# CS 180 - Big-O, Intro to Stacks

Note Title 9/19/2011

## Announcements

- HW due Monday
- Look for makefile on ~~announcement~~ schedule page → `make`

  or: `g++ SLinkedList.h`
  `g++ ~~testSLL~~.cpp`

- Lab on Thursday

# Algorithm Analysis

How do we compare two programs?

SPEED

→ time to run

# Speed

How fast an algorithm runs can be very dependent on variables in the system.

Examples:

- architecture
- language
- low level (assembly)
- inputs vary

# Primitive Operations

As a way to compare algorithms in a generic way, we instead count primitive operations.

Ex: add, load, shift, sub, comparison, multiplication & division

In addition, we (generally) only analyze the worst possible running time.

Why?

avoid misleading inputs

## Comparing

OK, so we have the worst case # of operations - usually a function of n.

↖ length of list, etc.

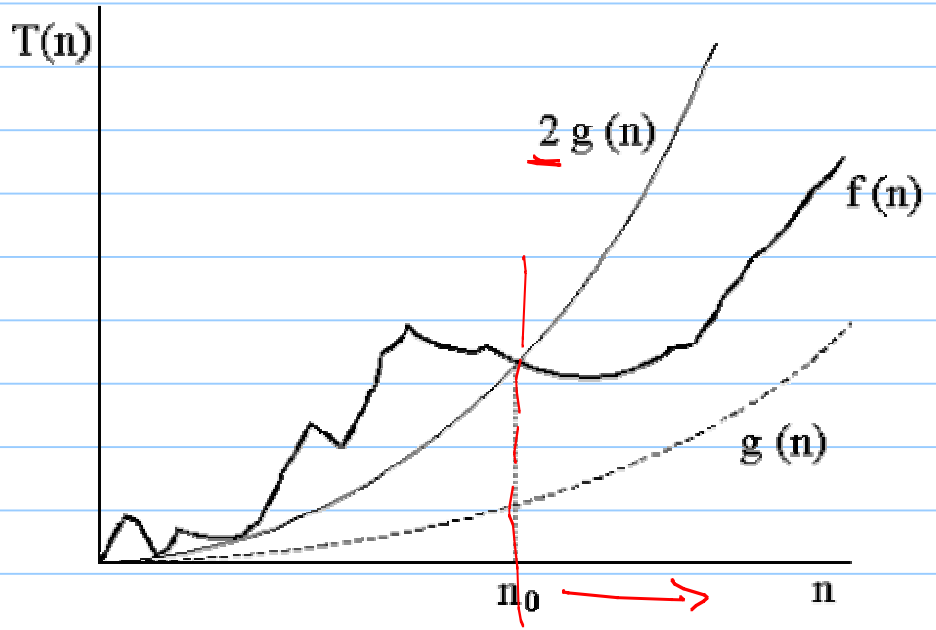
How to compare?

Big-O

# Big-O

We say $f(n)$ is $O(g(n))$ if $\forall n > n_0$,

$\exists\ c > 0$ such that $f(n) \leq c \cdot g(n)$.



