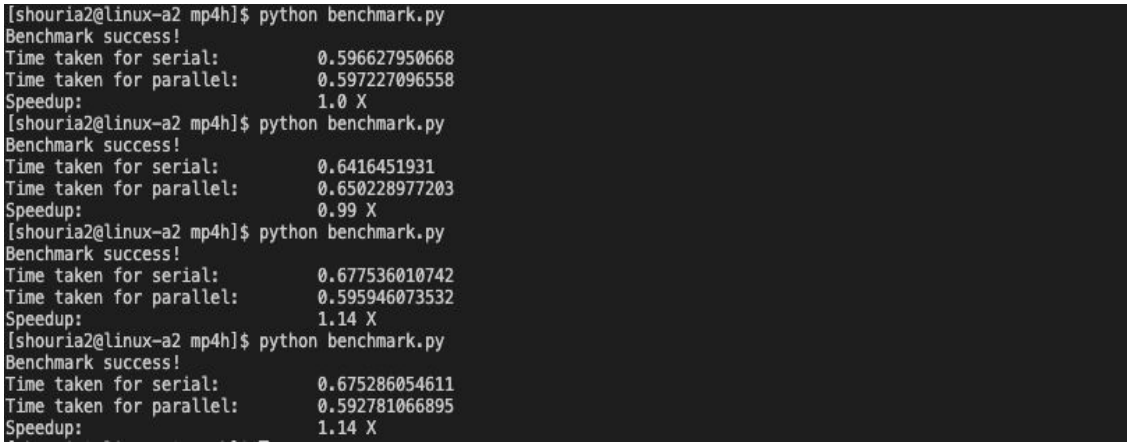
Shouri Addepally
ECE 220 Honors
05/03/2020

MP 4 Honors: Report

Questions:

1. Below are a few benchmark results:



2. While generating my threading model, I decided to keep track of the number of levels that had already searched in my binary tree using SWITCH_TO_SERIAL_FACTOR which was set to 4. Until the search crossed a level of 4, my threading model would continue to spawn parallel threads and would use the parallelized searching approach. If this "threshold" of levels was crossed, I used serialization by calling the recursive_helper which would stop spawning threads and use serialization to finish searching the rest of the binary tree. The reason why I used this threading model is because the 'pthread_create' and 'pthread_join' operations are costly and as we progress down the tree, we will slow down in the searching process if we try to parallelize. As a result, a more logical approach is to parallelize the initial levels of the tree and if we get to a larger amount of levels in the binary tree we can perform serialization.

3. While I built my tree, I used a random number generator to fill in values on both the right and left branches of each node. While building my tree, I also used recursion to continue filling out random values on every right and left branch of the tree. As part of the tree generation, I also built an array which stored all the values that were generated as part of the tree. It is important to note that I set the rightmost leaf-node value with -1000, and this was the value I would search for in both the find_serial and find_parallel functions.

4. If our input tree looks like a linked list we cannot improve the runtime. This is because the binary tree does not have the invariant of being generated in any particular order. As a result, there are possibilities where we repeatedly traverse the linked list inefficiently which will cause a slowdown in the time taken to find a particular element.