

*THE O(1) DICTIONARY*

DSA PROJECT

Faculty - Dr. Oswald C

Slot – F2

Course – Data structures and algorithms CSE2003

Mohammed Ali – 18BCE1327

Siddharth S -18BCE1171

**Motivation Behind Doing This Project**

Sometimes we are unable to find meaning of words online specially when trying to find translations or what is the word spoken in a different dialect specially in India where there not only multiple languages but even more variations in different dialects, hence our project could help a implement a dictionary which could find the meaning of the words in O(1) time.

This can also be used by anyone who wants to have his own personal dictionary where he can manage all the words and edit and delete as per his own will.

**Problem statement**

Hashing guarantees O(1) almost every time hence it only made sense that to implement a dictionary it would be the best choice which would allow easy and quick access to the words and their meaning allowing us to edit or delete or insert just by passing the word through the hash function . hence hashing was used instead BST which has complexity of O(logn).

**Algorithm of Solution**

Upon opening the file, you will be asked to choose from the given options

1. Insert
2. Find a word
3. Delete a word
4. Display entire dictionary

Upon selecting insert the program takes in the word and the meaning if the word is already present at the index then it will ask if you want to update it or not otherwise it is inserted at the hash index value.

If you want to find a word input is taken and passed though the hash function if the word is available at the position then it prints out the meaning else word does not exist.

If delete option is chosen, first check if no word is present then if only one word is present then traverse through the list until you find the word and next is not null otherwise if it’s the last word delete it.

If display is chosen iterate through the table size and display all the words with respect to its index.

**Data structure and Algorithm**

We have implemented an ASCII hash function as method to index the values into the table, Which uses all the letters of the word to minimize clashes rather than the popular one which just uses the first few letters.

The functions and data are initialized with the help of classes and friend class.

**The Pseudocode for the hash function is**

Hash(string word, int max)

{

Sum=0

For(i=0,i<word.length(),i++)

{

Sum=sum+word[i]

**//finding the sum of ascii values of all the letters for better entry**

}

Sum%=max;

**//modulo with the table size**

If sum<0

Sum+=max

**//just in case if it enter negative value**

Return sum

}

**Future Work Proposed**

To make the program more efficient we can implement separate chaining which allows us have linked list in the location where collision occurs when 2 or more keys have the same ASCII value.

We could also add a database which would help the user by saving the dictionary and allow access whenever he wants to.

**Time Complexity Analysis**

The time complexity of the code we have implemented in the best case is O (1), because when the number of elements is lesser than the number of holes, meaning the load factor is less than one, O (1) is achieved

But if the index is already filled up with some other word then the algorithm looks for the next empty space hence this could lead to the worst case which is O(n).