1. Module 1: Introduction to Data Structures and Algorithms

1.1 Data structures

data structure

A data structure is a way of organizing, storing, and performing operations on data.

record

A record is the data structure that stores subitems, often called fields, with a name associated with each subitem.

array

An array is a data structure that stores an ordered list of items, where each item is directly accessible by a positional index.

linked list

A linked list is a data structure that stores an ordered list of items in nodes, where each node stores data and has a pointer to the next node.

binary tree

A binary tree is a data structure in which each node stores data and has up to two children, known as a left child and a right child.

hash table

A hash table is a data structure that stores unordered items by mapping (or hashing) each item to a location in an array.

max-heap

A max-heap is a tree that maintains the simple property that a node's key is greater than or equal to the node's childrens' keys.

min-heap

A min-heap is a tree that maintains the simple property that a node's key is less than or equal to the node's childrens' keys.

graph

A graph is a data structure for representing connections among items, and consists of vertices connected by edges.

vertex

A vertex represents an item in a graph.

edge

An edge represents a connection between two vertices in a graph.

1.2 Introduction to algorithms

algorithm

An algorithm describes a sequence of steps to solve a computational problem or perform a calculation.

computational problem

A computational problem specifies an input, a question about the input that can be answered using a computer, and the desired output.

NP-complete

NP-complete problems are a set of problems for which no known efficient algorithm exists.

1.4 Abstract data types

abstract data type / ADT

An abstract data type (ADT) is a data type described by predefined user operations, such as "insert data at rear," without indicating how each operation is implemented.

list

A list is an ADT for holding ordered data.

dynamic array

A dynamic array is an ADT for holding ordered data and allowing indexed access.

stack

A stack is an ADT in which items are only inserted on or removed from the top of a stack.

queue

A queue is an ADT in which items are inserted at the end of the queue and removed from the front of the queue.

deque

A deque (pronounced "deck" and short for double-ended queue) is an ADT in which items can be inserted and removed at both the front and back.

bag

A bag is an ADT for storing items in which the order does not matter and duplicate items are allowed.

set

A set is an ADT for a collection of distinct items.

priority queue

A priority queue is a queue where each item has a priority, and items with higher priority are closer to the front of the queue than items with lower priority.

dictionary

A dictionary is an ADT that associates (or maps) keys with values.

2. Module 2: Searching and Algorithm Analysis

2.1 Algorithm efficiency

Algorithm efficiency

Algorithm efficiency is typically measured by the algorithm's computational complexity.

Computational complexity

Computational complexity is the amount of resources used by the algorithm.

runtime complexity

An algorithm's runtime complexity is a function, T(N), that represents the number of constant time operations performed by the algorithm on an input of size N.

best case

An algorithm's best case is the scenario where the algorithm does the minimum possible number of operations.

worst case

An algorithm's worst case is the scenario where the algorithm does the maximum possible number of operations.

space complexity

An algorithm's space complexity is a function, S(N), that represents the number of fixed-size memory units used by the algorithm for an input of size N.

auxiliary space complexity

An algorithm's auxiliary space complexity is the space complexity not including the input data.

2.2 Searching and algorithms

algorithm

An algorithm is a sequence of steps for accomplishing a task.

Linear search

Linear search is a search algorithm that starts from the beginning of a list, and checks each element until the search key is found or the end of the list is reached.

runtime

An algorithm's runtime is the time the algorithm takes to execute.

2.3 Binary search

Binary search

Binary search is a faster algorithm for searching a list if the list's elements are sorted and directly accessible (such as an array).

2.4 Constant time operations

constant time operation

A constant time operation is an operation that, for a given processor, always operates in the same amount of time, regardless of input values.

2.5 Growth of functions and complexity

Lower bound

Lower bound: A function f(N) that is ≤ the best case T(N), for all values of N ≥ 1.

Upper bound

Upper bound: A function f(N) that is ≥ the worst case T(N), for all values of N ≥ 1.

Asymptotic notation

Asymptotic notation is the classification of runtime complexity that uses functions that indicate only the growth rate of a bounding function.

O notation

O notation provides a growth rate for an algorithm's upper bound.

