[TAP:WNUME] Big-O

```
public static boolean contains(int[] nums, int x) {
    return containsHelper(nums,x,nums.length);
}
private static boolean containsHelper(int[] nums, int x,
                                      int curIdx){
    if (curIdx == 0)
       return false;
    return nums[curIdx]==x || containsHelper(nums,x,curIdx-1);
}
```

curIdx/2

- What is the time complexity of the code above?
  - A. O(n)
  - B. O(log n)
  - C. O(n log n)
  - D. $O(n^2)$
  - E. Whatever

$T(n) = k + T(n-1)$ (nums.length $\to \frac{n}{2}$)

$= k + k + T(n-2)$ ($\frac{n}{2^2}$)

$= k + k + \dots + T(0)$

$= O(n)\ O(\log n)$ $m = \log n$

$\frac{n}{2^m} = 1$

$n = 2^m$

$\log n = m$

Administrative Details

- Lab 1
  - I apologize for not having it returned yet
  - Feedback will show up on github as a Pull Request (PR)
  - PRs give you the option to view comments line-by-line, and respond to comments
- (New workflow this semester, so it is taking time to get the kinks worked out. It should be faster turnaround than printouts once it is working.)

Agenda

- Induction
  - List

(Proof by) Induction

- The mathematical cousin of recursion is induction:

Base case:
- prove for 0, 1, ...

Inductive case:
- assume true for n-1
  prove true for n

Recursion

- In recursion, we always use the same basic approach/structure
  - base case
  - recursive case

Mathematical Induction

- Prove that for every n ≥ 0

$$2^0 + 2^1 + 2^2 + \ldots + 2^n = 2^{n+1} - 1$$

Base case:

$n = 0$: $2^0 = 2^{0+1} - 1$

$1 = 2 - 1$

$1 = 1$ ✓

Inductive case:

assume $2^0 + 2^1 + \cdots + 2^{n-1} = 2^{n-1+1} - 1$ (IH)

show $2^0 + \cdots + 2^n = 2^{n+1} - 1$

$2^0 + \cdots + 2^n = 2^{n-1} + 2^n$ by IH

$= 2(2^n) - 1$

$= 2^{n+1} - 1$ ✓ □

Mathematical Induction

- Prove that for every n ≥ 0

$$0 + 1 + \ldots + n = \frac{n(n+1)}{2}$$

$n = 0$ $\quad 0 = \frac{0(0+1)}{2}$ ✓

assume $0 + \ldots + n-1 = \frac{(n-1)(\overbrace{n-1+1}^{n})}{2}$ (IH)

show $0 + \ldots + n = \frac{n(n+1)}{2}$

$0 + \ldots + n = \frac{n(n-1)}{2} + n$ by IH

$= \frac{n(n-1)}{2} + \frac{2n}{2}$

$= \frac{n^2+n}{2}$

$= \frac{n(n+1)}{2}$ □

Agenda

- Induction
- ⊙ List

The List Interface

```
interface List {
    size()
    isEmpty()
    contains(e)
    get(i)
    set(i, e)
    add(i, e)
    remove(i)
    addFirst(e)
    getLast()
    .
    .
    .
}
```

Vector implements List

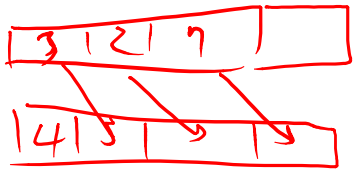
:

:

:

Singly Linked List

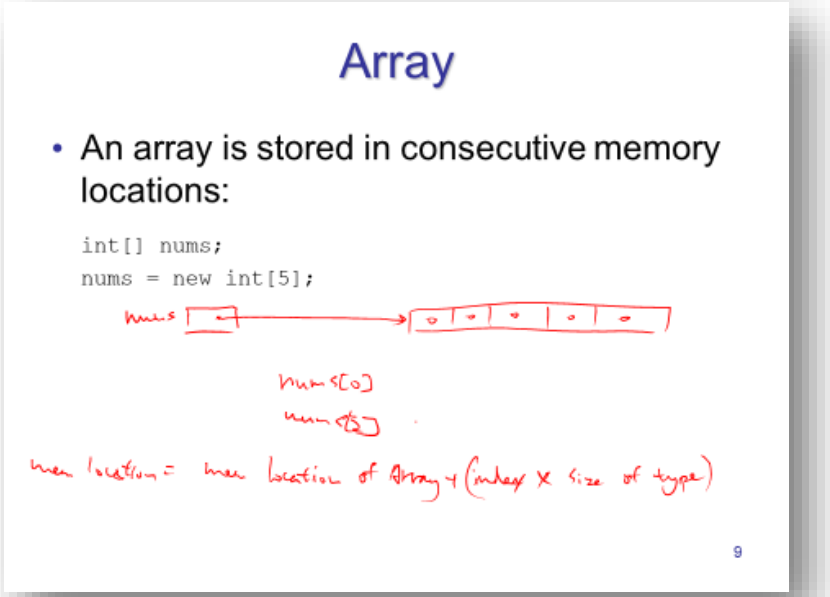
Pros and Cons of Vectors



Pros

constant time
Random access

- Fast access to elements



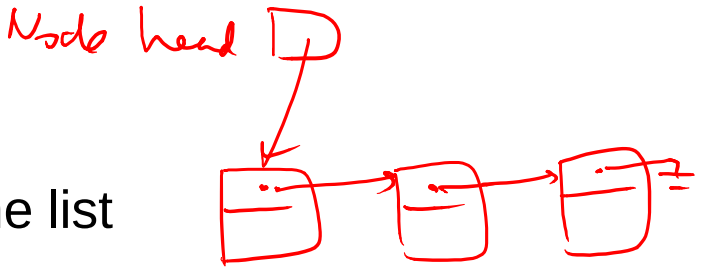
- Dynamically Resizeable? yes, but inefficient

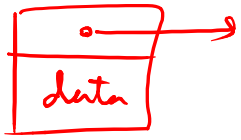
Cons

- Slow updates to front of list
- Hard to predict time for add (depends on internal array size)
- Potentially wasted space

Singly Linked List

- There are two key components of Lists
  - The list itself
    - Instance varibles
      - (Pointer to) the `head` node of the list
    - Methods
      - Those declared in the List interface
  - Nodes
    - Instance variables
      - data
      - (Pointer to) the “next” element
    - Define methods
      - Getters and setters



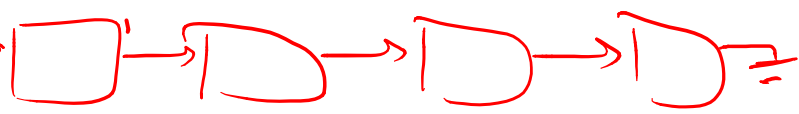


Singly Linked List Methods

head != null

//Pre: 0 ≤ index < numElements

```
public E get(int index) {
    Node finger = head;
    for(int i=0 ; i < index; i++)
        finger = finger.next();
    return finger.value();
}
```



Singly Linked List Methods

```
public E set(E d, int index) {
```

```
    E ori = finger.value();
    finger.setValue(d);
    return ori;
}
```

Singly Linked List Methods

```
public void add(E d, int index) {
```