COMP26120-S2 Algorithms and Data Structures (35)

Big-O (O) is *tight upper bound* - the most amount of time required (worst case performance).
Little-o (o) notation is used to describe an *loose upper bound* that cannot be tight.

Big Theta notation (Θ) is the *tightest bound* - the best of all the worst-case times.

Big Omega (Ω) is *tight lower bound* - the least amount of time required (best case).
Little Omega (ω) notation is used to describe an *loose lower bound* that cannot be tight.

Amortised Cost is the effort needed to preform each single action in a larger set, with total time divided by actions. It is not a magic tool, sometimes it does not make a difference or is not applicable, and it does not reflect the worst-case situation, nor does it reflect the total time needed as a whole.

Quick Sort work by a greedy divide-and-conquer. It separates the larger array into two smaller ones based on a (potentially) randomly selected divider.

Min-Heap is *essentially a binary tree* (not a search tree) where all children nodes have values larger than their sole parent. The smallest node will always be the root node. It is usually stored in an array, with children being 2n and 2n+1 (first location being 1 than 0). Heap is sorted by exchanging larger parents with its smallest child. Popping out an element will let the last value in array be filled in the first place and sort the heap.

Skip List is a data structure that has $O(\log n)$ complexity for searching. It is a multi-layer structure with higher layers having lower number of nodes (e.g. 50%) and serve as index by linking to the corresponding nodes in lower layers. Similar time complexity with AVL but it is simpler, faster and utilises less space.

Height-Balance Property exists when number of nodes on the left branch for each parent node is *at most* 1 node larger or smaller than that of the right branch. This property exists for all AVL trees or balanced trees. Do not necessarily exist with min-heap have depth higher than 2.

Adjacency List stores connections in an array(array()) manner. It usually takes less space when storing sparse graph compared to a Table.

Table stores connections in a two-dimensional NxN array. Usually have same size but with better speed for dense graphs compared to Adjacency list.

Sparse Graph is a graph that have much less edges compared to theoretical maximum.
Dense Graph is a graph that have an edge count close to the maximum edges possible.
There is no clear distinction or definition as of a graph is Sparse or Dense.

Simple Graph is an unweighted, undirected graph containing no graph loops (same node connection) or multiple edges. A simple graph may be either connected or disconnected and can have cycles.

Acyclic Graph is a graph that does not contain cycles (contain a path from one node to itself).

DFS and **BFS** uses different data structures. DFS uses a stack (recursion), while BFS uses a queue.

Tropological Sort is sorting aimed to get dependencies straight. It will put non depended node on the first and guarantees that later nodes will always have dependence requirements satisfied. It will not produce a result if there is a cycle in the graph.

Bellman-Ford is an algorithm that recursively find all shortest path from one node to all other nodes. It recursively computes the minimum length until nothing changes, which takes at most $|V| - 1$ times. This algorithm applies to all directed and weighted graphs, could contain negative edges but not negative cycles, or undirected graph with negative edges. It initially assumes all nodes needs infinity to reach and replace with better solutions.

A* Search is an algorithm to find shortest route towards a certain target. It relaxes nodes based on an estimate of distance of the total path, based on path already walked plus the best estimate length left.

Heuristic Function, opposite to an approximate or accurate function, is a quick shortcut to find a value, usually by trading optimality, completeness, accuracy, or precision for speed.

Admissibility of a heuristic function is the property that would never over-estimate.

Monotonicity is the property that a function value would only increase when the variable increase. It also implies the existence of admissibility.

Dijkstra is an algorithm that would find all shortest path from one node to all other nodes. It relaxes nodes based on the minimum length in priority queue. It does not allow any negative edge to appear as it would affect correctness. It uses loop invariant to ensure its correctness.

Minimum Spanning Tree (MST) is a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight. There may be more than one MST exist for a graph, generated by different algorithms.

Cut (S ,V-S) of an undirected graph G = (V,E), is a partition of V(as defined in CLRS Book). It divides graph into two disjoint sets of vertices on its either side.

Light edge is edge crossing a cut is light edge if its weight is the minimum of all the edge crossing the cut. Light edge is defined with respect to a particular Cut.

A cut **Respects** a set of edges A if no edge in A crosses the cut.

Safe edge is the edge which we can add to MST without any violation of MST's property. These are those edges which are the part of final MST.

