Recitation 1

June 7, 2018

Outline

Review Pointers and Dynamic Memory

Base Conversion

Big-O (Coming back to it)

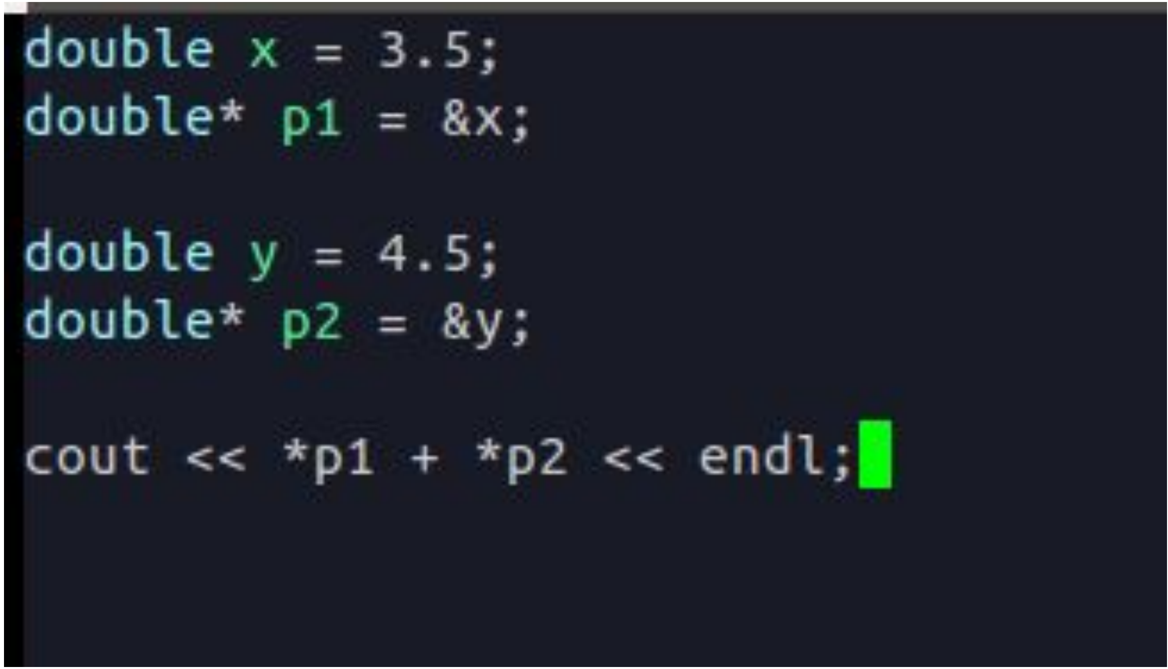
Case Study

Structs and classes (Helpful for the homework)

Pointers Review

Show the printout of the following code:

8



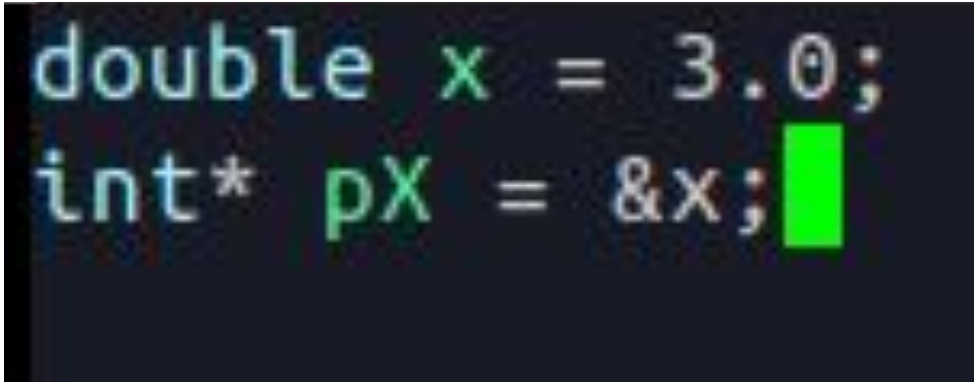
```
double x = 3.5;
double* p1 = &x;

double y = 4.5;
double* p2 = &y;

cout << *p1 + *p2 << endl;
```

Pointers Review

What's wrong with the following code?