Ex: $5n$ is $O(n^2)$

if $n > 5$, then $5n < n^2 = n \cdot n$

Ex: $5 \cdot n$ is $O(n)$

Let $c = 6$. Then $5n \leq c \cdot n$

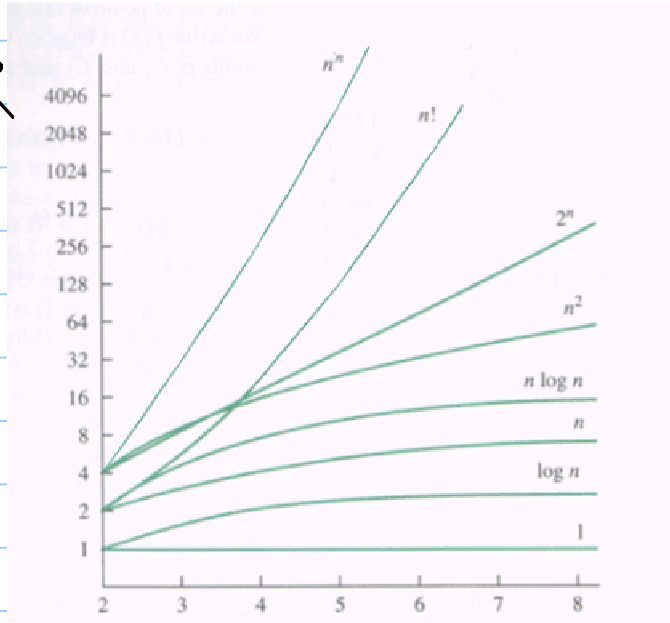
Ex: $16n^2 + 52$ is $O(n^2)$

$16n^2 + 52 \leq 16n^2 + 52n^2 \leq c \cdot n^2$

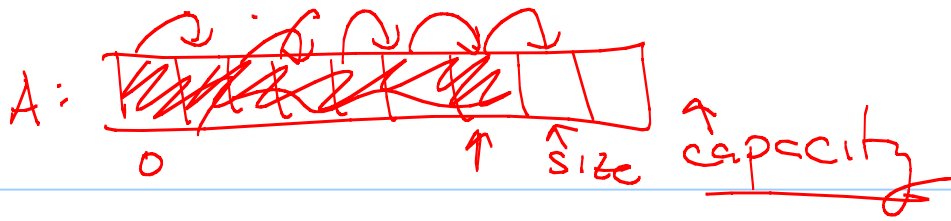
let $c = 68$

# Functions we will use

1. $O(1)$ - constant time
2. $O(\log n)$ - logarithmic time
3. $O(n)$ - linear time
4. $O(n \log n)$
5. $O(n^2)$ - quadratic time
6. $O(n^3)$ - cubic time
7. $O(2^n)$ - exponential time



# Algorithms

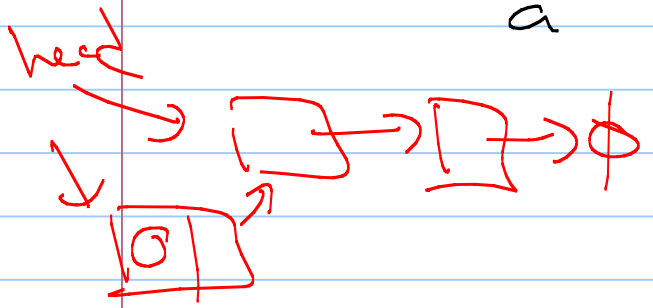



**Claim**: Inserting an element into the first spot in an array is $O(n)$ time.

```
for (int i = size-1; i >= 0; i--)
    A[i+1] = A[i];
A[0] = value;
```

$n$ times: $3 \cdot size + 4 = O(n)$

**Claim**: Inserting at the beginning of a list is $O(1)$ time.




allocate node
put value in it
2 pointer updates

4 operations $= O(1)$

# Common running times

- A for loop which goes from $i=0$ to $n-1$ and reads to an array

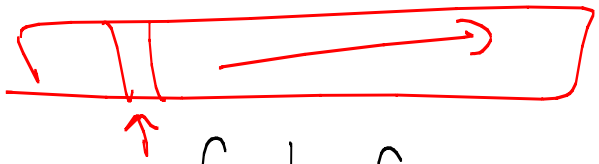
```
for (int i=0; i<n; i++)
    cin << array[i];
```

1 comparison, one addition, output, access to array

Analyze: $O(n)$ operations

$4 \cdot n + 2$

$$\sum_{i=0}^{n} 4 = \underbrace{(4+4\cdots+4)}_{n \text{ times}} = 4n$$



## Nested For loops : find if any 2 elements are identical

```
for (int i=0; i<n; i++)
    for (int j=i+1; j<n; j++)
        if (A[i] == A[j])
            cout << "Two items are the same" << endl;
```

3 operations

Analyze:

$$\sum_{i=0}^{n}\left[\sum_{j=i}^{n} 3\right] = \sum_{i=0}^{n}\left[3(n-i)\right]$$

$$= 3n + 3(n-1) + 3(n-2) + \cdots + 3 = 3\left[\sum_{i=1}^{n} i\right]$$

$$= 3\,\frac{n(n-1)}{2} = O(n^2)$$

Stack: a way to store a list of data

Ex: Web browser: store history for "back" button

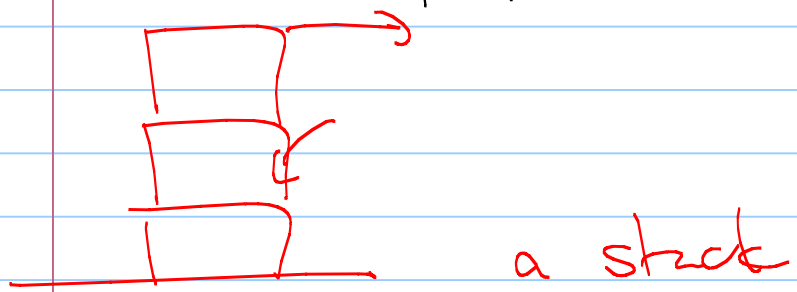
Ex: Text editors: store previously used commands

