Algorithms & Data Structures

Simon Hobbs

June 20, 2017

Contents

1

Data Structures

1.1 Elementary Data Structures

Arrays

- Indexing, inserting/retrieving values at an existing location.
- Returning the length, etc

Take an array A to be that of 8 characters:
Arrays differ in different languages, for instance in python an array element can hold any object, in C the array must contain all the same object. So in C A would be defined as char A[8]; with A being a pointer to the first character. In C the array declaration allocates the size of the object $\times$ <count> consecutively in memory. As elements are accessed it is automatic pointer arithmetic (leads to buffer overflow if unchecked). In python the list is not concretely defined in terms of memory position, but can still be consecutively accessed with an index acting like a c array.

Linked Lists

Sedgewick:: Page 17

- Insert: Adds an element to the chain breaking the links to the left and right and assigning those to itself.
- Delete: Removes an element from the list, remapping the pointers from the left to point to the one on the right.
- Empty: Responds if the head node points to the head node

A	B	C	D	E	F	G	H

Figure 1.1: An array of 8 characters. Each is a constant size and stored contiguously in memory.

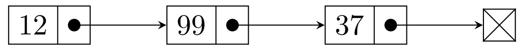

Figure 1.2: Diagram of a singley linked list. Each element contains some information and a pointer to the next element

A linked list is not contiguous memory like an array (in C). This means that the structure is not of a fixed size. Benifits include being able to reorder the list with only a constant handful of operations.
Versions of a linked list are singley linked, doubley linked, and cicularly linked.

Singley linked lists

Singley linked lists have to be traversed head to tail only.

Doubley linked lists

Doubley linked lists allow for the traversing of a link in both directions. Although increasing the flexability this doubles the number of interactions that an algo has to make with component rearangement. The head/tail points the the next/previous element twice. each element points to the previous and next element.

Circularly Linked Lists

The tail points to the head allowing traversal in 1 direction to all elements regardless of where the starting point is. Same number of actions as a singley linked list.

Pushdown Stacks

The Stack: Wiki Link
A restricted data structure disallowing arbitrary data access. Stacks are last in first out (LIFO) structures. A generalized queue/stack is a deque which is in the standard library of python under collections.

- Push: Adds an item to the stack
- Pop: Removes the last added item
- Empty: If the stack is empty $\rightarrow$ head $= =$ tail

Queue

Queue: Wiki Link
A generalized queue/stack is a deque which is in the standard library of python under collections. Fifo, any added item are added to the end/tail of the structure (inqueued). Items are only removed from the front/head (dequeued). This simulates waiting in line say waiting for input to be processsed in the correct order.

Table 1.1: Terminology used in tree structures

Nodes	A vertex connected to other vertecies by edges, in a tree nodes are connected by edges
Edge	Connections to other nodes
Path	List of distinct verticies where each successive one is connected to the next by an edge
Subtrees	A connected group of children nodes connected to a parent thats not the root
Digraph	
Ordered Tree	Order of children at every level is specified
Root	Top node of a tree
Child	A node directly connected to another, away from the root
Parent	Directly connected node towards the root. Can only have 1 parent as this causes cycles.
Sibling	Node with same parent
Decendant	Node accessible by parent to child
Ancestor	Node accessible by traversing child to parent
Leaf	(or external node) A node with no children (degree 1)
Branch	(or internal node) A node with children (degree >1)
Degree	The number of edges on a nodes
Path	Sequence of nodes and edges to reach another node
Level	1+connections to root of a node. (R)-( )-( )-(A) A has level4.
Node Height	Number of edges on the longest path between that node and a leaf.
Depth	Number of edges from the root to a given node
Forest	A set of disjointed trees
Branching Factor	Maximum number of children per node

- Put: Enque, add an element to the end of the queue.
- Get: Deque, remove an item from the front of the queue.
- Empty: head = = tail

Deque

Double ended queue. Combination of a stack and a queue. See pythons collections.deque

1.2 Trees

Sedgewick: : Chapter 4, pg. 35

Tree: Wiki Link

A tree is a data structure made up of nodes connected by edges without having a cycle in it (a node cannot call itself in anyway). A linear list is a trivial tree.

Tree Properties

1. There is exactly one path connecting any two nodes in a tree
2. A tree with N nodes has N-1 edges

3. Abinary tree with N internal nodes has N+1 external nodes

4. The external path length of any binary tree with N internal nodes is 2N greater than the internal path length

5. The height of a binary tree with N internal nodes is about $\log_2 N$. (Precisely floor($\log_2 N$))

Search Tree

Search Tree: Wiki Link
Tree data structure used for locating specific keys from within a set. Needs to be relatively balanced to be efficient.

