# Runtime Analysis and Evaluation

This document explains how three different data structures—Vector, Hash Table, and Binary Search Tree—work in the course management system for ABC University. We will look at how fast and how much memory each structure uses, and the pros and cons of each.

## 1. Vector Data Structure

A Vector is like a flexible array where you can add or remove items. It's great for quickly accessing items if you know their position.  
  
- Time Complexity:  
 - Getting an item by position: O(1)  
 - Adding an item: O(n) in the worst case (if it needs more space), but usually O(1)  
 - Removing an item: O(n) because you might need to shift other items  
 - Finding an item: O(n)  
   
- Space Complexity:  
 - O(n), where n is the number of items in the vector  
  
- Advantages:  
 - Fast access to items by position.  
 - Easy to understand and use.  
  
- Disadvantages:  
 - Not efficient for adding or removing items.  
 - Finding items takes longer since you have to look through the whole list.

## 2. Hash Table Data Structure

A Hash Table pairs keys with values using a special function called a hash function. It's really good at quickly finding, adding, or removing items.  
  
-Time Complexity:  
 - Getting an item by key: O(1) on average, but O(n) if there are too many overlaps (collisions)  
 - Adding an item: O(1) on average, but O(n) in the worst case  
 - Removing an item: O(1) on average, but O(n) in the worst case  
 - Finding an item: O(1) on average, but O(n) in the worst case  
   
- Space Complexity:  
 - O(n), where n is the number of items in the table  
  
- Advantages:  
 - Very fast for most operations.  
 - Great for quick lookups.  
  
- Disadvantages:  
 - Can slow down if there are too many overlaps.  
 - Uses extra space for storing the hash table.

## 3. Binary Search Tree (BST) Data Structure

A Binary Search Tree (BST) is a structure where each item is a node, and each node can have two children. One child is smaller, and the other is bigger.  
  
- Time Complexity:  
 - Getting an item: O(log n) on average, O(n) in the worst case (if the tree is not balanced)  
 - Adding an item: O(log n) on average, O(n) in the worst case  
 - Removing an item: O(log n) on average, O(n) in the worst case  
 - Finding an item: O(log n) on average, O(n) in the worst case  
   
- Space Complexity:  
 - O(n), where n is the number of nodes in the tree  
  
- Advantages:  
 - Keeps data in order, making it easy to go through.  
 - Efficient if the tree is balanced.  
  
- Disadvantages:  
 - Can get unbalanced, making it slower.  
 - Harder to set up compared to Vectors and Hash Tables.

## 4. Recommendation

Based on this analysis, the Hash Table is the best choice for ABC University's course management system. It works fast for most tasks like finding, adding, and removing courses, which is important for this program.  
  
However, if it's important to keep the courses in order, the Binary Search Tree might be a better choice, especially if steps are taken to keep the tree balanced.

## Runtime Analysis Chart

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Vector | Hash Table | Binary Search Tree |
| Access (get by key/index) | O(1) | O(1) average, O(n) worst | O(log n) average, O(n) worst |
| Insertion (add) | O(n) worst, O(1) average | O(1) average, O(n) worst | O(log n) average, O(n) worst |
| Deletion (remove) | O(n) | O(1) average, O(n) worst | O(log n) average, O(n) worst |
| Search (find) | O(n) | O(1) average, O(n) worst | O(log n) average, O(n) worst |

## Pseudocode: Opening and Reading the File

1. Open the file containing course data.  
 2. For each line in the file:  
 a. Parse the line to extract course information.  
 b. Check for any formatting errors.  
 c. If no errors, proceed to create a course object.

## Pseudocode: Creating Course Objects

1. Create a Course object class with the following attributes:  
 a. Course ID  
 b. Course Title  
 c. Prerequisites (as a list)  
 2. For each parsed line, create a Course object with the extracted data.

## Pseudocode: Printing Course Information

1. Prompt the user to enter a Course ID.  
 2. Search for the Course ID in the data structure.  
 3. If found, print the Course Title and Prerequisites.  
 4. If not found, display an error message.

## Pseudocode: Menu Options

1. Display the following menu options:  
 1. Load course data.  
 2. Print all courses (sorted alphanumerically).  
 3. Print course title and prerequisites.  
 9. Exit program.  
 2. Based on user input, execute the corresponding function.

## Pseudocode: Sorting and Printing Courses

1. Load all courses into the selected data structure.  
 2. Sort the courses by Course ID alphanumerically.  
 3. Print the sorted list to the display.