Ω notation

Ω notation provides a growth rate for an algorithm's lower bound.

Θ notation

Θ notation provides a growth rate that is both an upper and lower bound.

2.6 O notation

Big O notation

Big O notation is a mathematical way of describing how a function (running time of an algorithm) generally behaves in relation to the input size.

2.7 Algorithm analysis

worst-case runtime

The worst-case runtime of an algorithm is the runtime complexity for an input that results in the longest execution.

3. Module 3: Lists

3.1 List abstract data type (ADT)

list

A list is a common ADT for holding ordered data, having operations like append a data item, remove a data item, search whether a data item exists, and print the list.

3.2 Singly-linked lists

singly-linked list

A singly-linked list is a data structure for implementing a list ADT, where each node has data and a pointer to the next node.

head / tail

A singly-linked list's first node is called the head, and the last node the tail.

positional list

A singly-linked list is a type of positional list: A list where elements contain pointers to the next and/or previous elements in the list.

null

Null is a special value indicating a pointer points to nothing.

Append

Given a new node, the Append operation for a singly-linked list inserts the new node after the list's tail node.

Prepend

Given a new node, the Prepend operation for a singly-linked list inserts the new node before the list's head node.

3.3 Singly-linked lists: Insert

InsertAfter

Given a new node, the InsertAfter operation for a singly-linked list inserts the new node after a provided existing list node.

3.4 Singly-linked lists: Remove

RemoveAfter

Given a specified existing node in a singly-linked list, the RemoveAfter operation removes the node after the specified list node.

3.5 Linked list search

search

Given a key, a search algorithm returns the first node whose data matches that key, or returns null if a matching node was not found.

3.6 Array-based lists

array-based list

An array-based list is a list ADT implemented using an array.

append

Given a new element, the append operation for an array-based list of length X inserts the new element at the end of the list, or at index X.

Prepend

The Prepend operation for an array-based list inserts a new item at the start of the list.

InsertAfter

The InsertAfter operation for an array-based list inserts a new item after a specified index.

search

Given a key, the search operation returns the index for the first element whose data matches that key, or -1 if not found.

remove-at

Given the index of an item in an array-based list, the remove-at operation removes the item at that index.

4. Module 4: Lists Continued

4.1 Doubly-linked lists

doubly-linked list

A doubly-linked list is a data structure for implementing a list ADT, where each node has data, a pointer to the next node, and a pointer to the previous node.

positional list

A doubly-linked list is a type of positional list: A list where elements contain pointers to the next and/or previous elements in the list.

Append

Given a new node, the Append operation for a doubly-linked list inserts the new node after the list's tail node.

Prepend

Given a new node, the Prepend operation of a doubly-linked list inserts the new node before the list's head node and points the head pointer to the new node.

4.2 Doubly-linked lists: Insert

InsertAfter

Given a new node, the InsertAfter operation for a doubly-linked list inserts the new node after a provided existing list node.

4.3 Doubly-linked lists: Remove

Remove

The Remove operation for a doubly-linked list removes a provided existing list node.

4.4 Linked list traversal

list traversal

A list traversal algorithm visits all nodes in the list once and performs an operation on each node.

reverse traversal

A reverse traversal visits all nodes starting with the list's tail node and ending after visiting the list's head node.

4.6 Linked list dummy nodes

dummy node / header node

A linked list implementation may use a dummy node (or header node): A node with an unused data member that always resides at the head of the list and cannot be removed.

4.7 List data structure

list data structure

A list data structure is a data structure containing the list's head and tail, and may also include additional information, such as the list's size.

4.8 Circular lists

circular linked list

A circular linked list is a linked list where the tail node's next pointer points to the head of the list, instead of null.

5. Module 5: Stacks

5.1 Stack abstract data type (ADT)

stack

A stack is an ADT in which items are only inserted on or removed from the top of a stack.

push

The stack push operation inserts an item on the top of the stack.

pop

The stack pop operation removes and returns the item at the top of the stack.

last-in first-out

A stack is referred to as a last-in first-out ADT.