"Last in, first out"
LIFO

# The stack ADT:

Supports 2 main functions:

- push(e): add e to "top" of the stack
- pop(): remove e from the stack



a stack

## Others

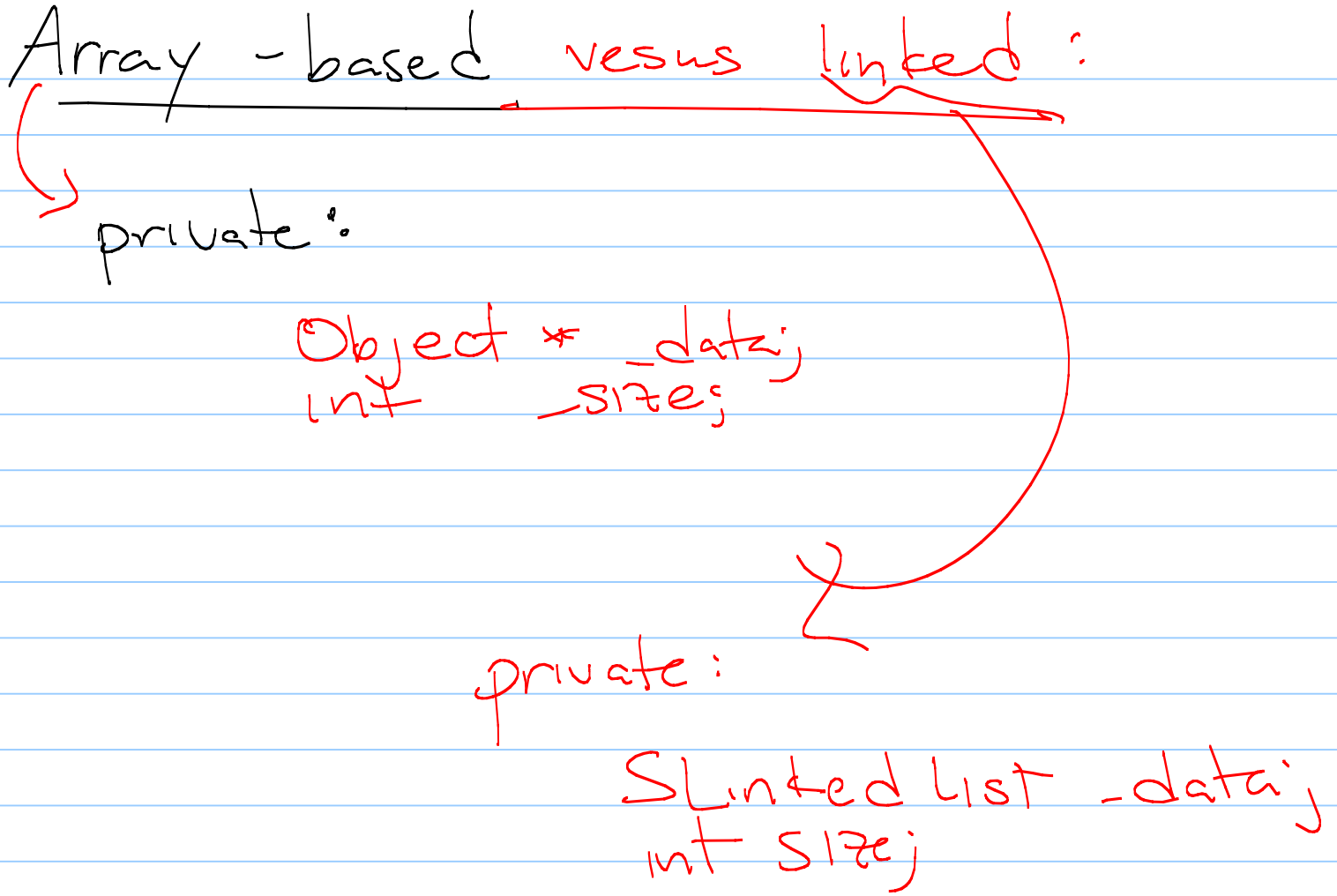
- top(): returns top element of the stack without removing it
- empty(): returns true if stack is empty
- size(): returns # of objects in the stack

# The Standard template library

- Has iostream, string, etc.

- Also has basic data structures!
  (We'll be coding our own anyway.)

- See cplusplus.com for documentation...


Array - based versus linked :
private :
Object * _data;
int _size;
private :
SLinkedList _data;
int size;

Plus other functions to code!