Course goals

- exposure to another language
  - C++
  - Object-oriented principles
- knowledge of specific data structures
  - lists, stacks & queues, priority queues, dynamic dictionaries, graphs
- impact of DS design & implementation on program performance
  - asymptotic complexity of algorithms

Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment will be introduced concurrently with the study of these topics, as appropriate

- Review of C++
- Introduction to Unix
- Review of program performance
  - time and space complexity
  - asymptotic notation
  - -- *searching* (linear vs binary) & *sorting* (insertion sort vs mergesort)
- Data representation and lists
- Stacks and Queues
- Hash tables
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
  - Heaps
- Search trees
  - Binary search trees
  - balanced binary search trees – AVL trees
- Graphs
  - representation
  - traversal

Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment will be introduced concurrently with the study of these topics, as appropriate

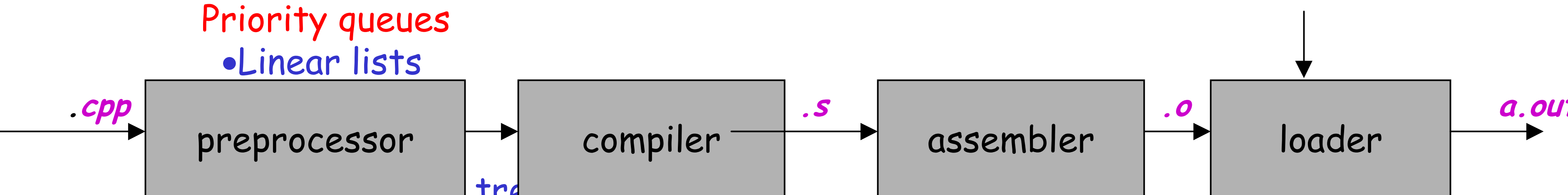
- Review of C++
  - objects
  - classes -- .h and .cpp files
  - templates
  - access control
    - public/ private/ protected methods
    - friend classes
  - inheritance
    - public/ private/ protected inheritance
    - multiple inheritance
  - the strings package
- Introduction to Unix
- Review of program performance
  - time and space complexity
  - asymptotic notation
  - -- *searching* (linear vs binary) & *sorting* (i
- Data representation and lists
- Stacks and Queues
- Hash tables
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
  - Heaps
- Search trees
  - Binary search trees
  - balanced binary search trees – AVL trees
- Graphs
  - representation
  - traversal

Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment will be introduced concurrently with the study of these topics, as appropriate

- Review of C++
- Introduction to Unix
- Review of program performance
  - time and space complexity
  - asymptotic notation
  - -- *searching* (linear vs binary) & *sorting* (i
- Data representation and lists
- Stacks and Queues
- Hash tables
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
  - balanced binary search trees – AVL trees
- Graphs
  - representation
  - traversal

- man pages
- the g++ compiler
  - stages in compilation
- makefiles
- environment variables
- the gdb debugger
- emacs? pico



Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment will be introduced concurrently with the study of these topics, as appropriate

- Review of C++
- Introduction to Unix
- Review of program performance
  - time and space complexity
  - asymptotic notation
  - -- *searching* (linear vs binary) & *sorting* (insertion sort vs mergesort)
- Data representation and lists
- Stacks and Queues
- Hash tables
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
  - Heaps
- Search trees
  - Binary search trees
  - balanced binary search tr
- Graphs
  - representation
  - traversal

- bigOh/ bigTheta notation
- asymptotic worst-case complexity of algorithms
- common complexities:
  - log n
  - n
  - n log n
  - n^2, n^3, ...
- determining complexities of algorithms
- example complexities -- sort/ search

Example: merge sort

```
mergeSort(A, i, j) // sort A[i,...j]
{
      if (i==j) return A[];
      mergeSort(A, i, (i+j)/2);
      mergeSort(A, (i+j)/2 + 1, j);
      merge(A, i, (i+j)/2, j)
}
```

```
merge(A, i, k, j)
//PreCond: A[i,...,k] and A[k,...,j] are sorted
//PostCond:A[i,...,j] is sorted
```

Recurrence : $T(n) \leq 2 \,.\, T(n/2) + c_1 \,.\, n + c_2$

$T(1) = c_3$

$T(n) = O(n \log n)$

Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment will be introduced concurrently with the study of these topics, as appropriate

- Review of C++
- Introduction to Unix
- Review of program performance
  - •time and space complexity
  - •asymptotic notation
  - -- *searching* (linear vs binary) & *sort*
- Data representation and lists
- Stacks and Queues
- Hash tables

- data representation:
  - array-based
  - linked/ pointer-based
  - simulated pointer (cursors)
- lists
  - ADT specification
  - representation using arrays
  - representation using linked lists
  - compare and contrast

```
class list{
  public:
     list();
     ~list();
     bool isEmpty();
     bool isFull();
     int length();
     bool Find(x,k);
     int Search(x);
     void delete(k,x);
     void insert(k,x);
  private:
};
```

VL

```
adt linearList{
     create()
     destroy()
     isEmpty()
     isFull()
     length()
     Find(x,k)
     Search(x)
     delete(k,x)
     insert(k,x)
}
```

Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment will be introduced concurrently with the study of these topics, as appropriate

- Review of C++
- Introduction to Unix
- Review of program performance
  - time and space complexity
  - asymptotic notation
  - -- searching (linear vs binary) & sort
- Data representation and lists
- Stacks and Queues
- Hash tables
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
  - Heaps
- Search trees
  - Binary search trees
  - balanced binary search trees – AVL trees
- Graphs
  - representation
  - traversal

