CSCI 200: Foundational Programming Concepts & Design Lecture 17

C++ STL:
`std::vector` & `std::string`

Big-O Notation

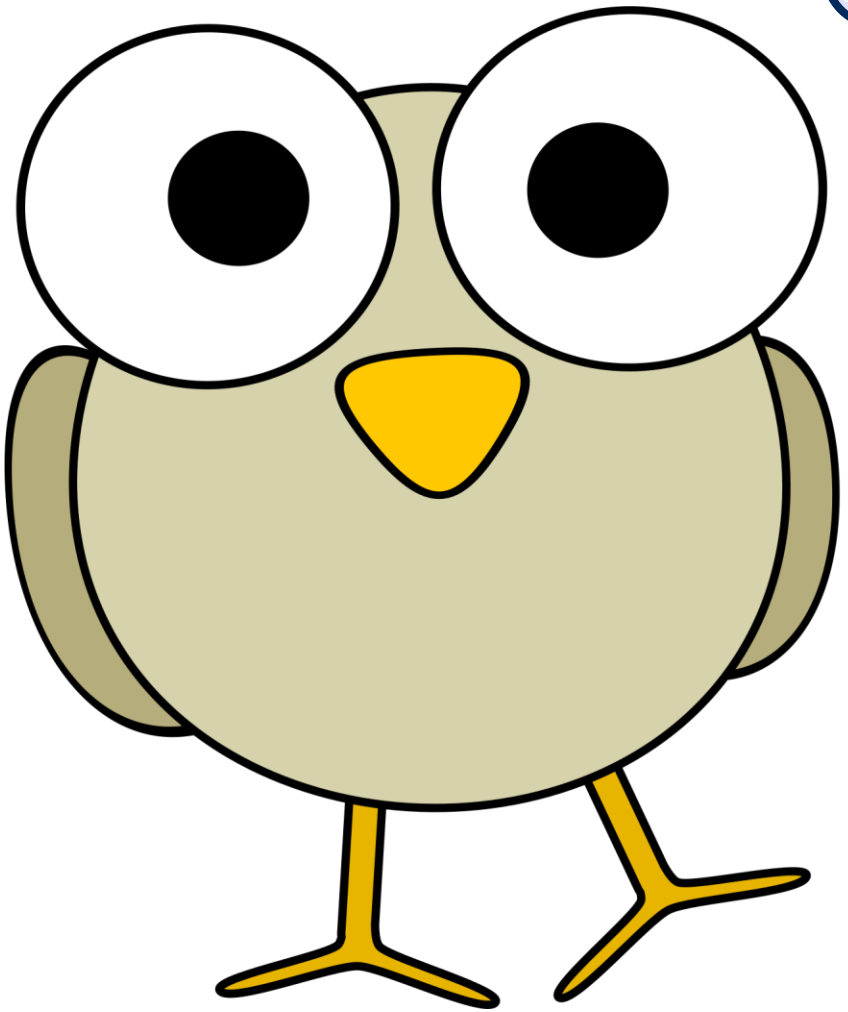
Previously in CSCI 200

- Input validation
  - Check value entered is of proper type
  - Check value entered is of proper range
- Do not assume a smart user
  - Cleanly handle invalid/bad input

Previously in CSCI 200

- **`iomanip`**
  - Used for formatting output
  - **`setw()`** applies only to the next output expression
  - The rest are set until changed

Questions?



Learning Outcomes For Today

- Describe the differences between & advantages of an array & vector.
- Construct a program that accesses an element in a vector, returns the length of a vector, changes the length of the vector, and other vector operations.
- Describe the differences between & advantages of a c-string & string.
- Construct a program that accesses an element in a string, returns the length of a string, changes the length of the string, and other string operations.
- Define Big O Notation and recite the dominance relations.

