LECTURE 2: LISTS, STACKS, AND QUEUES

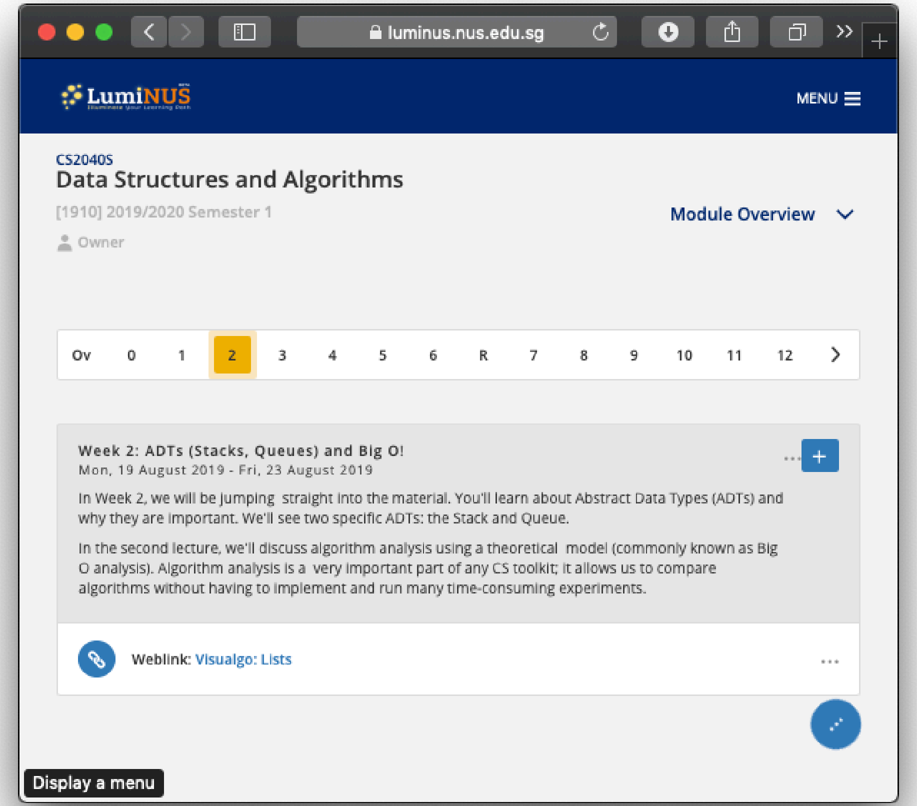
Harold Soh
harold@comp.nus.edu.sg

ADMINISTRATIVE ISSUES

87 registered students.

The slides from Lecture 1and the Java Quickstart are now online on Piazza and Luminus.

Lecture 2 (today) and 3 (tomorrow) will be up



CS2040S
Data Structures and Algorithms
[1910] 2019/2020 Semester 1
Owner

Week 2: ADTs (Stacks, Queues) and Big O!
Mon, 19 August 2019 - Fri, 23 August 2019

In Week 2, we will be jumping straight into the material. You'll learn about Abstract Data Types (ADTs) and why they are important. We'll see two specific ADTs: the Stack and Queue.

In the second lecture, we'll discuss algorithm analysis using a theoretical model (commonly known as Big O analysis). Algorithm analysis is a very important part of any CS toolkit; it allows us to compare algorithms without having to implement and run many time-consuming experiments.

Weblink: Visualgo: Lists

PROBLEM SET 0

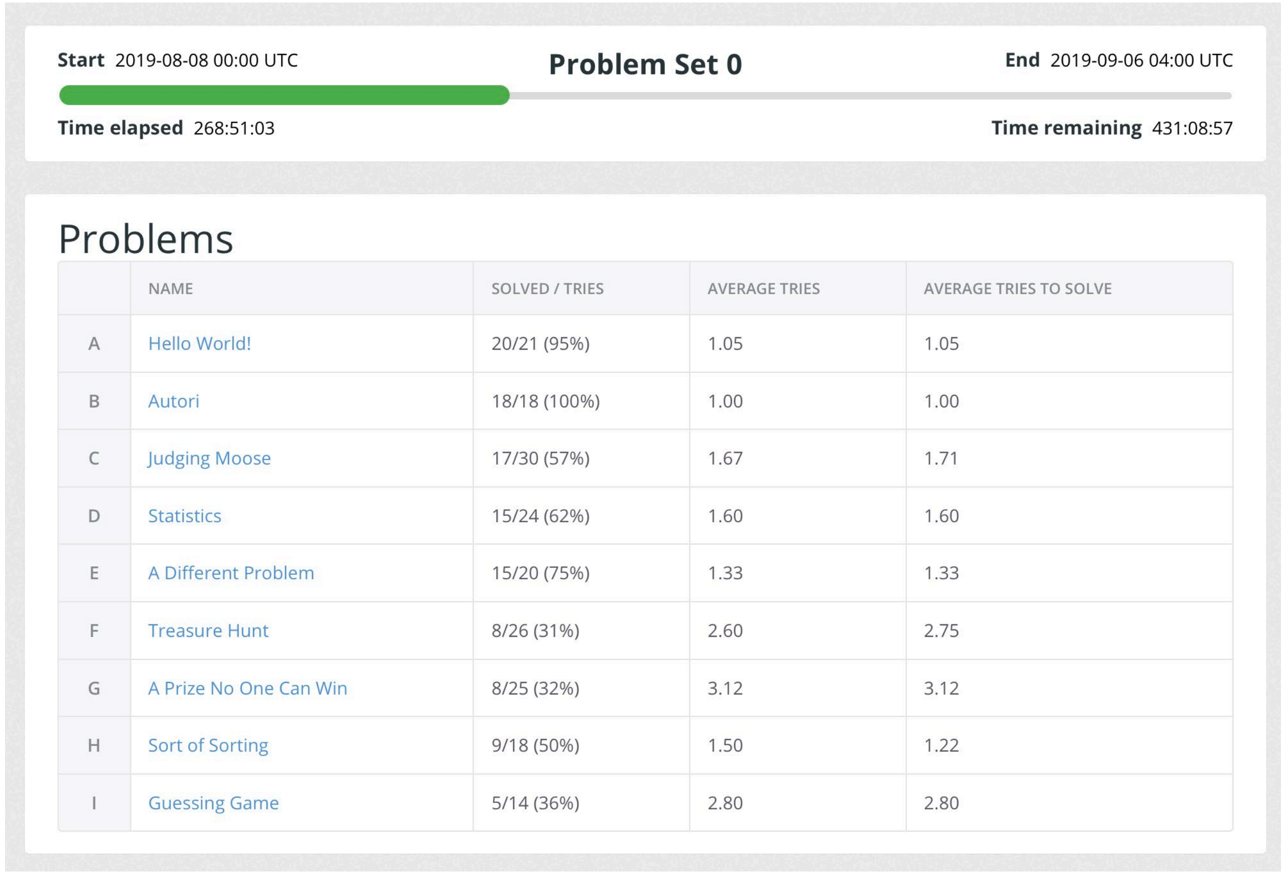
Ungraded Problem Set

- Getting started with Java and Kattis

Problem Set 1 will be released on Monday 26th Aug 2019.

Assignments will be released Monday and due in 2 weeks.

Late assignments will be penalized.



Start 2019-08-08 00:00 UTC **Problem Set 0** **End** 2019-09-06 04:00 UTC

Time elapsed 268:51:03 **Time remaining** 431:08:57

Problems

	NAME	SOLVED / TRIES	AVERAGE TRIES	AVERAGE TRIES TO SOLVE
A	Hello World!	20/21 (95%)	1.05	1.05
B	Autori	18/18 (100%)	1.00	1.00
C	Judging Moose	17/30 (57%)	1.67	1.71
D	Statistics	15/24 (62%)	1.60	1.60
E	A Different Problem	15/20 (75%)	1.33	1.33
F	Treasure Hunt	8/26 (31%)	2.60	2.75
G	A Prize No One Can Win	8/25 (32%)	3.12	3.12
H	Sort of Sorting	9/18 (50%)	1.50	1.22
I	Guessing Game	5/14 (36%)	2.80	2.80

QUESTIONS?

LEARNING OUTCOMES

By the end of this session, students should be able to:

- explain the **Linked List** Data Structure
- describe what is an **Abstract Data Type** (ADT)
- describe the **difference between ADTs and Data Structures** (DSes) and their relationship
- explain the **Stack** and **Queue** ADT

PEER-Q
76767
sms

DID YOU REVIEW VISUALGO?

FAIL

SMS to 7676

Code Respon

E.g.

ad6 A

Or via a browser:

https://peerq.nus.edu.sg/post.aspx

ERQ CODE: ad6

Hav ou ewed the Linked List mat l o sualgo?

Yes

B. No

C. Of course... not.

D. What is Visualgo?

TIME TO MOVE ON TO … ARCHIPELAGO

FAIL

https://archi go.comp.nu sg/

(Secretly hop ks!)

