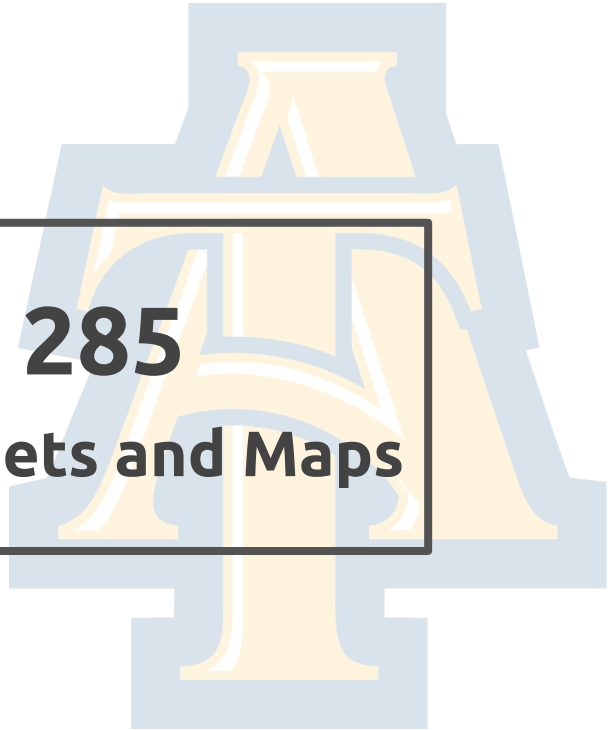
COMP - 285

Advanced Analysis of Algorithms

Welcome to COMP 285

Lecture 5: Stacks, Queues, Sets and Maps



Chris Lucas (cflucas@ncat.edu)

HW1 was due!

Today @ 1:59pm!

HW 2 released by EoD!

Quiz!

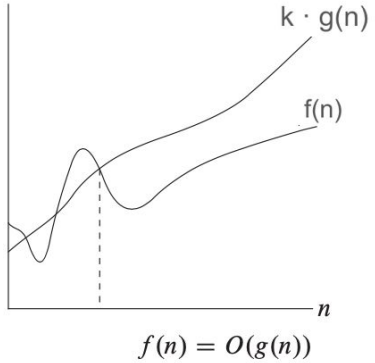
But first ...

Recall where we ended last lecture...

Recap

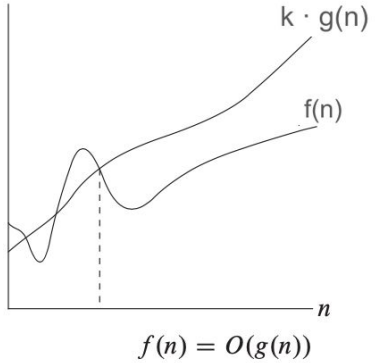
Recap

Big-Oh: Upper-bound | f = O(g) is similar to f ≤ g
“f grows no faster than g”

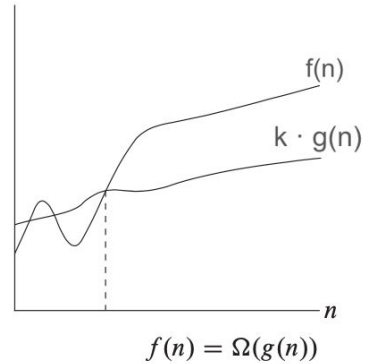


Recap

Big-Oh: Upper-bound | f = O(g) is similar to f ≤ g
“f grows no faster than g”



Big-Omega: Lower-bound | f = Ω(g) is similar to f ≥ g
“f grows no slower than g”

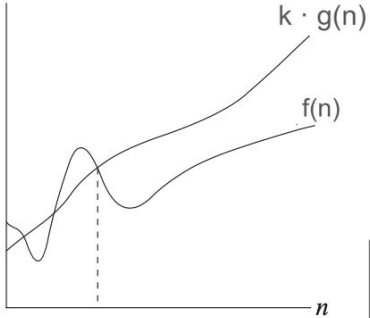


Recap

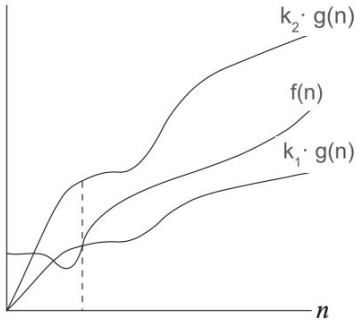
Big-Oh: Upper-bound | f = O(g) is similar to f ≤ g
“f grows no faster than g”

Big-Theta: Tight-bound | f = Θ(g) is similar to f = g
“f grows as fast as g”

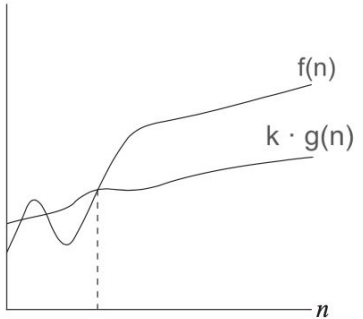
Big-Omega: Lower-bound | f = Ω(g) is similar to f ≥ g
“f grows no slower than g”



