School of Computer Science

COMP SCI 1103/2103 Algorithm Design & Data Structure

Quick Revision

Algorithmic Strategies

- Fundamentals of problem solving
- Strategies:
  - Brute force (exhaustive search)
  - Backtracking
  - Branch and bound
  - Divide-and-Conquer
  - Transform-and-Conquer
  - Dynamic Programming
  - Greedy Algorithms
  - Heuristic Algorithms

Tree

- Recursive definition
  - Root, parent-child, leaf, height, depth
- Binary tree
  - At most two child nodes
  - Example, binary heap
- Binary search tree
  - Complexity of search, insert and delete: O(height)
  - Self-balancing tree, e.g. AVL tree: height=O(log n)
- Heap
  - Height = log n, complexity of insert and deleteMin: O(log n)
  - We can implement priority queues with this data structure
  - Heap sort

Computational Complexity

- A few assumption
  - Algorithm
  - Input size matters
- Running time complexity bounds
  - Upper and lower bounds for whatever we are after
- Big-Oh notation formal definition
  - Big-Oh, Big-Omega, Big-Theta and Little-oh
  - How to prove things: by means of the formal definition
    - Sometimes a counter example can help
      if $f(n) = O(X)$ and $g(n) = O(Y)$ then is it always true: $f(n)/g(n) = O(X/Y)$?
- General rules

Searching and Sorting

- Searching
  - Exhaustive Search
  - Binary Search -> sorted array
- Sorting
  - Insertion sort
  - Selection sort
  - Bubble sort
  - Merge Sort (try to implement it as a practice)
  - Quick Sort
    - Make sure the size of the larger list is less than n
    - For this, keep pivot separate. Still, do every thing in-place!
    - Generally, it is sufficient to usually have both lists in $\Omega(n)$
  - Heap Sort
  - Bucket Sort

List ADT

- Data type and ADT
- Linked list
  - Dynamic size
- Stack
  - LIFO
  - Push, pop and empty
- Queue
  - FIFO
  - Add, remove and empty
- Stack and Queue can both be implemented by arrays and also linked-lists.
  - All basic operations need O(1)