Sample RAG Practice Document: Introduction to Data Structures

This document is created for practicing Retrieval-Augmented Generation (RAG). It contains structured text about common data structures and algorithms. Each section provides definitions, characteristics, and example use cases. The document is intentionally verbose to provide enough content for chunking and retrieval exercises. You can use this PDF to test embedding, chunking, vector search, and retrieval workflows.

Arrays (Page 1)

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element. Common operations are traversal, insertion, deletion, and searching.

Linked Lists (Page 1)

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. Types: singly linked list, doubly linked list, circular linked list.

Stacks (Page 1)

A stack is a linear data structure that follows the LIFO (Last In First Out) principle. Operations: push, pop, peek. Applications: expression evaluation, backtracking, function calls.

Queues (Page 1)

A queue is a linear structure which follows FIFO (First In First Out). Types: simple queue, circular queue, priority queue, deque. Applications: scheduling, buffering, breadth-first search.

Trees (Page 1)

A tree is a non-linear data structure made up of nodes connected by edges. Special cases: binary trees, binary search trees (BST), AVL trees, heaps. Applications: indexing, parsing, search optimization.

Graphs (Page 1)

A graph consists of a finite set of vertices and edges. Types: directed, undirected, weighted, unweighted, cyclic, acyclic. Applications: social networks, routing algorithms, recommendation systems.

Sorting Algorithms (Page 1)

Sorting is the process of arranging data in a particular format. Common algorithms: bubble sort, insertion sort, merge sort, quicksort, heapsort.

Searching Algorithms (Page 1)

Searching refers to finding the location of a given element in a list. Algorithms: linear search, binary search, depth-first search, breadth-first search.

Arrays (Page 2)

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element. Common operations are traversal, insertion, deletion, and searching.

Linked Lists (Page 2)

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. Types: singly linked list, doubly linked list, circular linked list.

Stacks (Page 2)

A stack is a linear data structure that follows the LIFO (Last In First Out) principle. Operations: push, pop, peek. Applications: expression evaluation, backtracking, function calls.

Queues (Page 2)

A queue is a linear structure which follows FIFO (First In First Out). Types: simple queue, circular queue, priority queue, deque. Applications: scheduling, buffering, breadth-first search.

Trees (Page 2)

A tree is a non-linear data structure made up of nodes connected by edges. Special cases: binary trees, binary search trees (BST), AVL trees, heaps. Applications: indexing, parsing, search optimization.

Graphs (Page 2)

A graph consists of a finite set of vertices and edges. Types: directed, undirected, weighted, unweighted, cyclic, acyclic. Applications: social networks, routing algorithms, recommendation systems.

Sorting Algorithms (Page 2)

Sorting is the process of arranging data in a particular format. Common algorithms: bubble sort, insertion sort, merge sort, quicksort, heapsort.

Searching Algorithms (Page 2)

Searching refers to finding the location of a given element in a list. Algorithms: linear search, binary search, depth-first search, breadth-first search.

Arrays (Page 3)

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element. Common operations are traversal, insertion, deletion, and searching.

Linked Lists (Page 3)

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. Types: singly linked list, doubly linked list, circular linked list.

Stacks (Page 3)

A stack is a linear data structure that follows the LIFO (Last In First Out) principle. Operations: push, pop, peek. Applications: expression evaluation, backtracking, function calls.

Queues (Page 3)

A queue is a linear structure which follows FIFO (First In First Out). Types: simple queue, circular queue, priority queue, deque. Applications: scheduling, buffering, breadth-first search.

Trees (Page 3)

A tree is a non-linear data structure made up of nodes connected by edges. Special cases: binary trees, binary search trees (BST), AVL trees, heaps. Applications: indexing, parsing, search optimization.

Graphs (Page 3)

A graph consists of a finite set of vertices and edges. Types: directed, undirected, weighted, unweighted, cyclic, acyclic. Applications: social networks, routing algorithms, recommendation systems.

Sorting Algorithms (Page 3)

Sorting is the process of arranging data in a particular format. Common algorithms: bubble sort, insertion sort, merge sort, quicksort, heapsort.

Searching Algorithms (Page 3)

Searching refers to finding the location of a given element in a list. Algorithms: linear search, binary search, depth-first search, breadth-first search.

Arrays (Page 4)

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element. Common operations are traversal, insertion, deletion, and searching.

