James Banh	52316090
Vince Cerda	23077304
Tyrone Minh Tong	39813123
Bao Tran	30972707

Assignment 3: M3 Report

Test Queries:

1. cristina lopes
2. master of software engineering
3. machine learning
4. python tutorial
5. for loop
6. thornton profile
7. linear algebra course information
8. ics45c project0
9. faculty awards
10. artificial intelligence courses
11. data science major
12. how to traverse a graph
13. computer science degree
14. computer science scholarships
15. research opportunities
16. shannon alfaro
17. academic dishonesty
18. recursion
19. uci hackathons
20. job opportunities for undergraduate students

In the beginning, we had trouble with the queries, Cristina lopes and master of software engineering. We kept getting website urls for html documents that were very short. It turned out that we weren't normalizing our term frequency scores and IDF scores. To normalize our term frequency scores, we used the formula 1 + log(tf) where tf is the raw term frequency of the term in the document. To normalize our IDF scores, we used the formula log(N/df) where N is the total number of documents in the Index and df is the number of documents that have the term in question. Additionally, we noticed that websites that had query terms which were in the title html tag or in headers, were ranked very poorly. To fix this, we added weight to words that were in tags we deemed important. We gave a score of 3 (3 * normalized term frequency) to words that were in the title tags, and a score of 2 (2 * normalized term frequency) to words that were in header (h1-h6), strong, and b tags.

Another problem we ran into is search time, which was very prominent in queries that had more than 3 terms (master of software engineering, linear algebra course information) and queries with terms that have a really high IDF score. Regarding the former, we decided to implement a champions list during indexing. For each token in the index, we sort the postings list based on each document’s tf-idf score. We then chose the top 2500 documents in the postings list of each term. This way, each token is guaranteed to have less than 2501 documents in their postings list, making searching significantly faster. The figure 2500 was chosen based on experimenting with the value of K in top K. We tried 500, 1000,2500 ,5000 , and no top K restriction and we found that below 1000, our results began to show different websites compared to the results shown when a champions list was not used. 5000 was a decent figure but it was still a tad bit slow so we brought K down to 2500, which outputted similar results as an Index with no champions list.

Regarding the search time problem with query terms having high IDF scores, we decided to filter those words out in search time. We only calculated the scores of each document in the postings list of terms that had an IDF score that was greater than 1. That way, the scores of documents of very common terms such as “and” or “or” would not skew the scores of documents of “rarer” terms.

We also ran into problems with our search engine outputting duplicate documents. As an example, the query “cristina lopes” would output:

“https://www.informatics.uci.edu/explore/faculty-profiles/cristina-lopes/”
“https://www.informatics.uci.edu/explore/faculty-profiles/cristina-lopes/#content”

These two urls led to pages with the exact same content. We encountered similar problems with the query “master of software engineering” and other queries on the list. We fixed this problem in the indexing process. We wrote a simhash function to determine the fingerprint of a document and used this fingerprint to check if the same fingerprint has already been indexed. If a document with the same fingerprint has already been indexed, we don’t index the document in question. This helped us filter documents that were exact duplicates which reduced the number of documents indexed from 55393 to 45793.