Fundamental Algorithms and Data Structures

Data Structures	Data Types	Algorithms
Dynamic Array	Stack	Binary search
Singly-Linked List	Queue	Mergesort
Doubly-Linked List	Priority Queue	Heapsort
Hash Table	List	Quicksort
Binary Search Tree	Map	Radix sort
Trie	Set	Depth-first search
Adjacency List		Breadth-first search
Adjacency Matrix		Dijkstra's algorithm
Binary Heap		
Bitvector		

We strongly recommend that you study these data structures and algorithms and understand the tradeoffs between them. You should also know the time complexities of each data structure and algorithm listed here. Rather than memorizing all of these details, we recommend taking the time to review each one individually and understand the ideas underlying each. While it's great to know that looking up a string of length L in a trie will take time O(L) in the worst-case and O(1) in the best-case, it's better to understand how the lookup works so that you can derive these time complexities on-the-fly if you need to.

Activity: Algorithms and Data Structures

Work together in a group to answer each of the following questions. We're not expecting that you know all of these answers on your own, so feel free to ask around or look up the answers online. We recommend using these questions as study practice going forward.

Data Structures

1. What is the time complexity of adding an element to the end of a dynamic array? What is the time complexity of adding an element to the front of a dynamic array?
2. Give one example of an operation on a doubly-linked list that is faster than the corresponding operation on a singly-linked list.
3. Give an example of when it would be useful to store head and tail pointers in a linked list.
4. Draw the binary search tree that results from inserting the elements 4, 1, 3, 2, 6, 5 into an empty tree. What is the height of this tree?
5. What is the maximum possible height of a binary search tree containing n elements? What is the minimum possible height?
6. What data structures might you use to implement a set? List some tradeoffs between each implementation.
7. What data structures might you use to implement a stack? List some tradeoffs between each implementation.
8. What is the average-case time complexity of looking up an element in a hash table? What is the worst-case time complexity of looking up an element in a hash table?
9. Give one advantage of an adjacency list over an adjacency matrix and vice-versa.

Algorithms

1. What are the best-case and worst-case runtimes of binary search on an array of length n?
2. What are the best-case and worst-case runtimes for quicksort?
3. Give one advantage of heapsort over quicksort and vice-versa.
4. Give one advantage of quicksort over radix sort and vice-versa.
5. Which graph algorithm would you use to find the shortest path in a graph between two nodes? Does your answer rely on any assumptions about the graph?