ECE 242
Data Structures and Algorithms

http://www.ecs.umass.edu/~polizzi/Teaching/ECE242/

Epilogue
Lecture 30
Prof. Eric Polizzi
Every computer program uses **data structures and algorithms**.

### Data Structures

**Why?** organize your data in computer's memory to efficiently store and retrieve **information**

**How?** using arrays, linked lists, stacks, queues, trees, matrix, etc.

### Algorithms

**Why?** manipulate the data in various way to perform **computation**

**How?** using strategy/method, operations and analysis → Importance of Big O notation

### Data-centric view of the world

- **Complex data structures** (from arrays to trees)
- **Basic 'non-numerical' algorithms act on database** (insert, remove, search, sort, traverse)

### Scientific Computing

- **Basic data structure**
- **Complex numerical algorithms** (computational modeling, HPC)
DATA STRUCTURES

Array
(1d, 2d, etc.)

Linked List
(simple, doubly)

Unsorted list

Sorted list

Hash-Tables

Trees
(BST, CBT, RBT)

Stacks

Queues

Heap
(Priority Q)

Graphs

ALGORITHMS

Insert
Remove
Search
Sort
Traverse

Pop, Push, dequeue, enqueue, etc.
General Purpose Data Structures

- Motivations
  - Store real-world data: personal records, inventories, media library, etc.

- Some comparisons between **general purpose** data structures

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Search</th>
<th>Insertion</th>
<th>Deletion</th>
<th>Traversal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>O(N)</td>
<td>O(1)</td>
<td>O(N)</td>
<td>—</td>
</tr>
<tr>
<td>Ordered array</td>
<td>O(logN)</td>
<td>O(N)</td>
<td>O(N)</td>
<td>O(N)</td>
</tr>
<tr>
<td>Linked list</td>
<td>O(N)</td>
<td>O(1)</td>
<td>O(N)</td>
<td>—</td>
</tr>
<tr>
<td>Ordered linked list</td>
<td>O(N)</td>
<td>O(N)</td>
<td>O(N)</td>
<td>O(N)</td>
</tr>
<tr>
<td>Binary tree (average)</td>
<td>O(logN)</td>
<td>O(logN)</td>
<td>O(logN)</td>
<td>O(N)</td>
</tr>
<tr>
<td>Binary tree (worst case)</td>
<td>O(N)</td>
<td>O(N)</td>
<td>O(N)</td>
<td>O(N)</td>
</tr>
<tr>
<td>Balanced tree (average and worst case)</td>
<td>O(logN)</td>
<td>O(logN)</td>
<td>O(logN)</td>
<td>O(N)</td>
</tr>
<tr>
<td>Hash table</td>
<td>O(1)</td>
<td>O(1)</td>
<td>O(1)</td>
<td>—</td>
</tr>
</tbody>
</table>
General Purpose Data Structures

- Tentative 'Guideline' on When to use What?
  - **Array**
    - Easy to implement, data locality
    - Static size
  - **Linked-list**
    - Still easy to implement
    - Dynamic size
  - **BST**
    - Balanced (difficult to implement)
    - Unbalanced (great if data inserted in random order)
  - **Hash-Tables**
    - Optimal for search, insert
    - No Traversal
    - Array-based (static size)
<table>
<thead>
<tr>
<th>Sorting Algorithm</th>
<th>Complexity</th>
<th>Additional Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bubble Sort</td>
<td>$O(N^2)$</td>
<td>$O(N^2)$ swaps, Slow, slow, slow</td>
</tr>
<tr>
<td>Selection Sort</td>
<td>$O(N^2)$</td>
<td>$O(N^2)$ swaps, Intuitive but still slow</td>
</tr>
<tr>
<td>Insertion Sort</td>
<td>$O(N^2)$</td>
<td>$O(N^2)$ swaps, half #comp. than Bubble, O(N) total if data almost sorted</td>
</tr>
<tr>
<td>Enhanced Insertion Sort</td>
<td>$O(N^2)$</td>
<td>O(NlogN) comp. + O(N^2) copies, Use binary search rather than linear search</td>
</tr>
<tr>
<td>List Insertion Sort</td>
<td>$O(N^2)$</td>
<td>O(N^2) comp. + O(N) copies, Only 2N copies, Does not work 'in-place'</td>
</tr>
<tr>
<td>MergeSort</td>
<td>$O(N\log N)$</td>
<td>O(N) copies by O(logN) levels, Divide &amp; Conquer + Recursive, Does not work 'in place'</td>
</tr>
<tr>
<td>ShellSort</td>
<td>$O(N(\log N)^2)$</td>
<td>In average, 'Insertion sort' using increment, Worst case not far from average</td>
</tr>
<tr>
<td>QuickSort</td>
<td>$O(N\log N)$</td>
<td>Comp. &gt; swaps, Divide and Conquer, Uses partitioning recursively</td>
</tr>
<tr>
<td>HeapSort</td>
<td>$O(N\log N)$</td>
<td>Require a heap data-structure, No worst case</td>
</tr>
</tbody>
</table>
Specialized Purpose Data Structures

- Motivations
  - Used as programming tools (conceptual aids)

- ADT
  - Stacks: LIFO
    - Array or Linked-list
    - Insert/delete in O(1)
  - Queue: FIFO
    - Array or Linked-List
    - Insert/delete in O(1)
  - Priority Queue: item with highest priority
    - Using Ordered array: O(1) deletion, O(N) insertion
    - Using Heap: O(logN) for both deletion/insertion
Motivations
- Do not store general-purpose data (Use in Modeling)
- Model real-world situations
- Tool for numerical modeling in scientific computing (via matrix representation)

Graph ↔ Matrix

Algorithms we have seen:
- Search: DFS, BFS
- Minimum Spanning Trees (MST)
- Shortest Path Problem (Dijkstra)

Efficiency of Algorithms depends on Dense vs Sparse matrix storage
Further Reading

- Your Textbook ;-) (e.g., Robert Lafore
- R. Sedgewick textbook
  - *Algorithms*, Fourth Edition
- Interview preparation (example G. Laakmann)
  - *Cracking the Coding Interview*, 6th Edition