$f(n) = O(g(n))$



$f(n) = \Theta(g(n))$



$f(n) = \Omega(g(n))$

Quiz!

www.comp285-fall22.ml

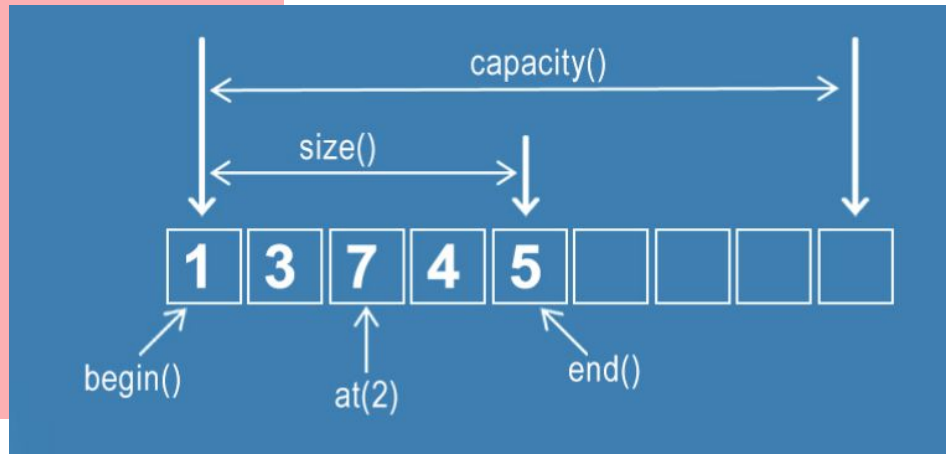
Big Questions!

- What are conventional data structures again? How fast are they?
- Which data structures use hashing? How fast are they?
- What is hashing?

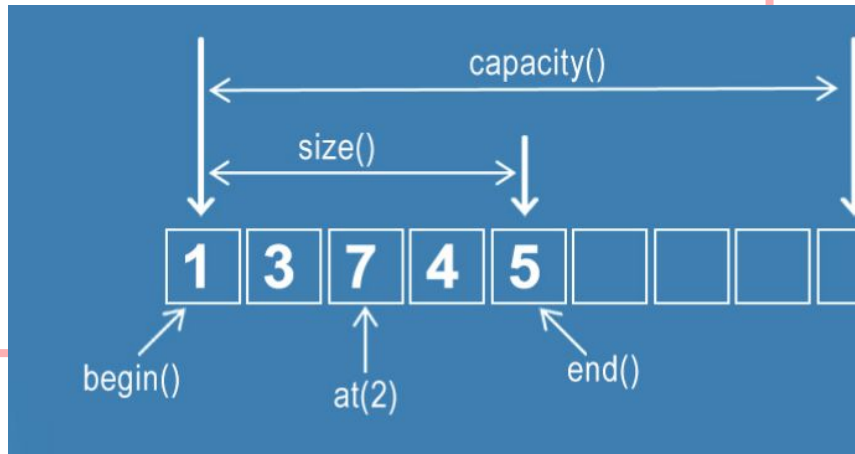
Big Questions!

- What are conventional data structures again? How fast are they?
- Which data structures use hashing? How fast are they?
- What is hashing?

