**BFS (breadth-first search)**: an algorithm that searches through all nodes connected to some starting node in a graph. Similar to **DFS**. Consider it like water spreading through a network of pipes - it will spread out all at once. *Can* find shortest paths in an unweighted graph.

**Binary search**: suppose you’re trying to find an element x in a sorted array. If x is less than the middle element, you know that x is in the first half of the array. Otherwise, x is in the second half. Repeat until you find x.

**Brute force**: trying all possibilities for the answer (for example, if you’re tasked with predicting who would win in a game, you could try iterating through every possible move sequence to find out whose win is forced).

**Combinatorics**: a subset of math that focuses on counting and techniques for counting (for example, counting the number of arrangements of people on a line where people of similar eye color must be together).

**DFS (depth-first search)**: an algorithm that searches through all nodes connected to some starting node in a graph. Similar to **BFS**. Consider it like the path you would take while trying to find your way out of a maze (with the right-hand rule). *Cannot* find shortest paths in any form of graph.

**Dijkstra**: an algorithm that finds the shortest path from some start node to every other node in a graph. Works for both directed and undirected graphs, but not with negative-weight edges.

**DSU (disjoint set union)**: a graph data structure that allows for fast insertion of edges and fast checking if two nodes are connected to each other.

**Dynamic programming**: a general problem solving mindset, where you eliminate useless information and use the answers for smaller versions of the task to build up to larger answers and eventually the full answer.

**Fenwick/segment tree**: two distinct data structures that work different ways but achieve a similar purpose: handling range queries quickly. For example, they can allow for quickly finding the sum of elements on some range [l, r] and quickly adding a value to any element.

**Greedy**: always making the immediately best decision, with no regard for how that decision will affect the future.

**Hashmap**: a data structure that uses hashing to map keys to values and allow for quick assignment (map[key] = value) and queries (get map[key]).

**Hashset**: a data structure that uses hashing to maintain a list of elements and: insert any element, ensure that no element is in the set more than once, and check if an element is in the set. Similar to **ordered set**, but doesn’t store its elements in sorted order.

**Kruskal/Prim**: two distinct algorithms that work different ways but achieve the same purpose: finding the **minimum spanning tree**.

**MST (minimum spanning tree)**: a tree with the minimum possible sum of edge weights that spans (connects) the whole graph.

**Ordered set**: a data structure that uses a balanced binary search tree to maintain a list of elements and: insert any element, ensure that no element is in the set more than once, and check if an element is in the set. Similar to **hashset**, but stores its elements in sorted order.

**Two pointers**: suppose you’re trying to find two elements that sum to x in a sorted array. As you increase the value of the first element (sweeping right), the value of [x - first element] (aka the desired second element) will only decrease, meaning that you can rule out any candidate second elements that you’ve already checked (and sweep left).