CS 2302 - Lab 4

Sofia Gutierrez

Olac Fuentes

**Introduction**

The objective of this lab was to create a program that compared the running times of two data structures, Binary Search Tree (BST) and B-tree. These two data structures were given a list of words that are most commonly used in the English language. Each of these words were accompanied by floating point numbers which served as the word embedding. This word embedding represented the word’s vector description or in other words, these numbers would represent the word. After the two data structures were created and were each filled with this list of words and their numbers, a second, separate file, which contained a list of more words, was created by the user. More specifically, this list had two words per line. The point of this was so that each of the two words per line would be compared to one another and their similarities would print. A cosine distance formula was used which would then help produce this similarity output. Again, this would then compare the running times that the outputs took to generate for both data structures.

**Design and Implementation**

This program had three sections in total which consisted of the word embedding class, the BST class, the B-tree class, and finally the main method. The word embedding class would of course contain the attributes which were ‘word’, and ‘emb’. These were needed in order to modify the basic methods which belong to BST and B-tree classes (insert method, search method, etc.). Then came the BST class which contained its appropriate attributes and methods: ‘InsertBST’, ‘HeightBST’, ‘NumofNodesBST’, and ‘SearchBST’. ‘InsertBST’, was of course needed to store the words and their embeddings in the appropriate position in the tree, while ‘HeightBST’ and ‘NumofNodesBST’ were functions used to check the code. The method ‘SearchBST’ was created to find two words within the tree and their location within the tree would be returned to the cosine distance equation so that the similarity output would complute.

Next was the BTree class. This began with naming its attributes: ‘data’, ‘child’, ‘isLeaf’. The methods included ‘FindChild’, ‘InsertInternal’, ‘Split’, ‘InsertLeaf’, ‘IsFull’, ‘Insert’, ‘HeightBTree’, ‘NumOfNodesBTree’, and ‘SearchBTree’. ‘FindChild’, ‘InsertInternal’, ‘Split’, ‘InsertLeaf’, ‘IsFull’, and ‘Insert’ were all functions needed to store the words and there floating point digits in their correct position within the tree. ‘HeightBTree’ and ‘NumOfNodesBTree’ were again functions that were used to check the code and the ‘SearchBTree’ method was used to find the two words within the tree and retrieve their location so that the cosine distance equation would operate successfully.

And finally, the last section was coded which contained the main method. This main method was split into two sections, one so that the BST would operate and the other so that the B-tree could operate. Within the BST section the larger file that was given would be opened and each line would be read and stored within the tree. Then the number of nodes within the tree and the height of the tree would be printed along with the time it took for each line to be read. Next, the second file, the one created by the user, would be read and the words within this smaller file would be searched within the tree all so the similarity of the two words given could be printed for the user. Again, the running time to compute the similarities would be displayed for the user.

Then came the next section which involved the B-tree. Again, each line from the larger file would be read and the data list of the tree was filled with a list of the word embedding objects which would be sorted alphabetically with the help of the functions found within the B-tree class. Then the second file was read through. Again, this file contained a list of two words. The two words were each stored as objects of the word embedding class to accommodate to the search function found within the B-tree class. The search function would return their location and the cosine distance formula would compute. The similarities were then printed out along with the time it took for these similarities to run.

**Experimental Results**

|  |  |  |
| --- | --- | --- |
| Number of Word Pairs | Binary Search Tree (BST) | B-Tree (max\_data = 5) |
| 15 | 0.007227897644042969 | 0.007316112518310547 |
| 30 | 0.01764512062072754 | 0.022891759872436523 |
| 50 | 0.022414684295654297 | 0.030154943466186523 |
| 120 | 0.05241584777832031 | 0.058554887771606445 |

