Performance Results

Table 1: Average contains() Execution Times (ms)

n	Vector	List	Hash	Tree
1000	1.2	1.5	0.01	1.8
2000	2.5	3.1	0.01	3.7
4000	5.3	6.4	0.01	7.9
8000	11.2	13.7	0.01	16.3

Table 2: Average query() Execution Times (ms)

n	Vector	List	Hash	Tree
1000	0.8	1.1	0.9	0.2
2000	1.7	2.3	1.8	0.3
4000	3.6	4.9	3.7	0.4
8000	7.5	10.2	7.8	0.6

Conclusion
The results of the performance verify each of the data structures' theoretical expectations. The hash-based inventory recorded an O(1) time complexity for contains() calls, making it 100 to 1000 times faster than other methods of name lookups. This is an extremely desirable attribute in games or programs where there are numerous item searches by name to conduct repeatedly. Alternatively, tree-based inventory offered improved range query efficiency of O(log n + k) that was between 10 and 20 times faster than linear searches for bigger inventories and thus more ideal for systems calling for constant filtered displays, such as sorting gears by weight. Surprisingly, the contains() of the was inferior to vector/list because the name search constraint required a complete scan of the tree.

This goes to underscore the necessity of choosing the right data structure based on the main use caseshash for lookups and tree for range queries. In the majority of RPG inventory systems, the ideal would be a hybrid approach: hash for quick access and tree for filtered displays, though this comes at the expense of memory usage. The results confirm that Big-O notation accurately forecasts real-world performance on a bigger scale.