Welcome to
Data Structures and Algorithms

Course description

- Algorithms and data structures emphasizes the following topics: data structures, abstract data types, recursive algorithms, algorithm analysis, sorting and searching, and problem-solving strategies. Labs alternate weeks.

Course objectives

- Introduce the student to the concept of data structures through abstract data structures including lists, sorted lists, stacks, queues, deques, sets/maps, directed acyclic graphs, and graphs; and implementations including the use of linked lists, arrays, binary search trees, *M*-way search trees, hash tables, complete trees, and adjacency matrices and lists.
- Introduce the student to algorithms design including greedy, divide-and-conquer, random and backtracking algorithms and dynamic programming; and specific algorithms including, for example, resizing arrays, balancing search trees, shortest path, and spanning trees.

Suggested texts and readings

- Cormen, Leiserson, Rivest, and Stein (CLRS), Introduction to Algorithms, 2nd Ed., MIT Press, 2001.

- Algorithm Design Book by Jon Kleinberg and Éva Tardos March 16, 2005Algorithm Design Book by Jon Kleinberg and Éva Tardos March 16, 2005

General overview of the topics

- Review of Mathematics and C++
- Asymptotic and Algorithm Analysis
  - Properties of data
  - Asymptotic Analysis
  - Algorithm Analysis
- Abstract Lists and Implementations
  - Linked lists and arrays
  - Stacks
  - Queues
  - Deques
- Abstract Sorted Lists and Implementations
  - General trees, binary (including binary and complete trees), *N*-ary trees, and tree traversals
  - Abstract Sorted Lists
  - Binary search trees
  - Balanced search trees
  - AVL trees
  - B-Trees
- Abstract Priority Queues
  - Heaps

General overview of the topics

- Abstract Sets/Maps
  - Chained Hash Tables
  - Linear Probing
  - Double Hashing
- Sorting Algorithms
  - Insertion and bubble sort
  - Heap, merge, and quick sort
  - Bucket and radix sort
- Graph and Direct Acyclic Graph Algorithms
  - Topological sort
  - Minimum spanning trees
  - Shortest path
- Algorithm Design
  - Greedy algorithms
  - Divide-and-conquer algorithms
  - Dynamic programming
  - Randomized algorithms
  - Backtracking algorithms
  - NP Completeness, Turing machines, and the halting problem
- Example of an advanced data structure

Data Structures and Algorithms

In this course, we will look at:

- *Algorithms* for solving problems efficiently
- *Data structures* for efficiently storing, accessing, and modifying data

We will see that all data structures have trade-offs

- There is no *ultimate* data structure...
- The choice depends on our requirements

Data Structures and Algorithms

Consider accessing the k^{th} entry in an array or linked list

- In an array, we can access it using an index `array[k]`
  - there is a single machine instruction for this
- We must step through the first $k - 1$ nodes in a linked list

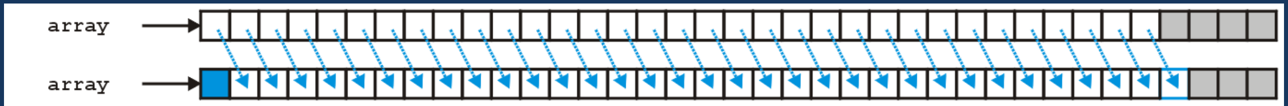
Consider searching for an entry in a sorted array or linked list

- In a sorted array, we use a fast binary search
  - Very fast
- We must step through all entries less than the entry we’re looking for
  - Slow

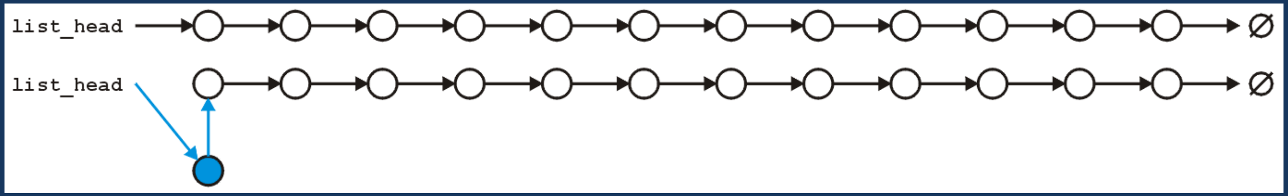
Data Structures and Algorithms

However, consider inserting a new entry to the start of an array or a linked list

- An array requires that you copy all the elements in the array over
  - Slow for large arrays

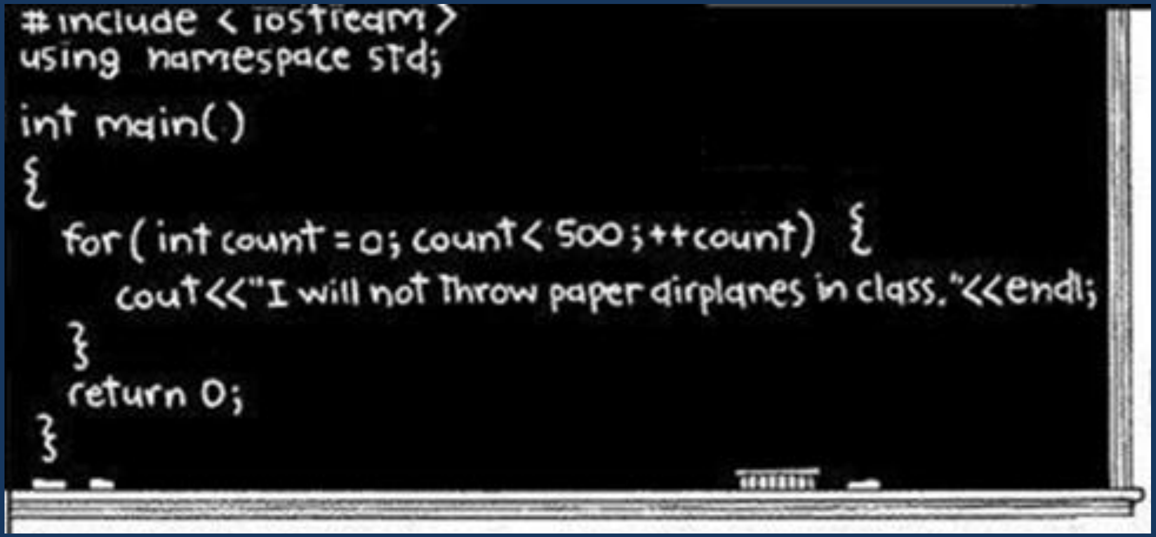


- A linked list allows you to make the insertion very quickly
  - Very fast regardless of size



C++

You will be using the C++ programming language in this course



C++

This course does not teach C++ programming

- You will use C++ to demonstrate your knowledge in this course

One lecture may be covered:

- About the Features of C++

Evaluation

Your evaluation in this course is based on three components:

- Assignments
- Quizzes (Mostly Announced)
- One mid-term examination
- One final examination
- Project