10 most known algorithms

Quicksort

Mergesort

Factorial

Longest increasing subsequence

Bfs

Dfs

Matrix multiplication

Traveling salesman problem

Dijktras

Min cut max flow problem

Datastructures

Stack

Queue

Arrays

Linked list

Tree

Graph

Priority queue

Maps also dictionary

Sets

Tuples

First start with Linear data structures and algorithms.

Arrays

Linked List

Stack

Queues

Then move to basic algorithms :

Sorting - Merge Sort, Insertion Sort, Quick Sort, Number of inversions

Matrix Multiplication (just know the algo if not implement it)

Prime Sieving

Modular Math including multiplication and division

Euclidean Algorithm for GCD, Modular Inverse, Fast Exponentiation

Fibonacci number with matrix multiplication

Probability distribution and expected value

Stats - Mean, Median, Variance, Bayes theorem

The one can learn some popular algorithmic techniques

Divide and Conquer - Binary Search, Maximum Subarray

Greedy Algorithms - Activity Selection, Huffman encoding

Dynamic Programming - Matrix Chain Multiplication, Knapsack,

Linear Programming - Variable Maximisation, Linear time sorting

String Algorithms - Manacher, LCS, Edit Distance

Then comes some typical non-linear data structures:

Trees - Binary Tree, General Tree, Lowest Common Ancestor

Binary Search Tree - Inorder Traversal, Level order traversal, finding kth largest element, diameter, depth, number of nodes, etc.

Heaps - Array Implementation, Heapify, Heap Sort

Union Find

Hash Table - Linear Probing, Open addressing, Collision avoidance

Then you can learn about Graphs:

Adjacency List, Adjacency Matrix, Weighted Edge Graphs

Basic Traversal algos - Breadth First Search, Depth First Search, etc

Shortest Path Finding Algorithm - Dijkstra, Floyd Warshal, Bellman Ford

Minimum Spanning Tree - Kruskal's Algo, Prim's Algo

Advance Tree and Graph :

Balanced Trees - AVL, Red-Black

Heavy Light Decomposition, B+ Trees, Quad Tree

Advance Graph - Min Cut, Max Flow

Maximum Matching - Hall's Marriage

Hamiltonian Cycle

Edge Graphs / Line Graphs

Strongly Connected Components

Dominant Sub-Graph, Vertex Cover, Travelling Salesman - Approx algos

Advance String Algorithms :

If you want to learn more and delve deep read more.

Advance Tree and Graph :

Balanced Trees - AVL, Red-Black

Heavy Light Decomposition, B+ Trees, Quad Tree

Advance Graph - Min Cut, Max Flow

Maximum Matching - Hall's Marriage

Hamiltonian Cycle

Edge Graphs / Line Graphs

Strongly Connected Components

Dominant Sub-Graph, Vertex Cover, Travelling Salesman - Approx algos

Advance String Algorithms :

Knuth Morris Pratt Algorithm

Rabin Karp Algorithm

Tries and Compressed Tries

Prefix Trees, Suffix Trees, Suffix Automation - Ukkonen Algorithm

Advance Math:

Fast Fourier Tranformation

Primality Testing

Computational Geometry - Closest point pair, Voronoi diagram, Convex Hull

General Advance topics :

Iterating through all combination / permutation

Bit manipulation

Algorithms

Graph Search (BFS, DFS) - DFS is especially important as it gives more structural information about the graph than one would initially think

Sorting ( Comparison Sorts and Bucketing Schemes)

The general method of Dynamic Programming

Matching algorithms and Network Flows

Regular Expressions and String Matching

Data Structures

Graphs - Trees are especially important

Maps

Heaps

Stacks / Queues

Tries

Extras include

- Greedy algorithms

- Probabilistic Methods

- Approximation Algorithms

Here is a list of common data structures

Array data structure and Sparse array

Linked list and Doubly linked list

Stack (abstract data type) and Queue (abstract data type)

and Double-ended queue

Binary tree and Treap

Red–black tree

Heap (data structure)

String (computer science) and Trie

B tree and B+ tree

Graph (abstract data type)

Hash table and Associative array

For most data structure, You need to learn how to perform Insertion, Deletion, traversal, Union, Search and Resize operations. Note that some data structures have very specific operations.

A List of common algorithms

Linear search

Binary search algorithm and Ternary search

Merge algorithm and Merge sort

Insertion sort and Timsort

Heapsort and Quicksort

Selection algorithm

Graph traversal (DFS, BFS)

Dijkstra's algorithm and Bellman–Ford algorithm

Topological sorting

Disjoint-set data structure (Union Find) and Minimum spanning tree

Flow network Algorithms

Binary Search

Quicksort

Merge Sort

Suffix Array

Knuth-Morris-Pratt Algorithm (KMP)

Rabin-Karp Algorithm

Tries

Depth First Traversal of a graph

Breadth First Traversal of a graph

Dijkstra's Algorithm

Binary Indexed Tree

Segment Tree (with lazy propagation)

Persistent Segment Tree

Z algorithm

Floyd Warshall Algorithm

Sparse Table(RMQ)

