CS180 - Stacks

Note Title 9/23/2011

Announcements

- HW due Monday
- Next program will be out Soon

Primitive Operations

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

of 2 #s

↳ addition, subtraction, memory access, return, mult + div

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

Why? guaranteeing a minimum performance

Big-O

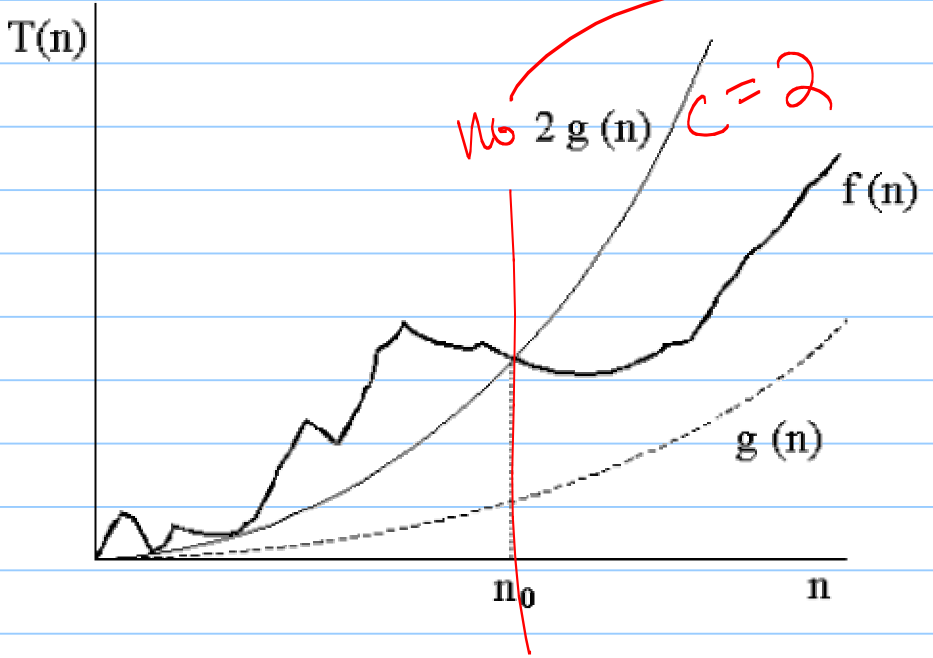
$f(n) \le g(n)$

for all

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

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

there exists



$\log_2 a + \log_2 b = \log_2(a \cdot b)$

$\log_2 x^c = c \log_2 x$

Functions we will use

① $O(1)$ – constant time

② $O(\log n)$ – logarthmic time

↳ Binary search

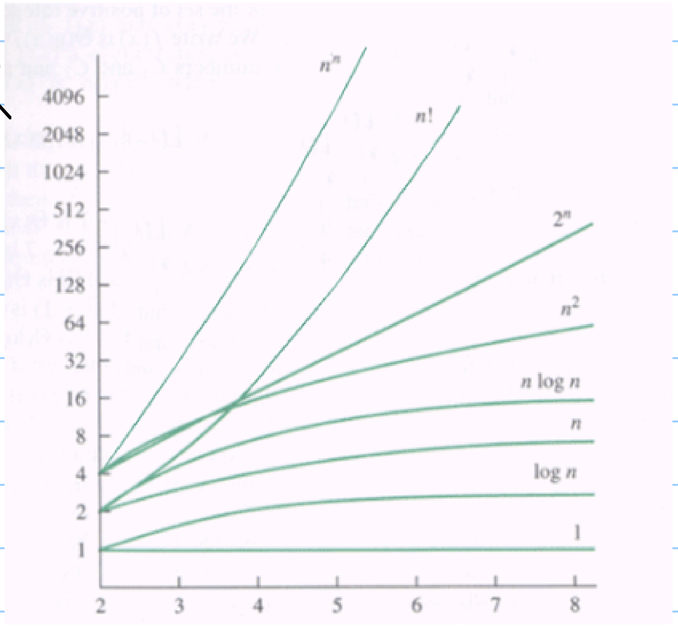
③ $O(n)$ – linear time

④ $O(n \log n)$

⑤ $O(n^2)$ – quadratic time

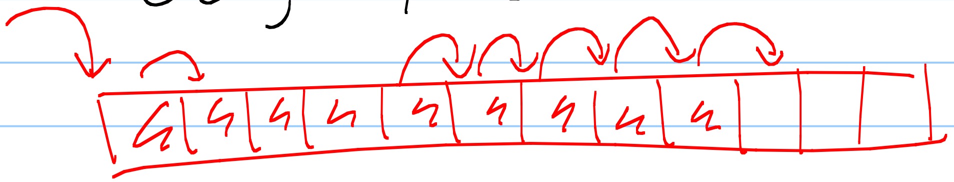
⑥ $O(n^3)$ – cubic time

⑦ $O(2^n)$ – exponential time

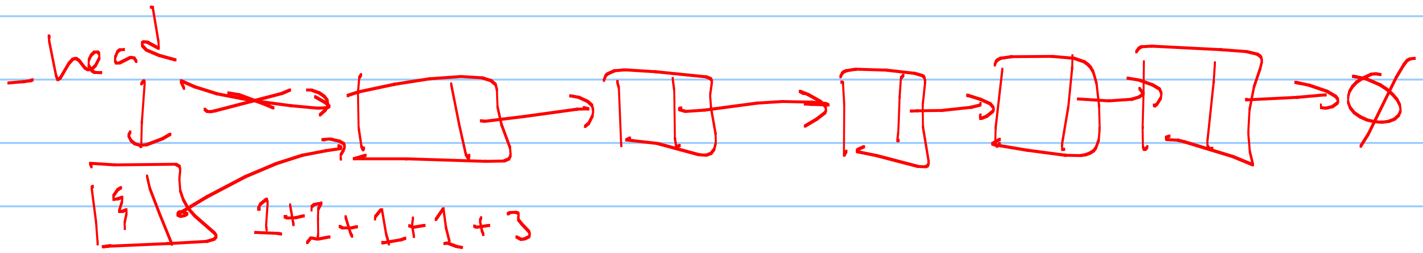


Algorithms

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



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



1+1+1+1+3

Common running times

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

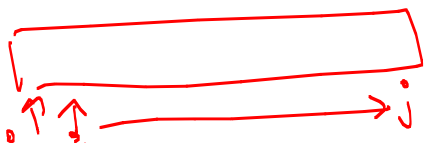
```
for (int i=0; i< n; i++)
    cin << array[i];
```

Analyze:

$$\sum_{i=0}^{n-1} 2 = \overbrace{(2+2+2+\cdots+2)}^{n \text{ times}}$$

