**CS-300 Project One Pseudocode**

Menu Pseudocode

Initialize linked list courses

Initialize empty courses object

Define int choice to 0

While choice doesn’t equal 4

Output menu

Output choice 1 to load in courses

Output choice 2 to print course list

Output choice 3 to print a singular course

Output choice 4 to exit program

Switch case with paremeter choice

Case 1

Call function loadCourses with file path parameter and courses linked list

Break

Case 2

Call printList with linked list courses

Break

Case 3

Output user to enter bid choice

Read in user input

Set empty course object equal to the returned course of searchCourses () call

If courses object isn’t empty

Output courses information

Else

Output that course was not found

Break

Print Course List Pseudocode

Initialize pointer node nodeptr equal to head

While nodeptr doesn’t equal NULL

Call display Course with current nodeptr->course

Nodeptr equals nodeptr->next

Print Course Hash Pseudocode

For auto nodeIterate equal to beginning node; nodeIterate doesn’t equal last node; increment nodeIterate

If nodeIterate->key doesn’t equal default UINT\_MAX

Output hash key and course information

Initialize new pointer node tempNode equal to nodeIterate->next

While tempNode doesn’t equal nullptr

Output hash key and course information

tempNode set equal to tempNode->next

Print Course Tree Pseudocode

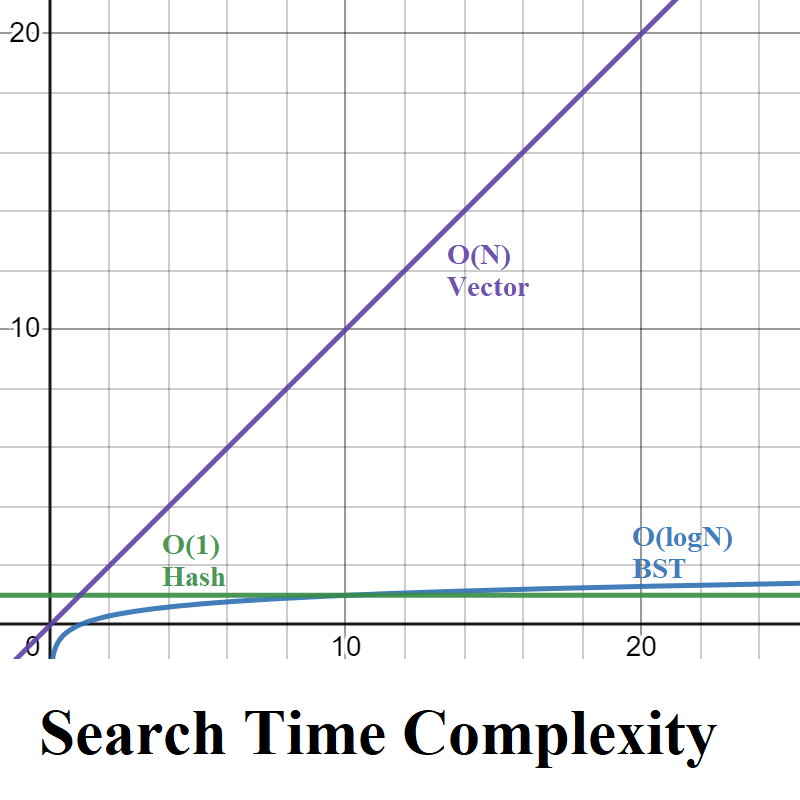
If pointer node doesn’t equal nullptr

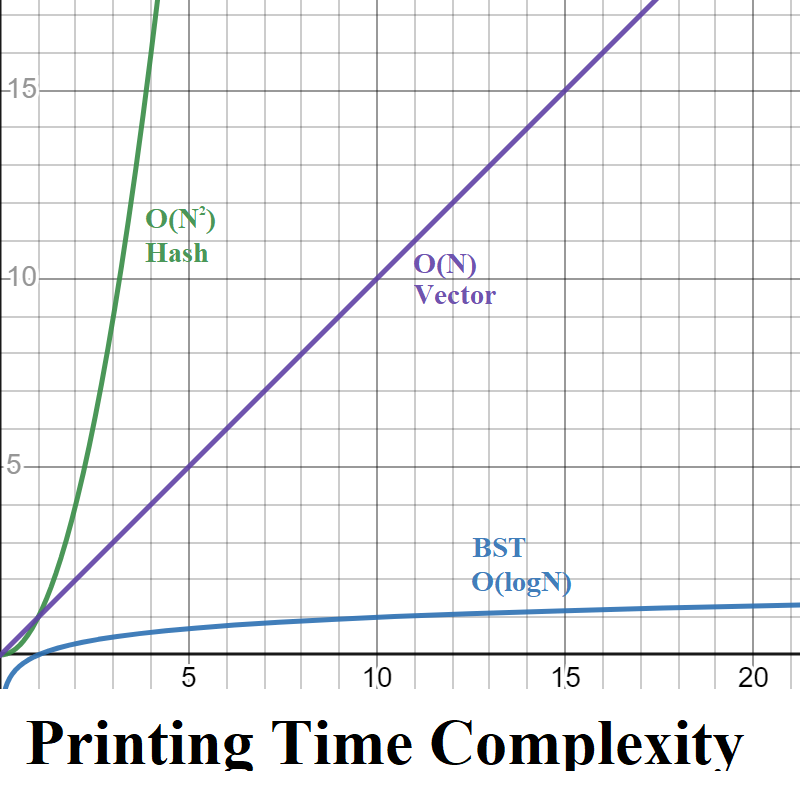
Recusively call inOrder function with pointer nodes->left

Output course information

Recusively call inOrder function with pointer nodes->right

Evaluation

* For the printing the list pseudocode
  + The runtime efficiency would be O(N) because:
    - 1 operation for the initialization of the pointer
    - N iterations for the while loop
    - 2 operations within the while loop
    - 1 final comparison operation for the while loop exiting
    - This equals 1 + N(2) + 1 = 2 + 2N = O(2 + 2N) = O(N)
  + While the space complexity would be S(N) = N + k because the list size is already fixed in the scenario that we’re printing, no courses are being added or removed during this process.