5.3 Array-based stacks

unbounded stack

An unbounded stack is a stack with no upper limit on length.

bounded stack

A bounded stack is a stack with a length that does not exceed a maximum value.

full

A bounded stack with a length equal to the maximum length is said to be full.

6. Module 6: Queue

6.1 Queue abstract data type (ADT)

queue

A queue is an ADT in which items are inserted at the end of the queue and removed from the front of the queue.

enqueue

The queue enqueue operation inserts an item at the end of the queue.

dequeue

The queue dequeue operation removes and returns the item at the front of the queue.

first-in first-out

A queue is referred to as a first-in first-out ADT.

6.3 Array-based queues

bounded queue

A bounded queue is a queue with a length that does not exceed a specified maximum value.

full

A bounded queue with a length equal to the maximum length is said to be full.

unbounded queue

An unbounded queue is a queue with a length that can grow indefinitely.

6.4 Deque abstract data type (ADT)

deque

A deque (pronounced "deck" and short for double-ended queue) is an ADT in which items can be inserted and removed at both the front and back.

peek

A peek operation returns an item in the deque without removing the item.

7. Module 7: Recursion

7.1 Recursion: Introduction

algorithm

An algorithm is a sequence of steps for solving a problem.

recursive algorithm

A recursive algorithm is an algorithm that breaks the problem into smaller subproblems and applies the same algorithm to solve the smaller subproblems.

base case

At some point, a recursive algorithm must describe how to actually do something, known as the base case.

7.2 Recursive methods

recursive method

A method that calls itself is a recursive method.

7.3 Recursive algorithm: Search

binary search

A binary search algorithm begins at the midpoint of the range and halves the range after each guess.

7.5 Creating a recursive method

7.6 Recursive math methods

Fibonacci sequence

The Fibonacci sequence is 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, etc.; starting with 0, 1, the pattern is to compute the next number by adding the previous two numbers.

greatest common divisor

The greatest common divisor (GCD) is the largest number that divides evenly into two numbers.

7.8 Stack overflow

stack frame

Each method call places a new stack frame on the stack, for local parameters, local variables, and more method items.

stack overflow

Deep recursion could fill the stack region and cause a stack overflow, meaning a stack frame extends beyond the memory region allocated for stack.

8. Module 8: Recursion Continued

8.1 Recursive definitions

algorithm

An algorithm is a sequence of steps, including at least 1 terminating step, for solving a problem.

recursive algorithm

A recursive algorithm is an algorithm that breaks the problem into smaller subproblems and applies the algorithm itself to solve the smaller subproblems.

base case

Base case: A case where a recursive algorithm completes without applying itself to a smaller subproblem.

recursive function

A recursive function is a function that calls itself.

8.2 Recursive algorithms

Fibonacci sequence

The Fibonacci sequence is a numerical sequence where each term is the sum of the previous 2 terms in the sequence, except the first 2 terms, which are 0 and 1.

Fibonacci number

Fibonacci number: A term in the Fibonacci sequence.

Binary search

Binary search is an algorithm that searches a sorted list for a key by first comparing the key to the middle element in the list and recursively searching half of the remaining list so long as the key is not found.

8.3 Analyzing the time complexity of recursive algorithms

recurrence relation

Recurrence relation: A function f(N) that is defined in terms of the same function operating on a value < N.

recursion tree

Recursion tree: A visual diagram of an operation done by a recursive function, that separates operations done directly by the function and operations done by recursive calls.

9. Module 9: Trees

9.1 Binary trees

binary tree

In a binary tree, each node has up to two children, known as a *left child* and a *right child*.

Leaf

Leaf: A tree node with no children.

Internal node

Internal node: A node with at least one child.

Parent

Parent: A node with a child is said to be that child's parent.

ancestors

A node's ancestors include the node's parent, the parent's parent, etc., up to the tree's root.

Root

Root: The one tree node with no parent (the "top" node).

edge

The link from a node to a child is called an edge.

depth

A node's depth is the number of edges on the path from the root to the node.

level

All nodes with the same depth form a tree level.

height

A tree's height is the largest depth of any node.

full

A binary tree is full if every node contains 0 or 2 children.

complete

A binary tree is complete if all levels, except possibly the last level, contain all possible nodes and all nodes in the last level are as far left as possible.

perfect

A binary tree is perfect, if all internal nodes have 2 children and all leaf nodes are at the same level.

