# INTRODUCTION:

A Word Ladder is the connection between 2 same letter words, such that each transition to a step can only be made by replacing only 1 letter in the source word and the target word is always a meaning word. A good solution has a number of steps equal to the number of places the given words differ in (Hamming Distance). E.g.

Cold -> Cord -> Card -> Ward -> Warm

# SOLUTION:

# LANGUAGE: JAVA

# LIBRARY: JAVA.SIMPLE

# DATA STRUCTURE: QUE AND STACK AND SET AND ARRAYLIST

# MY UNDERSTANDING:

We have to designed an optimize approach like we can implement breadth first search which guarantee the optimize solution or we can implement Depth first Search and I have implemented DFS in this case because it takes first one letter difference word and recursively calls the function

I am using java for this assingemnt.it will return NULL if no path exit between two word

Using java. Simple to open j son file then storing it in a set and then taking input from user (Start\_name,end\_name) validating words from dictionary then taking all the word of same length from dictionary and putting them in a set and sorting it

FIND(start\_name,end\_name,dictionary,visited)

It is a main function which does the logic

It find all the one word difference from start\_name and then push

It in a stack in a pair like

[start\_name,one\_word\_differnce]

[start\_name,one\_word\_differnce1]

Then push them all in Queue

Then pop the Queue and check peek() value and check if it match the end\_word then return that stack otherwise it will pop the word from that stack and recursive call the function find again with argument

FIND(one\_word\_differnce,end\_name,dictionary,visited)

If it return path then function will backtrack and return path otherwise if it is null then it pop the Queue again and again call the function until the path is found or Queue is empty and if Queue is empty and return is also empty then it will return empty otherwise it will return the path

This is DFS which FINDS LONGEST PATH

FOR BFS

I RENEMAED FIND FUNCTION TO BFS ALL THE CODE IS SAME JUST FEW LINES MAKE IT BFS WHICH IS:

Queue copy = new LinkedList(store); //THIS PART MAKE IT BFS REST IS SAME AS FIND

while(!copy.isEmpty()){

Stack current = new Stack();

current = (Stack) copy.remove();

if(current.peek().equals(end\_name)){

return current;

}

}//FROM 101 TO 144 IT MAKES BFS

I COPY A STORE QUEUE IN COPY QUEUE AND THEN REMOVE ITS ITEM WHILE IT IS NOT EMPRY AS THEY RETURN SKACK WHICH ARE PAIR AND I CHECK THAT IF PEEK() OF EVERY STACK MATCHES END WORD IT WILL RETURN THE STAKC

# OPTIMIZATION:

Yes it is possible to optimize it more and make it more perfect

FLAWS:

IT RETURN NULL FOR SAME WORDS LIKE USER ENTER HIT AND HIT IT WILL SAY THEY ARE NOT CONNECT MEANS I AM USING NO ONE WILL MAKE THAT STUPID MISTAKE

# GITHUN: https://github.com/rajazeeshanshaheen/Wordladder-