* For printing the Hash pseudocode
  + The Runtime efficiency would be O(N2) because:
    - 1 operation for the for loop nodeIterate initialization
      * N iterations for the for loop
      * 3 operations
      * 2 operations for worst case
      * N iterations for the while loop
        + 2 operations
      * 1 final operation for while loop exit
    - 1 final operation for the for loop exit
    - This equals 1 + N(3 + 2 + N(2) + 1) + 1 = N(2N + 6) + 2 = 2N2 + 6N + 2 = O(N2)
    - The runtime is quadratic because the hash has to go through each key which equates to the N length of the hash then it has to go through each linked list of each key equal to N length of the linked list
  + The space complexity would be S(N) = N2 + k
* For printing the BST pseudocode
  + The runtime efficiency would be $$$$ because:
    - 1 operation by the if
    - T(N/2) for the first recursive call on the left side
    - 1 operation for output course information
    - T(N/2) for the second recursive call on the right side
    - This equals 1 + T(N/2) + 1 + T(N/2) = O(logN)
  + The space complexity would be S(N) = logN + k



Advantages/Disadvantages and recommendation

For each runtime efficiency of the three data structures vector, Hash, and BST the BST has the most efficient runtime efficiency of O(logN) which has a better curve of computations compared to the quadratic or linear runtime of hash and vectors. The vector has the advantage of being simple and easily able to track size and resizable easily but with the caveat that resizing vectors has a linear time. While hash has a potential worse case is usually constant to insert, delete, and search vs BST’s runtime of O(logN). This runtime advantage allows faster operations if implemented properly for instance a hash requires additional memory to store hash values as well as deal with collisions while a BST doesn’t. As well a BST is more efficient at getting a sorted order simply by its ability to easily support recursion in an inOrder sort, unlike a hash. So ultimately the still more efficient runtime of O(logN) than linear and the ease of sorting I would recommend a BST for the courses in the Computer science program.