Heap / Priority Queue / Heapsort

Modular Multiplicative Inverse

nCr % M

Suffix Automaton

Lowest Common Ancestor

Counting Inversions

Euclid's Extended Algorithm

Suffix Tree

Dynamic Programming

Basic Data Structures

Logarithmic Exponentiation

Graphs

Minimum Spanning Tree

Efficient Prime Factorization

Combinatorics

Union Find/Disjoint Set

Knapsack problem

Aho-Corasick String Matching Algorithm

Strongly Connected Components

Bellman Ford algorithm

Heavy-light Decomposition

Convex Hull

Jarvis Algorithm Implementation

Line Intersection

Sieve of Erastothenes

Interval Tree

Counting Sort

Probabilities

Matrix Exponentiation

Network flow

Max Flow(Ford-Fulkerson)

K-d tree :

Deque

Binary Search Tree

Quick Select

Treap/Cartesian Tree

Game Theory

STL

Maximum Bipartite Matching

Manacher's Algorithm

Miller-Rabin Primality Test

Stable Marriage Problem

Hungarian Algorithm

Sweep line Algorithm

LCP

Gaussian Elimination

Pollard Rho Integer Factorization

Topological Sorting

Detecting Cycles in a Graph : Directed , Undirected

Geometry

Backtracking

Eulerian and Hamiltonian Paths

Graph Coloring

Meet in the Middle

Arbitrary Precision Integer(BigInt)

Radix Sort

Bucket Sort

Johnson's Algorithm

Maximal Matching in a General Graph

Recursion

Inclusion and Exclusion Principle

Coordinate Compression

Sqrt-Decomposition

Link-Cut Tree

Euler's Totient Function

Burnside Lemma

Edit/Levenshtein Distance

Branch and Bound

Math for Competitive Programming

Mo's Algorithm

If you want specific algorithms, my top 10 would be:

**Dijkstra's** - depending on the type of contest, you might see basic pathfinding problems, or you might see problems with non-obvious reductions to pathfinding problems. Whenever you have a cost minimization problem with a (reasonably small) finite number of states, an initial state and a target state, you can look at it as a pathfinding problem.

**Bellman-Ford** is useful for pathfinding when edges may have negative costs. For example if you're navigating a maze with potions which boost health and hazards which lower it, Bellman-Ford would be a great approach.

**Floyd-Warshall** is useful for computing all paths. It is sometimes used in problems where you don't need all paths, because it's so easy to implement. It is slower than other pathfinding algorithms though, so whether Floyd-Warshall is an option depends on the graph size.

**Edmonds-Karp** for max flow/min cut problems. One common application is bipartite matching problems. For example, given N people, M food items, and a list of each person's food allergies, how many people can you feed?

**The Hungarian** algorithm for assignment problems. Similar to the above, but in these problems the edges have weights, and we're maximizing the total weight rather than just the number of matchings.

**The sweep line "algorithm"** (more of a general approach really) is useful for various geometric problems, like the nearest pair problem. Also useful for a variety of intersection-related problems, like finding intersecting line segments, or conflicting calendar events.

**Graham scan** or another convex hull algorithm, for problems such as building a minimal fence to enclose animals.

An algorithm for finding strongly connected components, such as Tarjan's.

**Prim's algorithm** for minimum spanning trees.

**Knuth-Morris-Pratt** algorithm for string searching.

Other concepts worth studying, which aren't in the above list because they aren't specific algorithms:

Memoization/dynamic programming is quite useful. Some problems have obvious DP solutions, while others have very non-obvious ones which take practice to recognize.

**Binary search** is useful in many optimization problems, so make sure you're very comfortable implementing it.

**Combinatorial game** theory comes up now and then. I recommend Thomas Ferguson's introduction.

Tries are useful in a variety of text-related problems.

1. Arrays

Know how to use arrays. You will use them a lot.

Learn how to make them grow dynamically (for programming languages like C).

Know how to use a standard library array (example: vector)

Learn to use strings (they are arrays).

Know how to sort them with two good sorting algorithms (QuickSort and MergeSort)

Learn to search in an array. First linear search and then binary search.

Learn QuickSelect, it will be useful.

2. Linked Lists

Learn to create a linked list.

Know the basic operations: insert, print, delete.

You won't use linked lists a lot in the real word, but having a solid understanding of how they work will be of great use.

Know different versions of linked lists: circular and double, for example.

Know the advantages of a linked list vs an array

3-4. Stacks and Queues

Learn to implement stacks and queues with arrays and lists. Know the advantages.

For stacks, learn postfix conversion and evaluation. Also learn parenthesis matching. This will help you master stacks.

For queues, you should try to do some multithreading programming.

5. Trees

If you did a proper work with linked lists, trees should feel a little natural.

Learn to implement a binary tree (insert, delete, search).

Know the basic properties of a tree.

Learn at least one self-balancing tree (AVL, Splay, Red-Black).

I like a lot the Trie data structure. It has some pretty nice applications.

Another data structure used a lot is the heap.

Learn Breadth First Search and Depth First Search.

6. Hashmap

Hashmap is a really useful data structure.

Learn how to use them and how to implement one (so learn how to do a neat hashing function).

7. Graphs

Learn what is a graph and how to implement them with a matrices and lists.

Learn how to use them and how to implement one (so learn how to do a neat hashing function).

Learn how to find shortest path in a graph.

Implement DFS and BFS.

8-10. Algorithms

As I mentioned before, learn sorting and searching algorithms.

Learn Big O notation.

Search: MergeSort, QuickSort, HeapSort.

Diskstra. You won't use this, but it will give you a better understanding of graphs.

There are other nice algorithms that you might want to learn. Some are textbook algorithms (reverse string, substring, etc).