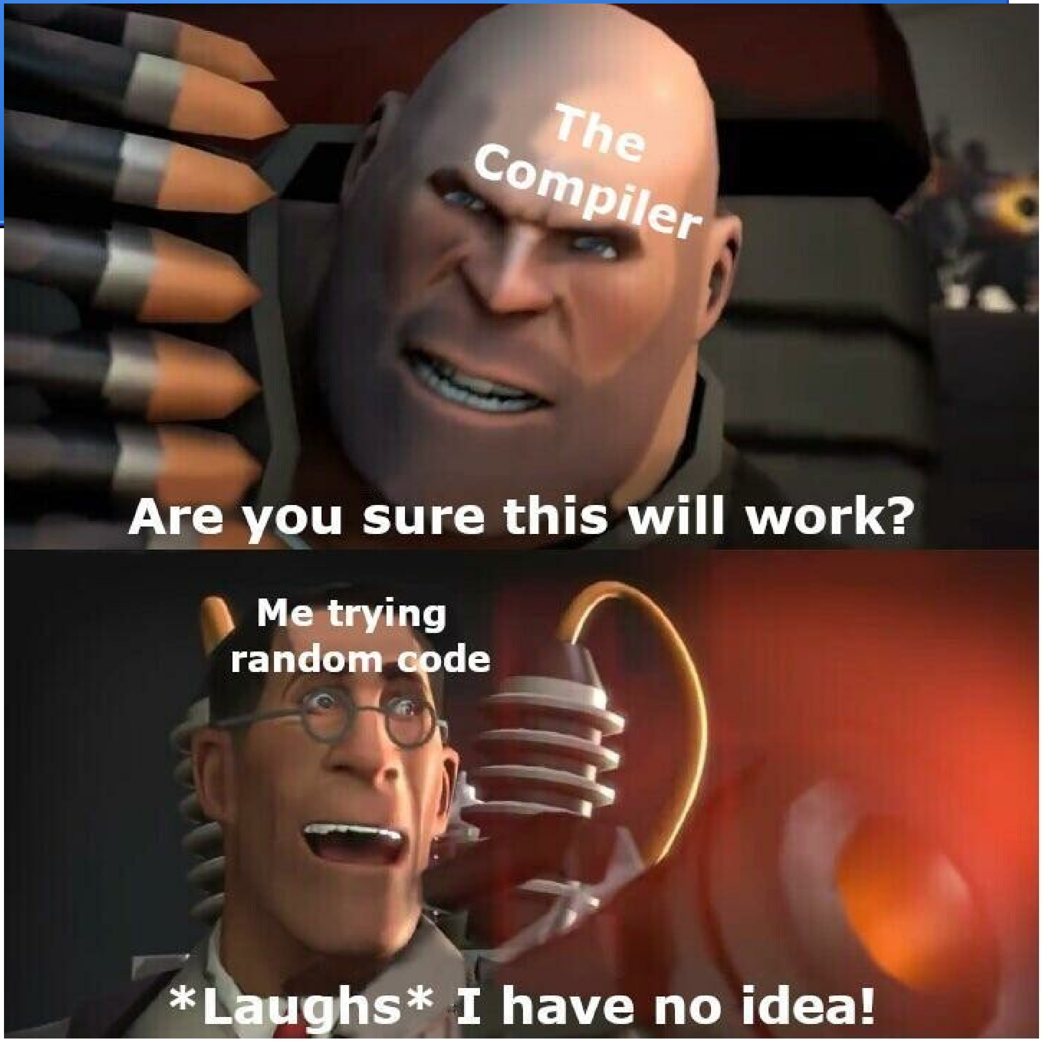
The Compiler
Are you sure this will work?
Me trying random code
Laughs I have no idea!

CS2308
Gentry Atkinson

Lecture 2.1
Algorithmic Analysis

What is an Algorithm

- **Step-by-step instructions that tell a computing agent how to solve some problem using resources.**
- **Resources:**
  - Memory
  - Time (measured as CPU cycles)
- **Type of Instructions:**
  - Sequential (1 action)
  - Conditional (0 or 1 action)
  - Loops (0 or more actions)

Psuedo-Code

- a human-language description of the sequential, conditional, and iterative operations of an algorithm.
- No rigid syntax.
- Clarity, organization, and completeness are important.
- Can be implemented in any computer language.