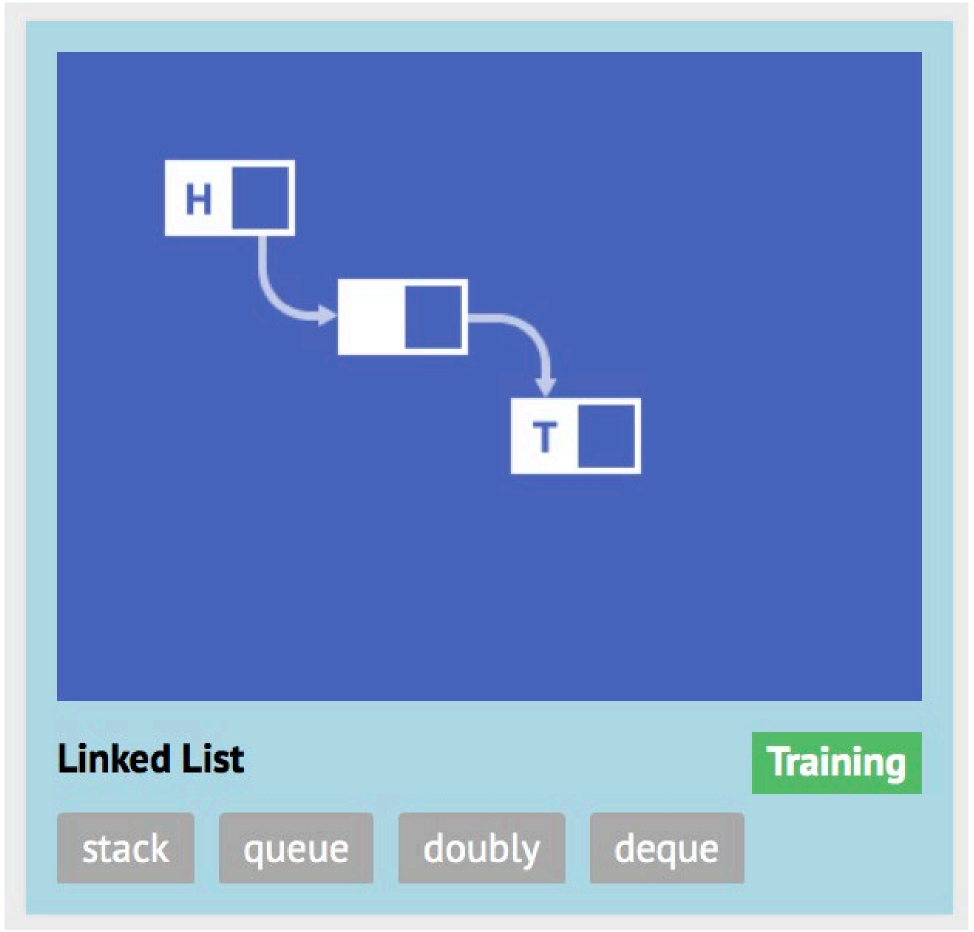
You should h receive n email

Sign in using your NUS email address, e.g., eXXXXXX@nus.edu.sg

Poll Everywhere

https://bit.ly/2LvG9bq

DID YOU REVIEW VISUALGO?



Via a browser:
PollEv.com/haroldsohsoo986
Or SMS: Text a code to +65 8241 0042

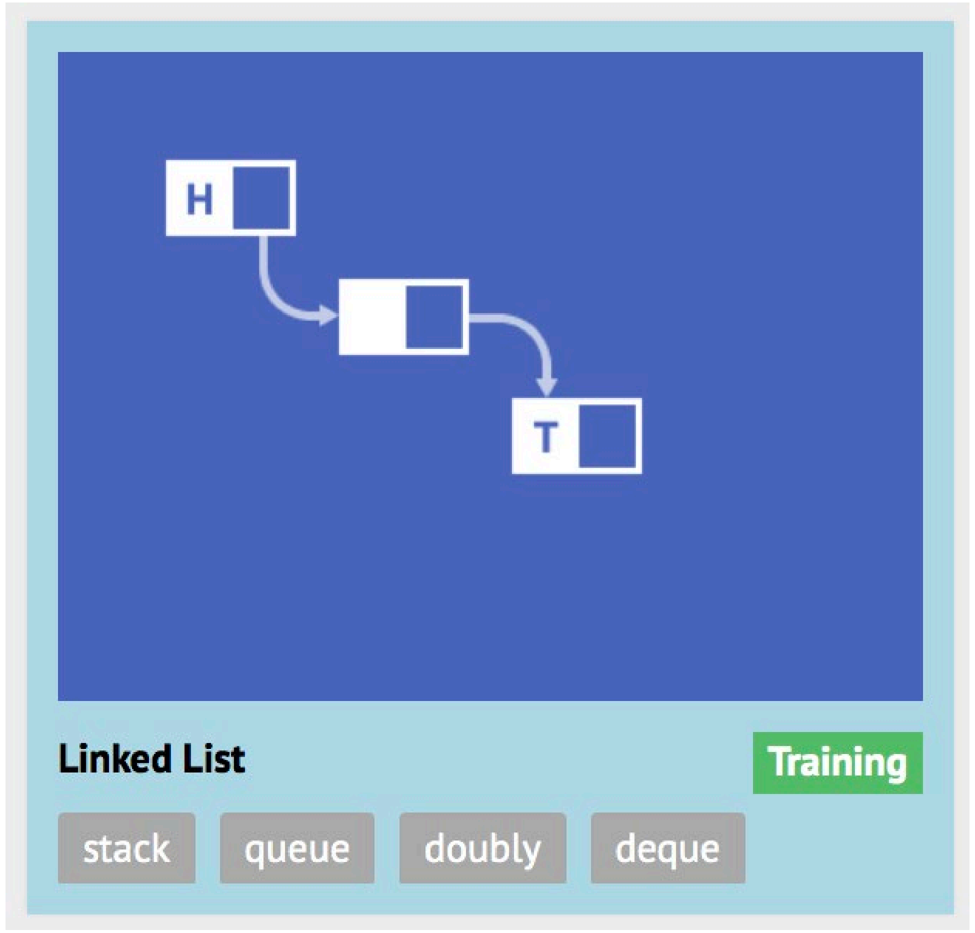
Have you reviewed the Linked List material on Visualgo?

A. Yes
B. No
C. Of course… not.
D. What is Visualgo?

Poll Everywhere

https://bit.ly/2LvG9bq

DID YOU REVIEW VISUALGO?



Via a browser:
PollEv.com/haroldsohsoo986
Or SMS: Text a code to +65 8241 0042

Have you reviewed the Linked List material on Visualgo?

A. **Yes**
B. No
C. Of course… not.
D. What is Visualgo?

REVISION: LINKED LIST DATA STRUCTURE

On to Visualgo!

Search the term 'algorithm visualization' in your favorite Search Engine, do you see VisuAlgo in the first page of the search result :)? Next level: Search that term again, but in your native language (if it is not English). Is VisuAlgo still listed in the first page? :). And get ready to be surprised: Search the name of your favorite data structure or algorithm without mentioning the keyword 'animation' or 'visualization'. Is VisuAlgo still listed in the first page? :):).

LINKED LIST V.S. ARRAYS

Which is faster (in the worst case)?

Operation	Compact Array	Linked List
Get(index i)		
Search(value)		
Insert(i, value)		
Remove(i) (non-lazy)		

LINKED LIST V.S. ARRAYS

Which is faster (in the worst case)?

Operation	Compact Array	Linked List
Get(index i)	✓ O(1)	O(N)
Search(value)		
Insert(i, value)		
Remove(i) (non-lazy)		

LINKED LIST V.S. ARRAYS

Which is faster (in the worst case)?

Operation	Compact Array	Linked List
Get(index i)	✓ O(1)	O(N)
Search(value)	Similar O(N)	
Insert(i, value)	Similar O(N)	
Remove(i) (non-lazy)	Similar O(N)	

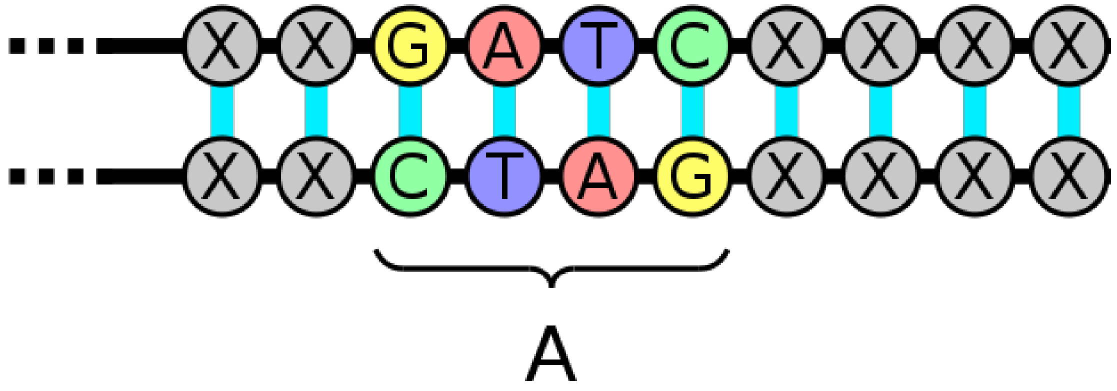
LINKED LIST V.S. ARRAYS

Which is faster (in the worst case)?