9.2 Applications of trees

Binary space partitioning / BSP

Binary space partitioning (BSP) is a technique of repeatedly separating a region of space into 2 parts and cataloging objects contained within the regions.

BSP tree

A BSP tree is a binary tree used to store information for binary space partitioning.

9.3 Binary search trees

binary search tree

A binary search tree (BST), which has an ordering property that any node's left subtree keys ≤ the node's key, and the right subtree's keys ≥ the node's key. That property enables fast searching for an item.

search

To search nodes means to find a node with a desired key, if such a node exists.

successor

A BST node's successor is the node that comes after in the BST ordering, so in A B C, A's successor is B, and B's successor is C.

predecessor

A BST node's predecessor is the node that comes before in the BST ordering.

9.4 BST search algorithm

search

Given a key, a search algorithm returns the first node found matching that key, or returns null if a matching node is not found.

9.5 BST insert algorithm

insert

Given a new node, a BST insert operation inserts the new node in a proper location obeying the BST ordering property.

9.6 BST remove algorithm

remove

Given a key, a BST remove operation removes the first-found matching node, restructuring the tree to preserve the BST ordering property.

9.7 BST inorder traversal

tree traversal

A tree traversal algorithm visits all nodes in the tree once and performs an operation on each node.

inorder traversal

An inorder traversal visits all nodes in a BST from smallest to largest.

9.8 BST height and insertion order

height

A tree's height is the maximum edges from the root to any leaf.

9.11 Tries

trie / prefix tree

A trie (or prefix tree) is a tree representing a set of strings.

terminal node

A terminal node is a node that represents a terminating character, which is the end of a string in the trie.

trie insert

Given a string, a trie insert operation creates a path from the root to a terminal node that visits all the string's characters in sequence.

trie search

Given a string, a trie search operation returns the terminal node corresponding to that string, or null if the string is not in the trie.

trie remove

Given a string, a trie remove operation removes the string's corresponding terminal node and all non-root ancestors with 0 children.

10. Module 10: Heaps and Treaps

10.1 Heaps

max-heap

A max-heap is a complete binary tree that maintains the simple property that a node's key is greater than or equal to the node's children's keys.

insert

An insert into a max-heap starts by inserting the node in the tree's last level, and then swapping the node with its parent until no max-heap property violation occurs.

percolating

The upward movement of a node in a max-heap is called percolating.

remove

A remove from a max-heap is always a removal of the root, and is done by replacing the root with the last level's last node, and swapping that node with its greatest child until no max-heap property violation occurs.

min-heap

A min-heap is similar to a max-heap, but a node's key is less than or equal to its children's keys.

10.3 Heap sort

Heapsort

Heapsort is a sorting algorithm that takes advantage of a max-heap's properties by repeatedly removing the max and building a sorted array in reverse order.

heapify

The heapify operation is used to turn an array into a heap.

10.4 Priority queue abstract data type (ADT)

priority queue

A priority queue is a queue where each item has a priority, and items with higher priority are closer to the front of the queue than items with lower priority.

enqueue

The priority queue enqueue operation inserts an item such that the item is closer to the front than all items of lower priority, and closer to the end than all items of equal or higher priority.

dequeue

The priority queue dequeue operation removes and returns the item at the front of the queue, which has the highest priority.

peek

A peek operation returns the highest priority item, without removing the item from the front of the queue.

EnqueueWithPriority

EnqueueWithPriority: An enqueue operation that includes an argument for the enqueued item's priority.

10.5 Treaps

treap

A treap uses a main key that maintains a binary search tree ordering property, and a secondary key generated randomly (often called "priority") during insertions that maintains a heap property.

search

A treap search is the same as a BST search using the main key, since the treap is a BST.

insert

A treap insert initially inserts a node as in a BST using the main key, then assigns a random priority to the node, and percolates the node up until the heap property is not violated.

delete

A treap delete can be done by setting the node's priority such that the node should be a leaf ($-\infty$ for a max-heap), percolating the node down using rotations until the node is a leaf, and then removing the node.