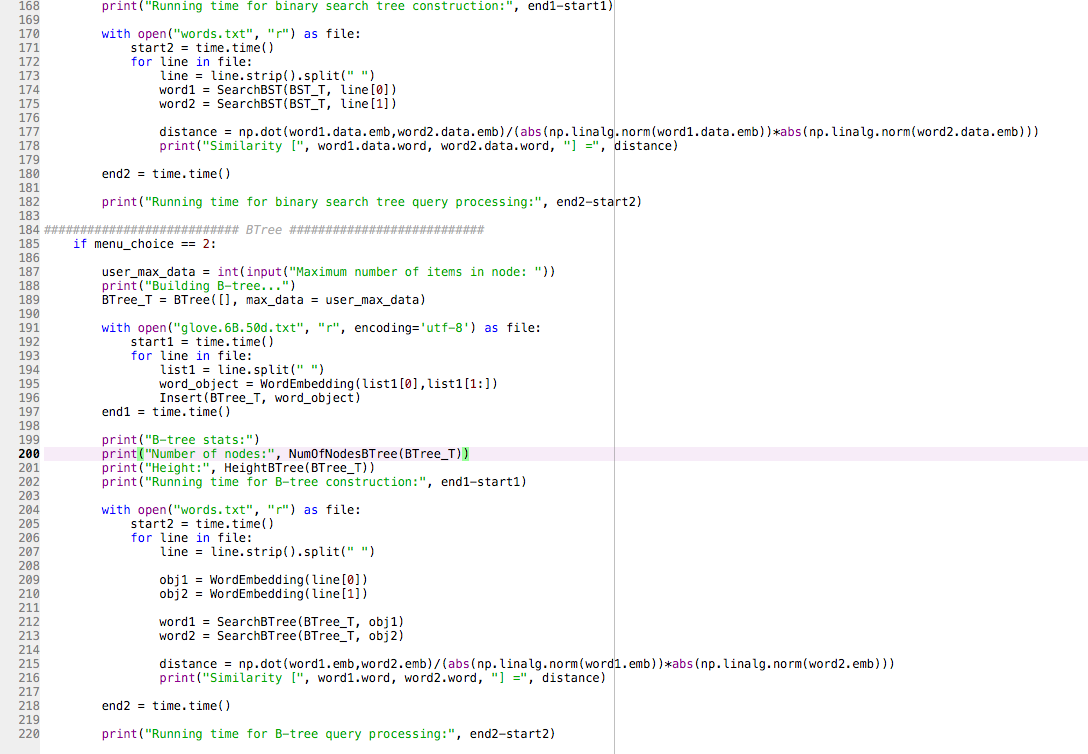
|  |  |  |
| --- | --- | --- |
| Number of Word Pairs | Binary Search Tree (BST) | B-Tree (max\_data = 3) |
| 15 | 0.0038268566131591797 | 0.0189361572265625 |
| 30 | 0.023446083068847656 | 0.015089035034179688 |
| 50 | 0.02054572105407715 | 0.028614044189453125 |
| 120 | 0.06256365776062012 | 0.058264732360839844 |