On Tap For Today

- Vectors
- Strings

- Big-O Notation

- Practice

On Tap For Today

- Vectors
- Strings

- Big-O Notation

- Practice

Vector Class

- `#include <vector>`
- `using namespace std;`

- Example of C++ Standard Template
- Can make a resizable array of any type

Vector Functions

- Numerous member functions defined for vectors
  - www.cplusplus.com/reference/vector/vector

```
pop_back()      front()    size()
push_back()     back()
insert()        at()
erase()
resize()
```

Benefits of Vectors

+ Vectors are safer to use (access protection)
+ Resizing of a vector is possible during runtime
+ Can add to front OR back OR middle of a vector

- Vectors are more cumbersome to use
  - Especially true with multi-dimensional vectors

Vector Example

```
#include <iostream>
#include <vector>
using namespace std;

int main() {

  vector< int > myVec;

  while( true ) {
    cout << "Enter a num (zero to stop): ";
    int x;
    cin >> x;
    if(x > 0) { myVec.push_back( x ); }
    else if (x == 0) { break; }
  }

  cout << "# positive values entered: " << myVec.size() << endl;
  cout << "Values are: ";
  for(size_t i = 0; i < myVec.size(); i++) {
    cout << myVec.at(i) << " ";
  }
  cout << endl;

  return 0;
}
```

On Tap For Today

- Vectors
- Strings

- Big-O Notation

- Practice

String Class

- `#include <string>`
- `using namespace std;`

- Implements concept of a character string
- Can increase/decrease its size dynamically

Using `string`

- Include the string library

```
#include <string>

using namespace std;
```

- Declare a variable of type string

```
string theEvilOne = "Emperor Palpatine";
```

- Use it like any other variable

```
cin >> theEvilOne;

theEvilOne += " lives.";
```

String Documentation

- http://www.cplusplus.com/reference/string/string/

Capacity:

size	Re
length	Re
max_size	Re
resize	Re
capacity	Re
reserve	Re
clear	Cl
empty	Te
shrink_to_fit C++11	Sh

std::string::length

C++98 | C++11

```
size_t length() const;
```

Return length of string

Returns the length of the string, in terms of bytes.

Inputting a string

- Just like any other variable

```
string myStr;

cin >> myStr;
```

Inputting a string

- Just like any other variable

```
string myStr;

cin >> myStr;
```

- But…

Inputting a sentence

- Get a line from cin

```
string myStr;

getline( cin, myStr );
```

Input Options

- Get a single value, excluding whitespace

```
T var;
istream >> var;
```

- Get a single character, including whitespace

```
char oneChar;
oneChar = istream.get();
```

- Get a single line, including space and tab

```
string myStr;
getline( istream, myStr );
```

On Tap For Today

- Vectors
- Strings

- Big-O Notation

- Practice

Vector/String Element Access

- Occurs in constant time - $O(1)$

Runtime Analysis

- Occurs in constant time - $O(1)$

- Uses "Big O Notation"
  - Measures asymptotic complexity of an algorithm
    - How does the function grow with size of *n*?
    - i.e. cost of running the algorithm

Vector Element Access

- Occurs in constant time - $O(1)$
- Given a vector of size *n*, how many elements need to be inspected to return the first element?
  - The last?
  - A random element *i* in [0, *n*)
- The minimum element?
- The maximum element?

Finding the Min/Max Value

- Pseudocode
  - Store the first value of the vector as our current min/max
  - For every element in the vector
    - Min – if an element is smaller than our current min, then that element is our new min
    - Max - If an element is larger than our current max, then that element is our new max

Printing A Vector

- Given a vector of size *n*, how many elements need to be inspected to print the entire vector?

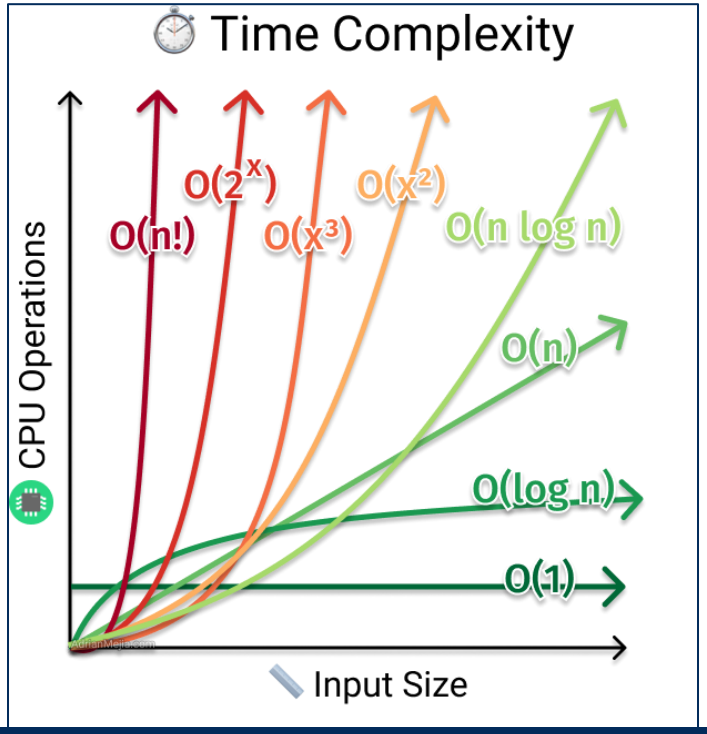
Printing A Vector

- Given a vector of size *n*, how many elements need to be inspected to print the entire vector?