Prim's Algorithm is the MST generation version of Dijkstra. It works by relaxing edges by selecting the best one within the priority queue contains all the edges that have a common edge with selected edges.

Kruskal's Algorithm is another greedy MST generation algorithm, pick up edges by choosing the one with least length and is safe to add.

Standard Form of a constraint in linear programming is $ax_1 + bx_2 + \cdots \leq c$.

Objective Function of a linear programming problem it is attempting to <u>maximise</u> the value of $ax_1 + bx_2 + \cdots$. For a minimise problem, it should be changed to maximise $-ax_1 - bx_2 - \cdots$.

Optimal Solution of a linear programming problem contains at least one vertex where several lines / planes created by constraints intersect.

Slack Form of a linear programming problem is a reformulation of the problem in which all constraints are converted to equality constraints. This is done by introducing slack variables, which are non-negative variables that are added to the constraints to make them equalities.

Basic Variables are those that only appear in one equation.
Basic Solution is the solution where the non-basic variables are equal to zero.

Entering Variable should be the variable with the largest negative coefficient in the objective function.
Leaving Variable should be the basic variable with the least amount of positive slack (RHS divided by the coefficients of the entering variable), or, coefficient when RHS is 0.

Pivot is a method to make entering variable a non-basic variable to become a basic variable, and otherwise for the leaving variable through gaussian elimination (entering variable become 1, and do row-based additions / subtractions). This makes the current solution moving from one crossing point to another.

Unbounded Feasible Region exists when solutions can be arbitrarily large. It is found when the Simplex algorithm detect all slack values to be infinite for a valid entry value.

Degeneracy in the simplex method is a situation in which a basic feasible solution has more than one basic variable equal to zero, maybe an inequality is effectively equality. It can cause the simplex method to take longer to converge, fail to converge, or cycling (same basic solution visited repetitively).

Primitive Root is a number g that $g^i \bmod n$ for a limited number of i would cover 0 to $n-1$.

Multiplicative Inverse is the number (under modulo n) multiplied by another number would yield 1. There is one for all numbers if n is a prime number.

Fast Modular Exponentiation: $A^2 \bmod C = (A * A) \bmod C = ((A \bmod C) * (A \bmod C)) \bmod C$

Extended Euclidean Algorithm is an extension beyond computing *Greatest Common Divisor (GCD)* and provide a solution to the problem of $ax + by = \gcd(a, b)$.

Public Key Pair (p, g, g^x mod p) is a prime number, a *primitive root*, and a 'random' number less than the prime derived by private key and the generator.
Encrypted Message is (a, b) where $a = g^k \bmod p$ and $b = My^k \bmod p$. K is client private key.
Decryption needs to find the multiplicative inverse of $a^x = g^{kx} = y^k \bmod p$. This will allow server to cancel out the y^k and get the value of M.

Class P is the set of problems that is solvable in polynomial time.
Class NP is the set of problems that is solvable in polynomial time on a *nondeterministic machine*. Or, this problem can generate a certificate to verify the results in polynomial time.
Nondeterministic Machine is a theoretical device that can concurrently go through all possibilities.
NP-Completeness is the property of a problem that is in *Class NP* and is harder than all other NP problems.

SAT problem: Given a Boolean expression B over variables V, does there exist an assignment A of truth values to V that makes B true? For K-SAT, it would be a 'K-clauses' Boolean expression.
Circuit-SAT problem: Given a (one-output) Boolean combinational circuit C, is there an assignment that causes the circuit to be 1.
Vertex Cover Problem: Given a graph and an integer k, is there a subset of the vertices containing at most k vertices that for every edge there is at least one node on either end is in this subset.
Clique Problem: Given a graph and an integer k, is there a subset of the vertices containing at most k vertices that for any pairs of nodes in the subset there is at least a one-way connection.
Hamiltonian Cycle Problem: Given a graph does there exist a cycle that visits each vertex exactly once.
Travelling Salesman Problem: Given a weighted graph and an integer k is there a Hamiltonian cycle whose weight is at most k.

$$3SAT \leq_p VertexCover \leq_p Clique \leq_p SetCover$$

$$VertexCover \leq_p SubsetSum \leq_p Knapsack$$

$$VertexCover \leq_p HamiltonianCycle \leq_p TravellingSalesman$$