Project 1 Submission

Alec Sieg

Southern New Hampshire University

CS-300: Analysis and Design

Professor: Leslie Gruner

June 22, 2025

Run Time Analysis

FUNCTION loadCourseData(fileName):

// Cost: 1, Executions: 1

OPEN file with filename

// Cost: 1, Executions: 1

IF file fails to open THEN RETURN false

// Cost: 1, Executions: 1

WHILE not end of file DO

// Cost: 1, Executions: n+1

read line, SPLIT by comma into tokens

// Cost: 1, Executions: n

// Check minimum 2 parameters

IF tokens.size() < 2 THEN

// Cost: 1, Executions: n

PRINT "Error: Need courseNumber and courseTitle"

// Cost: 1, Executions: worst case n

RETURN false

// Cost: 1, Executions: worst case 1

CREATE course with tokens[0], tokens[1]

// Cost: 1, Executions: n

ADD tokens[2+] as prerequisites

// Cost: p, Executions: n (p = avg prerequisites)

ADD course to tempVector

// Cost: 1, Executions: n

END WHILE

// Validate prerequisites exist as courses

FOR each course in tempVector DO

// Cost: 1, Executions: n+1

FOR each prerequisite DO

// Cost: 1, Executions: n\*p+1

IF prerequisite not found in tempVector THEN

// Cost: n, Executions: n\*p (linear search)

PRINT "Error: Prerequisite doesn't exist"

// Cost: 1, Executions: worst case n\*p

RETURN false

// Cost: 1, Executions: worst case 1

courses = tempVector

// Cost: 1, Executions: 1

RETURN true

// Cost: 1, Executions: 1

Runtime Analysis Structure

| **Data Structure** | **File Reading & Creation** | **Prerequisite Validation** | **Overall Complexity** |
| --- | --- | --- | --- |
| **Vector** | O(n) | O(n²×p) | **O(n²×p)** |
| **Hash Table** | O(n) | O(n×p) average, O(n²×p) worst | **O(n×p) average, O(n²×p) worst** |
| **Binary Search Tree** | O(n log n) average | O(n×p×log n) average | **O(n×p×log n) average, O(n²×p) worst** |

Advantages and Disadvantages

Vector Data Structure

Advantages: The vector data structure provides a simple implementation that is straightforward to implement and debug with minimal complexity. It offers memory efficiency with no additional overhead for pointers or hash table structure. The sequential memory access provides good cache locality and performance for iteration, with predictable memory usage that is easily calculated and consistent.

Disadvantages: The vector suffers from poor search performance with O(n) linear search for individual course lookup being inefficient. The expensive prerequisite validation with O(n²×p) complexity makes loading slow with many courses. External sorting is required with O(n log n) complexity needed for alphabetical course listing. The structure has scalability issues as performance degrades significantly when the course database grows.

Hash Table Data Structure

Advantage:

The hash table provides excellent search performance with O(1) average time for individual course lookup. It offers efficient prerequisite validation with O(n×p) average complexity during loading. The structure is ideal for frequent lookups when searching courses is the primary operation and remains scalable for large datasets as performance stays consistent as the course database grows.

Disadvantage: Hash tables carry the risk of hash collisions where performance can degrade to O(n) in worst-case scenarios. There is additional memory overhead as the hash table structure requires extra memory for buckets. The implementation is more complicated to implement correctly than vector structures. Sorting is still required as data must be extracted and sorted for alphabetical course listing.

Binary Search Tree

Advantage: The binary search tree offers balanced performance providing a good compromise between search (O(log n)) and insertion operations. It maintains natural ordering automatically with no external sorting needed. The structure provides efficient sorted traversal where in-order traversal gives O(n) alphabetical listing. There is moderate memory overhead with less overhead than hash tables while maintaining good performance.

Disadvantage: Binary search trees have potential for imbalance where performance can degrade to O(n) if the tree becomes unbalanced. There is higher loading complexity with O(n×p×log n) complexity during file loading and validation. The implementation complexity is more complex than vector requiring careful tree management. Performance in worst-case scenarios is heavily dependent on the insertion order of course data.