Linked Lists (Page 4)

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. Types: singly linked list, doubly linked list, circular linked list.

Stacks (Page 4)

A stack is a linear data structure that follows the LIFO (Last In First Out) principle. Operations: push, pop, peek. Applications: expression evaluation, backtracking, function calls.

Queues (Page 4)

A queue is a linear structure which follows FIFO (First In First Out). Types: simple queue, circular queue, priority queue, deque. Applications: scheduling, buffering, breadth-first search.

Trees (Page 4)

A tree is a non-linear data structure made up of nodes connected by edges. Special cases: binary trees, binary search trees (BST), AVL trees, heaps. Applications: indexing, parsing, search optimization.

Graphs (Page 4)

A graph consists of a finite set of vertices and edges. Types: directed, undirected, weighted, unweighted, cyclic, acyclic. Applications: social networks, routing algorithms, recommendation systems.

Sorting Algorithms (Page 4)

Sorting is the process of arranging data in a particular format. Common algorithms: bubble sort, insertion sort, merge sort, quicksort, heapsort.

Searching Algorithms (Page 4)

Searching refers to finding the location of a given element in a list. Algorithms: linear search, binary search, depth-first search, breadth-first search.

Arrays (Page 5)

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element. Common operations are traversal, insertion, deletion, and searching.

Linked Lists (Page 5)

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. Types: singly linked list, doubly linked list, circular linked list.

Stacks (Page 5)

A stack is a linear data structure that follows the LIFO (Last In First Out) principle. Operations: push, pop, peek. Applications: expression evaluation, backtracking, function calls.

Queues (Page 5)

A queue is a linear structure which follows FIFO (First In First Out). Types: simple queue, circular queue, priority queue, deque. Applications: scheduling, buffering, breadth-first search.

Trees (Page 5)

A tree is a non-linear data structure made up of nodes connected by edges. Special cases: binary trees, binary search trees (BST), AVL trees, heaps. Applications: indexing, parsing, search optimization.

Graphs (Page 5)

A graph consists of a finite set of vertices and edges. Types: directed, undirected, weighted, unweighted, cyclic, acyclic. Applications: social networks, routing algorithms, recommendation systems.

Sorting Algorithms (Page 5)

Sorting is the process of arranging data in a particular format. Common algorithms: bubble sort, insertion sort, merge sort, quicksort, heapsort.

Searching Algorithms (Page 5)

Searching refers to finding the location of a given element in a list. Algorithms: linear search, binary search, depth-first search, breadth-first search.

Arrays (Page 6)

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element. Common operations are traversal, insertion, deletion, and searching.

Linked Lists (Page 6)

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. Types: singly linked list, doubly linked list, circular linked list.

Stacks (Page 6)

A stack is a linear data structure that follows the LIFO (Last In First Out) principle. Operations: push, pop, peek. Applications: expression evaluation, backtracking, function calls.

Queues (Page 6)

A queue is a linear structure which follows FIFO (First In First Out). Types: simple queue, circular queue, priority queue, deque. Applications: scheduling, buffering, breadth-first search.

Trees (Page 6)

A tree is a non-linear data structure made up of nodes connected by edges. Special cases: binary trees, binary search trees (BST), AVL trees, heaps. Applications: indexing, parsing, search optimization.

Graphs (Page 6)

A graph consists of a finite set of vertices and edges. Types: directed, undirected, weighted, unweighted, cyclic, acyclic. Applications: social networks, routing algorithms, recommendation systems.

Sorting Algorithms (Page 6)

Sorting is the process of arranging data in a particular format. Common algorithms: bubble sort, insertion sort, merge sort, quicksort, heapsort.

Searching Algorithms (Page 6)

Searching refers to finding the location of a given element in a list. Algorithms: linear search, binary search, depth-first search, breadth-first search.

Arrays (Page 7)

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element. Common operations are traversal, insertion, deletion, and searching.

Linked Lists (Page 7)

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. Types: singly linked list, doubly linked list, circular linked list.

Stacks (Page 7)

A stack is a linear data structure that follows the LIFO (Last In First Out) principle. Operations: push, pop, peek. Applications: expression evaluation, backtracking, function calls.

Queues (Page 7)

A queue is a linear structure which follows FIFO (First In First Out). Types: simple queue, circular queue, priority queue, deque. Applications: scheduling, buffering, breadth-first search.

