

For Friday

- Finish Weiss chapter 3
 - The concepts here should be review. If you're struggling with the C++ aspects, you may wish to refer to Savitch, chapter 17.
- Homework:
 - Weiss, chapter 2, exercises 7, 11(a,c,d), 12(a,c,d)
 - Use the UNIX time command for exercise 7

Big Oh Notation

- $f(n) = O(g(n))$ iff positive constants c and n_0 exist such that $f(n) \leq cg(n)$ for all $n, n \geq n_0$.
- Provides an upper bound for function f .
- Standard functions used for g :

1	constant
$\log n$	logarithmic (base irrelevant)
n	linear
$n \log n$	$n \log n$
n^2	quadratic
n^3	cubic
2^n	exponential
$n!$	factorial

Finding Big Oh Bounds

- $3n + 2$
- $100n + 10$
- $10n^2 + 4n + 2$
- $6 * 2^n + n^2$
- 9
- 2045

Omega Notation (Ω)

- Lower bound analog of big oh notation.
- Definition:
 $f(n) = \Omega(g(n))$ iff positive constants c and n_0 exist such that $f(n) \geq cg(n)$ for all $n, n \leq n_0$.

Theta Notation (Θ)

- Theta notation can be used when a function can be bounded from above **and** below by the same function.

Little oh notation

- Little oh notation (o) applies when an upper bound is not also a lower bound.

Computing Running Times

- Rule 1: For loops
 - The running time is at most the running time of the statement inside the for loop (including tests) times the number of iterations.
- Rule 2: Nested loops
 - Analyze inside out.

- Rule 3: Consecutive statements
 - Add these (meaning the maximum is the one that counts)
- Rule 4: if/else
 - Running time of an if/else statement is no larger than the time of the test plus the larger of the running times of the bodies of the if and else.
- In general: work inside out.

Abstract Data Type

- A specification of a data type, including the operations of the data type
- Describes data items and operations in an **implementation-independent** way
- Can be implemented as a C++ class
- Could also be implemented as a set of variables and associated functions in C or another language

Linear List ADT

- AbstractDataType *LinearList*{
 - instances (or data)
 - ordered finite collection of zero or more elements
 - operations
 - Create()
 - Destroy()
 - IsEmpty()
 - Length()
 - Find(index) // returns an element
 - Search(key) // returns an element
 - DeleteAt(index)
 - DeleteValue(key)
 - Insert(index,element)
 - Output()}

Using an ADT

- ADTs are not directly usable
- They must be implemented
- Most ADTs can be implemented in more than one way
 - What would be different ways to represent the linear list ADT?

Linked Lists

- What's the basic concept?

What Is a Node?

- Two parts
 - Data
 - Pointer to the next one
- Make the data public
- Do provide a constructor with appropriate defaults

Creating Nodes

- We'll always create nodes dynamically.
- Note that we almost never actually work with a node itself; we use pointers to the nodes.
- Important:
 - ALWAYS initialize next to NULL when a node is created unless you are immediately assigning it another meaningful value.

C++ Details

- Deleting nodes . . .

Linked List Variations

- Doubly-linked lists
- Empty head node