Operation	Compact Array	Linked List
Get(index i)	✓ O(1)	O(N)
Search(value)	Similar O(N)	
Insert(i, value)	Similar O(N)	
Remove(i) (non-lazy)	O(N)	O(N)
Remove(Node)	~	O(1)

PROBLEM: DNA PALINDROMES

Consider the alphabets for a particular DNA sequence is stored as a singly linked list.



palindrome

noun

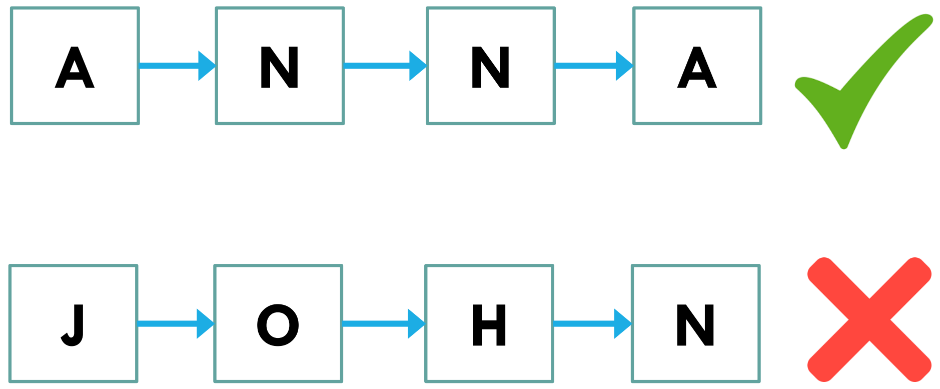
a word, phrase, or sequence that reads the same backwards as forwards, e.g.

- *madam*
- *anna*
- ACTGGTCA

Poll Everywhere

https://bit.ly/2LvG9bq

PROBLEM: DETECTING PALINDROMES



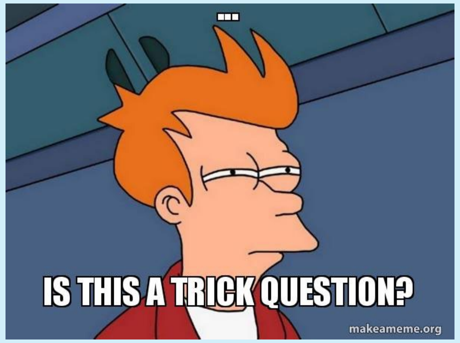
How many operations do we need to discover if the data in a singly linked list is a palindrome?

A. $\sim cN$ operations

B. $\sim cN^2$ operations

C. $\sim c2^N$ operations

D.



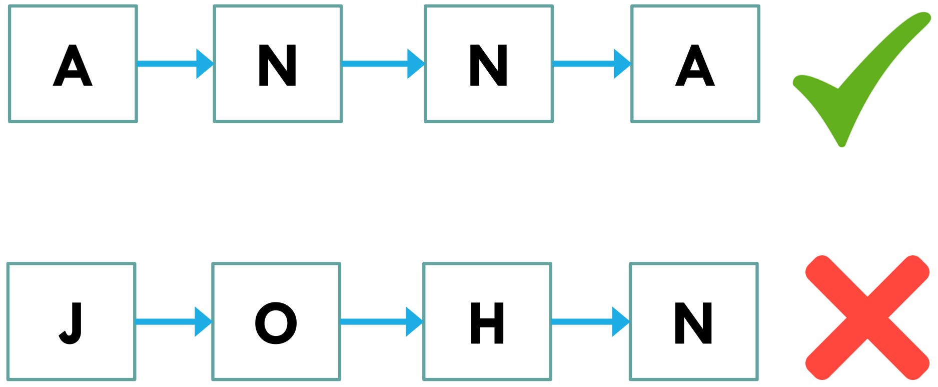
GENERAL STRATEGY

Think	Focus on	Ask	Improve
What is the most obvious way to solve the problem?	First focus on **correctness.** Then **efficiency.**	Ask yourself: • What operations are required? • What data structures do those operations well? • Is the data completely random (e.g., unsorted?) or is there structure I can leverage?	Improve your solution. • How can I make this faster? • What are my assumptions? • Does divide and conquer work?

Poll Everywhere

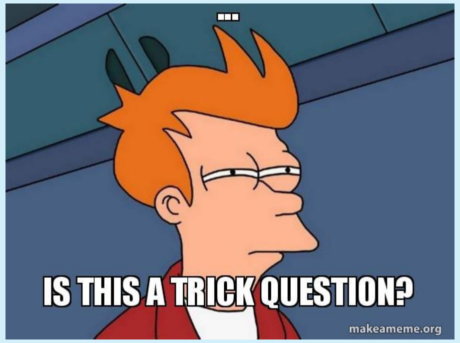
https://bit.ly/2LvG9bq

PROBLEM: DETECTING PALINDROMES



How many operations do we need to discover if the data in a singly linked list is a palindrome?

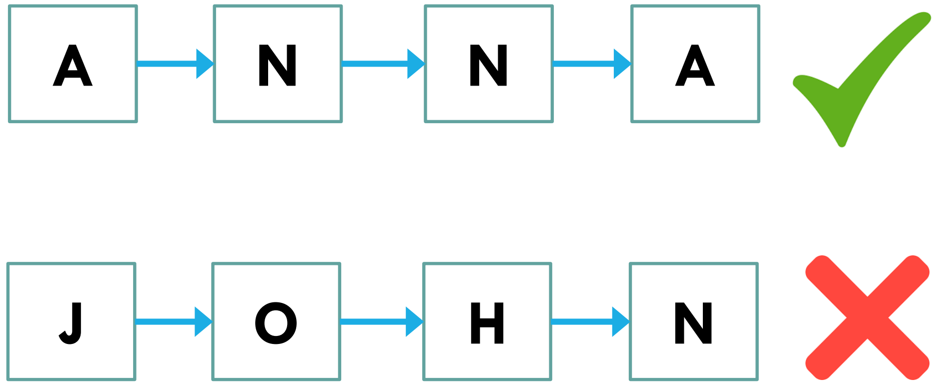
A. $\sim cN$ operations
B. $\sim cN^2$ operations
C. $\sim c2^N$ operations
D.



Poll Everywhere

https://bit.ly/2LvG9bq

PROBLEM: DETECTING PALINDROMES



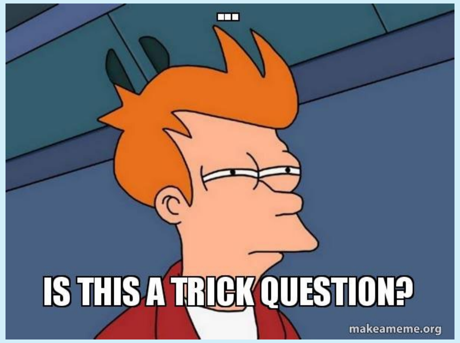
How many operations do we need to discover if the data in a singly linked list is a palindrome?

A. **$\sim cN$ operations**

B. $\sim cN^2$ operations

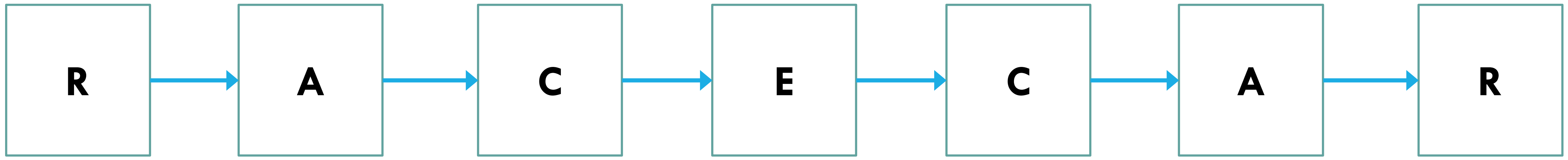
C. $\sim c2^N$ operations

D.



THE N^2 SOLUTION

For each element at the kth index, iterate to the $(N - (k + 1))$th element and compare.



FASTER: USE AN ARRAY AS A BUFFER

1. Run through the linked list and store each element in an array (~c*N* operations)
2. Check if data in the array is a palindrome (~c*N* Operations)

Cost: ~c*N* operations but extra c*N* memory!

FASTER: USE AN ARRAY AS A BUFFER

1. Run through the linked list and store each element in an array (~c*N* operations)
2. Check if data in the array is a palindrome (~c*N* Operations)

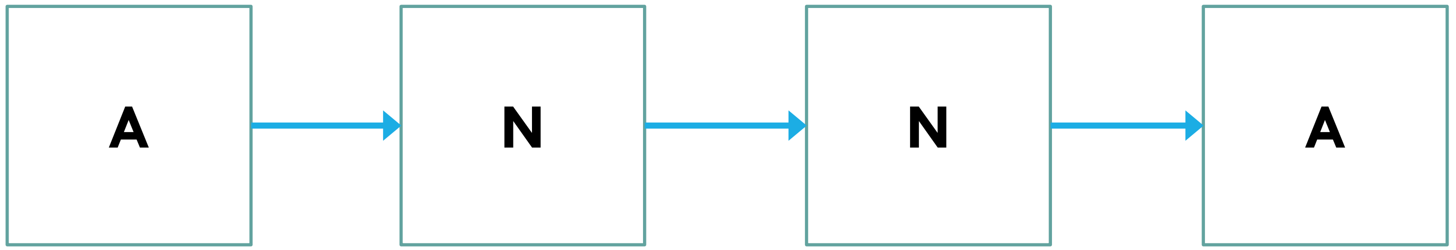
Cost: ~c*N* operations but extra c*N* memory!