Trees (Page 7)

A tree is a non-linear data structure made up of nodes connected by edges. Special cases: binary trees, binary search trees (BST), AVL trees, heaps. Applications: indexing, parsing, search optimization.

Graphs (Page 7)

A graph consists of a finite set of vertices and edges. Types: directed, undirected, weighted, unweighted, cyclic, acyclic. Applications: social networks, routing algorithms, recommendation systems.

Sorting Algorithms (Page 7)

Sorting is the process of arranging data in a particular format. Common algorithms: bubble sort, insertion sort, merge sort, quicksort, heapsort.

Searching Algorithms (Page 7)

Searching refers to finding the location of a given element in a list. Algorithms: linear search, binary search, depth-first search, breadth-first search.

Arrays (Page 8)

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element. Common operations are traversal, insertion, deletion, and searching.

Linked Lists (Page 8)

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. Types: singly linked list, doubly linked list, circular linked list.

Stacks (Page 8)

A stack is a linear data structure that follows the LIFO (Last In First Out) principle. Operations: push, pop, peek. Applications: expression evaluation, backtracking, function calls.

Queues (Page 8)

A queue is a linear structure which follows FIFO (First In First Out). Types: simple queue, circular queue, priority queue, deque. Applications: scheduling, buffering, breadth-first search.

Trees (Page 8)

A tree is a non-linear data structure made up of nodes connected by edges. Special cases: binary trees, binary search trees (BST), AVL trees, heaps. Applications: indexing, parsing, search optimization.

Graphs (Page 8)

A graph consists of a finite set of vertices and edges. Types: directed, undirected, weighted, unweighted, cyclic, acyclic. Applications: social networks, routing algorithms, recommendation systems.

Sorting Algorithms (Page 8)

Sorting is the process of arranging data in a particular format. Common algorithms: bubble sort, insertion sort, merge sort, quicksort, heapsort.

Searching Algorithms (Page 8)

Searching refers to finding the location of a given element in a list. Algorithms: linear search, binary search, depth-first search, breadth-first search.

Arrays (Page 9)

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element. Common operations are traversal, insertion, deletion, and searching.

Linked Lists (Page 9)

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. Types: singly linked list, doubly linked list, circular linked list.

Stacks (Page 9)

A stack is a linear data structure that follows the LIFO (Last In First Out) principle. Operations: push, pop, peek. Applications: expression evaluation, backtracking, function calls.

Queues (Page 9)

A queue is a linear structure which follows FIFO (First In First Out). Types: simple queue, circular queue, priority queue, deque. Applications: scheduling, buffering, breadth-first search.

Trees (Page 9)

A tree is a non-linear data structure made up of nodes connected by edges. Special cases: binary trees, binary search trees (BST), AVL trees, heaps. Applications: indexing, parsing, search optimization.

Graphs (Page 9)

A graph consists of a finite set of vertices and edges. Types: directed, undirected, weighted, unweighted, cyclic, acyclic. Applications: social networks, routing algorithms, recommendation systems.

Sorting Algorithms (Page 9)

Sorting is the process of arranging data in a particular format. Common algorithms: bubble sort, insertion sort, merge sort, quicksort, heapsort.

Searching Algorithms (Page 9)

Searching refers to finding the location of a given element in a list. Algorithms: linear search, binary search, depth-first search, breadth-first search.

Arrays (Page 10)

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element. Common operations are traversal, insertion, deletion, and searching.

Linked Lists (Page 10)

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. Types: singly linked list, doubly linked list, circular linked list.

Stacks (Page 10)

A stack is a linear data structure that follows the LIFO (Last In First Out) principle. Operations: push, pop, peek. Applications: expression evaluation, backtracking, function calls.

Queues (Page 10)

A queue is a linear structure which follows FIFO (First In First Out). Types: simple queue, circular queue, priority queue, deque. Applications: scheduling, buffering, breadth-first search.

Trees (Page 10)

A tree is a non-linear data structure made up of nodes connected by edges. Special cases: binary trees, binary search trees (BST), AVL trees, heaps. Applications: indexing, parsing, search optimization.

Graphs (Page 10)

A graph consists of a finite set of vertices and edges. Types: directed, undirected, weighted, unweighted, cyclic, acyclic. Applications: social networks, routing algorithms, recommendation systems.

Sorting Algorithms (Page 10)

Sorting is the process of arranging data in a particular format. Common algorithms: bubble sort, insertion sort, merge sort, quicksort, heapsort.

