# COP290: PLAGIARISM CHECKER
# ASSIGNMENT 7

**DIPEN KUMAR**
**2018CS50098**

## MEASURE OF SIMILARITY –

In this Assignment I have used cosine similarity as a measure of text similarity. [https://www.machinelearningplus.com/nlp/cosine-similarity]
Cosine is defined as dot product of two vector whole divided by magnitude of both vectors.

Here, vectors are array of weighted frequency of words.
Weighted frequency was calculated using TF times IDF [http://www.tfidf.com]

## ALGORITHM EXPLAINED–

I have read files character by character and when I got any particular character from some characters defined in code. I stop and take the array of characters obtained so far as a single word. After getting the word I have hashed it to calculate its index to store in array of structure with two attribute that are word and its frequency in the given file. Let's say this array a map. So basically I am storing words of the document in a map and for collision I have used linear probing.

My hash functions are-
[http://www.cse.iitd.ac.in/~csz188011/col106-a3.html]

```
import java.lang.Math;
public static long djb2(String str, int hashtableSize) {
    long hash = 5381;
    for (int i = 0; i < str.length(); i++) {
```

```
        hash = ((hash << 5) + hash) + str.charAt(i);
    }
    return Math.abs(hash) % hashtableSize;
}
import java.lang.Math;
public static long sdbm(String str, int hashtableSize) {
    long hash = 0;
    for (int i = 0; i < str.length(); i++) {
        hash = str.charAt(i) + (hash << 6) + (hash << 16) - ha
sh;
    }
    return Math.abs(hash) % (hashtableSize - 1) + 1;
}
```

Once I have map for every document i.e. corpus files + test file, then I am calculating number of documents containing every words obtained so far from different documents.

Then I am calculating-

TF(t) = (Number of times term t appears in a document)
IDF(t) = log(Total number of documents / Number of documents with term t in it)

Once I have both. I have calculated cosine similarity i.e. cos of angle between two vectors.

NOTE: TF(t) is different for different documents.

**TIME COMPLEXITY –**

1. Initializing array take O(n) where n is the length of array.
2. Reading file character by character take O(k) where k is the number of characters in the file.
3. Insertion in map with constant average time complexity.

Above there steps is done for all files i.e. test file plus all corpus files. Time complexity is O(n) + O(k) + constant times O(m) where m is the total number of documents in corpus folder, will give O(nm) + O(km)

Calculating IDF take O(nm) where n is the size of array of structure and m is the total number of documents in corpus folder.

Calculating cosine with every corpus file take O(n) where n is the size of the array of structure.

Hence Over all Time complexity of my program is O(nm) + O(km)
n is length of array
k is numbers of characters in file
m is total number of files.

**SPACE COMPLEXITY –**

For each file I am using array of fixed length i.e. 10000.
Array of structure of length = 10000. Let's say it n.
Let say number of file be m.
Then my data take O(nm) space in program.

Also during run time I am using three long long int data type to store sum and square sums needed to calculate cosine similarity.
I am using same space for every pair. Hence it is a constant space.
Before calculating new cosine and overwriting the previous value I am printing it in console.

**THANK YOU**