Can we do **constant space** and **~c*N* operations?**

(let's hold off for now)

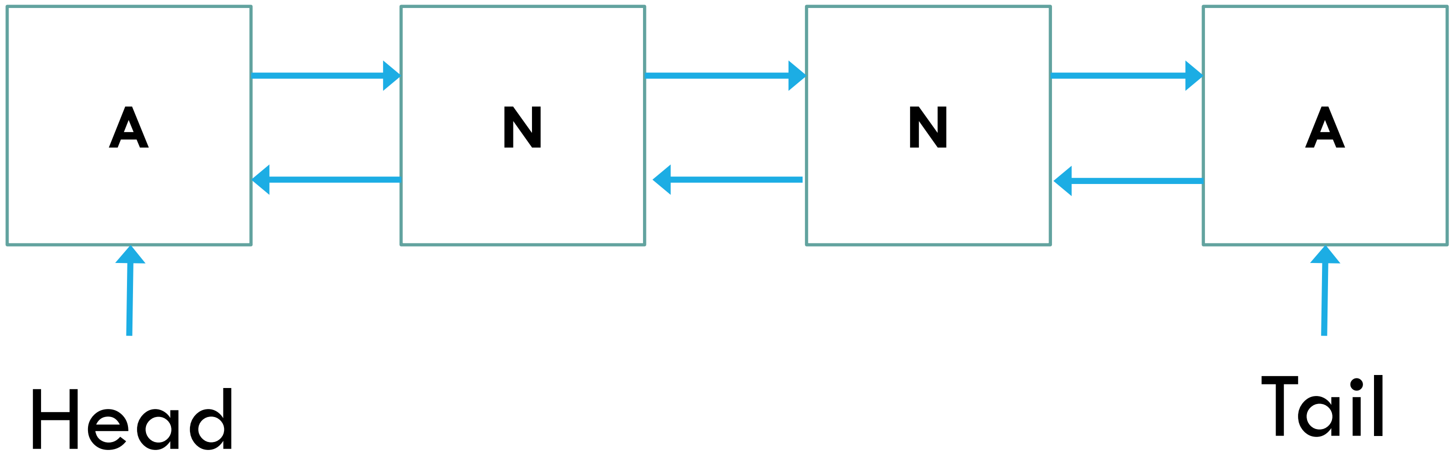
ADAPTING THE DATA STRUCTURE

How can we change the singly linked list to enable checks using ~N operations without using an additional buffer?



ADAPTING THE DATA STRUCTURE

How can we change the singly linked list to enable checks using ~N operations without using an additional buffer?



Poll Everywhere

https://bit.ly/2LvG9bq

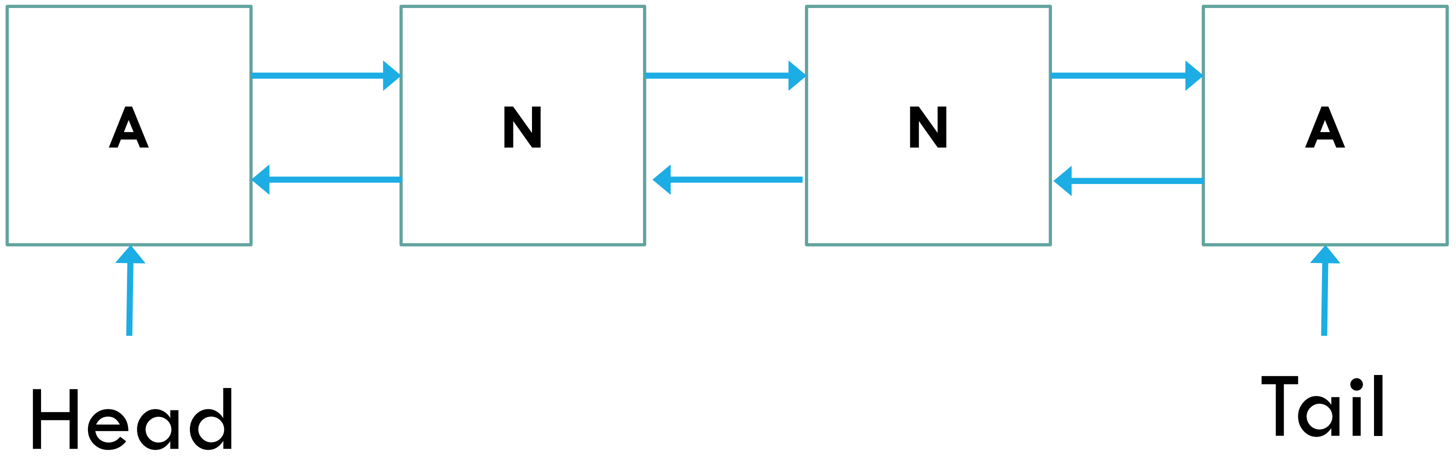
ADAPTING THE DATA STRUCTURE

Does this save space compared to the buffer solution?

A. Y

B. N

C. Ummm.. It depends?



Poll Everywhere

https://bit.ly/2LvG9bq

ADAPTING THE DATA STRUCTURE

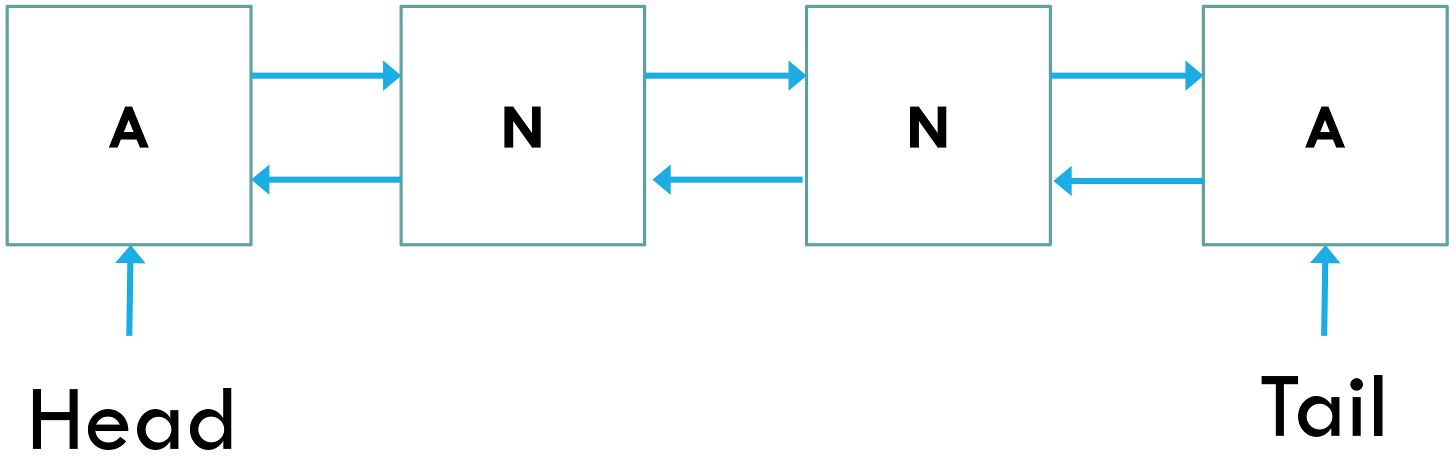
Does this save space compared to the buffer solution?

A. Y

B. N

C. **Ummm.. It depends?**

Why?



PROBLEM: REVERSE A SINGLY LINKED LIST

$$\begin{tikzcd}
\boxed{L} \arrow[r] & \boxed{U} \arrow[r] & \boxed{K} \arrow[r] & \boxed{E}
\end{tikzcd}$$

$$\begin{tikzcd}
\boxed{E} \arrow[r] & \boxed{K} \arrow[r] & \boxed{U} \arrow[r] & \boxed{L}
\end{tikzcd}$$

Recursive solution?

PROBLEM: RECURSIVE LIST REVERSAL

RECURSIVE SOLUTION TO REVERSING LISTS

```
function reverseList(Node head)
    if head is null or head.next is null
        return head
    new_head = reverseList(head.next)
    head.next.next = head
    head.next = null
    return new_head
```

Can we do **constant space** and ~*cN* **operations?**

TAKE AWAYS

You can sometimes trade memory for computation time.

If a given data structure doesn't work for your problem, **Change it!**

We can operate recursively on data structures.

QUESTIONS?

PART 2: ABSTRACT DATA TYPES

Harold Soh
harold@comp.nus.edu.sg

PROBLEM: NARUTO THE NEW HIRE

Description of Naruto:

- "Nice guy!"
- "Really likes bananas!"
- "Not the smartest fellow… but friendly!"
- "I'm afraid he's going to $^%& up our code, man!"

Poll Everywhere

https://bit.ly/2LvG9bq

WHAT SHOULD WE DO?

We want to welcome Naruto to our team.

He wants to contribute.

We want him to contribute.

But we don’t want him to wreck our software.

A. Give him some pointless work...
B. The forest is nearby...
C. Monkeys taste pretty good...
D. All of the above!
E. None of the above!

Poll Everywhere

https://bit.ly/2LvG9bq

WHAT SHOULD WE DO?

We want to welcome Naruto to our team.

He wants to contribute.

We want him to contribute.

But we don't want him to wreck our software.