Searching Algorithms (Page 10)

Searching refers to finding the location of a given element in a list. Algorithms: linear search, binary search, depth-first search, breadth-first search.

Arrays (Page 11)

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element. Common operations are traversal, insertion, deletion, and searching.

Linked Lists (Page 11)

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. Types: singly linked list, doubly linked list, circular linked list.

Stacks (Page 11)

A stack is a linear data structure that follows the LIFO (Last In First Out) principle. Operations: push, pop, peek. Applications: expression evaluation, backtracking, function calls.

Queues (Page 11)

A queue is a linear structure which follows FIFO (First In First Out). Types: simple queue, circular queue, priority queue, deque. Applications: scheduling, buffering, breadth-first search.

Trees (Page 11)

A tree is a non-linear data structure made up of nodes connected by edges. Special cases: binary trees, binary search trees (BST), AVL trees, heaps. Applications: indexing, parsing, search optimization.

Graphs (Page 11)

A graph consists of a finite set of vertices and edges. Types: directed, undirected, weighted, unweighted, cyclic, acyclic. Applications: social networks, routing algorithms, recommendation systems.

Sorting Algorithms (Page 11)

Sorting is the process of arranging data in a particular format. Common algorithms: bubble sort, insertion sort, merge sort, quicksort, heapsort.

Searching Algorithms (Page 11)

Searching refers to finding the location of a given element in a list. Algorithms: linear search, binary search, depth-first search, breadth-first search.

Arrays (Page 12)

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element. Common operations are traversal, insertion, deletion, and searching.

Linked Lists (Page 12)

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. Types: singly linked list, doubly linked list, circular linked list.

Stacks (Page 12)

A stack is a linear data structure that follows the LIFO (Last In First Out) principle. Operations: push, pop, peek. Applications: expression evaluation, backtracking, function calls.

Queues (Page 12)

A queue is a linear structure which follows FIFO (First In First Out). Types: simple queue, circular queue, priority queue, deque. Applications: scheduling, buffering, breadth-first search.

Trees (Page 12)

A tree is a non-linear data structure made up of nodes connected by edges. Special cases: binary trees, binary search trees (BST), AVL trees, heaps. Applications: indexing, parsing, search optimization.

Graphs (Page 12)

A graph consists of a finite set of vertices and edges. Types: directed, undirected, weighted, unweighted, cyclic, acyclic. Applications: social networks, routing algorithms, recommendation systems.

Sorting Algorithms (Page 12)

Sorting is the process of arranging data in a particular format. Common algorithms: bubble sort, insertion sort, merge sort, quicksort, heapsort.

Searching Algorithms (Page 12)

Searching refers to finding the location of a given element in a list. Algorithms: linear search, binary search, depth-first search, breadth-first search.

Arrays (Page 13)

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element. Common operations are traversal, insertion, deletion, and searching.

Linked Lists (Page 13)

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. Types: singly linked list, doubly linked list, circular linked list.

Stacks (Page 13)

A stack is a linear data structure that follows the LIFO (Last In First Out) principle. Operations: push, pop, peek. Applications: expression evaluation, backtracking, function calls.

Queues (Page 13)

A queue is a linear structure which follows FIFO (First In First Out). Types: simple queue, circular queue, priority queue, deque. Applications: scheduling, buffering, breadth-first search.

Trees (Page 13)

A tree is a non-linear data structure made up of nodes connected by edges. Special cases: binary trees, binary search trees (BST), AVL trees, heaps. Applications: indexing, parsing, search optimization.

Graphs (Page 13)

A graph consists of a finite set of vertices and edges. Types: directed, undirected, weighted, unweighted, cyclic, acyclic. Applications: social networks, routing algorithms, recommendation systems.

Sorting Algorithms (Page 13)

Sorting is the process of arranging data in a particular format. Common algorithms: bubble sort, insertion sort, merge sort, quicksort, heapsort.

Searching Algorithms (Page 13)

Searching refers to finding the location of a given element in a list. Algorithms: linear search, binary search, depth-first search, breadth-first search.

Arrays (Page 14)

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element. Common operations are traversal, insertion, deletion, and searching.

Linked Lists (Page 14)

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. Types: singly linked list, doubly linked list, circular linked list.

Stacks (Page 14)

A stack is a linear data structure that follows the LIFO (Last In First Out) principle. Operations: push, pop, peek. Applications: expression evaluation, backtracking, function calls.

