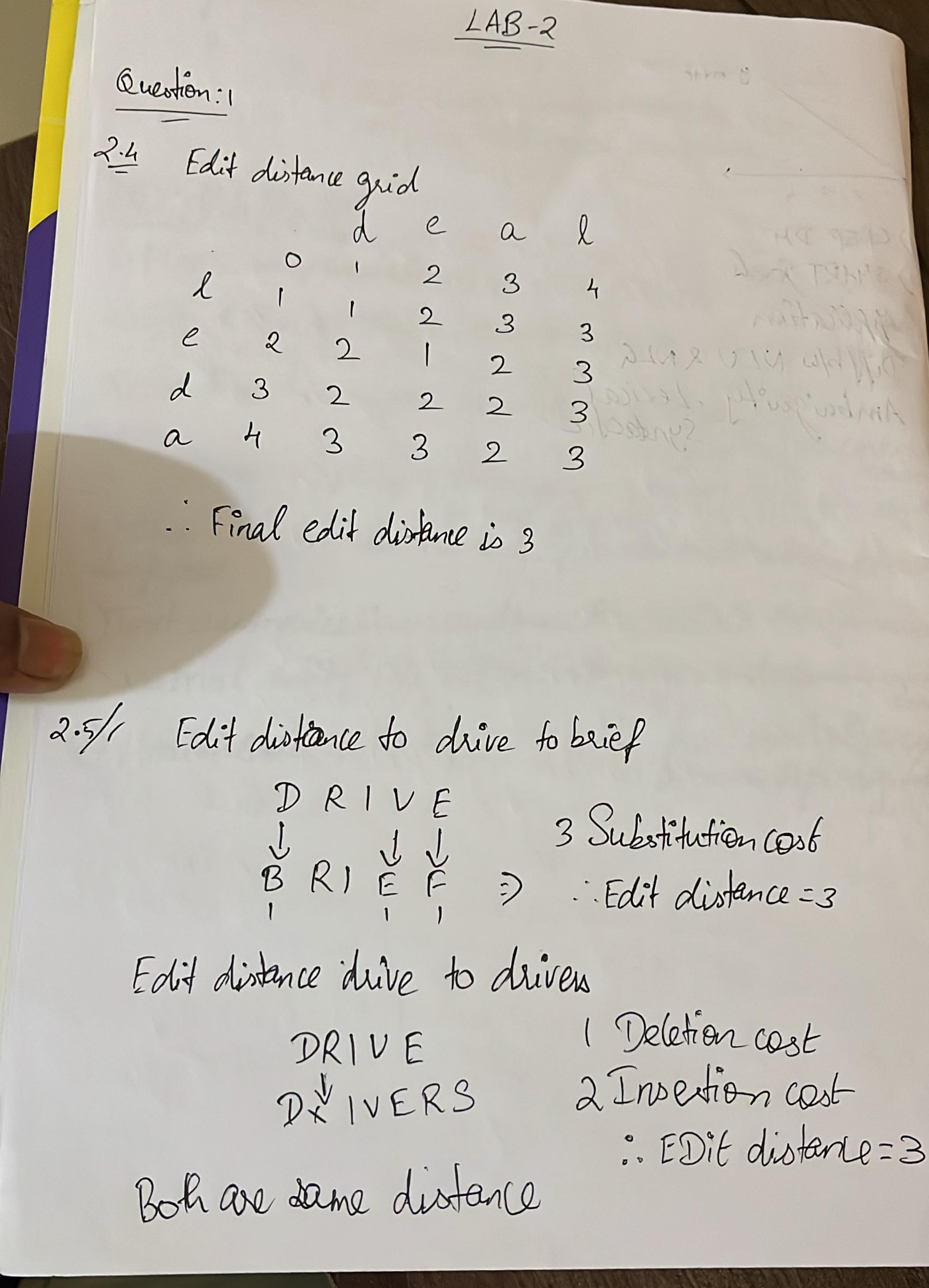
# Lab2:Edit Distance and Applications

**Report**

**Question 1**

## Question 2: Levenshtein Edit Distance

Library: NLTK

Function: nltk.metrics.distance.edit\_distance

This function calculates the Levenshtein distance, the minimum number of edits (insertions, deletions, substitutions) required to transform one string into another.

