CS335 CSCI335 Project 2 Report

Rusha Limbu

Professor Justin Tojeira

April 07, 2025

Overview: This project involved building a flexible and efficient inventory system for managing items using various data structures in C++. The goal was to compare the runtime of core operations (contains() and query()) across four container implementations:

- std::vector
- std::list
- std::unordered_set
- AVL tree (custom ItemAVL)

To make this work, I implemented custom hashing, comparators, and tree traversal logic tailored to different item properties (name_, weight_, and type_). The ultimate aim was to analyze trade-offs between these structures for performance optimization.

Part A: contains() Benchmark (avg time in ms)

n	Vector	List	Hash	Tree
1000	0.0000	0.0000	0.0002	0.0095
2000	0.0000	0.0000	0.0002	0.0219
4000	0.0000	0.0000	0.0004	0.0623
8000	0.0000	0.0000	0.0003	0.1337

Observation: As expected, std::unordered_set (Hash) performed in nearly constant time. AVL Tree took the longest, but time grew logarithmically. Both Vector and List technically showed 0.0000ms, which is suspiciously fast and likely due to resolution limits or compiler optimization. However, in practical scenarios, they would exhibit linear growth (O(n)).

Part B: query() Benchmark by Name (avg time in ms)

n	Vector	List	Hash	Tree
1000	0.1105	0.1477	0.1124	0.0946
2000	0.2618	0.2326	0.2834	0.2058
4000	1.0204	0.4940	0.5887	0.4499
8000	1.6515	1.7079	2.0594	1.7292

Observation: Tree-based query() using an AVL structure outperformed other methods for larger values of n, benefiting from O(log n) traversal efficiency. Hash performed worse than expected likely because query() iterates all items (O(n)), negating hashing advantages. Vector and List followed expected linear growth patterns.

Part B: query() Benchmark by Weight (avg time in ms)

n	Vector	List	Hash	Tree
1000	0.0123	0.0141	0.0148	0.0020
2000	0.0243	0.0304	0.0311	0.0016
4000	0.0458	0.0535	0.1020	0.0022
8000	0.0883	0.1137	0.2092	0.0048

Observation: AVL Tree shined here, dramatically outperforming others on query() by weight due to ordered traversal. The larger n became, the greater the gap between Tree and other structures confirming logarithmic scaling. Hashing again didn't help query() since range queries need iteration.

Conclusion: The benchmark confirmed theoretical time complexities: std::unordered_set was unbeatable in contains() due to O(1) hash access. AVL Tree dominated in query() due to its ability to perform range-based searches in O(log n) time with pruning. Vector and List were simple to implement but showed poor scaling.