- Occurs in linear time - $O(n)$
  - $O(n) > O(1)$

Big O Dominance Relations

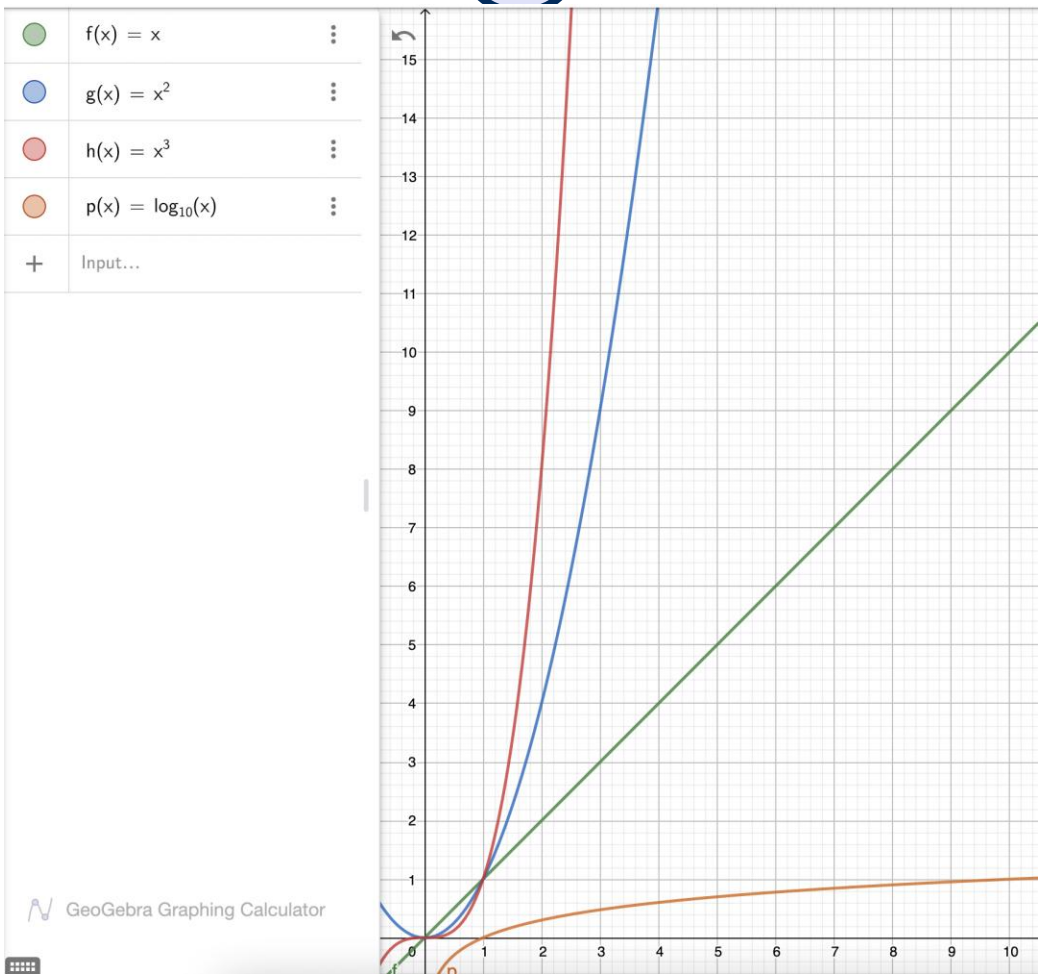
- Higher order polynomials dominate lower order
  - $O(n^n) > O(n!) > O(2^n) > O(n^3) > O(n^2) > O(n \log n) > O(n) > O(\log n) > O(1)$



