s


INTERNATIONAL ISLAMIC UNIVERSITY
ISLAMABAD

DSA PROJECT : STRUCTURE ANALYSIS

Tables & Chart Description

Our project uses different data structures and algorithms. The tables and chart help compare them in terms of speed and usefulness.

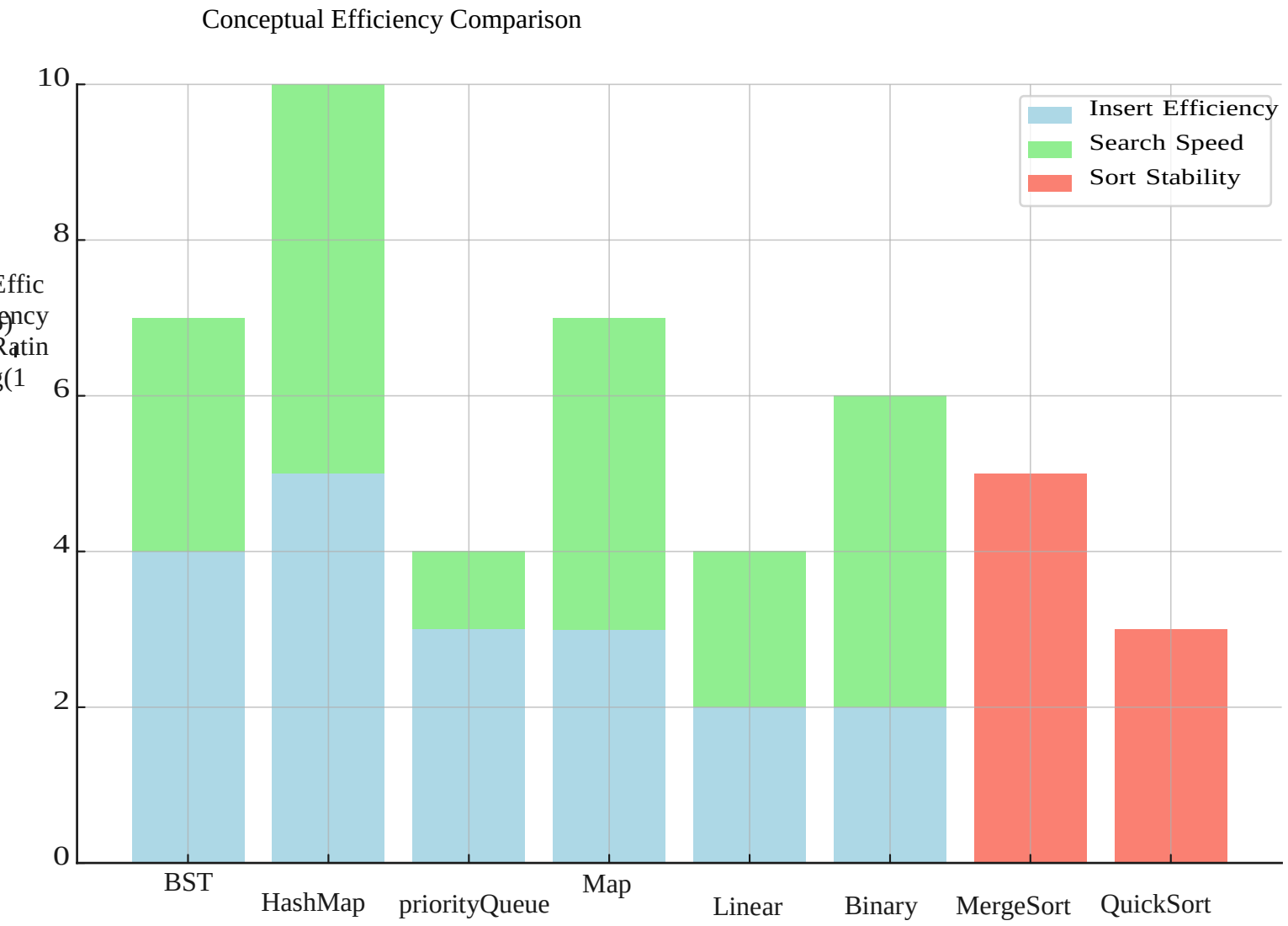
What the Tables Show:

- **Data Structures Table** shows how fast each structure is for adding and finding data.
  Example: Hash Map is very fast for searching orders.
- **Sorting Algorithms Table** compares how sorting methods work.
  Example: Merge Sort is stable and safe for big data, Quick Sort is faster but riskier.
- **Searching Algorithms Table** explains when to use Linear Search or Binary Search.
  Example: Linear Search works on any list, Binary Search only works on sorted lists but is faster.

Data Structure	Average Time	Worst Time	Space Complexity
BST<MenuItem*>	O(log n)	O(n)	O(n)
OrderQueue<Order*>	Push: O(n), Pop: O(1)	Push: O(n), Pop: O(1)	O(n)
unordered_map<string, Order*>	O(1)	O(n)	O(n)
map<string, MenuItem*>	O(log n)	O(log n)	O(n)
list<DeliveryDriver*>	O(1)	O(n)	O(n)

Algorithm	Average Time	Worst Time	Space	Stable	In-Place
Merge Sort	O(n log n)	O(n log n)	O(n)	Yes	No
Quick Sort	O(n log n)	O(n²)	O(log n)	No	Yes
std::sort()	O(n log n)	O(n log n)	O(log n)	No	Yes

Algorithm	Time Complexity	Sorted Required	Use Case
Linear Search	O(n)	No	Unsorted Order ID Search
Binary Search	O(log n)	Yes	Sorted Order ID Search



What the Chart Shows:

- It compares how each method performs in three ways:
  - Adding new data (Insert)
  - Searching data (Speed)
  - Keeping order while sorting (Stability)

Summary:

We chose each method because it fits the job:

- Fast search = Hash Map
- Sorted menu = BST
- Stable sorting = Merge Sort

These choices make the system faster and easier to use.