# CS 240 Data Structures and Algorithms I

Alex Vondrak

ajvondrak@csupomona.edu

October 26, 2011

- For the sake of discussion, we've been implementing stacks by hand
- Java has many data structures included with its libraries
- Because it's so extensive, it's important to be familiar with the Java libraries' APIs
- http://download.oracle.com/javase/7/docs/api/java/util/ Stack.html

- Q: How efficient are the stack operations?
- A: Depends on the implementation!
- Let's take a look at our ArrayStack:

Operation	Worst-Case Running Time
size	
isEmpty	
top	
рор	
push	

- Q: How efficient are the stack operations?
- A: Depends on the implementation!
- Let's take a look at our ArrayStack:

Operation	Worst-Case Running Time
size	O(1)
isEmpty	
top	
pop	
push	

- Q: How efficient are the stack operations?
- A: Depends on the implementation!
- Let's take a look at our ArrayStack:

Operation	Worst-Case Running Time
size	O(1)
isEmpty	O(1)
top	
pop	
push	

- Q: How efficient are the stack operations?
- A: Depends on the implementation!
- Let's take a look at our ArrayStack:

Operation	Worst-Case Running Time
size	O(1)
isEmpty	O(1)
top	<i>O</i> (1)
pop	
push	

- Q: How efficient are the stack operations?
- A: Depends on the implementation!
- Let's take a look at our ArrayStack:

Operation	Worst-Case Running Time
size	O(1)
isEmpty	O(1)
top	O(1)
рор	O(1)
push	

- Q: How efficient are the stack operations?
- A: Depends on the implementation!
- Let's take a look at our ArrayStack:

Operation	Worst-Case Running Time
size	O(1)
isEmpty	O(1)
top	O(1)
рор	O(1)
push	<i>O</i> ( <i>n</i> )

#### Amortized Analysis

- At worst, push takes O(n) time to resize the array...
- ... But how often does that happen?
- Idea: average out the cost of *n* operations performed in sequence

```
public void push(E value) {
    if (size() == data.length)
        grow();
    data[++top] = value;
}
```

Q: What if grow() increases the size of the array by 1?Q: What if grow() doubles the size of the array?

Suppose, for simplicity:

- The internal array starts with a capacity of 1
- The cost of writing/copying an array element is 1 operation
- Q: What is the average cost of a sequence of *n* pushes? A:

 $\underbrace{1}_{\text{store}}$  +

Suppose, for simplicity:

- The internal array starts with a capacity of 1
- The cost of writing/copying an array element is 1 operation
- Q: What is the average cost of a sequence of n pushes?

A:

store grow & store

Suppose, for simplicity:

- The internal array starts with a capacity of 1
- The cost of writing/copying an array element is 1 operation

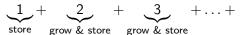
Q: What is the average cost of a sequence of *n* pushes? A:



Suppose, for simplicity:

- The internal array starts with a capacity of 1
- The cost of writing/copying an array element is 1 operation

Q: What is the average cost of a sequence of *n* pushes? A:



Suppose, for simplicity:

- The internal array starts with a capacity of 1
- The cost of writing/copying an array element is 1 operation

Q: What is the average cost of a sequence of *n* pushes? A:



Suppose, for simplicity:

- The internal array starts with a capacity of 1
- The cost of writing/copying an array element is 1 operation

Q: What is the average cost of a sequence of *n* pushes? A:



Averaged out, that's O(n)

Suppose, for simplicity:

- The internal array starts with a capacity of 1
- The cost of writing/copying an array element is 1 operation
- Q: What is the average cost of a sequence of n pushes?
- A: The growths will happen less frequently, but will cost

first growth

Suppose, for simplicity:

- The internal array starts with a capacity of 1
- The cost of writing/copying an array element is 1 operation
- Q: What is the average cost of a sequence of *n* pushes?
- A: The growths will happen less frequently, but will cost



first growth second growth

Suppose, for simplicity:

- The internal array starts with a capacity of 1
- The cost of writing/copying an array element is 1 operation
- Q: What is the average cost of a sequence of *n* pushes?
- A: The growths will happen less frequently, but will cost



first growth second growth third growth

Suppose, for simplicity:

- The internal array starts with a capacity of 1
- The cost of writing/copying an array element is 1 operation
- Q: What is the average cost of a sequence of *n* pushes?
- A: The growths will happen less frequently, but will cost



first growth second growth third growth fourth growth

Suppose, for simplicity:

- The internal array starts with a capacity of 1
- The cost of writing/copying an array element is 1 operation
- Q: What is the average cost of a sequence of *n* pushes?
- A: The growths will happen less frequently, but will cost



first growth second growth third growth fourth growth

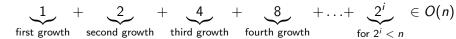
Suppose, for simplicity:

- The internal array starts with a capacity of 1
- The cost of writing/copying an array element is 1 operation
- Q: What is the average cost of a sequence of *n* pushes?
- A: The growths will happen less frequently, but will cost



Suppose, for simplicity:

- The internal array starts with a capacity of 1
- The cost of writing/copying an array element is 1 operation
- Q: What is the average cost of a sequence of n pushes?
- A: The growths will happen less frequently, but will cost



Adding that to the cost of storing (1 for each push), that's still O(n). Averaged out over *n* operations, that's O(1).