Algorithm	Essence	Time Complexity	Datastructures	Pardigm
Tarjan	Strongly connected components. Assign **lowlink** values to nodes and keep track if on **stack** or not. Same lowlink => same component. Do a DFS from "every" node. **Independent sets**.	O(V+E) (from DFS)	Boolean array (is on stack), stack, graph	Tree Search Algorithm
Dijkstras	Finds shortest path from root node to a specific node (or every other node in graph). Faster if implemented with fib- or hollow heap. General case can't handle negative edges due to **select**.	O(|E| + |V|log|V|)	Graph, priority queue (choose implementation)	Greedy
Bellman-Ford	Loop through edgelist **relaxing** every edge. if d[u] + c(u,v) < d[v] then d[v] = d [u] + c(u,v). Loops at most **n-1** times. If a relaxation occurs at n-1th iteration then **negative cycle** exists.	O(n^2) best case, O (n^3) worst case	Array, graph	DP
Goldberg-Tarjan	**Prepush-flow.** Intuitively like pouring a full bucket into a system and then looking for leaks + redirecting flow. Work with **excess flow**. Using operations **relabe**l, **saturated push** and **nonsaturated push**. Saturated push takes the longest time to perform. Push based on height of given node + the node it's pushing to.	O(|V|^2 * |E|)	Graph, priority queue (choose implementation)	Maxflow
Ford-Fulkerson	Method for finding max flow in graph. Not specified how to find augmented paths. Find **bottlenecks** and increment flow. If no s-t path exists then flow := maxflow. **Mincut theorem**.	O(f*|E|) or O(C*m)	Graph, residual graph	Maxflow
Simplex	Min/max for mathmatical models. Use of **slack** variables. **Basic/nonbasic**. Solve linear equation system.			
Edmond-Karp	Implementation of Ford-Fulkerson with use of BFS. Improves the time complexity by **elminating dependency** on maxflow constant.	O(|V|*|E|^2)	Graph, residual graph	Maxflow
DFS	Graph search. Searches depth over breadth. Will explore as far as possible along a path before backtracking. **Back-, forward-, cross-** and **tree edge.**	O(|V| + |E|)	Graph, Set, Stack	Graph search
BFS	Graph search. Searches breadth over depth. Will explore graph in **layers**. Like peeling an onion.	O(|V| + |E|)	Graph, Set, Queue	Graph search
Prims	Finds **MST** of weighted undirected graph. Builds tree one vertex at a time. Each step adds the cheapest connection to the tree.	O(|E| + |V|log|V|)	Heap, Graph, Priority Queue	Greedy
Kruskal	Finds **MST** of weighted undirected graph. Constructs a forest and connectes vertices that does not create cycles. **Union-Find.**	O(|E|log|V|)	Set, Union-Find	Greedy
Dynamic Programming	Saves previously calculated values, **memorization**. Improving time complexity but also increases space complexity.	Depends	-	-
Knapsack Algorithm	Dynamic algorithm for finding the optimal amount of groceries one can carry depending on weight while maximizing cost. Reduce problem to a matrix, precompute and backtrack through matrix to find solution steps. **Overlapping smaller problems.**	O(n*m)	Matrix, array	DP
Closest Points	Sort points on x- and y-coordinates. Divide points into sublists and solve independent smaller problems. Create set S sorted on y and compute closest points at midline. Iterative at small n.	O(n*log(n))	Array, sets	Divide and Conquer
Backtracking	Track back through smaller calculations in order to reach global min/max. Used in dynamic programming and others.	-	Matrix (?)	-
Gale-Shapley	Solves stable marriage problem by matching two sets where each item in these sets have a **preference list** over the other set.	O(n^2)	Array, inverted priority list	Greedy

Data Structure	Essence	Operations	Time Complexity	
Stack	Last in, First out	pop, push, search, space	O(1),O(1), O(n), O(n)	
Queue	First in, First out	pop, push, search, space	O(1),O(1), O(n), O(n)	
Binary Tree	Tree where each leaf has at most two children	search, delete, insert	O(logn), O(n), O(logn) => worst case O(n) for nonbalanced tree	
AVL-tree	Self balancing tree with rotations to keep the height equal among subtrees.	search, delete, insert	O(logn) for all	
Min/max Heap	Always min/max out first. Heap property (min): if P is parent of C, then key(P) =< key(C).	getMin, extractMin, decreaseKey	O(1), O(logn) (heapify), O(logn)	
Fibonacci Heap				
Hollow Heap	Improved Fibonacci Heap. Implemented with a tree or **DAG**. Many pointers. If **rank(n) = r**, then n has **at least F_r+3 - 1 descendants.** Time complexity is dependant of **link(v,w)** which i called when we call **delete_min().**	link, delete, delete_min, decrease_key, insert	All operations take O(1), except delete and delete_min => takes O(logn). *(log is log of phi)*	
Union-Find	Set with **disjoint** subsets. Can merge sets with **union** and keep track of subsets with parent array in **find**.	find, union	O(1), O(logn)	
Residiual Graph	Graph with **backward** edges. Good when redirecting flow in maxflow algorithms.	-	-	
DAG	Used in hollow heap for keeping track of nodes.	-	-	