A. Give him some pointless work…
B. The forest is nearby…
C. Monkeys taste pretty good…
D. All of the above!
E. **None of the above!**

ABSTRACTION

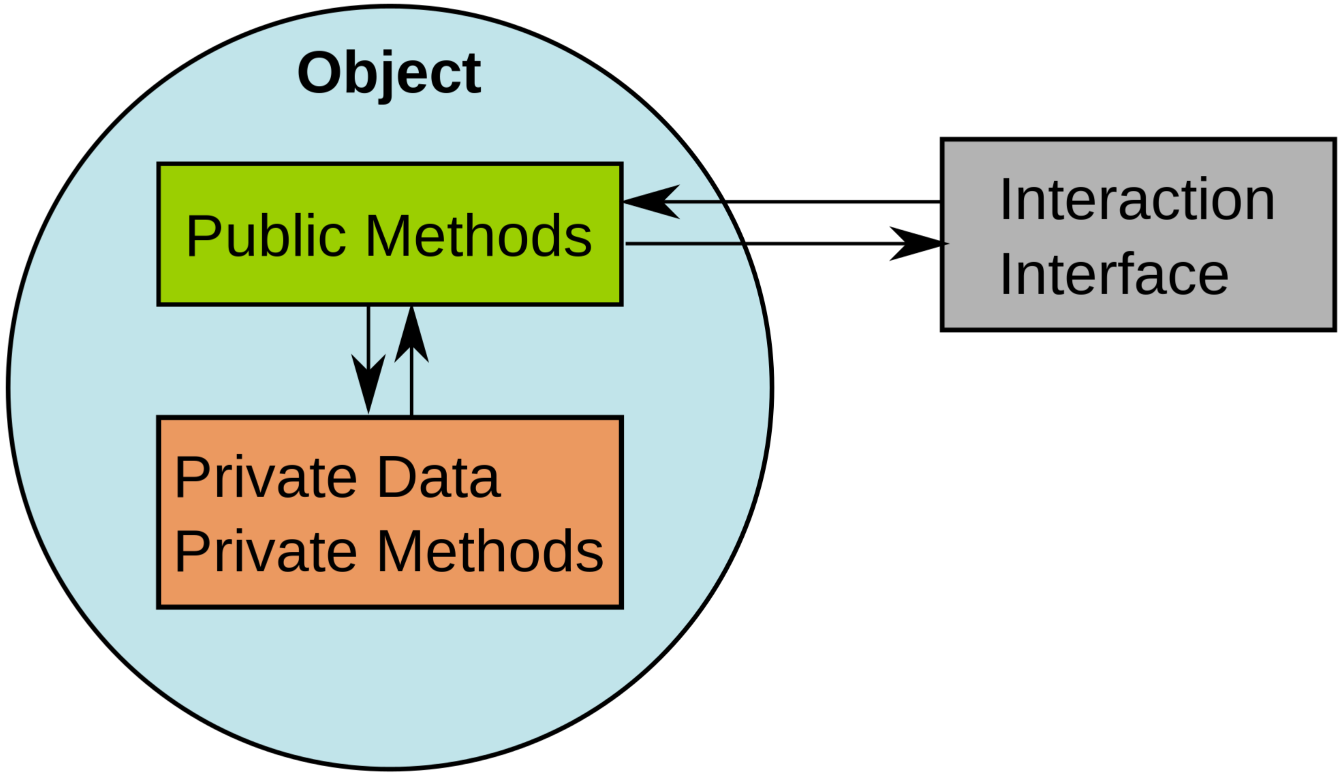
Remove all unnecessary elements

- What Naruto Needs-to-Know? **Expose!**
- What Naruto *doesn't* Need-to-Know? **Hide!**

Keep things simple!

ENCAPSULATION AND INFORMATION HIDING

Implementer



User

[Image from Wikipedia commons]

ABSTRACT DATA TYPES (ADTs)

Define **behavior**, *not* internal operation.

Interface `SimpleStore`

`add(x)`

- **Description:** adds x to the stored items.
- **Pre:** x is the item to be added
- **Post:** x is added to the store
- **Computational cost:**
  - takes constant time
  - adds constant space (memory).

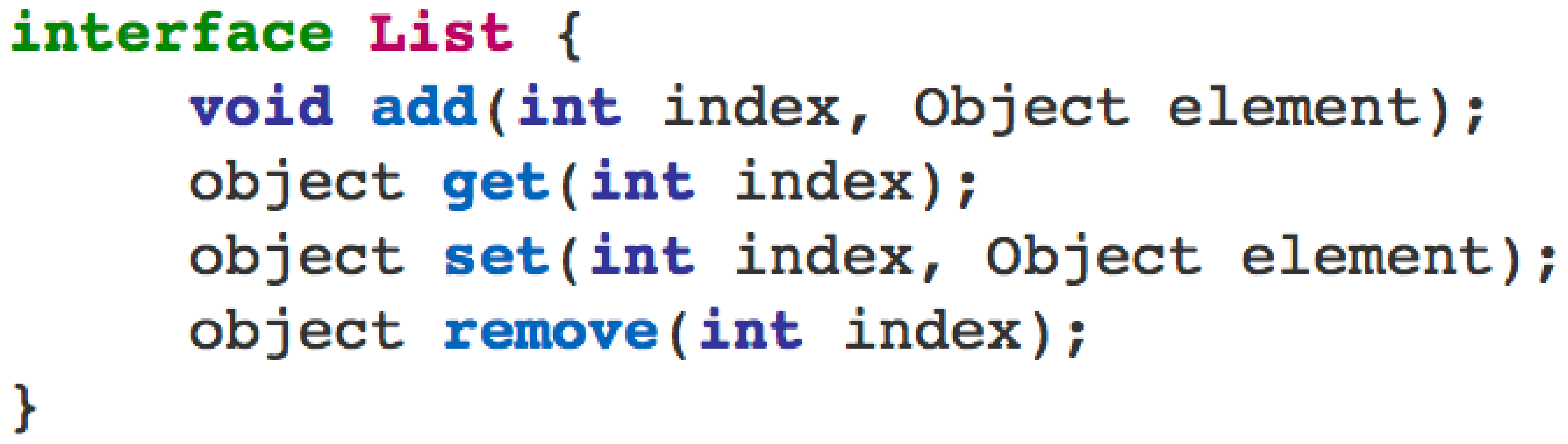
V.S.

Class `SimpleStore`

`add(x)`

- adds x to the stored items.
- if arr has already reached its current maximum size,
  - create a new temporary array
  - copy the original array over
  - add the x
  - update the end marker
- and so on ....