Vectors/Arrays

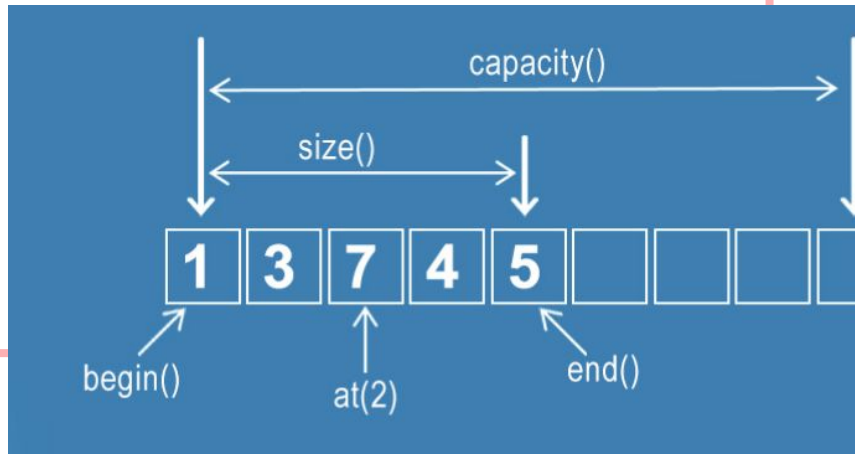


Array Operations Chart



Array Operations Chart

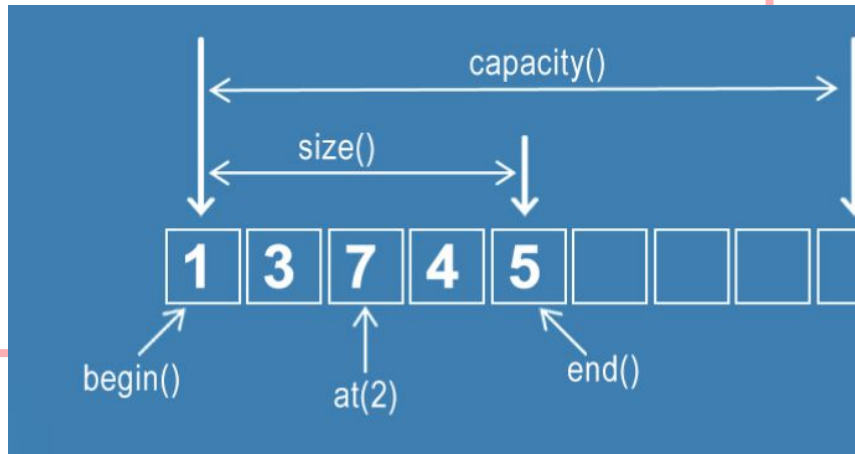
Data Structure	Access	Search	Insertion	Deletion
Array				



Array Operations Chart

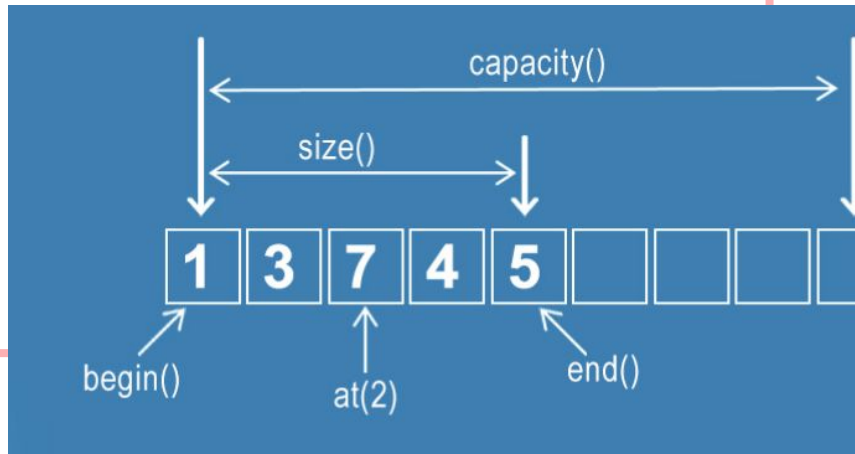
Data Structure	Access	Search	Insertion	Deletion
Array				

```
int thirdItem = arr[2];
```



Array Operations Chart

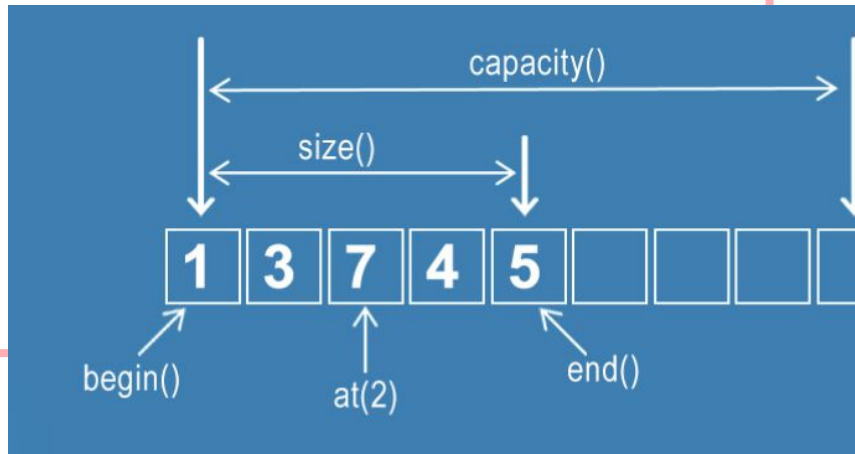
Data Structure	Access	Search	Insertion	Deletion
Array	O(1)			



Array Operations Chart

Data Structure	Access	Search	Insertion	Deletion
Array	O(1)			

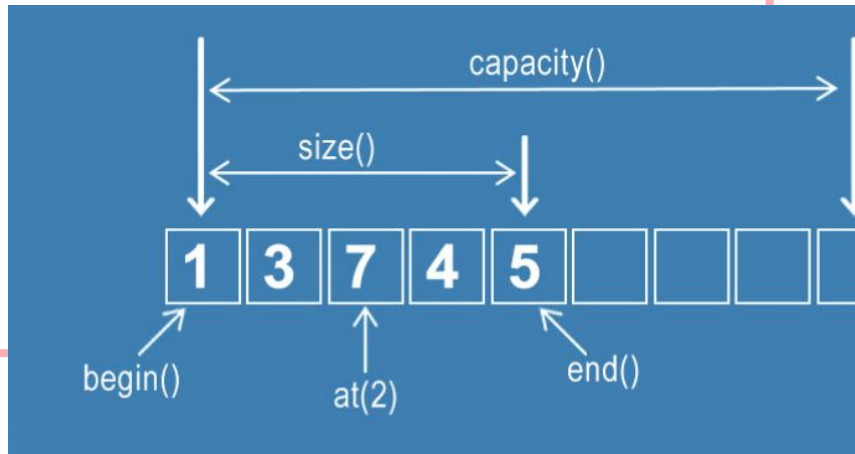
```
for(int i = 0; i < arr.size(); i++) {
    If (arr[i] == target_value) {
        …
    }
}
```



