| **Spanning Tree Data Structure** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **Initialize variables** | 3 | n | n+3 |
| **While inputfile not end of line** | 1 | n | n |
| Get line of file | 1 |  | n |
| Initialize variables | 3 | n | n+3 |
| **For loop to iterate through file** | 1 | n | n |
| If then logic to add subsections to element | 10 | n | n+10 |
| Check if info in proper format | 3 | n | n+3 |
| Verify that first prereq exist | 8 | N^2 | N^2+8 |
| Verify that second prereq exist | 8 | N^2 | N^2+8 |
| While loop to pull data from file | 2 | N | N+2 |
| While loop to make temp item | 5 | N | N+5 |
| For loop to create temp element values | 11 | N | N+11 |
| **Recursive loop to find and insert new node with current node that has the temp element values** | **12** | **Log n** | **N+1** |
| **Total Cost** | | | 2N^2+9n+log n + 68 |
| **Runtime** | | | O(n^2 + log n) |

**Recommendation**

My recommendation for the data structure to use for this project would be the spanning tree data structure.

**Recommendation**

While it might involve more lines of code and a different implementation scheme, the benefits far outweigh the negatives. While my analysis has n^2 as the runtime, that cost is from the importing and verifying that the data is valid and not from the structure runtime itself. The spanning tree has a worse case runtime of log n, which is better than the other two and since the data is sorted as it is inserted into the data structure, the code to show all courses in order, search for a course and provide information of prerequisites will be far simpler and faster.