Reading

On the topics Cuckoo hashing, Splay trees and Count min sketches from the following resource:-

1. SPLAY TREES

<http://web.stanford.edu/class/archive/cs/cs166/cs166.1166/lectures/10/Slides10.pdf>

1. COUNT MIN SKETCH AND COUNT SKETCH

<http://web.stanford.edu/class/archive/cs/cs166/cs166.1166/lectures/11/Slides11.pdf>

1. CUCKOO HASHING

<http://web.stanford.edu/class/archive/cs/cs166/cs166.1166/lectures/13/Slides13.pdf>

1. MARKOV’S INEQUALITY AND CHEBYCHEV INEQUALITY

[https://en.wikipedia.org/wiki/Markov%27s\_inequality](https://en.wikipedia.org/wiki/Markov's_inequality)

[https://en.wikipedia.org/wiki/Chebyshev%27s\_inequality](https://en.wikipedia.org/wiki/Chebyshev's_inequality)

1. OPEN QUESTIONS RELATED TO CUCKOO HASHING

<http://www.eecs.harvard.edu/~michaelm/postscripts/esa2009.pdf>

IMPLEMENTATION

1. Attached cuckoo hashing code along with the mail
2. Used the ideas of splay tree and frequency estimator (count min sketch) in the project on data analysis.

The project aims at analyzing big data and based on the history of input would implement the following functionalities   
1. **Search for a query**.  
2. **Recognize the input pattern**  
3 **Estimation of frequency of search of a query**.  
4. Give suggestions to the user taking into consideration the previous history of search.  
5 Use of concurrent programming in Haskell in order to cut down the time required for the above functions.

For **Search for a query, recognize the input pattern**, splay tree is used. When user searches for some query .It splays up the query to the root. In case not present splays the parent to the root.

After splaying it displays the root and 2 its children and 4 grandchildren. As they tend to have the most recently searched queries and also queries searched nearby to the current search.

For **Estimation of frequency of search of a query** we use count min sketch. Which gives rough yet very good estimate of frequency of a query. Its makes use of four hash tables each with an independent hash function. That too in considerable less memory

The GitHub link is:- <https://github.com/Adildangui/HASKELL->

https://github.com/himanshu-sopho/research-internship#research-internship

REPORT OF THRESHOLD AFTER WHICH PROBABLITY OF REHASH IS HIGH WITH N NUMBER OF TABLES IN THE HASHING

1.For only one hashtable with number of buckets 11 of all the inputs given to be hashed the need to rehash the table arises at the 8th number input.

2.For 2 tables with the same parameter each as previous hash table, the need to rehash arises at 13th number input.

3.For 3 tables with the same parameter each as previous hash table, the need to rehash arises at 17th number input.

4.For 4 tables with the same parameter each as previous hash table, the need to rehash arises at 19th number input.

5.For 5 tables with the same parameter each as previous hash table, the need to rehash arises does not arrive at all with the same set of inputs.