Example: Find the Largest Number

1. Given a list **L** of integers of size **n**.
2. Set **Largest** to the first value in **L, L0.**
3. For all remaining elements in **L**:
    a) If **Li** is larger than largest:
        i. Set **Largest** to **Li**
4. Output **Largest**

Measuring Time Efficiency

- For this list : **{2, 6, 3, 4, 8}**
  - Step 1 happens 0 times.
  - Step 2 happens 1 time.
  - Step 3a makes 4 comparisons.
  - Step 3ai does 2 assignments.
  - Step 4 happens 0 times.
- For a list with length n=5, our algorithm performs 7 actions.

1. Given a list **L** of integers of size **n**.
2. Set **Largest** to the first value in **L, L0.**
3. For all remaining elements in **L**:
   a) If **Li** is larger than largest:
      i. Set **Largest** to **Li**
4. Output **Largest**

Generalizing This Measure

- For a list with length **n**:
  - Step 0 always happens 0 times.
  - Step 1 always happens 1 time.
  - Step 3a always does n-1 comparisons.
  - Step 3ai will happen at most n-1 times or as little as 0 times.
- Our algorithm will perform at least **n** actions and at most **2n-1** actions.

Big O notation

- It would be cumbersome to tell a colleague that our algorithm uses “n to 2n-1” steps.
- Big O is a simplified notation:
  - Drop all constants
  - Keep only the highest order operation
- Examples:
  - **2n-1 is O(n)**
  - **$n^2 + 5n$ is $O(n^2)$**
  - **1000000 is O(1) or O(c)**

Orders of Big O

- $O(1)$ constant time
- $O(\log(n))$ log time
- $O(n)$ linear time
- $O(n\log(n))$ log linear time
- $O(n^2)$ polynomial time
- $O(2^n)$ exponential time
- $O(n!)$ factorial time

Fastest

Slowest

Comparison of Orders

Let’s assume that one operation takes one second.

	2 step	8 step	32 step	128 step	512 step	2048 step	8192 step
constant	1 sec.	1 sec.	1 sec.	1 sec.	1 sec.	1 sec.	1 sec.
log(n)	1 sec.	3 sec.	5 sec.	7 sec.	9 sec.	11 sec.	13 sec.
n	2 sec.	8 sec.	32 sec.	2 min.	8 min.	34 min.	2 hours
nlog(n)	2 sec.	24 sec.	2 min.	14 min.	1 hour	6 hours	1 day
n^2	4 sec.	1 min.	17 min.	4 hours	3 days	48 days	2 years
2^n	4 sec.	1 min.	136 years	1e31 years	...	...	...
n!	2 sec.	11 min.	8e27 years	1e208 year	...	...	...

2^{nd} Example: Find the most frequent number

1) Given a list of integers **L** of length **n**
2) Create lists **V** and **C** of length **n**
3) For every value in **L**:
   a) Try to find Li in V
      i. If Li is stored at Vj, increment Cj
      ii. Otherwise set Vm to Li and set Cm to 1 where m is the next empty position in the two lists.
4) Find the index k of the maximum value in **C**
5) Output Vk

Space Complexity

- We are given the list **L,** it doesn't count against the space complexity.
- We create 2 lists **V** and **C** which both have size **n**.
- To process **n** values, we need to store **2n** values in memory.
- Out algorithm is **O(n)** space complexity.

1) Given a list of integers **L** of length **n**
2) Create lists **V** and **C** of length **n**
3) For every value in **L**:
    a) Try to find Li in V
        i. If Li is stored at Vj, increment Cj
        ii. Otherwise set Vm to Li and set Cm to 1 where m is the next empty position in the two lists.
4) Find the index k of the maximum value in **C**
5) Output Vk

Big O Considerations

- Not meaningful for small values of n. The Big O measurement only needs to be true after some **threshold.**
- We have assumed so far that our computer only has one processor.
- Big O is an **upper bound** so a linear time algorithm is also in the set of polynomial time algorithms.

Questions or Comments?