JAVA INTERFACE EXAMPLE

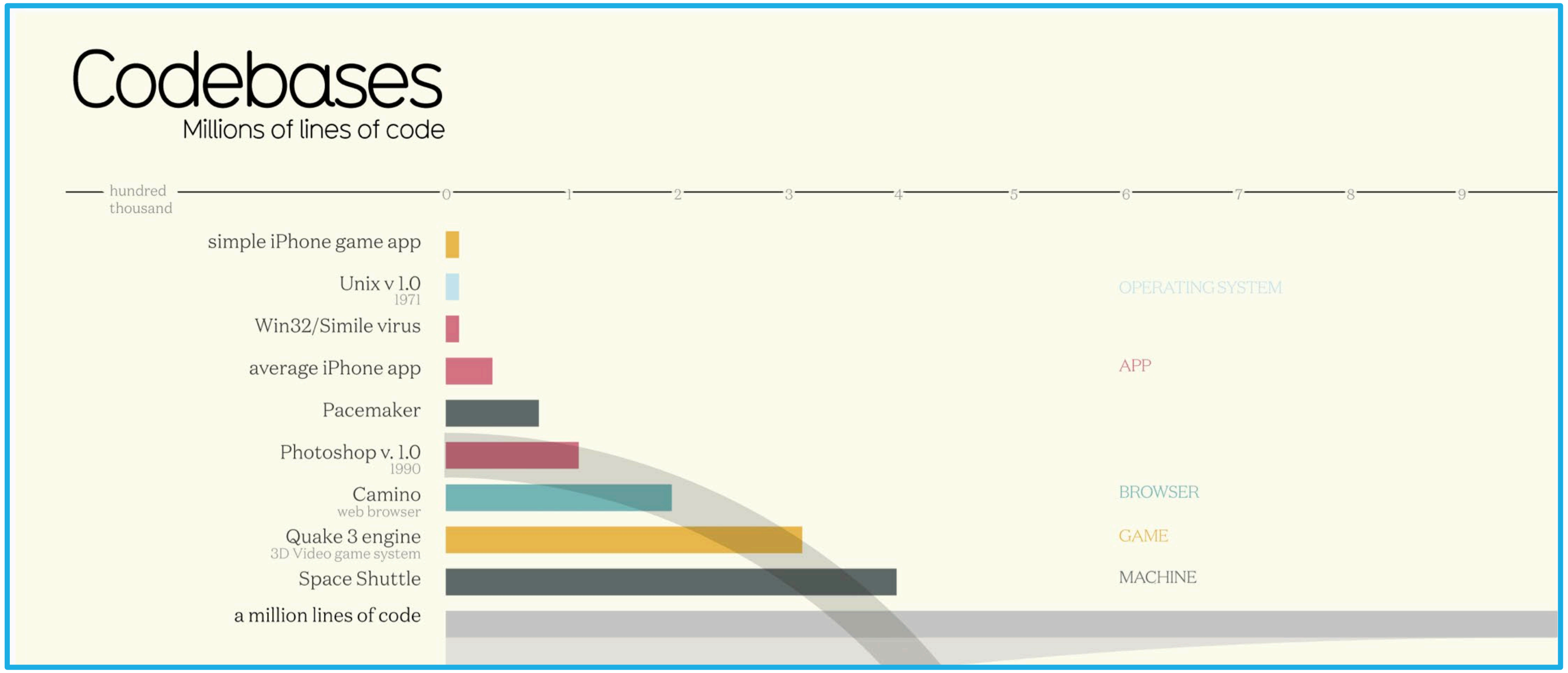


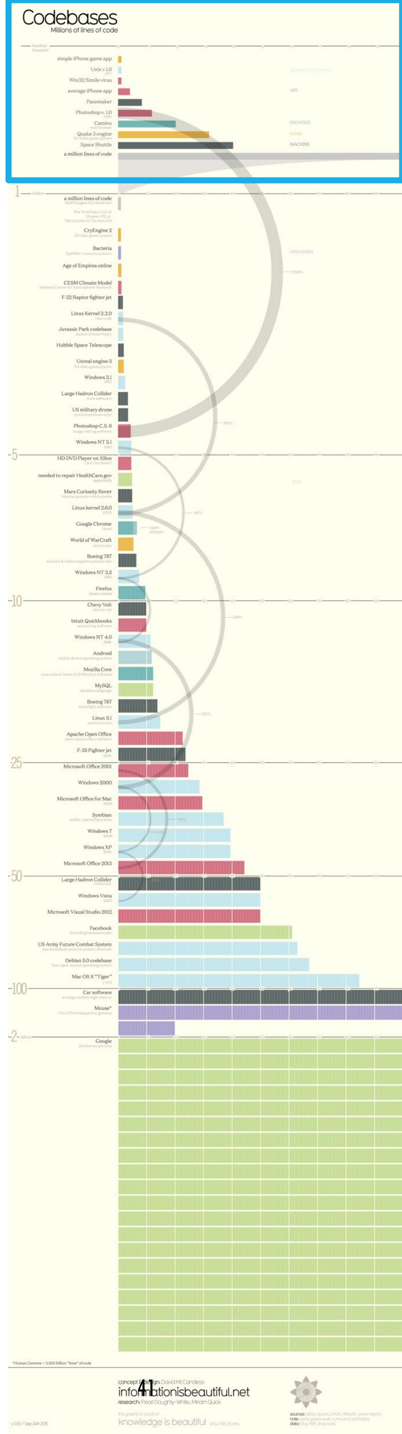
```
interface List {
        void add(int index, Object element);
        object get(int index);
        object set(int index, Object element);
        object remove(int index);
}
```

CLAIM: WE ARE ALL NARUTO!

=

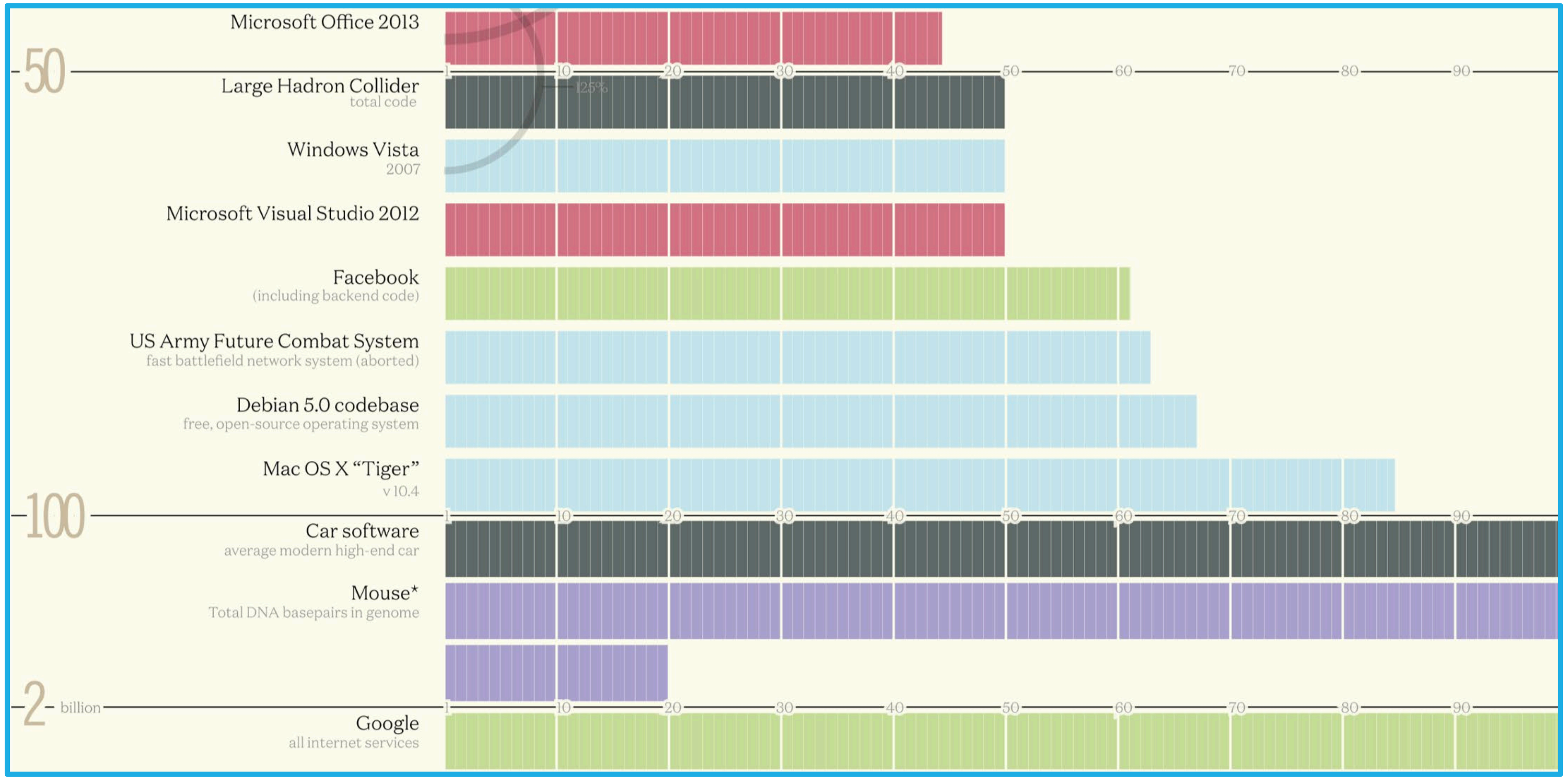
SOFTWARE IS GETTING VERY COMPLEX

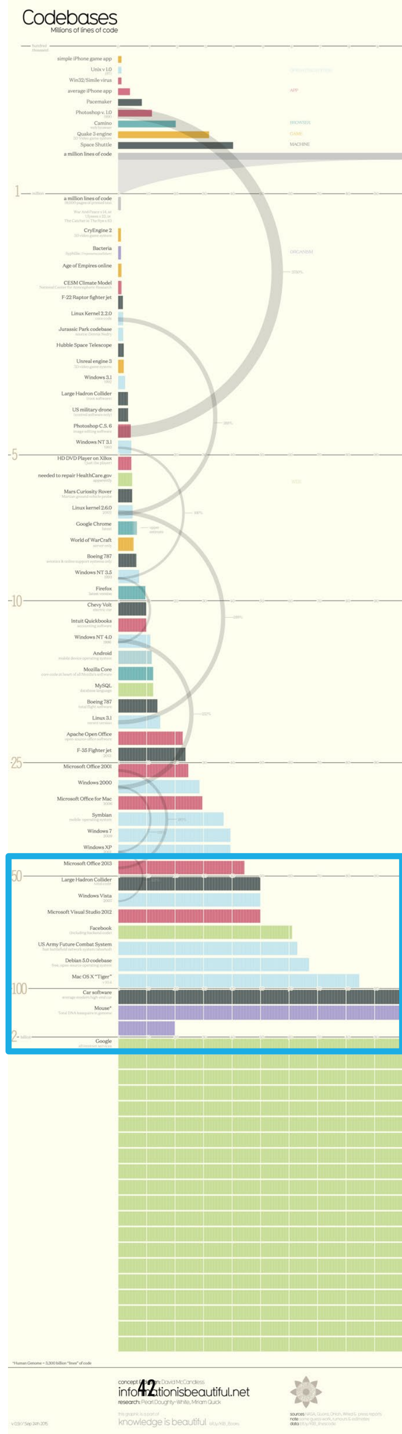




[http://www.informationisbeautiful.net/visualizations/million-lines-of-code/]

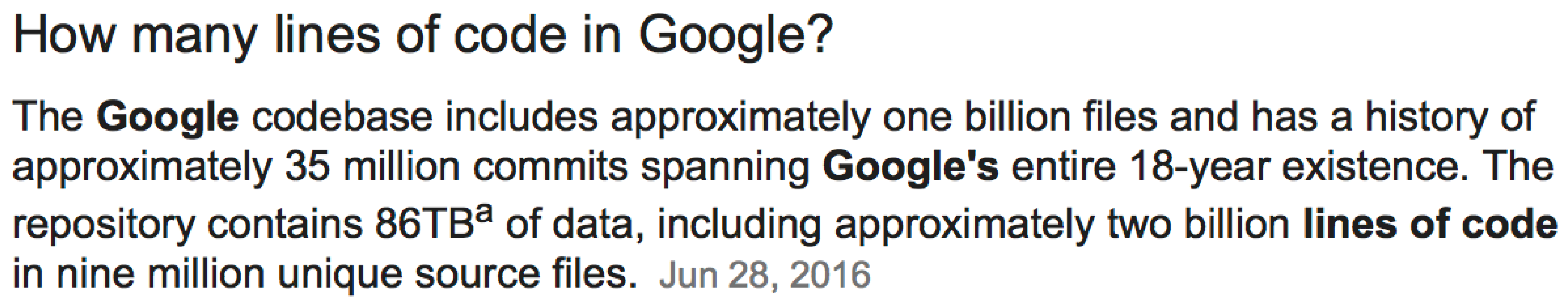
SOFTWARE IS GETTING VERY COMPLEX





[http://www.informationisbeautiful.net/visualizations/million-lines-of-code/]

SOFTWARE IS GETTING VERY COMPLEX


How many lines of code in Google?