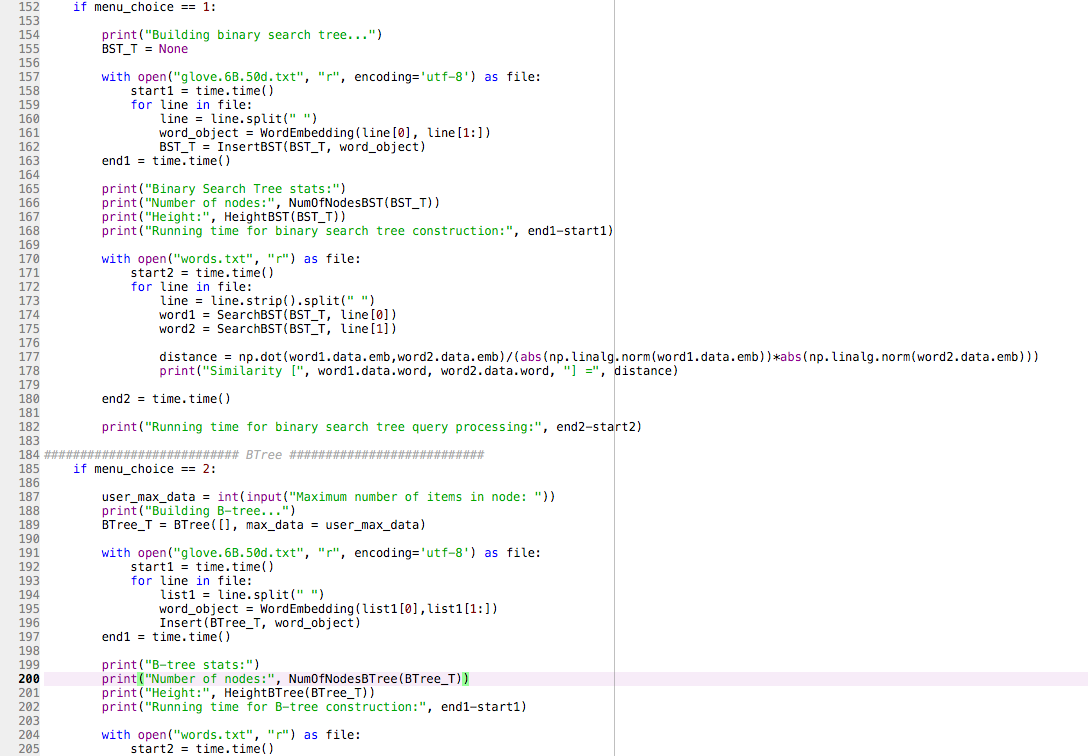
The figures above display the running times for both data structures used in this program with a different max\_data for B-tree. This shows that both Binary Search Tree and B-Tree have running times that are very similar however, when taking a closer look at the time the BST has a slightly faster running time than the B-tree when the max\_data was set to 5. The B-tree had a slightly better time than the BST when the max\_data was set to 3.

