CSS 342

Dr. Carol Zander

April 2, 2007

Functional abstraction: separation of algorithm from syntax.

Modular abstraction: isolating functional components that communicate over public interfaces

ADT: a description of a collection of data and its methods

Data structure: the implementation of an ADT

OO: Abstraction, polymorphism, inheritance, encapsulation

OO: Interaction of ADTs

Interface: contract

Class: user-defined type

Defines the memory layout at instantiation

Like a blueprint

April 4, 2007

Object: instance of a class

Laid out in memory according to the class definition

April 9, 2007

STL – standard since 1994

Vector

Lists

Deque

Sets

Multisets

Map

Multimap

Stack

Queue

PriorityQueue

Sequence containers – O(n)

Associative containers – O(log n)

Interator – traverse a collection (design pattern)

Forward

Bidirectional

Random access

List ADT:

Definition 1: Finite sequence of zero or more items (Collection)

Definition 2: An ordered collection of Definition 1

Methods:

Insert

Remove

Retrieve/find/lookup/locate/search

Display

isEmpty

getcount

merge

getFirst

getLast

sort

Implementation:

Arrays

Linked lists

April 16, 2007

Choosing an algorithm:

Memory consumption

Bandwith consumption

I/O requests

CPU consumption, AKA running time

Why consider efficiency?

Helps you choose an optimal solution for the dataset

Complexity analysis allows programmer to consider general perf case instead of specific case

Predict algorithm performance

Determine expensive code blocks at runtime

Growth rates: analyze algorithm running time without knowing input details

Running time, T(n):

Number of time units required to execute a code block over a sufficiently large dataset

Vocab:

Algorithm analysis/complexity/order/running time/efficiency/Big O/Big Theta/Big Omega

O(1): constant

O(n): linear

O(n^2): quadratic

O(n^3) cubed

O(2^n): exponential

O(log n): logarithmic

T(n) = O(f(n)):

Proof: There exists 2 positive constants n0, c > 0, such that T(n) <= cf(n) for all n >=n0.

Prove if T(n) = 3n^2 + 5n – 4, then T(n) = O(n^2)

Goal: 3n^2 + 5n – 4 <= cn^2

Steps:

3n^2 + 5n – 4 <= 3n^2 + 5n

3n^2 + 5n – 4 <= 3n^2 + 5n^2, n >= 1

3n^2 + 5n – 4 <= 8n&2, n >= 1

Let C = 8, n0 = 1, then

3n^2 + 5n – 4

8n^2 >= 2n^2 + 5n^2

Find the complexity of the following:

sum = 0;

for (int i = 0; i < n; i++)

{

sum += a[i];

}

T(n) = O(1 +