Array Operations Chart

Data Structure	Access	Search	Insertion	Deletion
Array	O(1)	O(n)		

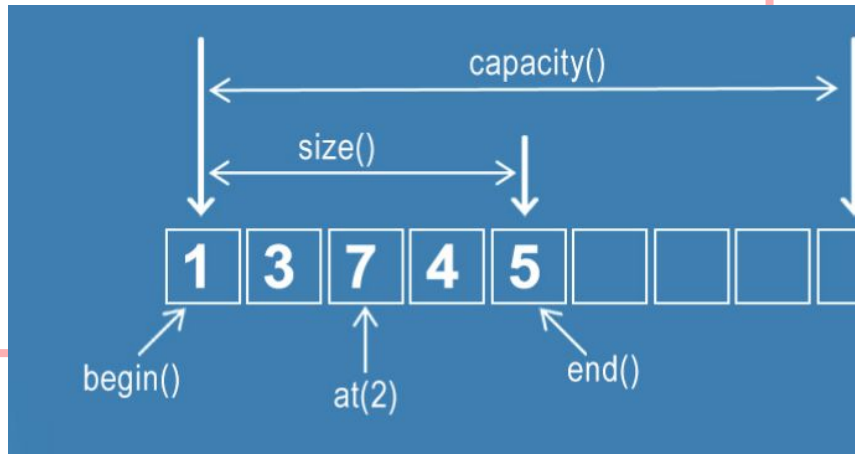
arr.insert(...),
arr.erase(...)



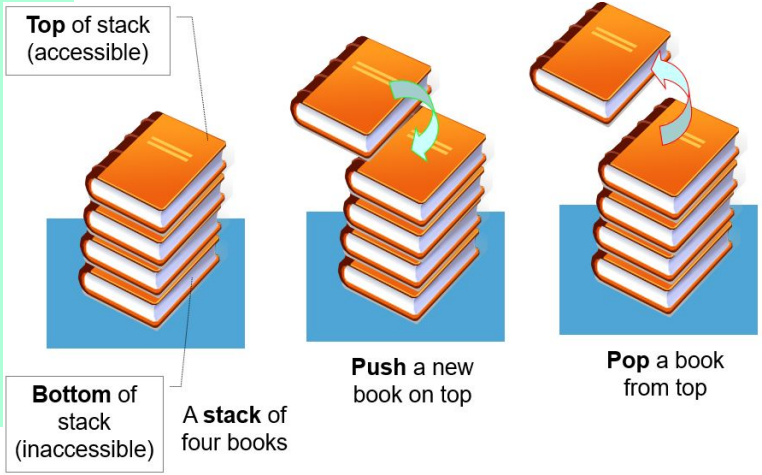
Array Operations Chart

Data Structure	Access	Search	Insertion	Deletion
Array	O(1)	O(n)	O(n)	O(n)

arr.insert(...),
arr.erase(...)

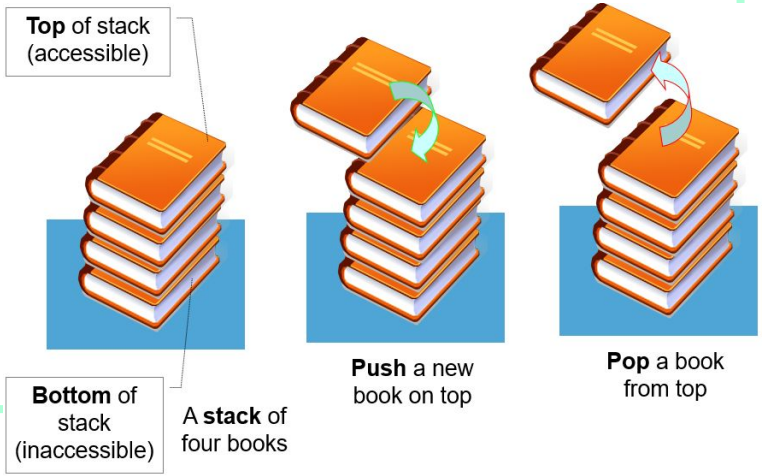


Stacks


Top of stack (accessible)
Bottom of stack (inaccessible)
A **stack** of four books
Push a new book on top
Pop a book from top

