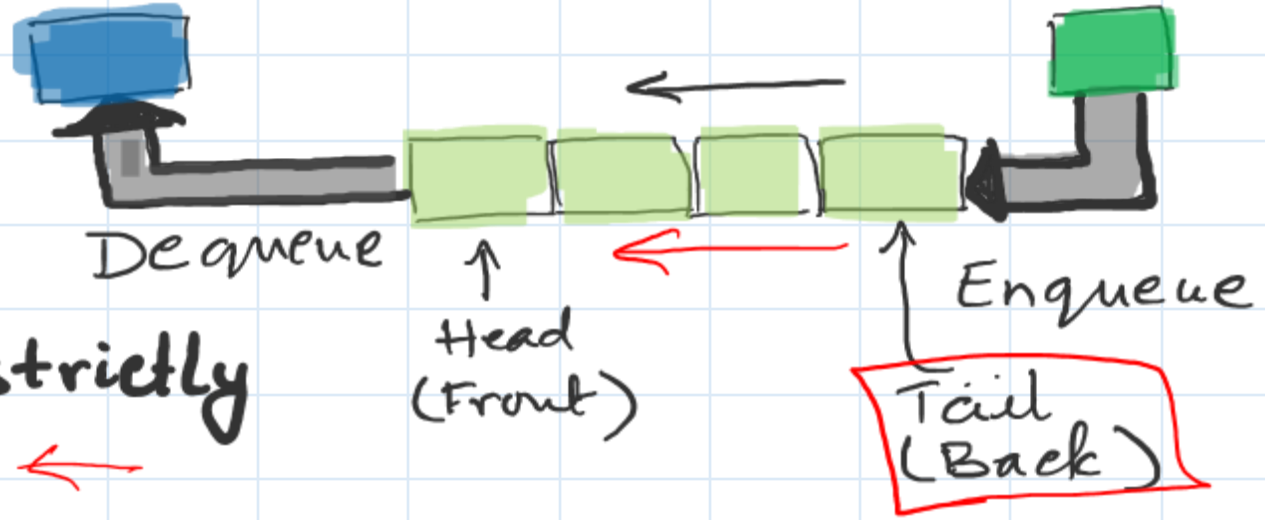
  separate stack for each tab

- HTML parsing

- Convex hull (Graham's Scan)

https://en.wikipedia.org/wiki/Graham_scan

Queue ADT



- Stores Objects ✓
- Insertions & deletions are strictly "First in - First out" ←

Two operations on Queues

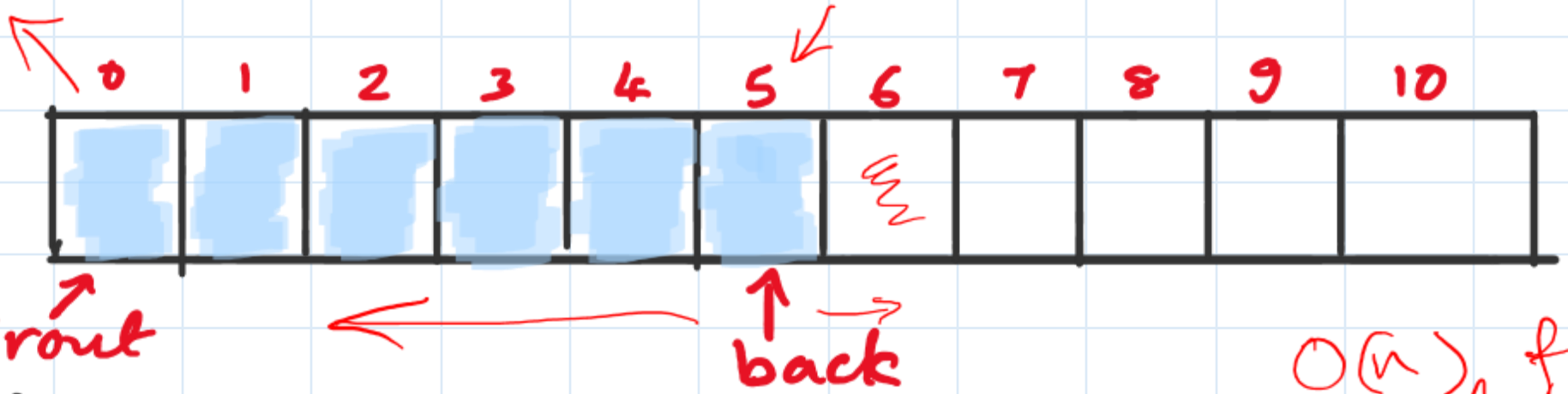
enqueue(x) } : inserts object x at the tail.
OR add(x) }

O(1)

dequeue() } : removes the objects from the head
OR remove() } of the queue and returns it. O(1)

peek() } : returns the object at the front
OR first() } of the queue without removing it. O(1)

Array based Implementation of a Queue.



