**National University of Computer & Emerging Sciences**

**Karachi Campus**



**Project Proposal**

**Ashanti:**

**A sophomore take on the modern search engine**

**[]**

**Group Members:**

**Fawad J.Fateh 19K-1449**

**Zaid Bin Shahab 19K-1512**

**Arun Jai 19K-1437**

* **Introduction**

Ashanti is our take on an efficient, fast and compact search engine using the concepts we learned in the data structures course. Today’s world and the internet would be incomplete without searching. A search engine achieves the task of searching through various techniques and algorithms. As the data threshold continues to grow day by day, the need for a space and time efficient search engine becomes more and more necessary. An ideal search engine should have near constant search time no matter how dense the data set. But for this the conditions and the data set itself needs to be ideal(i.e. clean). It is difficult and time consuming to process raw, unfiltered data. Before processing large chunks of data it is advisable to clean and organize it in a form which is suitable enough to be processed without wasting processing power on junk and bad sectors. We have used the concepts of data structures such as hash tables and linked lists and trees in order to accomplish our goal of an efficient search engine which can process and category handle a very large data set in almost constant time( best to average case ) and always less than 0(n) (Worst case) search time. The brute force method of linear searching has been avoided in this approach. In order to first clean and filter the large data set into smaller and less complex data sets we have also developed a file cleaner utility which works in lieu of the search engine.

Text

Description automatically generated

* **Existing System**

Existing search engines like those used in web browsers and pdf display applications and other relevant text editors.

* **Problem Statement**

Avoid linear searching in the worst case in a corpus of more than 35k words and have a utility to filter out bad words(bad data).

* **Proposed Solution**

A hash table is used to store the individual words in the data set. The hash table on average provides almost a near constant 0(1) lookup time and in the worst case is close to 0(logn) (in case of collisions which are handled by linear probing). A file cleaner utility has also been developed in lieu of the search engine which can break any document which contains upwards of 6000 lines into six smaller documents all while removing bad- incomplete words and lowercasing them(if there is a need for a uniform data). A 26-m arry(prefix tree) is used to store the words for the dictionary which always finds the meaning of the given word in 0(L) where L = number of alphabets of the word. Ascii art is used to generate runtime bar graphs of the document’s wise frequency of the searched word. A linked list is used to maintain the postings list which contain the number of documents the word is present in.

Text

Description automatically generated

* **Salient Features**
* Single word search
* Multiple term search
* Bar graphs of the frequencies of the word in the document corpus
* A file cleaner which breaks a big document into six smaller documents on some preset conditions
* A prefix tree to display the meanings of search words
* Ascii art for general aesthetics
* **Tools & Technologies**

C++ and visual studio.

* Accept
* Reject

**Course Teacher:** Sir Basit Jasani

**Signature:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_