Runtime Analysis

Felie Marie Magbanua

Southern New Hampshire University

CS-300

DSA: Analysis and Design

Jonathan Norman

June 22, 2025

Runtime Analysis Chart: Vector

Code Step	Line Cost	# Times Execute	Total Cost
open file	1	1	1
for each line in file	1	n	n
split line by ',' into tokens	1	n	n
create new Course object	1	n	n
add tokens[i] to course.prerequisites (per prereq)	1	n*p	n*p
add course to courses vector	1	n	n
for each course: for each prereq: validate existence	1	up to n^2	n^2
Total Cost			n^2 + 4n + n*p + 1
Runtime			O(n^2)

- *n = number of courses, p = avg prerequisites per course*

Runtime Analysis Chart: Hash Table

Code Step	Line Cost	# Times Execute	Total Cost
open file	1	1	1
for each line in file	1	n	n
split line by ',' into tokens	1	n	n
create new Course object	1	n	n
add tokens[i] to course.prerequisites (per prereq)	1	n*p	n*p
insert course into hash table	1	n	n
for each course: for each prereq: validate existence	1	up to n*p	n*p
Total Cost			4n + 2n*p + 1
Runtime			O(n*p) = O(n^2) worst, O(n) avg

- *n = number of courses, p = avg prerequisites per course*
- *Note: Hash table lookup is O(1) average, but O(n) worst case with collisions*

Runtime Analysis Chart: Binary Search Tree (BST)

Code Step	Line Cost	# Times Execute	Total Cost
open file	1	1	1
for each line in file	1	n	n
split line by ',' into tokens	1	n	n
create new Course object	1	n	n
add tokens[i] to course.prerequisites (per prereq)	1	n*p	n*p
insert course into BST (by courseNumber)	log n*	n	n log n (avg), n^2 (worst)
for each course (in-order): for each prereq: validate	n	n*p	n^2
Total Cost			n^2 + n log n + n*p + 3n + 1
Runtime			O(n^2) worst, O(n log n) avg

log n for balanced BST, up to n for degenerate/unbalanced tree