EXTRA TP1 - Midterms

Name: Jarilla, Angelo Christiene M.
Section: BSCS401
Subject: Design and Analysis of Algorithm

BIG O NOTATIONS:

Algorithm	Efficiency (Compile-time memory size & Complexity)	Use-case (Best-fit scenario)	Optimization (Execution speed & Improvements)
Linked List	- Memory-efficient as it dynamically allocates memory per node. - Requires extra storage for pointers. - Time Complexity: Access/Search: Θ(n), O(n) Insertion/Deletion: Θ(1), O(1) - Space Complexity: O(n)	- Best for dynamic data structures where frequent insertions/deletions occur. - Used in memory-efficient applications like undo/redo functionality, hash table chaining, and polynomial arithmetic.	- Insertion/Deletion is O(1) at the head/tail, O(n) for searching. - Optimized by using Doubly Linked Lists (fast traversal in both directions) or Skip Lists (faster search).
Queue	- Uses contiguous memory in array-based queues or pointers in linked lists. - Time Complexity: Access/Search: Θ(n), O(n) Insertion/Deletion: Θ(1), O(1) - Space Complexity: O(n)	- Best for scheduling problems (CPU scheduling, print spooling, BFS traversal). - Used in networking (packet scheduling), operating systems, and customer service lines.	- Enqueue/Dequeue operations O(1) in linked list queues. - Can be optimized using Circular Queues (O(1) space complexity) or Double-ended Queues (Deque, O(1) insertion/removal at both ends).
Graph	- Uses adjacency lists (efficient for sparse graphs) or adjacency matrix (fast lookups but uses more space). - Time Complexity: Access/Search: Θ(log(n)), O(n) Insertion/Deletion:	- Best for modeling relationships like social networks, transportation maps, web crawling, and recommendation systems.	- BFS & DFS run in O(V + E). - Dijkstra's algorithm (shortest path) runs in O((V + E) log V). - Optimized using priority queues (Dijkstra) or *A* search (heuristic pathfinding)

	Θ(log(n)), O(n) - Space Complexity: O(n) for adjacency lists, $O(n^2)$ for adjacency matrix		
Binary Tree	- Each node requires memory for left/right child pointers. - Time Complexity: Access/Search: Θ(log(n)), O(log(n)) Insertion/Deletion: Θ(log(n)), O(log(n)) - Space Complexity: O(n)	- Best for efficient searching, sorting, and hierarchical storage (e.g., file systems, databases, decision trees). - Used in AI, game development (decision trees), and compilers.	- Balanced trees (AVL, Red-Black Trees) improve search to O(log n). - B-Trees (used in databases) improve search efficiency for large datasets.

EFFICIENCY TABLE:

Info:	Linkedlist	Queue	Graph	Trees
User time (seconds):	0.03	0.02	0.02	0.02
System time (seconds):	0.00	0.00	0.00	0.00
Percent of CPU this job got:	70%	69%	66%	66%
Elapsed (wall clock) time (h:mm:ss or m:ss):	0:00.04	0:00.05	0:00.05	0:00.04
Average shared text size (kbytes):	0	0	0	0
Average unshared data size (kbytes):	0	0	0	0
Average stack size (kbytes):	0	0	0	0
Average total size (kbytes):	0	0	0	0
Maximum resident set size (kbytes):	10280	10248	10232	10248
Average resident set size (kbytes):	0	0	0	0
Major (requiring I/O) page faults:	0	0	0	0
Minor (reclaiming a frame) page faults:	1125	1120	1120	1135
Voluntary context switches:	40	41	40	41
Involuntary context switches:	0	0	0	0
Swaps:	0	0	0	0
File system inputs:	0	0	0	0
File system outputs:	0	0	0	0
Socket messages sent:	0	0	0	0
Socket messages received:	4096	4096	4096	4096
Signals delivered:	0	0	0	0
Page size (bytes):	-	-	-	-
Exit status:	4.0K	4.0K	4.0K	4.0K
File Size (Disk usage of the file):	0.03	0.02	0.02	0.02

ALGORITHM USE CASE

	Linkedlist	**Queue**	**Graph**	**Trees**
Best-fit Use-case	Dynamic memory allocation, frequent insertions/deletions	FIFO processes, task scheduling	Modeling relationships, network problems	Hierarchical data, efficient searching
Example Use-cases	Undo/redo operations, real-time systems	Task scheduling, BFS in graph traversal, request handling	Social networks, GPS routing, web crawlers	File systems, binary search trees, decision trees in machine learning

EXECUTION SPEED:

	LinkedList	Queue	Graph	Trees
real	0m0.056s	0m0.050s	0m0.053s	0m0.056s
user	0m0.013s	0m0.028s	0m0.027s	0m0.019s
sys	0m0.025s	0m0.009s	0m0.010s	0m0.021s