Wrong type.

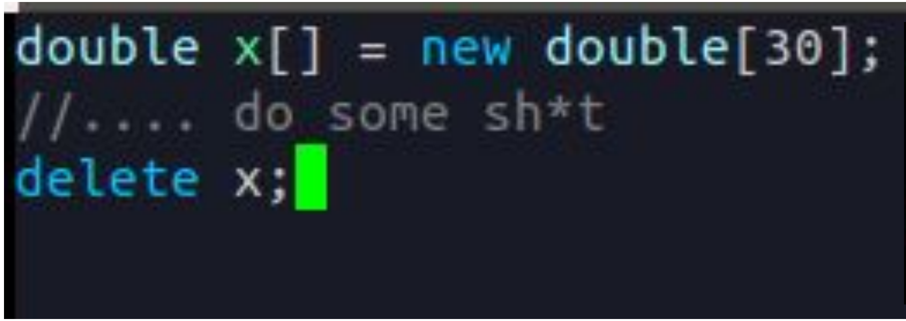
x is double, but *px is a pointer variable for int variables.



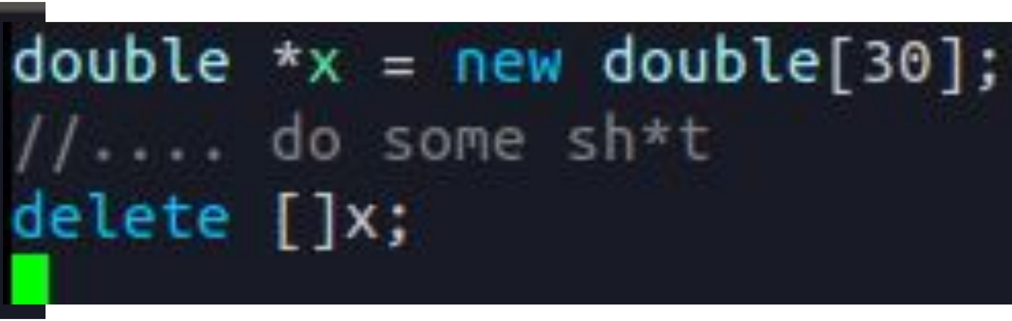
```
double x = 3.0;
int* pX = &x;
```

Dynamic Memory Review

Suppose you want a dynamic array and later need to release it. Identify two errors in the following code.



```
double x[] = new double[30];
//.... do some sh*t
delete x;
```



```
double *x = new double[30];
//.... do some sh*t
delete []x;
```

Pass by...

Reference- changes the value of the variable stored in the referred spot.

Value (default in C++)- “Takes” the value

Converting numbers from base to base

Write 124 in base 3

11121 = 81 + 27 + 9 + 6 + 1

Convert 1011 0100 0000 1010 from binary to hex (skipping the decimal translation we like)

0xB40A

What the heck is *Asymptotic* Complexity?

O(f(n)) measures the *upper bound* on the number of “steps” an algorithm takes to terminate, relative to the size of the input, *n*.

f(n) is your algorithm

O(f(n)) upper bound

Ω(f(n)) lower bound

Θ(f(n)) tight bound

Big-O Cheat Slide

Drop the constant

Only use the “fastest growing” term if they’re separated by addition.

Ex: n^2 grows faster than n.

$$O(n^2+30n + 5\log n) = O(n^2)$$

2^n is very often not a good sign… (ex. $2^{20} = 1,048,576$)