11. Modules 11-12: Hash Tables & Sets

11.1 Hash tables

hash table

A hash table is a data structure that stores unordered items by mapping (or hashing) each item to a location in an array (or vector).

key

In a hash table, an item's key is the value used to map to an index.

bucket

Each hash table array element is called a bucket.

hash function

A hash function computes a bucket index from the item's key.

modulo operator %

A common hash function uses the modulo operator %, which computes the integer remainder when dividing two numbers.

collision

A collision occurs when an item being inserted into a hash table maps to the same bucket as an existing item in the hash table.

Chaining

Chaining is a collision resolution technique where each bucket has a list of items (so bucket 5's list would become 55, 75).

Open addressing

Open addressing is a collision resolution technique where collisions are resolved by looking for an empty bucket elsewhere in the table (so 75 might be stored in bucket 6).

11.2 Chaining

Chaining

Chaining handles hash table collisions by using a list for each bucket, where each list may store multiple items that map to the same bucket.

11.3 Linear probing

linear probing

A hash table with linear probing handles a collision by starting at the key's mapped bucket, and then linearly searches subsequent buckets until an empty bucket is found.

empty-since-start

An empty-since-start bucket has been empty since the hash table was created.

empty-after-removal

An empty-after-removal bucket had an item removed that caused the bucket to now be empty.

11.4 Quadratic probing

quadratic probing

A hash table with quadratic probing handles a collision by starting at the key's mapped bucket, and then quadratically searches subsequent buckets until an empty bucket is found.

probing sequence

Iterating through sequential i values to obtain the desired table index is called the probing sequence.

11.5 Double hashing

Double hashing

Double hashing is an open-addressing collision resolution technique that uses 2 different hash functions to compute bucket indices.

probing sequence

Iterating through sequential i values to obtain the desired table index is called the probing sequence.

11.6 Hash table resizing

resize

A hash table resize operation increases the number of buckets, while preserving all existing items.

load factor

A hash table's load factor is the number of items in the hash table divided by the number of buckets.

11.7 Common hash functions

perfect hash function

A perfect hash function maps items to buckets with no collisions.

modulo hash

A modulo hash uses the remainder from division of the key by hash table size N.

mid-square hash

A mid-square hash squares the key, extracts R digits from the result's middle, and returns the remainder of the middle digits divided by hash table size N.

multiplicative string hash

A multiplicative string hash repeatedly multiplies the hash value and adds the ASCII (or Unicode) value of each character in the string.

11.8 Direct hashing

direct hash function

A direct hash function uses the item's key as the bucket index.

direct access table

A hash table with a direct hash function is called a direct access table.

search

Given a key, a direct access table search algorithm returns the item at index key if the bucket is not empty, and returns null (indicating item not found) if empty.

11.9 Hashing Algorithms: Cryptography, Password Hashing

Cryptography

Cryptography is a field of study focused on transmitting data securely.

encryption

Encryption: alteration of data to hide the original meaning.

decryption

Decryption: reconstruction of original data from encrypted data.

cryptographic hash function

A cryptographic hash function is a hash function designed specifically for cryptography.

password hashing function

A password hashing function is a cryptographic hashing function that produces a hash value for a password.

11.10 Set abstract data type

set

A set is a collection of distinct elements.

add

A set add operation adds an element to the set, provided an equal element doesn't already exist in the set.

key value

Key value: A primitive data value that serves as a unique identifier for the element.

remove

Given a key, a set remove operation removes the element with the specified key from the set.

search

Given a key, a set search operation returns the set element with the specified key, or null if no such element exists.

subset

A set X is a subset of set Y only if every element of X is also an element of Y.

11.11 Set operations

union

The union of sets X and Y, denoted as X ∪ Y, is a set that contains every element from X, every element from Y, and no additional elements.

intersection

The intersection of sets X and Y, denoted as X ∩ Y, is a set that contains every element that is in both X and Y, and no additional elements.

difference

The difference of sets X and Y, denoted as X \ Y, is a set that contains every element that is in X but not in Y, and no additional elements.

filter

A filter operation on set X produces a subset containing only elements from X that satisfy a particular condition.

filter predicate

The condition for filtering is commonly represented by a filter predicate: A function that takes an element as an argument and returns a Boolean value indicating whether or not that element will be in the filtered subset.

map

A map operation on set X produces a new set by applying some function F to each element.