The **Google** codebase includes approximately one billion files and has a history of approximately 35 million commits spanning **Google's** entire 18-year existence. The repository contains 86TB[a] of data, including approximately two billion **lines of code** in nine million unique source files. Jun 28, 2016

Also look at : http://www.informationisbeautiful.net/visualizations/million-lines-of-code/

ADTS ALLOW US TO:

Abstract away unnecessary details

Better understand complex software

Save us from ourselves

QUESTIONS?

PROBLEM: WE HAVE TOO MANY CUSTOMERS!

We need to manage a list of our orders.

Naruto to design order management app

Policy: serve the earliest customers first.

NOW IN OG ORCHARD POINT, LOWER LEVEL 1
NOW THERE'S ONE FOR EVERYONE
SINGAPORE DESIGN WEEK

PROBLEM: A CUSTOMER QUEUE

Design an ADT for Naruto to use

QUEUE VISUALIZATION

Back to Visualgo!

Do You Know?

Search the term 'algorithm visualization' in your favorite Search Engine, do you see VisuAlgo in the first page of the search result :)? Next level: Search that term again, but in your native language (if it is not English). Is VisuAlgo still listed in the first page? :). And get ready to be surprised: Search the name of your favorite data structure or algorithm without mentioning the keyword 'animation' or 'visualization'. Is VisuAlgo still listed in the first page? :):).

Poll Everywhere

https://bit.ly/2LvG9bq

QUEUE

Operations:

- `enqueue(x)`
- `peek()`
- `dequeue()`

First-in-First-Out (FIFO)

Which data structure should we use to implement our Queue ADT?

A. Linked List

B. Array

C. What is a data structure?

D. Wah! So many people.. must queue up also!

COMPUTATIONAL COST

	Array	Singly Linked List
enqueue(x)		
peek()		
dequeue()		
Other issues?		

COMPUTATIONAL COST

	Array	Singly Linked List
enqueue(x)	1	1
peek()	1	1
dequeue()	N	1
Other issues?	Fixed sized	Variable sized

Poll Everywhere

https://bit.ly/2LvG9bq

QUEUE

Operations:

- `enqueue(x)`
- `peek()`
- `dequeue()`

First-in-First-Out (FIFO)

Which data structure should we use to implement our Queue ADT?

A. **Linked List**

B. Array

C. What is a data structure?

D. Wah! So many people.. must queue up also!

BUSINESS IS BOOMING!

Queue Worked!

Customers are happy!

Naruto (and you) get a promotion

We’ve started home deliveries!

PROBLEM: NEW HIRES DELIVERING BOXES

New hires need very precise instructions

Policy:

- Deliver box from top to bottom
- New packages from factory goes on top

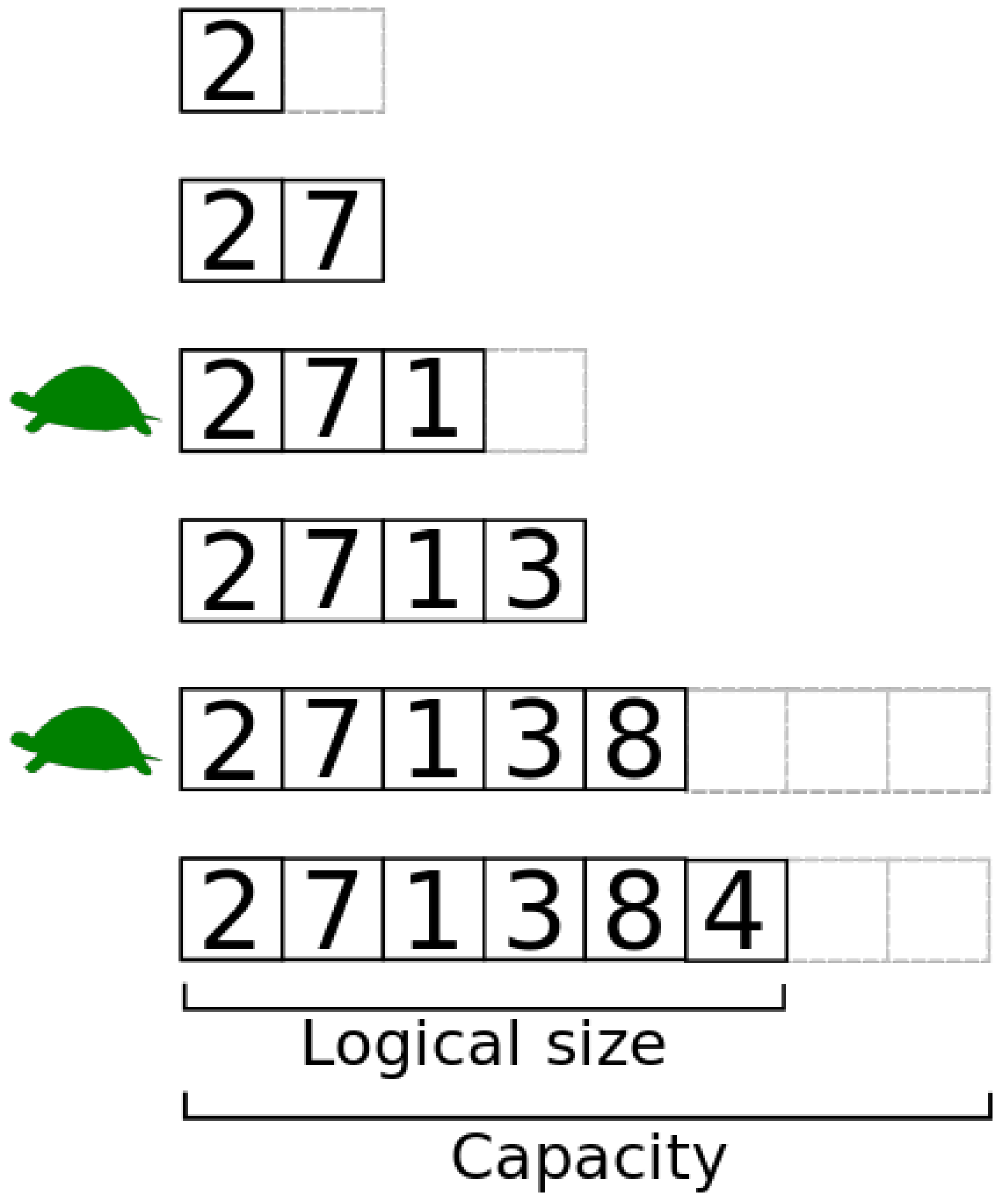
DYNAMIC ARRAYS

Arrays that can grow.

Amortized constant time for adding at the end (*more about this later in the course!*)

Different strategies for growing, e.g.

- Double the space each time you need to grow
- increase by a pre-set amount



Poll Everywhere

https://bit.ly/2LvG9bq

STACK

Operations:

- `push(x)`
- `peek()`
- `pop()`

Last-in-First-Out (LIFO)

Which is more time efficient for implementing a Stack: a Linked List or a Dynamic Array?

A. Dynamic Array
B. Singly Linked List
C. What is an ADT?
D. Ask Naruto!

COMPUTATIONAL COST

	Dynamic Array	Singly Linked List
push(x)	1 (Amortized)	1
peek()		
pop()		

COMPUTATIONAL COST

	Dynamic Array	Singly Linked List
push(x)	1 (Amortized)	1
peek()	1	1
pop()	1	1
	push to the back	push to the front

Poll Everywhere

https://bit.ly/2LvG9bq

STACK

Operations:

- push(x)
- peek()
- pop()

Last-in-First-Out (LIFO)

Which is more time efficient for implementing a Stack: a Linked List or a Dynamic Array?

A. **Dynamic Array**
B. **Singly Linked List**
C. What is an ADT?
D. Ask Naruto!

DELIVERY PROBLEM!

Stacks worked!

Customers are getting their deliveries!

Naruto (and you) get another promotion!

STACKS ARE VERY USEFUL!