O(n) for dequeue

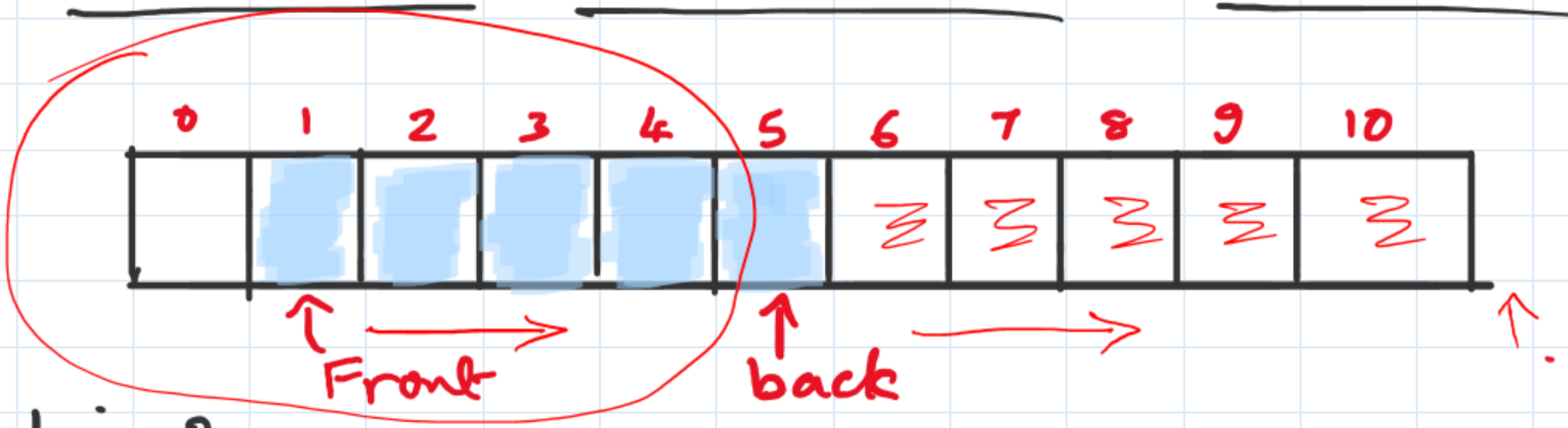
Option 1

store first element in index 0
second —"— —— "— 1
⋮

add(x): increment back index, add x

remove(): remove the element at front
and shift all elements right

Array based Implementation of a Queue.



Option 2

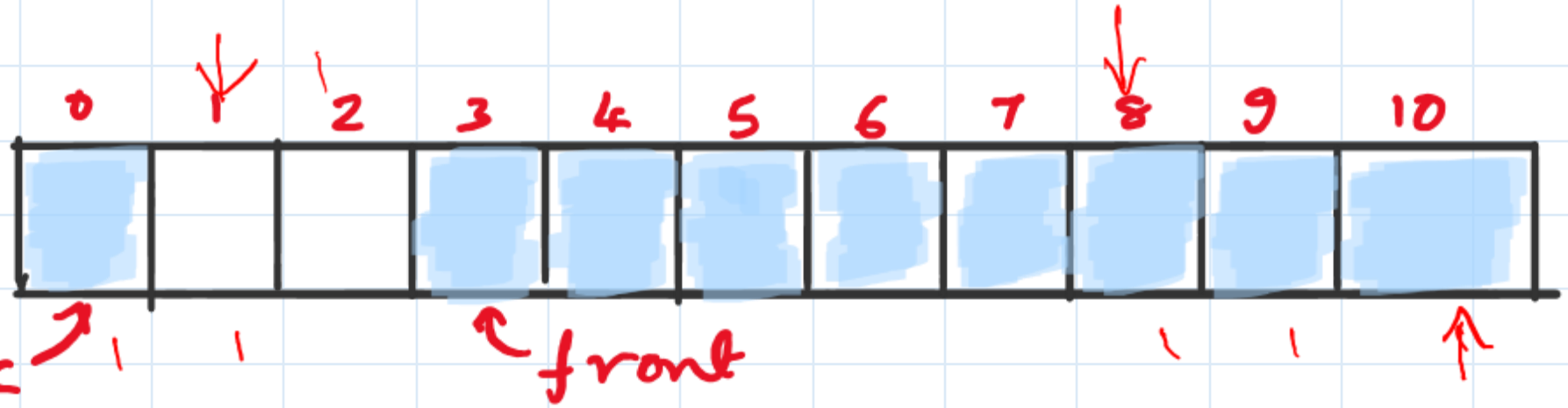
store first element in index 0
second —"— —— "— 1

add(x): increment back index, add x

remove(): remove the element at front.

increment the front index

Array based Implementation of a Queue.



Option 3 Treat array as being "circular"
Implementable using modulo arithmetic.

add(x): insert x at (front + size) % N (size of the queue) O(1)

remove(): remove the element at front.

front = (front + 1) % N O(1)

Linked List Implementation of Queue

- Often we need unbounded queue
- Linked List implementation is similar in approach to the implementation of stack.

- In general, linked list implementations are more expensive in implementation

instantiation of new objects are expensive

Double Ended Queue ADT (deque)

Combines the features of stacks and Queue

addLast(x)

addFirst(x)

first()

removeFirst()

removeLast()

last()

size()

isEmpty()

Implementation is left as an exercise