11.12 Static and dynamic set operations

dynamic set

A dynamic set is a set that can change after being constructed.

static set

A static set is a set that doesn't change after being constructed.

12. Module 13: Sorting Algorithms

12.1 Sorting: Introduction

Sorting

Sorting is the process of converting a list of elements into ascending (or descending) order.

12.2 Selection sort

Selection sort

Selection sort is a sorting algorithm that treats the input as two parts, a sorted part and an unsorted part, and repeatedly selects the proper next value to move from the unsorted part to the end of the sorted part.

12.3 Insertion sort

Insertion sort

Insertion sort is a sorting algorithm that treats the input as two parts, a sorted part and an unsorted part, and repeatedly inserts the next value from the unsorted part into the correct location in the sorted part.

nearly sorted

A nearly sorted list only contains a few elements not in sorted order.

12.4 Shell sort

Shell sort

Shell sort is a sorting algorithm that treats the input as a collection of interleaved lists, and sorts each list individually with a variant of the insertion sort algorithm.

gap value

A gap value is a positive integer representing the distance between elements in an interleaved list.

12.5 Quicksort

Quicksort

Quicksort is a sorting algorithm that repeatedly partitions the input into low and high parts (each part unsorted), and then recursively sorts each of those parts.

pivot

The pivot can be any value within the array being sorted, commonly the value of the middle array element.

12.6 Merge sort

Merge sort

Merge sort is a sorting algorithm that divides a list into two halves, recursively sorts each half, and then merges the sorted halves to produce a sorted list.

12.7 Radix sort

bucket

A bucket is a collection of integer values that all share a particular digit value.

Radix sort

Radix sort is a sorting algorithm specifically for an array of *integers*: The algorithm processes one digit at a time starting with the least significant digit and ending with the most significant.

12.8 Overview of fast sorting algorithms

fast sorting algorithm

A fast sorting algorithm is a sorting algorithm that has an average runtime complexity of O($NlogN$) or better.

element comparison sorting algorithm

A element comparison sorting algorithm is a sorting algorithm that operates on an array of elements that can be compared to each other.

12.9 Bubble sort

Bubble sort

Bubble sort is a sorting algorithm that iterates through a list, comparing and swapping adjacent elements if the second element is less than the first element.

12.10 Quickselect

Quickselect

Quickselect is an algorithm that selects the k^{th} smallest element in a list.

12.11 Bucket sort

Bucket sort

Bucket sort is a numerical sorting algorithm that distributes numbers into buckets, sorts each bucket with an additional sorting algorithm, and then concatenates buckets together to build the sorted result.

bucket

A bucket is a container for numerical values in a specific range.

13. Balanced Trees

13.1 AVL: A balanced tree

AVL tree

An AVL tree is a BST with a height balance property and specific operations to rebalance the tree when a node is inserted or removed.

height balanced

A BST is height balanced if for any node, the heights of the node's left and right subtrees differ by only 0 or 1.

balance factor

A node's balance factor is the left subtree height minus the right subtree height.

13.2 AVL rotations

rotation

A rotation is a local rearrangement of a BST that maintains the BST ordering property while rebalancing the tree.

13.4 AVL removals

remove

Given a key, an AVL tree remove operation removes the first-found matching node, restructuring the tree to preserve all AVL tree requirements.

13.5 Red-black tree: A balanced tree

red-black tree

A red-black tree is a BST with two node types, namely red and black, and supporting operations that ensure the tree is balanced when a node is inserted or removed.

13.7 Red-black tree: Insertion

insert

Given a new node, a red-black tree insert operation inserts the new node in the proper location such that all red-black tree requirements still hold after the insertion completes.

13.8 Red-black tree: Removal

remove
Given a key, a red-black tree remove operation removes the first-found matching node, restructuring the tree to preserve all red-black tree requirements.