Queues (Page 14)

A queue is a linear structure which follows FIFO (First In First Out). Types: simple queue, circular queue, priority queue, deque. Applications: scheduling, buffering, breadth-first search.

Trees (Page 14)

A tree is a non-linear data structure made up of nodes connected by edges. Special cases: binary trees, binary search trees (BST), AVL trees, heaps. Applications: indexing, parsing, search optimization.

Graphs (Page 14)

A graph consists of a finite set of vertices and edges. Types: directed, undirected, weighted, unweighted, cyclic, acyclic. Applications: social networks, routing algorithms, recommendation systems.

Sorting Algorithms (Page 14)

Sorting is the process of arranging data in a particular format. Common algorithms: bubble sort, insertion sort, merge sort, quicksort, heapsort.

Searching Algorithms (Page 14)

Searching refers to finding the location of a given element in a list. Algorithms: linear search, binary search, depth-first search, breadth-first search.

Arrays (Page 15)

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element. Common operations are traversal, insertion, deletion, and searching.

Linked Lists (Page 15)

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. Types: singly linked list, doubly linked list, circular linked list.

Stacks (Page 15)

A stack is a linear data structure that follows the LIFO (Last In First Out) principle. Operations: push, pop, peek. Applications: expression evaluation, backtracking, function calls.

Queues (Page 15)

A queue is a linear structure which follows FIFO (First In First Out). Types: simple queue, circular queue, priority queue, deque. Applications: scheduling, buffering, breadth-first search.

Trees (Page 15)

A tree is a non-linear data structure made up of nodes connected by edges. Special cases: binary trees, binary search trees (BST), AVL trees, heaps. Applications: indexing, parsing, search optimization.

Graphs (Page 15)

A graph consists of a finite set of vertices and edges. Types: directed, undirected, weighted, unweighted, cyclic, acyclic. Applications: social networks, routing algorithms, recommendation systems.

Sorting Algorithms (Page 15)

Sorting is the process of arranging data in a particular format. Common algorithms: bubble sort, insertion sort, merge sort, quicksort, heapsort.

Searching Algorithms (Page 15)

Searching refers to finding the location of a given element in a list. Algorithms: linear search, binary search, depth-first search, breadth-first search.

Arrays (Page 16)

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element. Common operations are traversal, insertion, deletion, and searching.

Linked Lists (Page 16)

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. Types: singly linked list, doubly linked list, circular linked list.

Stacks (Page 16)

A stack is a linear data structure that follows the LIFO (Last In First Out) principle. Operations: push, pop, peek. Applications: expression evaluation, backtracking, function calls.

Queues (Page 16)

A queue is a linear structure which follows FIFO (First In First Out). Types: simple queue, circular queue, priority queue, deque. Applications: scheduling, buffering, breadth-first search.

Trees (Page 16)

A tree is a non-linear data structure made up of nodes connected by edges. Special cases: binary trees, binary search trees (BST), AVL trees, heaps. Applications: indexing, parsing, search optimization.

Graphs (Page 16)

A graph consists of a finite set of vertices and edges. Types: directed, undirected, weighted, unweighted, cyclic, acyclic. Applications: social networks, routing algorithms, recommendation systems.

Sorting Algorithms (Page 16)

Sorting is the process of arranging data in a particular format. Common algorithms: bubble sort, insertion sort, merge sort, quicksort, heapsort.

Searching Algorithms (Page 16)

Searching refers to finding the location of a given element in a list. Algorithms: linear search, binary search, depth-first search, breadth-first search.

Arrays (Page 17)

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element. Common operations are traversal, insertion, deletion, and searching.

Linked Lists (Page 17)

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. Types: singly linked list, doubly linked list, circular linked list.

Stacks (Page 17)

A stack is a linear data structure that follows the LIFO (Last In First Out) principle. Operations: push, pop, peek. Applications: expression evaluation, backtracking, function calls.

Queues (Page 17)

A queue is a linear structure which follows FIFO (First In First Out). Types: simple queue, circular queue, priority queue, deque. Applications: scheduling, buffering, breadth-first search.

Trees (Page 17)

A tree is a non-linear data structure made up of nodes connected by edges. Special cases: binary trees, binary search trees (BST), AVL trees, heaps. Applications: indexing, parsing, search optimization.

