Definitions and Concepts for AQA Computer Science A-level

Topic 2: Fundamentals of Data Structures

2.1 Data Structures and Abstract Data Types

2.1.1 Data Structures

Data Structures: A data structure is a format used to store, organise and manage data in a way that allows efficient access and modification for the needs of the program.

2.1.2 Single- and Multi-Dimensional Arrays

Arrays: A data structure for storing a finite, ordered set of data of the same data type within a single identifier.

Multi-Dimensional Arrays: An array where each data item is located using multiple indices.

Single-Dimensional Arrays: An array where each data item can be located using a single index.

2.1.3 Fields, Records and Files

Binary File: An organised collection of records where data is stored in binary.

Fields: A single item of data.

Records: A data structure that stores multiple fields, organised based on attributes, within a single line of a file.

Text File: An organised collection of records where data is stored in human-readable characters.

2.1.4 Abstract Data Types/Data Structures

Dictionaries: A data structure consisting of set of keys that are mapped to their corresponding values.

Dynamic Structures: A data structure whose memory allocation size can change throughout the execution of the program.

Graphs: A data structure consisting of a set of vertices/nodes connected by a set of edges/arcs.

Hash Tables: A data structure where a hashing algorithm creates a mapping between keys and values. The data item can then be directly accessed by recalculation, without any search.

Queues: A first-in-first-out (FIFO) data structure. The first item added/pushed on to the queue is the first to be removed/popped off.

Stacks: A last-in-first-out (LIFO) data structure. The last item added/pushed is the first to be removed/popped off.

Static Structures: A data structure that is allocated a fixed amount of memory space, which does not change throughout the execution of the program.

Trees: A data structure that uses a set of linked nodes to form a hierarchical structure starting at a root node. Each node is a child/sub-node of a parent node.

Vectors: A data structure representing a quantity with both magnitude and direction. It can be represented as a list, function or geometric point.

2.3 Stacks

Peek/Top: An operation that allows the user to view the top element of the stack without modifying it.

Pop: An operation that removes the most recently added element that was not yet removed from the stack.

Push: An operation that adds an element to the top of the stack.

2.4 Graphs

Adjacency List: A representation of a graph by storing a list of connected nodes to each node.

Adjacency Matrix: A matrix representation of a graph that stores the edges connecting all possible nodes.

Directed Graphs: A graph where the order of the vertices paired in an edge matter. The edges are one way.

Edge/Arc: A connection that represents a relationship between two nodes.

Undirected Graphs: A graph where the order of the vertices paired in an edge does not matter. The edges are bidirectional.

Vertex/Node: The representation of an object on a graph that is capable of being related to

other such objects.

Weighted Graphs: A graph where each edge/arc has an associated value (known as its weight).

2.5 Trees (Including Binary Trees)

Binary Trees: A rooted tree in which each node has, at most, 2 children.✢

Rooted Trees: A tree in which one node has been designated as the root.✢

Root Node: The only node in a rooted tree without a parent.✢

Trees: A connected, undirected graph with no cycles.✢

2.6 Hash Tables

Collisions: The phenomenon when two key values compute to the same hash.✢

Hashing Algorithms: An algorithm that calculates a value to determine the unique index where a data item is to be stored in a hash table.

Rehashing: The process of rerunning the hashing algorithm in the event of a collision.

2.8 Vectors

Convex Combination of two Vectors: Any vector that can be expressed as a linear combination of the two vectors.

Dot/Scalar Product of two Vectors: The sum of the products of components with the same index of two vectors.

Scalar-Vector Multiplication: An operation that multiplies all the components of a vector by the same scalar quantity.

Vector Addition: An operation that adds two vectors by component-wise addition, resulting in another vector as the output.

Definitions with a '✢' taken from AQA AS and A-level Computer Science specification version 1.5