14. Graphs

14.1 Graphs: Introduction

graph
A graph is a data structure for representing connections among items, and consists of vertices connected by edges.

vertex
A vertex (or node) represents an item in a graph.

edge
An edge represents a connection between two vertices in a graph.

adjacent
Two vertices are adjacent if connected by an edge.

path
A path is a sequence of edges leading from a source (starting) vertex to a destination (ending) vertex.

path length
The path length is the number of edges in the path.

distance
The distance between two vertices is the number of edges on the shortest path between those vertices.

14.3 Graph representations: Adjacency lists

adjacent
Two vertices are adjacent if connected by an edge.

adjacency list
In an adjacency list graph representation, each vertex has a list of adjacent vertices, each list item representing an edge.

sparse graph
A sparse graph has far fewer edges than the maximum possible.

14.4 Graph representations: Adjacency matrices

adjacent

Two vertices are adjacent if connected by an edge.

adjacency matrix

In an adjacency matrix graph representation, each vertex is assigned to a matrix row and column, and a matrix element is 1 if the corresponding two vertices have an edge or is 0 otherwise.

14.5 Graphs: Breadth-first search

graph traversal

An algorithm commonly must visit every vertex in a graph in some order, known as a graph traversal.

breadth-first search

A breadth-first search (BFS) is a traversal that visits a starting vertex, then all vertices of distance 1 from that vertex, then of distance 2, and so on, without revisiting a vertex.

discovered

When the BFS algorithm first encounters a vertex, that vertex is said to have been discovered.

frontier

In the BFS algorithm, the vertices in the queue are called the frontier, being vertices thus far discovered but not yet visited.

14.6 Graphs: Depth-first search

graph traversal

An algorithm commonly must visit every vertex in a graph in some order, known as a graph traversal.

depth-first search

A depth-first search (DFS) is a traversal that visits a starting vertex, then visits every vertex along each path starting from that vertex to the path's end before backtracking.

14.7 Directed graphs

directed graph / digraph

A directed graph, or digraph, consists of vertices connected by directed edges.

directed edge

A directed edge is a connection between a starting vertex and a terminating vertex.

adjacent

In a directed graph, a vertex Y is adjacent to a vertex X, if there is an edge from X to Y.

path

A path is a sequence of directed edges leading from a source (starting) vertex to a destination (ending) vertex.

cycle

A cycle is a path that starts and ends at the same vertex.

cyclic / acyclic

A directed graph is cyclic if the graph contains a cycle, and acyclic if the graph does not contain a cycle.

14.8 Weighted graphs

weighted graph

A weighted graph associates a weight with each edge.

weight / cost

A graph edge's weight, or cost, represents some numerical value between vertex items, such as flight cost between airports, connection speed between computers, or travel time between cities.

path length

In a weighted graph, the path length is the sum of the edge weights in the path.

cycle length

The cycle length is the sum of the edge weights in a cycle.

negative edge weight cycle

A negative edge weight cycle has a cycle length less than 0.

14.9 Algorithm: Dijkstra's shortest path

Dijkstra's shortest path algorithm

Dijkstra's shortest path algorithm, created by Edsger Dijkstra, determines the shortest path from a start vertex to each vertex in a graph.

distance

A vertex's distance is the shortest path distance from the start vertex.

predecessor pointer

A vertex's predecessor pointer points to the previous vertex along the shortest path from the start vertex.

14.10 Algorithm: Bellman-Ford's shortest path

Bellman-Ford shortest path algorithm

The Bellman-Ford shortest path algorithm, created by Richard Bellman and Lester Ford, Jr., determines the shortest path from a start vertex to each vertex in a graph.

distance

A vertex's distance is the shortest path distance from the start vertex.

predecessor pointer

A vertex's predecessor pointer points to the previous vertex along the shortest path from the start vertex.

14.11 Topological sort

topological sort

A topological sort of a directed, acyclic graph produces a list of the graph's vertices such that for every edge from a vertex X to a vertex Y, X comes before Y in the list.

14.12 Minimum spanning tree

minimum spanning tree

A graph's minimum spanning tree is a subset of the graph's edges that connect all vertices in the graph together with the minimum sum of edge weights.

connected

A connected graph contains a path between every pair of vertices.

