**COMP 5327**

|  |  |
| --- | --- |
| ***Assignmrnt 03*** | ***Student Name : Meerim Samakova*** |
| ***Assignment Date: 11/30/2023*** | ***Student id : 10003737*** |

**Abstract Data Type (ADT):** Think of it as having a box that comes with rules, on how you can use it. For instance imagine a toy box where you can only add and remove toys following guidelines like a list where you can add or remove items following specific rules.

**Acyclic Graph:** Picture drawing lines between friends without creating any loops. It's like having a map that shows who is friends with whom without forming any connections.

**Adjacency List:** Imagine having a list where each persons name is accompanied by a list of their friends names to them. It's similar to writing down who's friends with whom.

**Adjacency Matrix:** Envision a table with peoples names on one side. The same names on the top. Wherever the row and column intersect mark it if those individuals are friends. It's like having a table that displays who's friends, with whom.

**Cost:** Think of it as spreading out the cost of a toy over days instead of paying for it all at once. Amortized cost helps in distributing expenses over time.

**Array Merging:**Picture combining two piles of toys to create one pile.

Array merging involves taking two lists and combining them into a larger list.

Backtracking; Think of it as navigating through a maze and attempting paths until you find the way, out. If you reach an end you. Explore another route. Backtracking is akin to exploring options and retracing your steps when one approach doesn't yield the desired result, in order to try something

**Best Case:** Picture yourself playing a game and achieving the highest possible score. The best case scenario refers to when everything goes well or smoothly without encountering any issues.

**Big O Notation:** It's a way to measure how quickly something happens or how time and space it requires as it grows. For instance think about how the length of a line increases as more people join it.

**Binary Heap:** Imagine a stack of toys where the largest toy sits on top and each subsequent toy below is smaller. This arrangement represents a heap! It assists in identifying either the smallest toy, within the stack.

**Branch and Bound:** It's like solving a puzzle by exploring parts (branches) without wasting time on unhelpful sections. You can visualize it as navigating through a maze while searching for the path.

**Breadth First Search:** Picture trying to find an exit in a maze by examining all paths on the same level before delving deeper. Breadth first search involves prioritizing areas before venturing away.

**Brute Force Attack:** Envision attempting every combination to unlock a lock because you don't know the code. A brute force attack exhaustively tries all options until it discovers the one to guessing passwords.

**Clique Problem:** It's akin, to identifying the group of friends where everyone is connected to every person within that group.

Imagine a scenario where every individual, in a group is interconnected with every person.

**Collision:** It's akin to two cars unexpectedly colliding because they both attempted to navigate through the road simultaneously. In the realm of computing a collision occurs when two entities accidentally converge in the location.

**Complete Graph:** Picture a group where each person is acquainted with every individual. A complete graph resembles drawing lines that connect all the members within the group ensuring no one is left out.

**Complexity Class:** It's comparable to sorting types of puzzles, into the box based on their level of difficulty to solve. Complexity classes categorize problem types according to their level of challenge for computers to solve.

**Algorithm:** Think of it as following a recipe step, by step, where you always perform the actions for the same input. For example imagine making cookies using a recipe that you faithfully follow each time to achieve delicious results.

**Dynamic Programming:** Picture solving a problem by breaking it down into manageable parts and remembering the solutions to those parts. It's like solving a puzzle piece by piece and utilizing the solutions you have already found to solve the problem efficiently.

**Edge:** Visualize it as a connection between two places on a map. In the context of graphs edges are similar to those lines that connect points or locations.

**Euler Cycle:** Imagine taking a walk where you explore every street in your neighborhood and eventually end up back where you started. An Euler cycle is like embarking on a journey through every path in a network or graph and ultimately returning to the point.

**Fibonacci Number:** It refers to a sequence of numbers where each number is obtained by adding the two preceding numbers – for instance, 0 1, 1 2, 3 5... It's akin, to observing an established pattern where you add up the two numbers in order to determine the one.

**Full Array:** Consider it as a list wherein every position is occupied or filled with something significant.

Just picture a bookshelf with every compartment filled. A full binary tree is similar, to a family tree, where each parent has either two children or no children all.