Stack Operations Chart

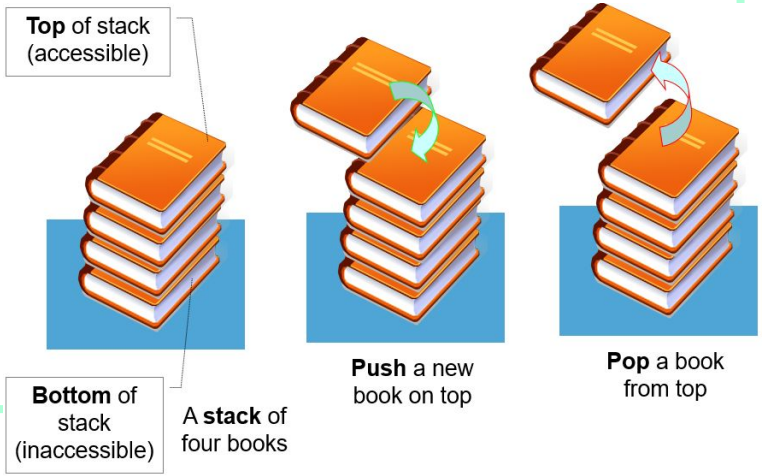
Data Structure	Access	Search	Insertion	Deletion
Stack				



Stack Operations Chart

Data Structure	Access	Search	Insertion	Deletion
Stack				

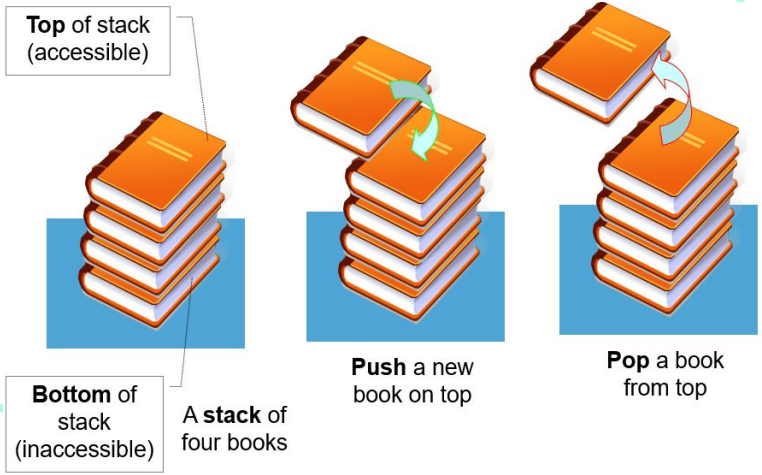
```
while (i < indexToAccess) {
    stack.pop();
    i--;
}
```



Stack Operations Chart

Data Structure	Access	Search	Insertion	Deletion
Stack	O(n)			

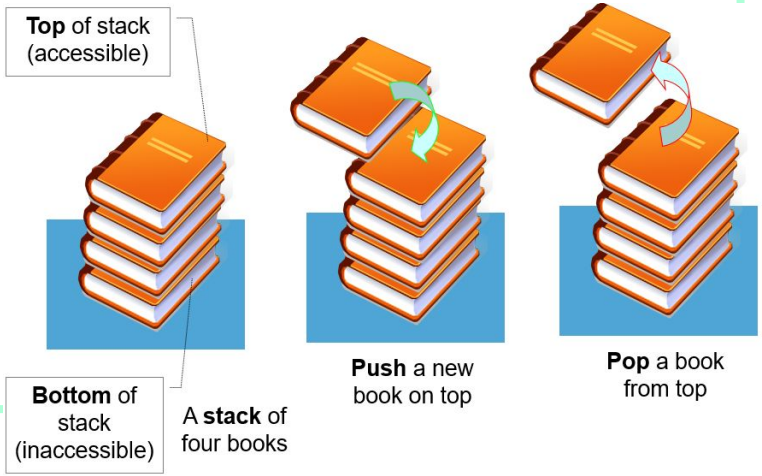
```
while (i < indexToAccess) {
    stack.pop();
    i--;
}
```



Stack Operations Chart

Data Structure	Access	Search	Insertion	Deletion
Stack	O(n)			

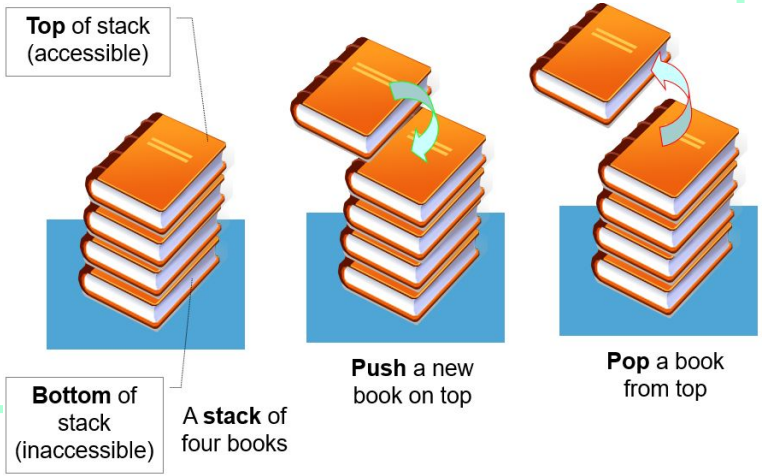
```
while (value != targetValue) {
    value = stack.pop();
}
```



Stack Operations Chart

Data Structure	Access	Search	Insertion	Deletion
Stack	O(n)	O(n)		

