CS 008
Lecture notes
3/19/24

1 Outline

- Announcements
- Template classes
- Data structures & algorithms
  - Big-O notation
  - Stacks
  - Queue

2 Announcements

Exam:

Exam is next Thursday (3/28/24)

3 Template classes

As the name implies, the idea of a **template** class unlike a regular class would be the ability to pass in the data type as a parameter so that there is no need to create the same class for different parameters.

4 Big-O notation review

Consider a function: $f(n) = a_j n^j + a_{j-1} n^{j-1} + ... + a_1 n + a_0$, what would be the complexity time of this function? (If all variables of a are constants)

Looking at the function, you only consider the **largest** term in the function (which would be $a_j n^j$) and then drop the constant resulting in a time complexity of $O(n^j)$.

Now, consider the same function but with a small change: $f(n) = a_j n^j + a_{j-1} n^{j-1} + ... + a_1 n + a_0 + b \log(n)$. The complexity class of the function given is still $O(n^j)$ since the largest term has not changed. However, if b is a constant of 10^{20}! (a very large number), what would happen to the complexity time?

It will still be $O(n^j)$.

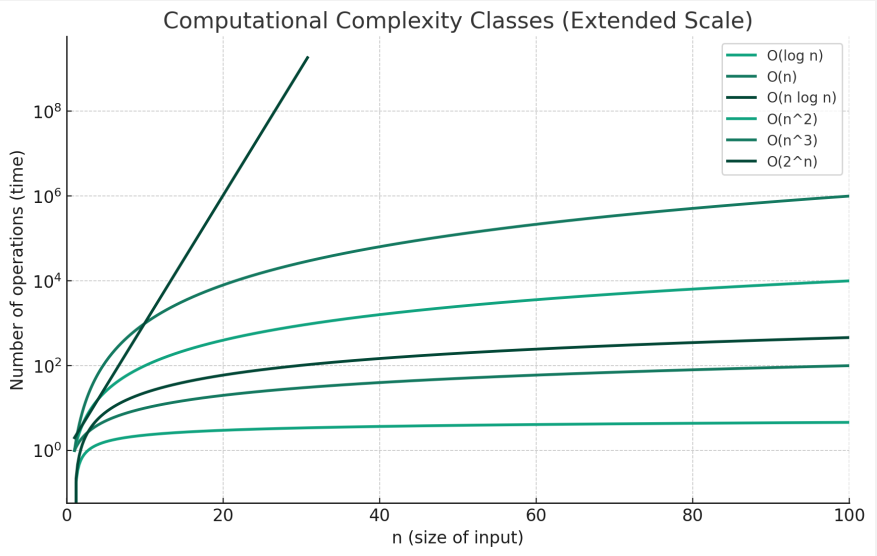
Question:

If we added $c*2^n$ to the end of the function, would the time complexity change?

Answer: Yes, at a large enough value for n, the value of $c * 2^n$ would exceed the largest term in the original function.

Big-O classes represent *upper bounds* on runtime.

Figure 1:



Note:

Sometimes, you might encounter something like $f(n) = O(n)$ however, a better way to describe a function that has a certain time complexity would be: $f(n) \in O(n)$.

Definition:

If $f(n) \in O(n^k)$, then there exists an integer N and a real number G which depends on N such that for all values of $n \geq N$: $f(n) \leq G * n^k$.

5 Stacks

A **stack** is an abstract data type and is a 1-dimensional container class containing many objects of one certain type which can only add and remove from one end and is known as **LIFO** (*last in, first out*). Consider a stack of books, intuitively we know that a stack of books would need to be removed from the top to bottom. New books would be added to the top and will also be the first ones to be removed.

A **stack** contains a few important functions:

- pop(): remove top item
- push(): add item to top
- top(): view top item
- empty(): check if stack is empty
- size(): check the size of the stack

Question:

What's the difference between an **abstract data type** and a **data structure**?

Answer: A data structure is an implementation of an abstract data type.

Note:

C++ has a built-in stack (include stack) that already exists. *insert example code*

Possible implementations of a stack:

- static-length array
- dynamic array
- linked-list