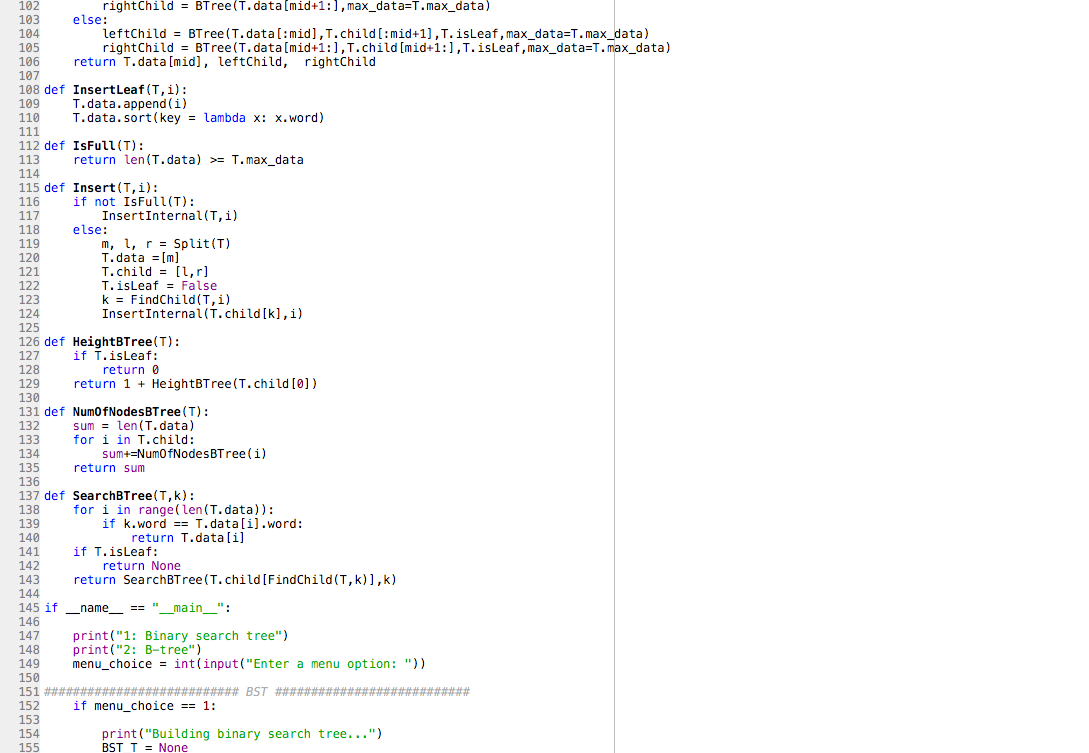
**Conclusion**

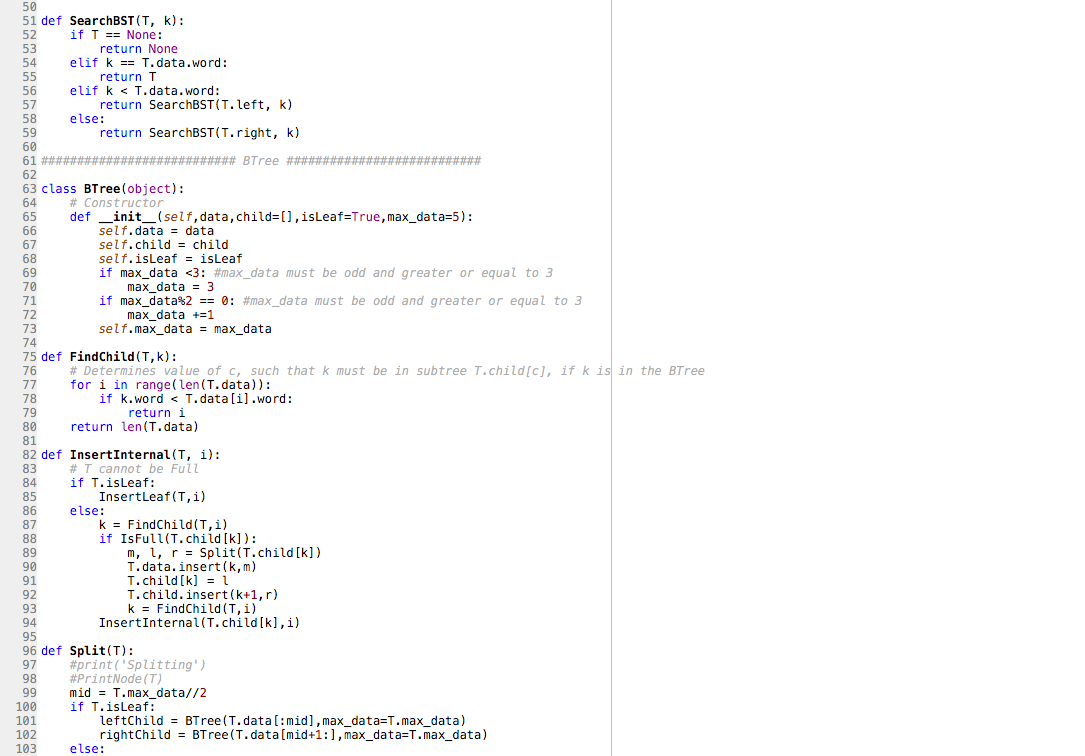
This program successfully completed its task which again was to compare the running times of two data structures, Binary search tree and B-tree. In this lab, I became more familiar with the basic functions found within BST and B-tree. I also became more comfortable with creating classes, attributes, and objects. The most trouble I had during this lab was accommodating the Search function for both data structures to the word embedding, specifically the B-tree, but once I became more familiar with objects and their purpose I was able to successfully perform this task.

**Appendix**









**Academic Honesty Certification**

I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class.

Sofia Gutierrez October 26, 2019