Binary Trees

Binary Trees: Allisons Link
Binary Trees: Wiki Link

Hash Tree

Hash Array Tree: Wiki Link
Merkle Tree: Wiki Link

Trie

Trie: Wiki Link
Type of search tree. No node stores the key associated with its node, the position in the tree decides its key. All of the decendants of a node have the same prefix, the root is an empty string.
Commonly used for autocomplete and predictive text.
A trie can replace hash table:

- Worst case lookup is better
- No key collisions
- Buckets are only necessary if a key identifies multiple values
- Can provide alphabetical ordering

Drawbacks:

- Tends to be slower than a hash for lookups
- Floats can cause nasty long search chains
- Can require more memory as the keys are split up instead of contiguous

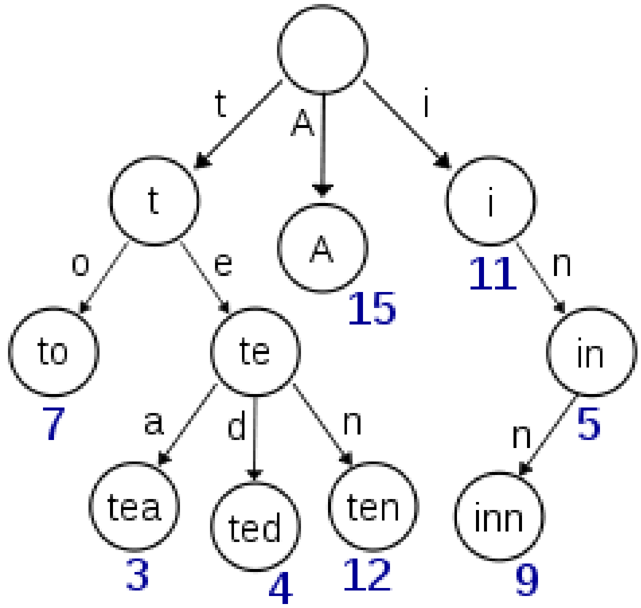

Figure 1.3: A trie data type visualization

Heap

The Heap: Wiki Link

Tree data type, subtype of a priority queue. two types → min and max. In a min/max heap the root is the lowest/highest value in the tree.
The binary heap was introduced for the heap sort algorithm. The heap is partially ordered.

- Heap Property: with P → parent node, C → child node, then the key of P is ordered with respect to C. This applies for every child and parent.
- The root is the lowest or highest value
- Items always go in the next free slot. If it isnt in the right place compare to its parent and swap is the parent is smaller/larger.

Sets

Sets: Wiki Link

Table 1.2: **Time Complexity**

Algo	Ave	Worst
Space	O(n)	O(n)
Search	O(1)	O(n)
Insert	O(1)	O(n)
Delete	O(1)	O(n)

Table 1.3: Hash table terms

Key	The name of a value/ attribute
Bucket	The array elements. Typically a dynamic array
Slot	Synonym for bucket
Hash function	computes the index from a key
Load Factor	$\frac{entries}{buckets}$ The higher the load factorthe slower the search, the lower the load facter the more memory wasted.

Union

Tagged Union

1.3 Hashes

Hash function

Hash Table

Hash Table: Wiki Link

Hash tables tend to be faster than other table data structures, degrading to the same average lookup time of an unordered array only in the worst case senario. If the hash function is complex and the entry count small, this advantage can be lost.

A hash table is an associative array (i.e. a dictionary) that maps keys to values. Puts a key through a hash function to find/retrieve an index to an array of buckets

Collision Resolution

Separate Chaining	Buckets have a list of their own entries to a single has, if the hash matches and the key doesn't do a linear search of the list
Open Addressing	On a collision, the new entry takes the next open slot. Useful if memory is an issue and the entries are smaller that $\sim 4\times$ sizeof(*)

Heuristics $\rightarrow$ efficient algos that get a good although not necessarily perfect solution

2

Algorithms

NIST Dictionary of Algos and Data Structs: NIST Link

2.1 Asymptotics

How the algorithm grows as N $\rightarrow \infty$.