- data representation:
  - array–based
  - linked/ pointer–based
  - simulated pointer (cursors)
- lists
  - ADT specification
  - representation using arrays
  - representation using linked lists
  - compare and contrast

Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment will be introduced concurrently with the study of these topics, as appropriate

- Review of C++
- Introduction to Unix
- Review of program performance
  - time and space complexity
  - asymptotic notation
  - -- *searching* (linear vs binary) & *sort*
- Data representation and lists
- Stacks and Queues
- Hash tables
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
  - Heaps
- Search trees
  - Binary search trees
  - balanced binary search trees – AVL trees
- Graphs
  - representation
  - traversal

- ADT specification
  - stack– LIFO
  - queue – FIFO
  - (dequeue)
- implementation
  - representation using arrays
    - “circular” for queues
  - representation using linked lists
  - $\Theta(1)$ time operations
  - min and nextMin operations

Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment will be introduced concurrently with the study of these topics, as appropriate

- Review of C++
- Introduction to Unix
- Review of program performance
  - time and space complexity
  - asymptotic notation
  - -- *searching* (linear vs binary) & s
- Data representation and lists
- Stacks and Queues
- Hash tables
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
  - Heaps
- Search trees
  - Binary search trees
  - balanced binary search trees – AVL trees
- Graphs
  - representation
  - traversal

- a **recursive** definition
  - root
  - left [sub]tree
  - right [sub]tree
- implementation
  - representation using arrays
    - inefficient, except for complete trees
  - representation using linked structures
  - O(h) time operations (h: height of the tree)
- tree **traversals** -- recursively defined
  - preorder/ inorder/ postorder
  - each takes O(n) time (n: # elements)

Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment will be introduced concurrently with the study of these topics, as appropriate

- Review of C++
- Introduction to Unix
- Review of program performance
  - time and space complexity
  - asymptotic notation
  - -- *searching* (linear vs binary) & *sorting* (insertion sort vs mergesort)
- Data representation and lists
- Stacks and Queues
- Hash tables
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
  - Heaps
- Search trees
  - Binary search trees
  - balanced binary search tre
- Graphs
  - representation
  - traversal

- ADT specification
  - create/ destroy/ isEmpty
  - insert
  - min
  - deleteMin
- implementation
  - linear list -- one of the operations is O(n)
  - binary tree -- a *complete* tree
    - represented using array
      - O(log n) operations
      - fast implementations (bit-manipulation)
- other operations --
  - max
  - decrease/ increase
  - delete

Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment
of these topics, as appropriate

Review of C++

Introduction to Unix

Review of program performa

- time and space complexity
- asymptotic notation

-- *searching* (linear vs binar

Data representation and lists

Stacks and Queues

Hash tables

Binary trees

- representation
- traversal

Priority queues

- Linear lists
- Heaps

Search trees

- Binary search trees
- balanced binary search trees – AVL trees

Graphs

- representation
- traversal

- dynamic dictionaries -- ADT
  - create/ destroy
  - insert
  - delete
  - find
- implementation using binary trees
  - bst's --operations are O(h)
    - inorder traversal sorts the elements
  - <u>balanced</u> bst's -- the AVL tree
    - height is always O(log n)
    - insert/ delete may involve rotations
      - RR/ LL/ RL/ LR

Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment
of these topics, as appropriate

Review of C++

Introduction to Unix

Review of program performa

- time and space complexity
- asymptotic notation
- -- *searching* (linear vs binar

Data representation and lists

Stacks and Queues

Hash tables

Binary trees

- representation
- traversal

Priority queues

- Linear lists
- Heaps

Search trees

- Binary search trees
- balanced binary search tre

Graphs

- representation
- traversal

- dynamic dictionaries -- ADT
  - create/ destroy
  - insert
  - delete
  - find

- implementation using binary trees
  - bst's --operations are O(h)
    - inorder traversal sorts the elements
  - balanced bst's -- the AVL tree
    - height is always O(log n)
    - insert/ delete may involve rotations
      - RR/ LL/ RL/ LR

- implementation: representation using arrays as tables
  - a hash function maps *keys* to *buckets*
  - collisions may result in overflow
  - handling overflows:
    - open addressing
      - linear probing
      - quadratic probing
    - chaining
  - performance: worst-case O(n), average-case O(1)

Course outline

Features of C++, object-oriented programming principles, and features of the Unix programming environment will be introduced concurrently with the study of these topics, as appropriate

- Review of C++
- Introduction to Unix
- Review of program performance
  - time and space complexity
  - asymptotic notation
  - -- *searching* (linear vs binary) & s
- Data representation and lists
- Stacks and Queues
- Hash tables
- Binary trees
  - representation
  - traversal
- Priority queues
  - Linear lists
  - Heaps
- Search trees
  - Binary search trees
  - balanced binary search trees – AVL trees
- Graphs
  - representation
  - traversal

- definition: $G=(V,E)$, $|V| = n$; $|E|=m$;
  - lots of terminology
- representation
  - adjacency matrices
  - adjacency lists
  - **compare and contrast**
- example operations
  - traversals
    - depth first (DFS)
    - breadth–first (BFS)
  - topological sort of DAG's
  - cycle detection
    - directed and undirected graphs
  - shortest paths
    - the Warshall–Floyd algorithm