# Documents

**Why We Use This Data Structures :**

When the Project was assigned to Us That Day We Discuss Which Data Structure was more Suitable For This Code From the Discussion We finalized the Doubly Linked List. In Github First We Write Code In a Doubly Linked List Data Structure. And Code Run In Good Time But After 2 Day We Know That Some File Is Not Opening In This Data Structure We Take Meeting But We Not Find Any Error So We Take New Data Structure As VECTOR AND UNORDERED MAP AND LINK LIST. This Code Has No Issue About File Handling So It Works In Good Time Complexity. This Data structure has a very Low Time Complexity As Other And Neutral Space Complexity.

## Time Complexity:

- O(N) For Add Data(Without Extra Words) In Linkedlist
- O(M log(M)) For Sorting Top K Word In Vectors
- O(K) For Print Top K Word
- O(K) For Search Specific Word's Frequency
- O(K) For Search Specific Word's Frequency
- O(N/2) Because For We Have To Travell In Full Linked List

## Space Complexity:

O(K) For Add Data In Linkedlist, Where K < N . Beacuse Some Node Will Repeat Then We Only Plus Count So All Data Will Not Add

- O(N) For Add Data In Vector.

**Why We Do Not Use Other Data Structures :**

**Stack And Queue:** We Have Not Use This Data Structure Because Of Time Complexity Of This Data Structure In This Project Is O(N^2) ( For Searching In Functions)

**Binary:** In Our Group, All Are Not Study This So.

**WHAT WE HAVE TO DO IN PROJECT :**

In the Project, We Have To Print the Top K Words From a File With As Small As Small Time Complexity And Space Complexity.

**WHAT WE ADD EXTRA IN THIS PROJECT :**

----In This Project, We Add 2 functions so users can Use This Program For Multiple Use.

-- 1. Search_Freq():: This Function Will Give A Frequency Of the given Word If the Word Is In a File.

-- 2. Search_Word():: This Function Will give words that have the same Frequency As User Given.

**FUNCTION WORKING :**

1 . Lower_String Function: Give Lower String So If User Mistakenly Write Word In Capital Or Mixed So We Use This Function

2 . Make Node Function: This Will Make a Linked List And Add Node If It Was Not Extra Or A Single Character.

3 . Compare Function: It Will Compare Node's Count So We Can Sort Easily

4 . Print Top K Function: It Will Print Top K Word If K Is Valid

5 . Search_Frequency Function: It Will Search A User Given Word's Frequency

6 . Search_Word Function: It Will Give All Word Of User Given Frequency

7. Delete_All Function: It Will Free the Memory Of the Linked List

8. Main Function: We Will Call All Functions According To User.

**TEST FILE :**

-- We Have Given You A Input File Which Give You In Github.

-- This File Has 698 Pages.

-- This File Has a Total of 223791 Words.

-- When We Do Not Include Extra Work Then It Works In an Average Time Of 10 Sec. But When We Include Extra Work Then It Increases To 13.5 Seconds.

-- First We Do a Project With Only a LinkedIn list This Takes 26 Seconds.