Kruskal's minimum spanning tree algorithm

Kruskal's minimum spanning tree algorithm determines subset of the graph's edges that connect all vertices in an undirected graph with the minimum sum of edge weights.

14.13 All pairs shortest path

all pairs shortest path

An all pairs shortest path algorithm determines the shortest path between all possible pairs of vertices in a graph.

Floyd-Warshall all-pairs shortest path algorithm

The Floyd-Warshall all-pairs shortest path algorithm generates a |V| x |V| matrix of values representing the shortest path lengths between all vertex pairs in a graph.

negative cycle

A negative cycle is a cycle with edge weights that sum to a negative value.

15. Algorithms

15.1 Huffman compression

compression

Given data represented as some quantity of bits, compression transforms the data to use fewer bits.

Huffman coding

Huffman coding is a common compression technique that assigns fewer bits to frequent items, using a binary tree.

15.2 Heuristics

heuristic

Heuristic: A technique that willingly accepts a non-optimal or less accurate solution in order to improve execution speed.

heuristic algorithm

A heuristic algorithm is an algorithm that quickly determines a near optimal or approximate solution.

0-1 knapsack problem

0-1 knapsack problem: The knapsack problem with the quantity of each item limited to 1.

self-adjusting heuristic

A self-adjusting heuristic is an algorithm that modifies a data structure based on how that data structure is used.

15.3 Greedy algorithms

greedy algorithm

A greedy algorithm is an algorithm that, when presented with a list of options, chooses the option that is optimal at that point in time.

fractional knapsack problem

The fractional knapsack problem is the knapsack problem with the potential to take each item a fractional number of times, provided the fraction is in the range [0.0, 1.0].

activity selection problem

The activity selection problem is a problem where 1 or more activities are available, each with a start and finish time, and the goal is to build the largest possible set of activities without time conflicts.

15.4 Dynamic programming

Dynamic programming

Dynamic programming is a problem solving technique that splits a problem into smaller subproblems, computes and stores solutions to subproblems in memory, and then uses the stored solutions to solve the larger problem.

longest common substring

The longest common substring algorithm takes 2 strings as input and determines the longest substring that exists in both strings.

16. B-trees

16.1 B-trees

B-tree

A B-tree with order K is a tree where nodes can have up to K-1 keys and up to K children.

order

The order is the maximum number of children a node can have.

full

A 2-3-4 tree node containing exactly 3 keys is said to be full, and uses all keys and children.

2-node

A node with 1 key is called a 2-node.

3-node

A node with 2 keys is called a 3-node.

4-node

A node with 3 keys is called a 4-node.

16.2 2-3-4 tree search algorithm

search

Given a key, a search algorithm returns the first node found matching that key, or returns null if a matching node is not found.

16.3 2-3-4 tree insert algorithm

insert

Given a new key, a 2-3-4 tree insert operation inserts the new key in the proper location such that all 2-3-4 tree properties are preserved.

split

An important operation during insertion is the split operation, which is done on every full node encountered during insertion traversal.

preemptive split

The preemptive split insertion scheme always splits any full node encountered during insertion traversal.

16.4 2-3-4 tree rotations and fusion

rotation

A rotation is a rearrangement of keys between 3 nodes that maintains all 2-3-4 tree properties in the process.

right rotation

A right rotation on a node causes the node to lose one key and the node's right sibling to gain one key.

left rotation

A left rotation on a node causes the node to lose one key and the node's left sibling to gain one key.

fusion

A fusion is a combination of 3 keys: 2 from adjacent sibling nodes that have 1 key each, and a third from the parent of the siblings.

16.5 2-3-4 tree removal

merge

A B-Tree merge operates on a node with 1 key and increases the node's keys to 2 or 3 using either a rotation or fusion.

remove

Given a key, a 2-3-4 tree remove operation removes the first-found matching key, restructuring the tree to preserve all 2-3-4 tree rules.

preemptive merge

The preemptive merge removal scheme involves increasing the number of keys in all single-key, non-root nodes encountered during traversal.