Practical No. 2

Aim - Implement Dynamic programming algorithm for computing the edit distance between strings s1 and s2. (Hint. Levenshtein Distance)

Source Code -

```
import numpy as np

def levenshtein(s1, s2):
    if (s1 == ""):
        return len(s2)

    if (s2 == ""):
        return len(s1)

    if (s1[-1] == s2[-1]):
        cost = 0

    else:
        cost = 1

    res = min([levenshtein(s1[:-1], s2) + 1,
           levenshtein(s1, s2[:-1]) + 1,
           levenshtein(s1[:-1], s2[:-1]) + cost])

    return res



print(levenshtein("Execution", "Intention"))
```

Output -

```
runfile('C:/Users/ckt/Documents/KUNAL-workspace/P2.py', wdir='C:/Users/
ckt/Documents/KUNAL-workspace')
5

In [1]:

In [2]:
```