Example: 'leda' → 'deal'  
Edit Distance = 3

### Draft Plan:

1. Import required function  
 - Load edit\_distance from nltk.metrics.distance.  
2. Define input strings  
 - Assign two strings (e.g., 'leda' and 'deal') to variables.  
3. Call edit\_distance() function  
 - Pass both strings as arguments to compute the edit distance.  
4. Store the result  
 - Save the output of the function in a variable.  
5. Display the result  
 - Print the edit distance with a descriptive message.

### Test Cases:

1. edit\_distance('kitten', 'sitting') → 3

2. edit\_distance('flaw', 'lawn') → 2

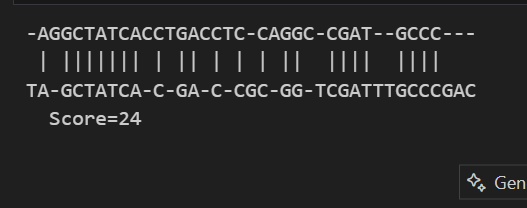
3. edit\_distance('intention', 'execution') → 5

## Question 3: DNA Sequence Alignment

Library: Biopython

Functions: pairwise2.align.globalxx, format\_alignment

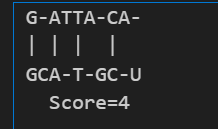
This performs global alignment of two DNA sequences using match scores only. The alignment output shows matches, sequences, and the alignment score.

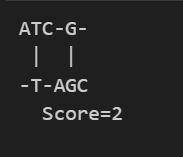
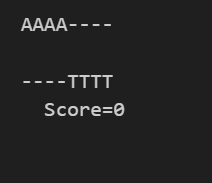
Example Sequences:  


**Draft Plan:**

1. Import Biopython modules  
 - Use pairwise2 for alignment and format\_alignment for readable output.  
2. Define two DNA sequences  
 - Assign the nucleotide strings to textA and textB.  
3. Perform global alignment  
 - Use pairwise2.align.globalxx() to align both sequences based on common base matches.  
4. Store all possible alignments  
 - The function returns a list of alignments with equal best scores.  
5. Display the best alignment  
 - Use format\_alignment() to show the first (best) result.

### Test Cases:



**Question 4: Word Correction Using Edit Distance**

Library: NLTK

Functions: nltk.corpus.words.words(), nltk.download('words'), nltk.metrics.distance.edit\_distance

This approach corrects misspelled words by comparing them to dictionary words using edit distance. It filters words of similar length, computes distance, and returns the top 5 closest matches.

Example Input: 'leda'  
Expected Output:

Suggestions for 'leda':

lea (edit distance = 1)

led (edit distance = 1)

Leda (edit distance = 1)

lede (edit distance = 1)

lepa (edit distance = 1)

### Draft Plan:

1. Import NLTK modules  
 - Import words corpus and edit\_distance function.  
2. Download the corpus  
 - Use nltk.download() to ensure the words list is available.  
3. Load the English word list  
 - Get the full vocabulary using words.words().  
4. Define a function to suggest corrections  
 - Accept a word and vocabulary list.  
 - Filter vocabulary to include only those with similar length (±2 characters).  
 - Compute edit distance between the input word and each filtered word.  
 - Sort by distance and return the top 5 closest matches.  
5. Input a misspelled word  
 - For example: 'leda'  
6. Call the correction function  
 - Get and print the top 5 closest English words based on edit distance.

### Test Cases:

1. Input: 'speling'

Expected Output:

Suggestions for 'speling':

apeling (edit distance = 1)

spelding (edit distance = 1)

spelling (edit distance = 1)

sperling (edit distance = 1)

spewing (edit distance = 1)

1. Input: 'korrectud'

Expected Output:

Suggestions for 'korrectud':

corrected (edit distance = 2)

porrectus (edit distance = 2)

correct (edit distance = 3)

correctly (edit distance = 3)

corrector (edit distance = 3)

1. Input: 'bycycle'

Expected Output:

Suggestions for 'bycycle':

bicycle (edit distance = 1)

bicycler (edit distance = 2)

bicyclo (edit distance = 2)

biocycle (edit distance = 2)

cycle (edit distance = 2)