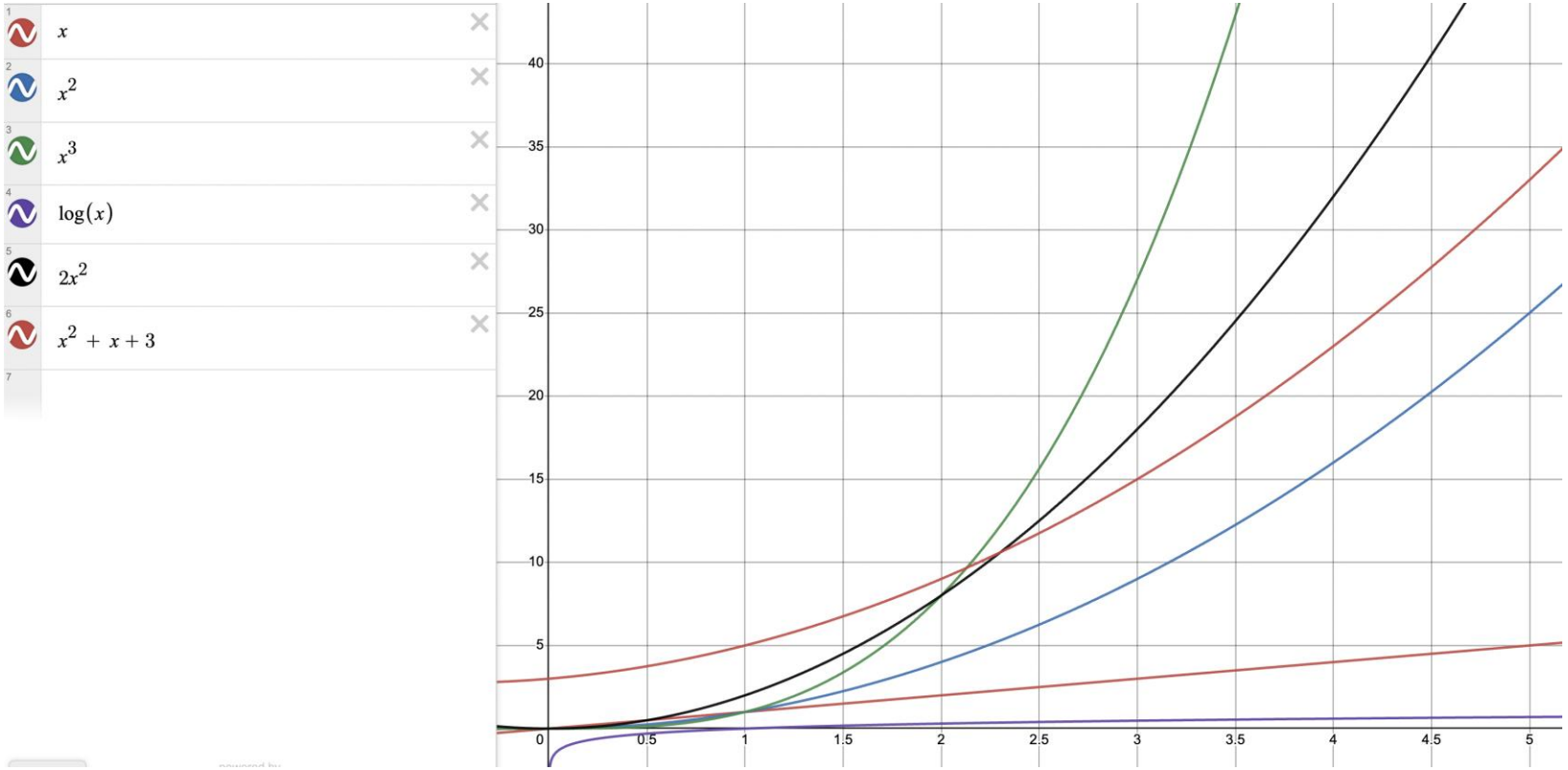
Big O Dominance Relations

- Higher order polynomials dominate lower order
  - $O(n^n) > O(n!) > O(2^n) > O(n^3) > O(n^2) > O(n \log n) > O(n) > O(\log n) > O(1)$
- Therefore, ignore scalar factors and lower order
  - $O(n^2 + n + c) \rightarrow O(n^2)$
  - $O(2n) \rightarrow O(n)$