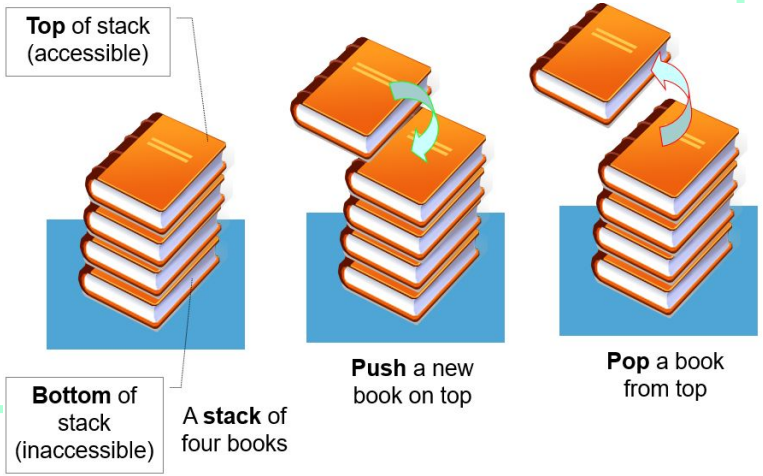
```
while (value != targetValue) {
    value = stack.pop();
}
```



Stack Operations Chart

Data Structure	Access	Search	Insertion	Deletion
Stack	O(n)	O(n)		

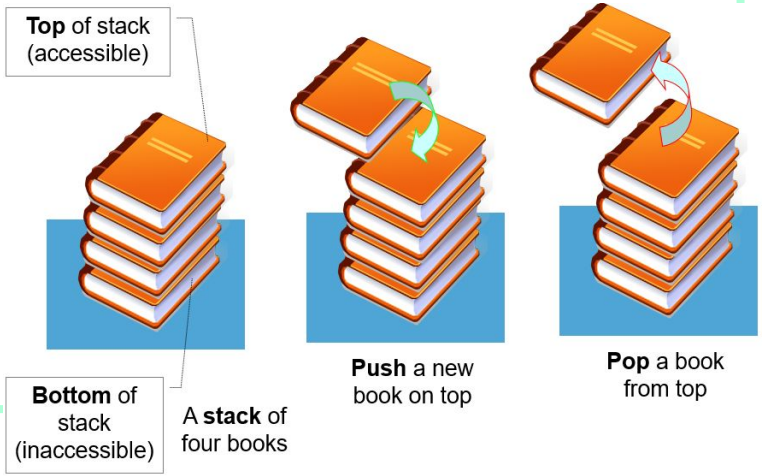
stack.push(),
stack.pop()



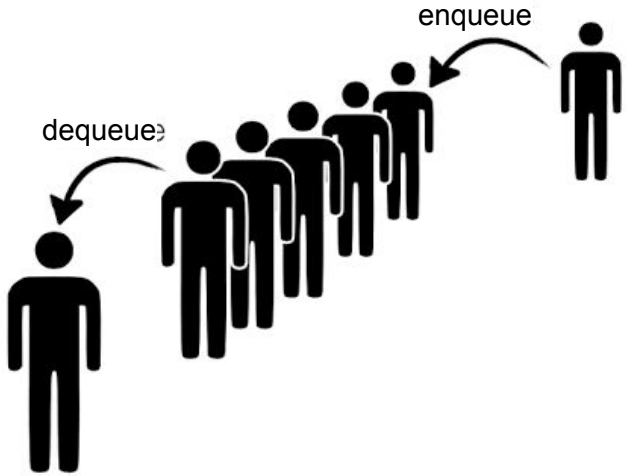
Stack Operations Chart

Data Structure	Access	Search	Insertion	Deletion
Stack	O(n)	O(n)	O(1)	O(1)

stack.push(),
stack.pop()