Formal definition

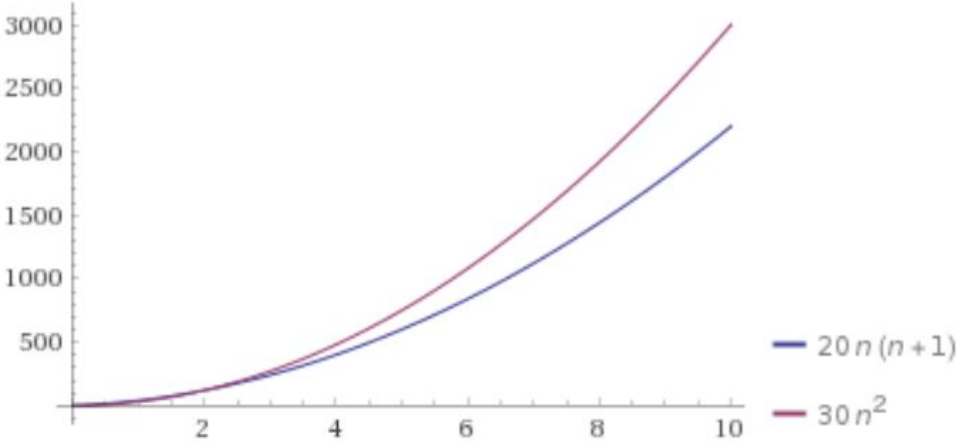
f(n) = O(g(n)) means there are positive constants c and k, such that 0 ≤ f(n) ≤ cg(n) for all n ≥ k.

The values of c and k must be fixed for the function f and must not depend on n.

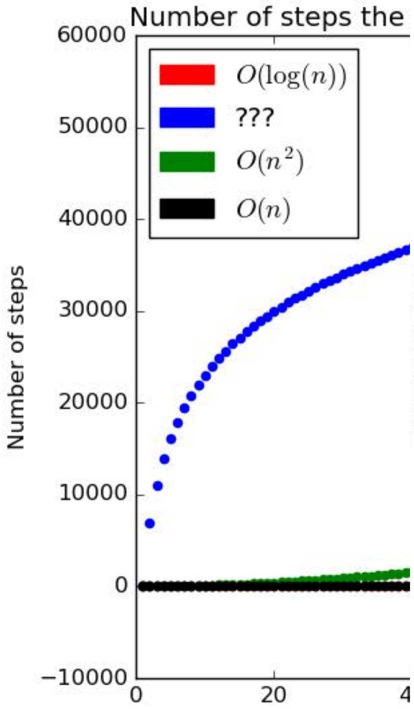
Ex: $f(n) = 20\ n^2 + 20n$, $g(n) = n^2$