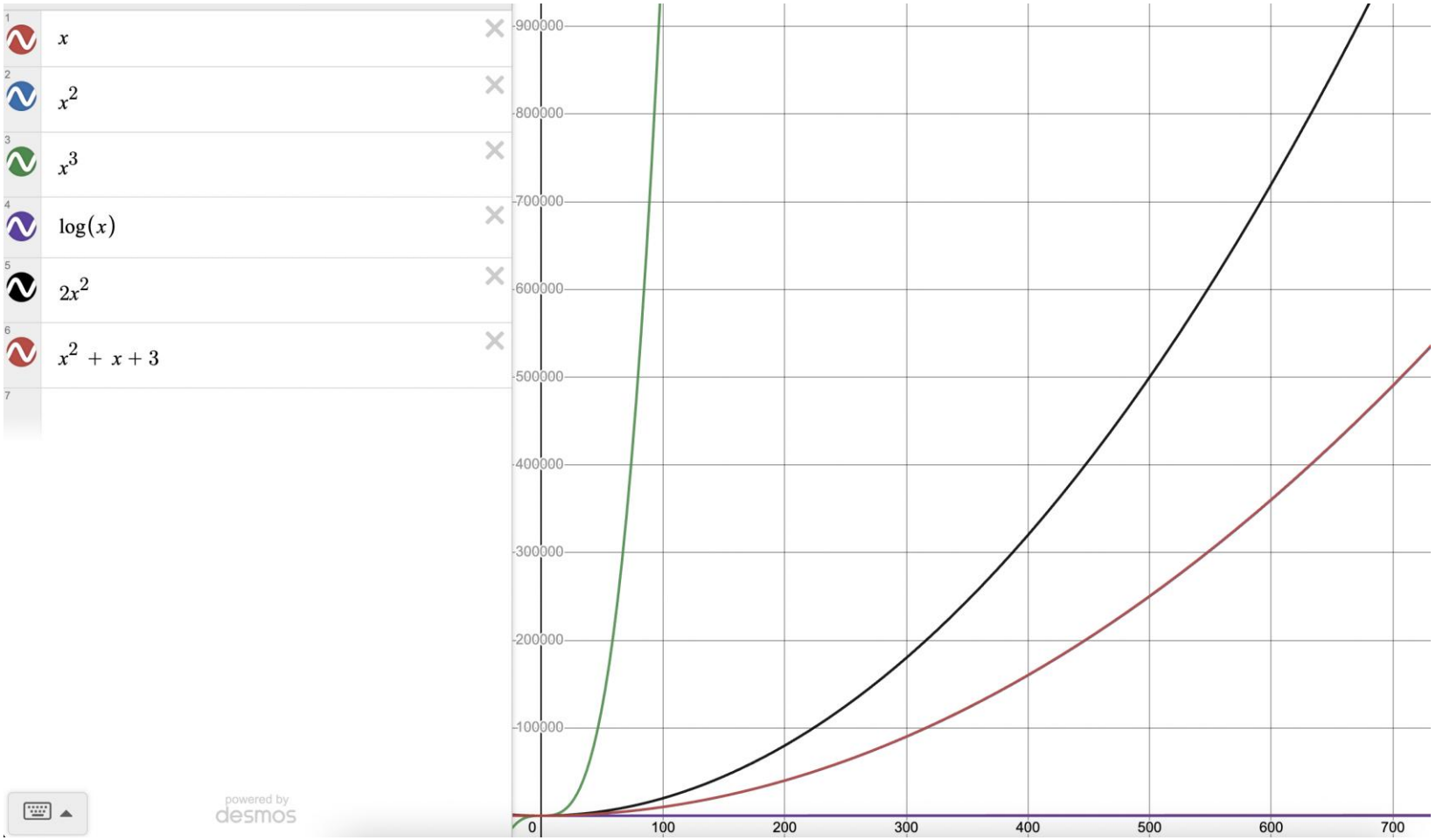
Asymptotic Complexity



Asymptotic Complexity



Asymptotic Complexity



Big O Dominance Relations

- Higher order polynomials dominate lower order
  - $O(n^n) > O(n!) > O(2^n) > O(n^3) > O(n^2) > O(n \log n) > O(n) > O(\log n) > O(1)$
- Therefore, ignore scalar factors and lower order
  - $O(n^2 + n + c) \rightarrow O(n^2)$
  - $O(2n) \rightarrow O(n)$

Vector Operations

- Element Access – O(1)
- Vector Traversal – O(n)

- (Will continue to add to)

On Tap For Today

- Vectors
- Strings

- Big-O Notation

- Practice

To Do For Next Time

- L3B – string test suite
  - Implement functions such that all tests pass
  - Leverage string class documentation!!

- Complete Lecture 17 Post Class Quiz before next class
  - Open notes and resources
  - Graded on correctness