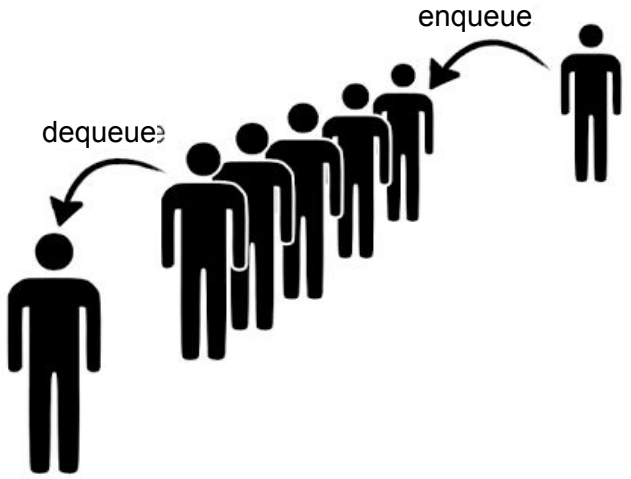


Queues



Queue Operations Chart

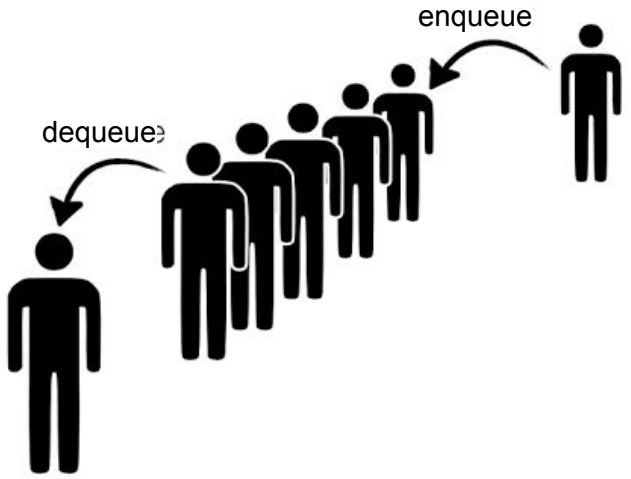
Data Structure	Access	Search	Insertion	Deletion
Queue				



Queue Operations Chart

Data Structure	Access	Search	Insertion	Deletion
Queue				

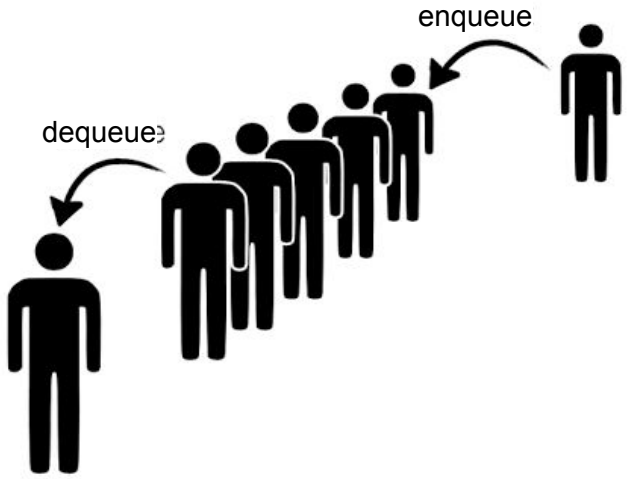
```
while (i < indexToAccess) {
    queue.dequeue();
    i--;
}
```



Queue Operations Chart

Data Structure	Access	Search	Insertion	Deletion
Queue	O(n)			

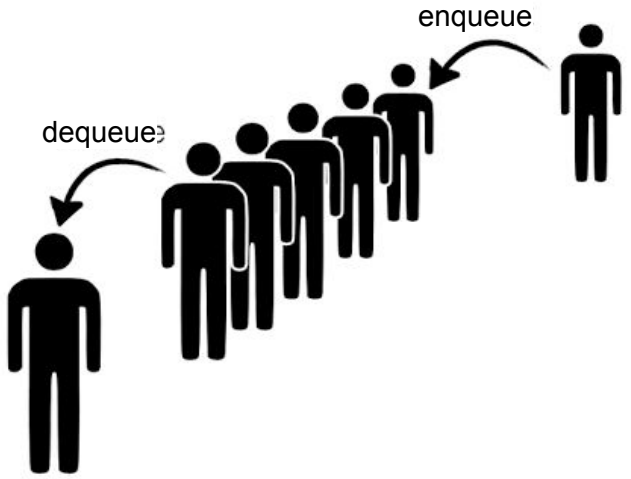
```
while (i < indexToAccess) {
    queue.dequeue();
    i--;
}
```



Queue Operations Chart

Data Structure	Access	Search	Insertion	Deletion
Queue	O(n)			

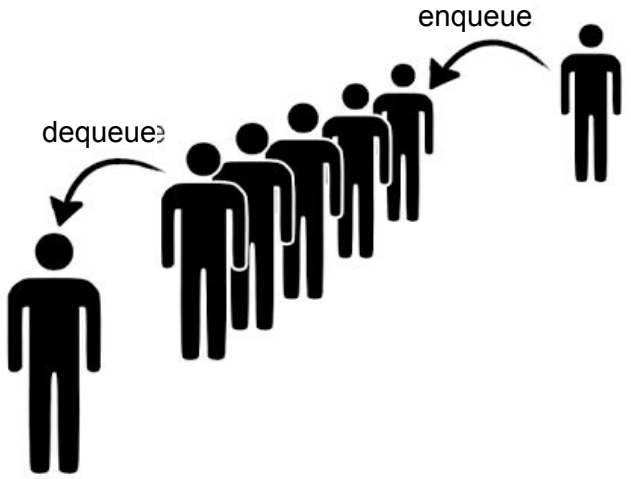
```
while (value != targetValue) {
   value = queue.dequeue();
}
```



Queue Operations Chart

Data Structure	Access	Search	Insertion	Deletion
Queue	O(n)	O(n)		