Graphs (Page 17)

A graph consists of a finite set of vertices and edges. Types: directed, undirected, weighted, unweighted, cyclic, acyclic. Applications: social networks, routing algorithms, recommendation systems.

Sorting Algorithms (Page 17)

Sorting is the process of arranging data in a particular format. Common algorithms: bubble sort, insertion sort, merge sort, quicksort, heapsort.

Searching Algorithms (Page 17)

Searching refers to finding the location of a given element in a list. Algorithms: linear search, binary search, depth-first search, breadth-first search.

Arrays (Page 18)

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element. Common operations are traversal, insertion, deletion, and searching.

Linked Lists (Page 18)

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. Types: singly linked list, doubly linked list, circular linked list.

Stacks (Page 18)

A stack is a linear data structure that follows the LIFO (Last In First Out) principle. Operations: push, pop, peek. Applications: expression evaluation, backtracking, function calls.

Queues (Page 18)

A queue is a linear structure which follows FIFO (First In First Out). Types: simple queue, circular queue, priority queue, deque. Applications: scheduling, buffering, breadth-first search.

Trees (Page 18)

A tree is a non-linear data structure made up of nodes connected by edges. Special cases: binary trees, binary search trees (BST), AVL trees, heaps. Applications: indexing, parsing, search optimization.

Graphs (Page 18)

A graph consists of a finite set of vertices and edges. Types: directed, undirected, weighted, unweighted, cyclic, acyclic. Applications: social networks, routing algorithms, recommendation systems.

Sorting Algorithms (Page 18)

Sorting is the process of arranging data in a particular format. Common algorithms: bubble sort, insertion sort, merge sort, quicksort, heapsort.

Searching Algorithms (Page 18)

Searching refers to finding the location of a given element in a list. Algorithms: linear search, binary search, depth-first search, breadth-first search.

Arrays (Page 19)

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element. Common operations are traversal, insertion, deletion, and searching.

Linked Lists (Page 19)

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. Types: singly linked list, doubly linked list, circular linked list.

Stacks (Page 19)

A stack is a linear data structure that follows the LIFO (Last In First Out) principle. Operations: push, pop, peek. Applications: expression evaluation, backtracking, function calls.

Queues (Page 19)

A queue is a linear structure which follows FIFO (First In First Out). Types: simple queue, circular queue, priority queue, deque. Applications: scheduling, buffering, breadth-first search.

Trees (Page 19)

A tree is a non-linear data structure made up of nodes connected by edges. Special cases: binary trees, binary search trees (BST), AVL trees, heaps. Applications: indexing, parsing, search optimization.

Graphs (Page 19)

A graph consists of a finite set of vertices and edges. Types: directed, undirected, weighted, unweighted, cyclic, acyclic. Applications: social networks, routing algorithms, recommendation systems.

Sorting Algorithms (Page 19)

Sorting is the process of arranging data in a particular format. Common algorithms: bubble sort, insertion sort, merge sort, quicksort, heapsort.

Searching Algorithms (Page 19)

Searching refers to finding the location of a given element in a list. Algorithms: linear search, binary search, depth-first search, breadth-first search.

Arrays (Page 20)

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element. Common operations are traversal, insertion, deletion, and searching.

Linked Lists (Page 20)

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. Types: singly linked list, doubly linked list, circular linked list.

Stacks (Page 20)

A stack is a linear data structure that follows the LIFO (Last In First Out) principle. Operations: push, pop, peek. Applications: expression evaluation, backtracking, function calls.

Queues (Page 20)

A queue is a linear structure which follows FIFO (First In First Out). Types: simple queue, circular queue, priority queue, deque. Applications: scheduling, buffering, breadth-first search.

Trees (Page 20)

A tree is a non-linear data structure made up of nodes connected by edges. Special cases: binary trees, binary search trees (BST), AVL trees, heaps. Applications: indexing, parsing, search optimization.

Graphs (Page 20)

A graph consists of a finite set of vertices and edges. Types: directed, undirected, weighted, unweighted, cyclic, acyclic. Applications: social networks, routing algorithms, recommendation systems.

Sorting Algorithms (Page 20)

Sorting is the process of arranging data in a particular format. Common algorithms: bubble sort, insertion sort, merge sort, quicksort, heapsort.

Searching Algorithms (Page 20)

Searching refers to finding the location of a given element in a list. Algorithms: linear search, binary search, depth-first search, breadth-first search.