$$= 2n = O(n)$$

$$(4+4+4\cdots+4) = 4(1+1+1\cdots+1)$$



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;
```

Analyze:

$$\sum_{i=0}^{n-1}\left[\underbrace{\sum_{j=i+1}^{n-1} 4}\right] = 4\sum_{i=0}^{n-1}(n-1-i) = 4\big((n-1)+(n-2)+(n-3)+\cdots+1\big)$$

$$= 4\sum_{i=0}^{n-1} i = \frac{4n(n-1)}{2} = O(n^2)$$

$$4\sum_{j=i+1}^{n-1} 1 = 4(\underbrace{1+1+\cdots+1}_{(n-1)-(i)}) = 4(n-1-i)$$

Stack: a way to store a list of data

same behavior

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

Ex: Text editors: store previously used commands

type "undo"

abstract data type

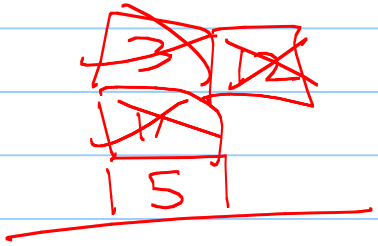
The stack ADT :

Supports 2 main functions:

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

push(5)
push(11)
push(3)
pop()
push(12)

pop()
pop()



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

Now, how to build?

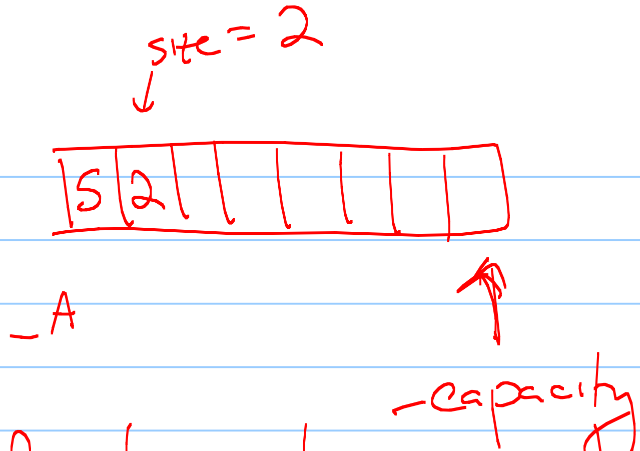
We know the functions to list in the .h file.

But how to do private data to store things on the stack?

Ideas?

- Use an array
- Use a singly linked list

Array-based:



private:

```
    int _size;        // # of elements in stack
    Object * _A;
    int _capacity;    // max # stack can hold
```

Other functions to code

Housekeeping functions!

- Destructor
- Operator =
- Copy Constructor