Comparison And Optimization:-

In our Student Management System, we chose to use a Binary Search Tree (BST) to handle searching and sorting by roll number. This structure allows us to quickly find records, with search times generally much faster than scanning through a list, and it also lets us easily get the students in order by simply doing an inorder traversal.

For operations involving CGPA, we used a Doubly Linked List (DLL). This gave us the flexibility to move forwards and backwards through the data easily, which makes tasks like updating or traversing the list more convenient.

While we did use simpler algorithms like Bubble Sort and Linear Search, these were sufficient for the smaller datasets we were working with and kept the implementation straightforward.

Overall, combining the BST and DLL helped us strike a good balance between speed and simplicity, and by adding a custom undo feature using a stack, we gave users better control over their actions in the system.

1) Data Structures Comparison:-

Data Structure	Use Case	Time Complexity (Avg)	Space Complexity	Why You Used It
BST	Roll No search/sort	Search: O(log n) Sort: O(n) (inorder)	O(n)	Fast for ordered data (roll no)
Doubly Linked List	CGPA search & sort	Search: O(n) Sort: $O(n^2)$	O(n)	Easy node traversal + undo support
Stack (via DLL)	Undo last operation	Push/Pop: O(1)	O(n)	Simple to implement undo feature

2) Searching Algorithms Comparison:-

Algorithm	Applied To	Technique	Time Complexity	Space Complexity	Pros	Cons
Binary Search (BST)	Roll No	Tree Traversal	O(log n) (avg), O(n) (worst if unbalanced)	O(h)	Fast, organized structure	Needs balanced tree
Linear Search (DLL)	CGPA	Sequential Search	O(n)	O(1)	Simple and flexible	Slow for large datasets

3) Sorting Algorithms Comparison:-

Algorithm	Applied To	Time Complexity	Space Complexity	Stable?	Pros	Cons
Inorder BST Sort	Roll No (BST)	O(n)	O(h)	Yes	No swapping, natural ordering	Needs BST insert first
Bubble Sort	CGPA (DLL)	$O(n^2)$	O(1)	Yes	Easy to code, stable	Slow for large n