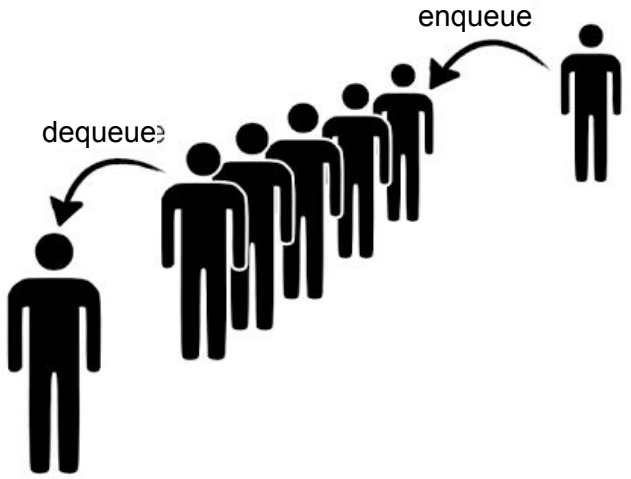
```
while (value != targetValue) {
   value = stack.pop();
}
```



Queue Operations Chart

Data Structure	Access	Search	Insertion	Deletion
Queue	O(n)	O(n)		

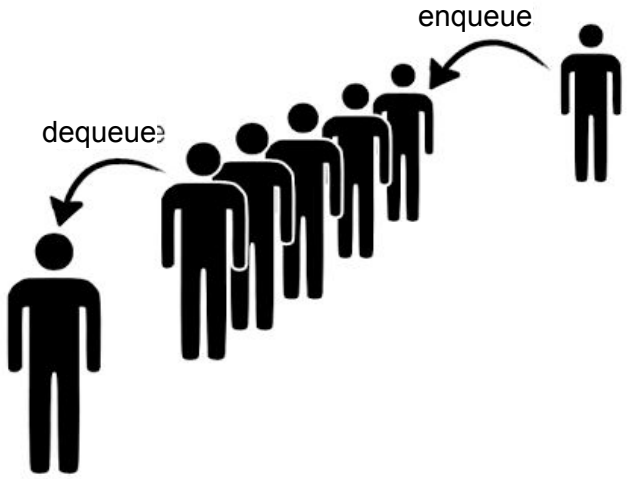
queue.enqueue(),
queue.dequeue()



Queue Operations Chart

Data Structure	Access	Search	Insertion	Deletion
Queue	O(n)	O(n)	O(1)	O(1)

queue.enqueue(),
queue.dequeue()



Concrete Examples

› to enhance my programming abilities. In this course I hope to be able to understand and solve more complex algorithms

e I want to better my interviewing and coding skills.

s I want to be able to actually code and like it.

Concrete Examples

n building efficient algorithms to solve coding interview questions.

enhance my problem solving skills, c

, I want to enhance my coding skills, specifically in C++ to help me earn an internship and a job as a software engineer post grad.

Balance Parentheses

Given a string containing just char ‘(‘ and char ‘)’, return whether or not the parentheses are valid.

Examples:

() -> true

)(-> false

()(()()) -> true

(() -> false

Balance Parentheses

Given a string containing just char ‘(‘ and char ‘)’, return whether or not the parentheses are valid.

Examples:

() -> true

)(-> false

()(()()) -> true

(() -> false

Let’s code it!!!

Kahoot!

www.kahoot.it, Code: 520608

Enter your @aggies.ncat email

Balance Parentheses V2

Given a string containing chars ‘(‘, ‘{‘ , ‘[‘, ‘)’, ‘}’, ’]’ return whether or not the parentheses are valid.

Examples:

([]{}]) -> true

}{(-> false

([{()}])(([])({})) -> true

(({}) -> false

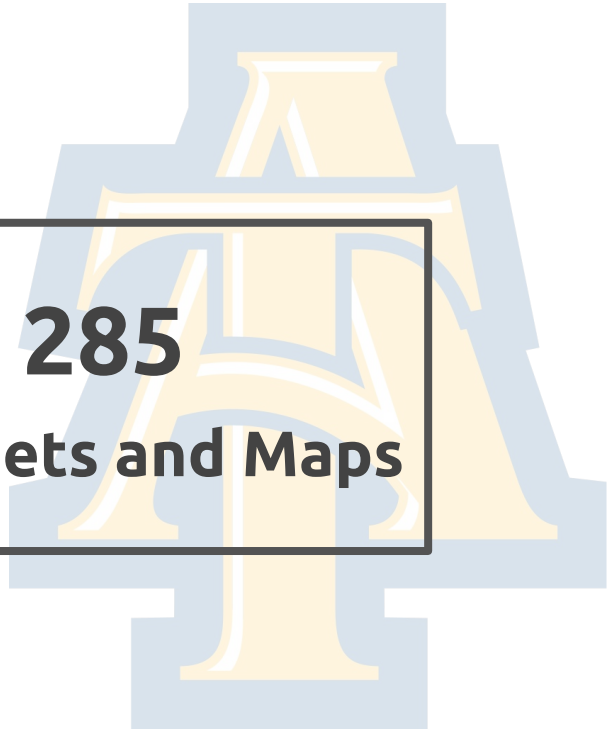
Let’s code it!!!

COMP - 285

Advanced Analysis of Algorithms

Welcome to COMP 285

Lecture 5: Stacks, Queues, Sets and Maps



Chris Lucas (cflucas@ncat.edu)