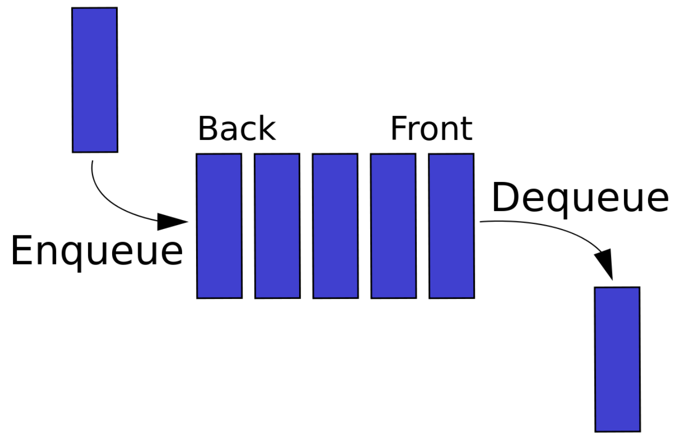
Implement Recursion

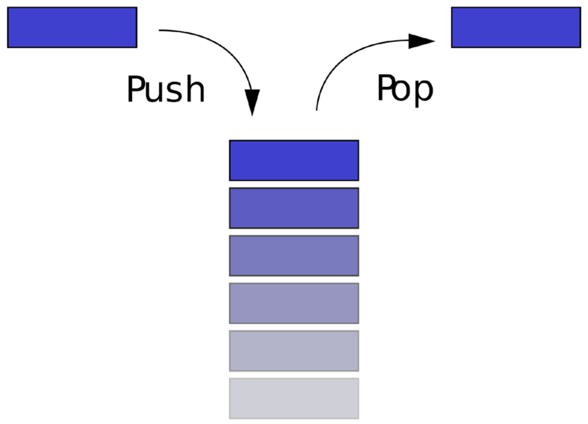
Expression Evaluation: (x+2)*(2/45) + 2

Depth-First-Search (later in the semester!)

and others…

SUMMARY: QUEUE & STACK





FIFO

LIFO

Poll Everywhere

https://bit.ly/2LvG9bq

NARUTO POSES A PUZZLE

Can we implement a Queue using Stacks?

A. Yes
B. No
C. Naruto is smarter than me...
D. What is a Queue again?

Poll Everywhere
https://bit.ly/2LvG9bq

NARUTO POSES A PUZZLE

Can we implement a Queue using Stacks?

A. **Yes** [✓]
B. No
C. Naruto is smarter than me…
D. What is a Queue again?

USE 2 STACKS

Enqueue(x): Add it to the top of the Stack A

Dequeue():

- Pop off all elements from Stack A to Stack B.
- Remove the last element (which is the first to be enqueued).
- Pop off all elements from Stack B to Stack A.

Computational Cost:

- Constant time for the Enqueue.
- ~2N Operations for the Dequeue.

But there is a faster way…

TWO SPECIALIZED STACKS

Poll Everywhere

https://bit.ly/2LvG9bq

NARUTO POSES ANOTHER PUZZLE

How can we quickly reverse a singly linked list?

A. Use a Stack
B. Use a Queue
C. Naruto is definitely smarter than me…
D. What is a Stack again?

NARUTO POSES ANOTHER PUZZLE

Poll Everywhere

https://bit.ly/2LvG9bq

How can we quickly reverse a singly linked list?

A. **Use a Stack**
B. Use a Queue
C. Naruto is definitely smarter than me…
D. What is a Stack again?

Poll Everywhere

https://bit.ly/2LvG9bq

NARUTO POSES ANOTHER PUZZLE

How can we quickly reverse a singly linked list?

A. **Use a Stack**
B. Use a Queue
C. Naruto is definitely smarter than me...
D. What is a Stack again?

But there is a more space efficient way...

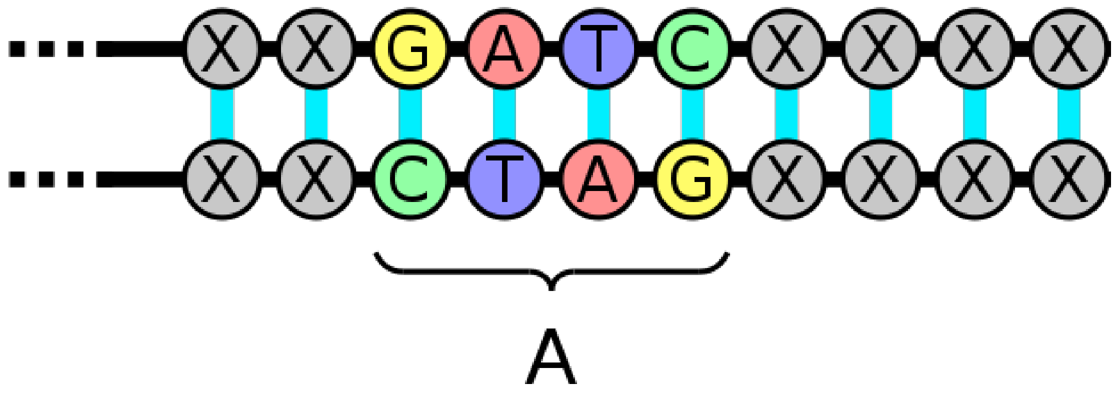
REVERSING A SINGLY LINKED LIST

REVERSING A SINGLY LINKED LIST PSEUDOCODE

```
Node reversed_list = null
Node current = head
while current is not null
    Node next = current.next
    current.next = reversed_list
    reversed_list = current
    current = next
head = reversed_list
```

PROBLEM: DNA PALINDROMES

Consider the alphabets for a particular DNA sequence is stored as a linked list.



palindrome

noun

a word, phrase, or sequence that reads the same backwards as forwards, e.g.

- *madam*
- *anna*
- ACTGGTCA

USE A **STACK**

1. ~~Get the midpoint of an list (~cN operations)~~
2. Run through first half the linked list and **push each element on a stack** (~cN operations)
3. Iterate through the remaining half and check against each popped element of the stack (~cN Operations)

Cost: ~cN operations but extra ~$N/2$ memory!

REVERSE THE SECOND HALF

1. Get the midpoint of an list (~c*N* operations)
2. Reverse the second half (~c*N* operations)
3. Iterate from the beginning and the midpoint, comparing each element

Cost: ~c*N* operations and 1 memory slot!

NARUTO TURNS OUT TO BE AWESOME!

NEW! Description of Naruto:

- “Nice guy!”
- “Really likes bananas!”
- “Friendly and super smart!”
- “I was totally wrong! he turned out to be an awesome coder, man!”

LEARNING OUTCOMES

By the end of this session, students should be able to:

- explain the **Linked List** Data Structure
- describe what is an **Abstract Data Type** (ADT)
- describe the **difference between ADTs and Data Structures** (DSes) and their relationship
- explain the **Stack** and **Queue** ADT

OTHER TAKE AWAYS

Change the data structure to suit your problem

ADTs can save us from complexity

Don’t judge people / monkeys by their appearance.

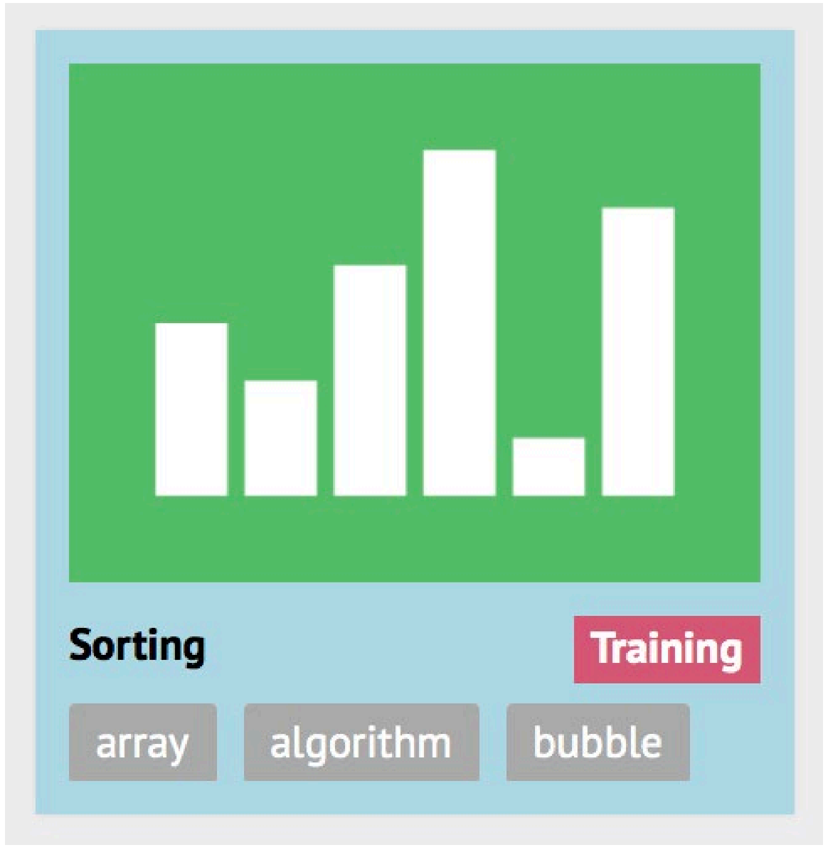
BEFORE LECTURE TOMORROW

Go to Visualgo.net and do the Sorting Module:

https://visualgo.net/en/sorting

Required: Sections 1- 8

Optional: 9 onwards



QUESTIONS?