# Cmput 291
# Project 2
# Sarah Morris, Victoria Bobey, Eldon Lake

**Experimental Results**

Hash Table times:

| Search #/Search Type | Key | Data | Range |
|---|---|---|---|
| 1 | 5.1021575927734375 | 1086629042.1485901 | 111.38916015625 |
| 2 | 6.413459777832031 | 366598895.072937 | |
| 3 | 4.792213439941406 | 1265428450.1075745 | |
| 4 | 5.412101745605469 | 1431917526.960373 | |
| 5 | 4.9114227294921875 | 311798863.17253113 | |
| Average | 5.326274843 | 836350755.5 | |

B-Tree Times:

| Search #/Search Type | Key | Data | Range |
|---|---|---|---|
| 1 | 7.700920104980469 | 539477543.592453 | 116.20521545410156 |
| 2 | 4.696846008300781 | 964701429.8439026 | 97.70393371582031 |
| 3 | 6.890296936035156 | 461164918.422699 | 90.28911590576172 |
| 4 | 7.2002410888671875 | 539477543.592453 | 121.80805206298828 |
| 5 | 7.0095062255859375 | 862524826.7650604 | 102.71072387695312 |
| Average | 6.699562073 | 673469252.4 | 105.9447008 |

Index File times:

| Search # /Search Type | Key | Data | Range |
|---|---|---|---|
| 1 | 45.490264892578125 | 30.58910369873047 | 82.89813995361328 |
| 2 | 5.984306335449219 | 36.88335418701172 | 121.9034194946289 |
| 3 | 5.698204040527344 | 35.71510314941406 | 89.00165557861328 |
| 4 | 5.793571472167969 | 31.614303588867188 | 79.29801940917969 |
| 5 | 5.602836608886719 | 5.412101745605469 | 104.28428649902344 |
| Average | 13.71383549 | 15.39707177 | 76.70593261 |

**IndexFile:**

Our IndexFile consists of a normal b-tree table that is used for key search and range search and a secondary index b-tree that is used for data search.

The b-tree was fast with both key and range search but very slow with the data search. The hash table proved to be no better at this. The secondary index allows the data search to be preformed in much the same way that the key search is preformed, but instead of the keys being the index that is being searched through, the data is.

**Analysis:**

The hash table proved to be efficient for the key search. On average it only took 5 microseconds. The data search was much slower, averaging a runtime of 83635075.55 microseconds. The range search is also quit slow. It took on average **(INSERT AVERAGE HERE)** microseconds.

The b-tree proved to be efficient for both key and range search but very slow for data search. Key search can be preformed with an average time of 5 microseconds. Range search can be performed with an average time of 106 microseconds. The data search took longer at an average time of 673469252.4 microseconds.

The data search should be much slower for the hash table and the b-tree since each key/data combination must be checked until one is found that matches the given data. This could potentially mean that all of the key/data pairs must be checked.

The range search was slow for the hash table since the data is stored in no order. The program must search through all of the keys looking for any that fall between the given upper and lower bounds. The b-tree did not have to do this since the data is stored in more of a sequential order.

Since the b-tree was able to preform range searches significantly faster than the hash table, we choose to make our indexFile a b-tree. The b-tree preforms fast key searches as well as range searches. The data search was preformed on a b-tree with a secondary index. Having this allows for the completion of all three searches in very little time.

The secondary index in the indexFile allows the program to run through the b-tree like any other key search but instead of the keys being the normal keys from the key/data pair the key is the data portion of this pair. This means that the data search is just as fast as the key search. This is a significant improvement from the hash table and b-tree (with no secondary index) performance of the data search.

As you can see from the experimental results the key search and data search that were preformed by the indexFile have very similar averages for time. Both are very quick with run time averages of 14 and 15 microseconds respectively. The Range search takes more time, but is still very fast at only 77 microseconds on average.