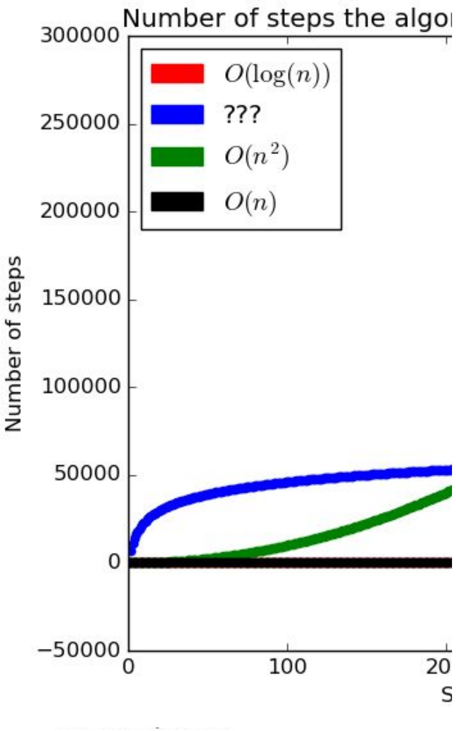
0 ≤ f(n) for all n ≥ 0 ✓

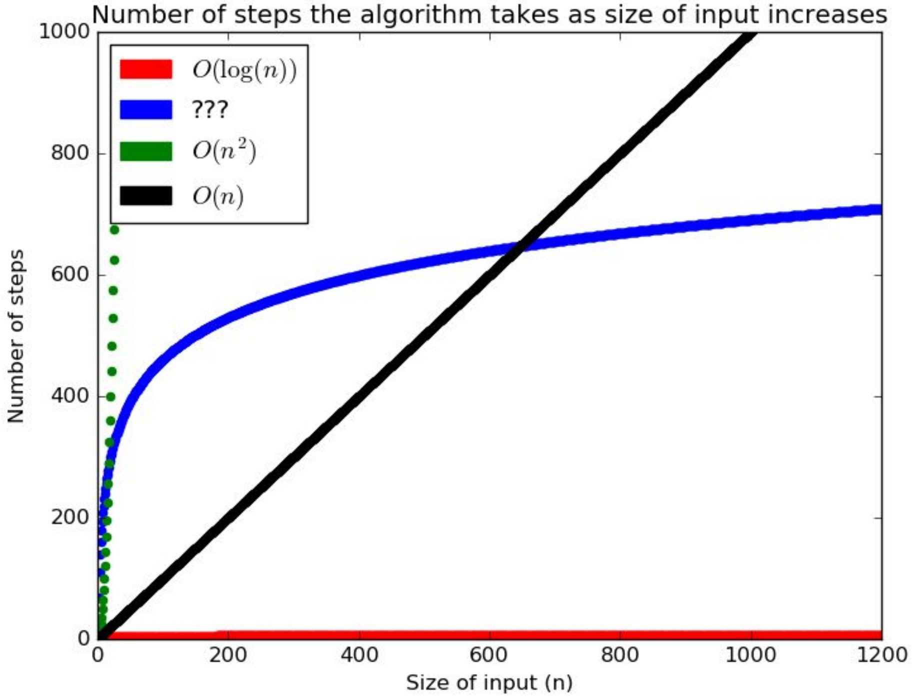
$20\ n^2 + 20\ n \le 30\ n^2$ for all n ≥ 2 ✓



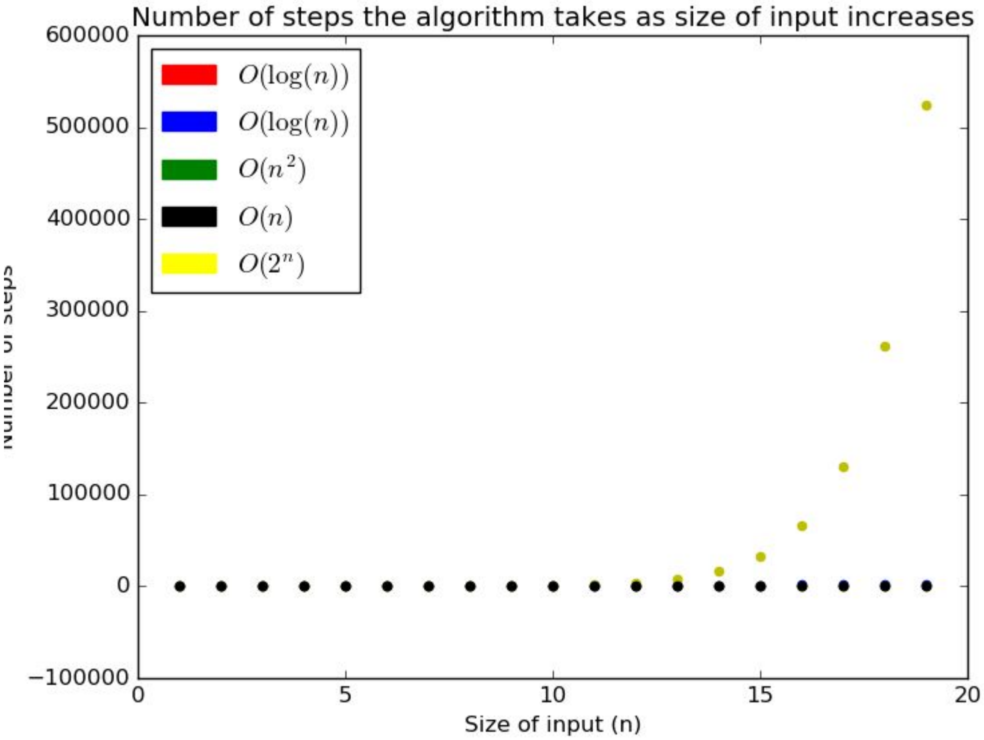
What about constants?







Exponentials are (generally) bad



Case Study: Simplex Algorithm

Maxmize $c^Tx = c_1\ x_1 + c_2\ x_2 + c_3\ x_3 + \dots\ c_n\ x_n$

Subject to Ax <= b (budget constraints)

Simplex: Exponential Algorithm

Known Polynomial algorithm (interior point methods)

Structs / Classes

Initialized with different “properties”

Can have functions associated with them

Recitation Quiz and Survey

Survey Link in moodle with the recitation quiz.