Hayden Walpole
U1210998
5/19/2024

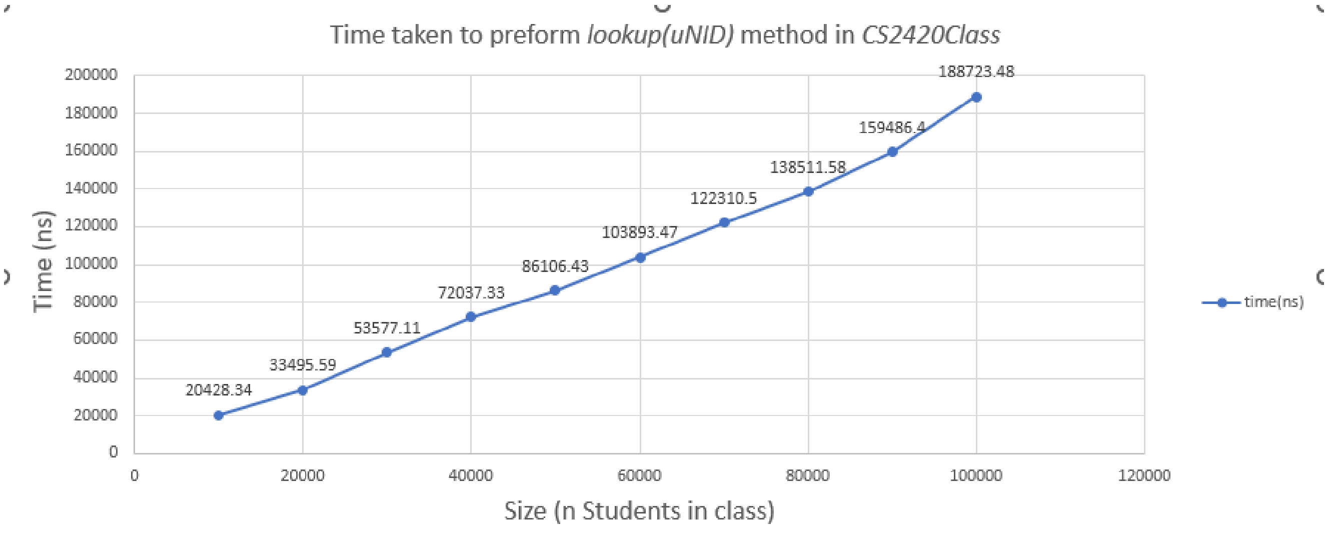
Analysis Document — Assignment 2

1. My Partner for this assignment was Zijia (Harry) Xie, and they submitted our assignment on Gradescope as a group.

2. This was a new experience for me to coordinate and do pair programming with a partner. I enjoyed both roles, but I liked being a navigator better because I felt like I was able to think about the problems more instead of also having to focus on typing. I could also tell Zijia liked being the Driver better, but we would still switch off. We wrote all of the code through pair programming, but I would say that Zijia ended up writing about 70% of the code, and me the other 30%. I attribute this to the fact that we had more discussion while I was driving on some more challenging portions, and Zijia can type faster than I can. We spent about 4 hours programming, and 6 hours writing tests for this assignment. I Enjoyed working with my partner, and we are planning to work on Assignment 3 together as well. On Assignment three I want to do better with tracking time and make sure we are switching about every 30 minutes and I am contributing enough verbally while I am driving.

3. Comparable and Comparator are both used for comparing objects in java, but there are differences between the two. Comparable takes one object as an input and compares it with the object it is called from to define natural ordering. Usually this is added to an existing class as the way we will do comparison for this class. Comparator is its own class with only one method that takes two inputs to compare with one another. For example, in this assignment we wrote multiple Comparable type functions that would order student objects based on a characteristic like UID, or Alphabetical order. An example of a situation when it would be better to used Comparator would be for two shapes, where we want to know which one has more sides. In our CS2420ClassGeneric extra features Comparator cannot be used instead of Comparable, because we need to define was we are comparing about each student in each feature.

4. For this Timing experiment we used a value of 10,000 times to loop with each N starting at N=10,000 to N=100,000 increasing N in steps of 10,000. We used the provided code in "Collect_running_times.txt" assignment description and put it in a new class called CS2420ClassTimer to run our experiment. For each value of N we would randomly generate a number to use as a UID and call lookup 10,000 averaging the time it took to find the student with that UID number. Code was also include in the provided code to account for the cost of running the "timesToLoop" loop. I expect the running times to gradually increase for this function as N increases, and my hypothesis would be that they will increase linearly.



5. The Big-O notation appears to be O(N) since the line appears to be linear and increasing at a steady rate. This means that the time it take to preform the lookup(uNID) will always be directly proportional to the size of students contained in that class. To know with a better confidence that this is true we could expand our size of N and make sure that the data still appears to be linear with more points, or take more tests with each size N and take an average time between them for each size N time.