Daniel Collins

CS 300

Professor Ricardo Scarello

April 22, 2024

**Runtime Analysis**

**Runtime Analysis Chart and Evaluation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Structure** | **Function** | **Cost per line** | **Times executed** | **Worst-case runtime** |
| **Vector** | Reading file | 1 | n | O(n) |
|  | Creating objects | 1 | n | O(n2) |
|  | **Big-O** |  |  | **O(n2)** |
| **Hash Table** | Reading file | 1 | n | O(n) |
|  | Creating objects | 1 | n | O(n) |
|  | **Big-O** |  |  | **O(n2)** |
| **Tree** | Reading file | 1 | n | O(n) |
|  | Creating objects | O(log n) | n | O(n log n) |
|  | **Big-O** |  |  | **O(n log n)** |

Each data structure has the same runtime for reading a file, O(n), because each file is read row by row, in constant time. The vector data structure creates course objects with a O(n2) runtime when it reads each course, searching for prerequisites; this process is a selection sort, so it uses nested loops, therefore the runtime is quadratic. The hash table has an O(n2) worst-case runtime for creating course objects because it probes all elements that have been inserted. The binary search tree creates course objects with a O(n log n) runtime because it must ensure the height of the tree is proportional to the number of elements.