Algorithms by David Sedgewick: Page 67
David Mount Notes: Link
Big-O Notation: Wiki Link
Time Complexity: Wiki Link

Notation

Θ: The asymptotic class of an algo.

$$\Theta(g(n)) \equiv \left\{ f(n), 0 <= c_1 g(n) <= f(n) <= c_2 g(n) | c_1, c_2, n_0 \in |\Re| \text{ and } n_0 <= n \right\} \tag{2.1}$$

For a algo to be Θ(g(n)) it needs to be both O(g(n)) and Ω(g(n)). A Θ(g(n)) grows at exactly g(n).

O: Upper asymptotic bound for an algo. An algo that has O(g(n)) grows at or slower than that rate.

$$O(g(n)) \equiv \left\{ f(n) | 0 <= f(n) <= cg(n) | c, n_0 \in |\Re| \text{ and } n_0 <= n \right\} \tag{2.2}$$

Ω: The lower bound on the growth. An algo w/ Ω(g(n)) grows at or faster than g(n).

$$\Omega(g(n)) \equiv \left\{ f(n) | 0 <= cg(n) <= f(n) | c_1, c_2, n_0 \in |\Re| \text{ and } n_0 <= n \right\} \tag{2.3}$$

Performace

Some algorithms, classically quicksort, have drastically different average and worst case performance. The best case is seldom used.

Table 2.1: Asymptotic growth types

Notation	Name	Example
O(1)	Constant Time	Seeing if a binary number is even or odd
O(log log n)	Double Logarthmic	
O(log n)	Logarithmic	Finding and item in a sorted array with binary search
O((log n)^c) w/ $c > 1$	Polylogarithmic	
O(n^c) w/ $0 < c < 1$	Fractional power	Searching in a kd-tree
O(n)	Linear	Find an item in an unsorted list
O(n log*n)	n log-star n	Union-find
O(n log n)	quasilinear/linearithmic	Theoretical limit on sorting based on comparison (heapsort, mergesort, quicksort)
O(n^2)	Quadratic	Simple comparison sorting (bubble, selection, etc)
O(n^c)	Polynomial	LU decomposition
O(2^n),O(n^n),O(n!)	Exponential	

Worst Case: $T_{worst}(n) \equiv max_{|I|=n} T(I)$. The worst possible performance with legal input.

Best Case: The fastest performance $\rightarrow$ a sorting algorithm realizing its already sorted

Average Case: $T_{avg}(n) \equiv \sum_{|I|=n} p(I)T(I)$ Where p(I) is the probability weight of T(I) occuring.

Asymptotic Analysis

(Strong) Induction

Iteration

Recurrance

Bounding

Integration

Master's Theorem

Masters Theorem: Wiki Link

Akra-Bazzi Method: Wiki Link

For divide and conquer algos. The generalization of this is the Akra-Bazzi method Let a >= 1, b> 1 be constants and let T(n) be the recurrence:

$$T(n) = aT(n/b) = N^k n >= 0 \tag{2.4}$$

Assume n is a power of b and the basis case T(1) be a constant. Then:

Case 1: if $a > b^k$ then T(n) $\in \Theta(n^{log_b a})$

Case 2: if $a == b^k$ then T(n) $\in \Theta(n^k logn)$

Case 3: if $a < b^k$ then T(n) $\in \Theta(n^k)$

Merge sort has a=2, b=2, k=1 so $T(n) \in \Theta(nlogn)$ Binary search a=1, b=2, k=0 so $T(n) \in \Theta(logn)$

2.2 Algorithm Types

Divide & Conquer

Selection

2.3 Searching

Binary Search

Binary Search: Wiki Link

Linear Search

Linear Search: Wiki Link

2.4 Selection

Sieve Technique

2.5 Sorting

Merge Sort

Heap Sort

Quick Sort

Bubble Sort

Insertion Sort

Selection Sort

Count Sort

Radix Sort

3

Math

3.1 Series

Arithmetic:

$$\sum_{i=1}^{n} i = 1 + 2 + ... + n = \frac{n(n+1)}{2} \tag{3.1}$$

Geometric:

$$\sum_{i=0}^{n} x^i = 1 + x + x^2 + ... + x^n = \frac{x^{n+1} - 1}{x - 1} \quad \text{c} \neq 1 \tag{3.2}$$

Harmonic:

$$H_n \equiv \sum_{i=1}^{n} \frac{1}{i} = 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{n} \approx ln(n) \tag{3.3}$$

Constant:

$$\sum_{i=1}^{n} 1 = n \tag{3.4}$$