**(GCD)** it's like searching for the largest number that can divide into two given numbers without leaving any remainder. For instance if we have 12 and 18 the GCD would be 6 because its the number that can evenly divide both 12 and 18.

Think of an algorithm as making the best choice at each step in order to reach a solution with the hope that this will lead to an overall optimal outcome. It's, like selecting the attractive candy at every turn aiming to collect the most delicious candies overall.

**Hamiltonian Cycle:**  It's, like going on a sightseeing tour of a city making sure to visit every spot once without revisiting any place.

**Hash Function:** Think of it as a formula that magically transforms something like a word or number into a code. It's similar to having a code that converts your name into something only you can decipher

**Heap:** Imagine a stack of toys where the largest toy sits on top and each subsequent toy is smaller than the one above it. A heap is like a way to arrange things so that the biggest or smallest item is easily accessible.

**Heuristic:** It's akin to having some guiding principles or rules to assist you in making decisions when you're uncertain about the choice. It's comparable to following clues in search of treasure even if you don't know its location.

**Huffman Encoding:** Picture having your language where frequently used letters have shorter codes while less common ones have longer codes. Huffman encoding is a method, for compressing information by assigning codes to frequently occurring elements.

**In Place Algorithm:** Imagine rearranging your room without requiring any space.An algorithm that operates, in place modifies elements directly without requiring space.

**Karnaugh Map:** Imagine solving a puzzle by grouping elements to determine the solution. A Karnaugh map aids in simplifying expressions by grouping elements.

**Linear:** It's akin to drawing a line on a graph, where values steadily increase or decrease. In computing a linear algorithm implies that the time required to perform a task increases as the input size grows.

**Linked List:** Visualize it, as a chain where each link's aware of the link. A linked list resembles a sequence where each element points to the element forming an interconnected chain of information.

**Little o Notation:** It's similar, to saying that one thing grows slower compared to another. For instance if you have two lines on a graph one line might grow at a rate than the other and thats precisely what little o notation helps us describe.

**Logarithm:** Picture it as a kind of power that tells you how times you need to divide something in half to reach a specific number. For example when we take the logarithm base 2 of 8 we get 3 because it takes three divisions by 2 to reach 1.

**Markov Chain:** Think of it as a story where what comes next depends on the situation and not on what happened previously. It represents a sequence of events where the next event is determined by the event and not influenced by the entire past.

**Multigraph:** Imagine having a map where there can be routes between two locations. A multigraph works similarly representing a network where there can be connections between the pair of nodes.

**Node:** Consider it as a point or position, within a network or graph. In a family tree scenario each person can be seen as a node and the lines connecting them depict their relationships.

It's similar, to a type of puzzles where its simple to verify if an answer's correct. Challenging to actually find the answer. It's like attempting to crack a code without having knowledge of the underlying rules.

**Queue:** Think of it as a queue at a ticket counter, where the person who arrives first is served first. A queue is basically a waiting list where things wait in line for their turn.

**Redix sort:** Now picture sorting items based on their digits like organizing numbers, by examining each digit. Radix sort arranges things by comparing their digits from right to left in order to establish the order.

**Red Black Tree:** Imagine a kind of tree where the nodes are assigned colors. Either red or black. These trees are designed in a way that ensures they remain balanced which makes searching for elements faster.

**Relaxation:** Picture taking a break and finding comfort. In the context of algorithms relaxation refers to updating information to make it more accurate or finding the solution.

**Tree Traversal:** It's similar, to exploring a family tree by visiting each individual. Tree traversal is an approach to visit or examine every node in a tree structure.

**Turing Machine:** Think of it as a machine that can perform various tasks by following predefined rules. It's a device of executing any computational task when provided with appropriate instructions.

**Vertex:** Imagine a point. Dot on a graph. In maps cities can be compared to vertices while lines connecting cities represent the roads between them.

**Weighted Graph:** Envision a map where each road has a number indicating its length. A weighted graph is like a network where each edge carries values representing some quantity such, as distance or cost.