**Project proposal**

Name: Omer Rastgar

ID: or05554

Value set: (Example)

|  |  |
| --- | --- |
| **Problem string set** | **Solution string set** |
| “The compressor inlet valve is blocked” | “Servicing required ” |
| “High temperature in compressor room ” | “Install an exhaust fan to reduce temperature” |
| “Reduce flow due to bends in the pipe” | “Change design of piping to increase efficiency” |
| “Dryer inlet blocked” | “Servicing required ” |
| “The Compressor filter is choked” | “Servicing required ” |

Aim:

Designing an algorithm that would be able to search the problem set for key words like “blocked”, “inlet” and provide us with all problems containing these keywords.

1. Design both algorithms
2. Get theoretical time complexity
3. Get time from time it function
4. Display distribution with graphs on large data sets

Analysis: (Rabin karp algorithm)

While using Rabin karp we have to traverse through each and every string in the problem set and find the key word that is provided by the user. It uses naïve approach to traverse through the data set thus has a worst time complexity is O(n\*m).

Best case= O(n+m); where n is number of characters in the problem set and m is the number in keyword.

According to the problem set above:

n= 142 characters (includes spaces)

m= 7 characters

Example: (my algorithm)

In the preprocessing phase a hash table (not dictionary) is created with all the words present. They also contain a tuple that contains group numbers of all common words. The hash function would be SHA 256 would be used to decrease collisions.

If a word is too common meaning it is present in 0.25 of the groups, it will not be added therefore:

Words like:

1. “is”
2. “compressor”
3. “the”

As they are too common within the problem set.

When a keyword is entered it will split in words, each word in a separate thread goes through the function and all group numbers stored and then compared with each thread, the common group would be shown to the person.

Analysis of my algorithm

The worst case would be O(n) due to the collision that can arise, where n are the words in the hash table.

We can see that there were 142 characters in the example set above but only 27 words, furthermore common words would be deleted from the table so we only have 21 words. Thus even the worst case of my algorithm is better than Rabin karp algorithm at normal conditions.

Best case= O (1)

This is because we are searching a hash table which has a complexity of O(1)

Data structure utilized in my algorithm:

1. Lists
2. tuple
3. Hash table (I will be building my own)
4. Linked list

Libraries used:

1. Hash lib

Benefits

1) Reduce time searching for required string

2) Can also be used for searching:

1. Cross word puzzles
2. Product search
3. Encyclopedia word search

Disadvantages

1. High space complexity
2. Requires a preprocessed hash table which has to be updated every time a new string is added

**Add-ons:**

1. Fuzzy search implementation (Soundex algorithm) (problem: increases space complexity)

* Fuzzy search helps in spelling errors identification