 # Payal Chavan
 # Homework-6
 # Summer 2024
 # CS 5800 Algorithms (Seattle)
 # Date: 06/28/2024

Question 1: Implementation of edit-Distance for two words.

Output: Demonstration of 5 word-pair examples:

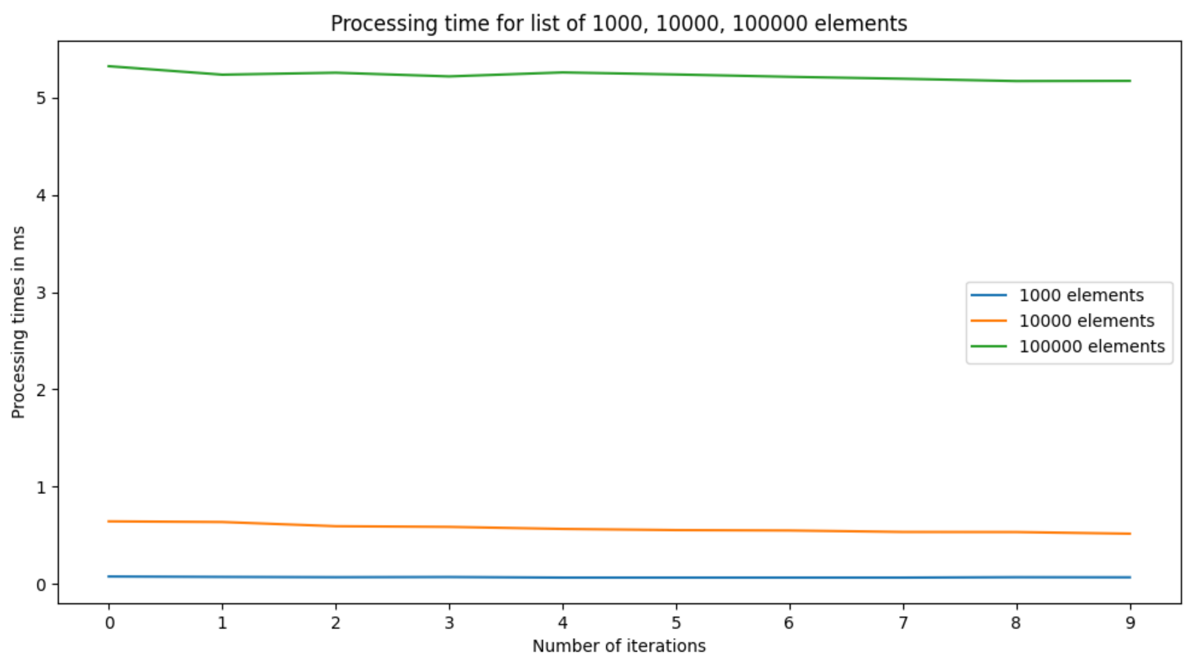
```
payalchavan@Payals-MacBook-Air /Users/payalchavan/Documents/Algorithms/Assignment6
⚡ /usr/bin/python3
/Users/payalchavan/Documents/Algorithms/Assignment6/editDistance.py
usage: python3 /Users/payalchavan/Documents/Algorithms/Assignment6/editDistance.py
<word A> <word B>

/usr/bin/python3 /Users/payalchavan/Documents/Algorithms/Assignment6/editDistance.py
seattle seahawk
Edit distance between 'seattle' and 'seahawk'
Distance: 4
Table:
    _  s  e  a  h  a  w  k
 _  0  1  2  3  4  5  6  7
 s  1  0  1  2  3  4  5  6
 e  2  1  0  1  2  3  4  5
 a  3  2  1  0  1  2  3  4
 t  4  3  2  1  1  2  3  4
 t  5  4  3  2  2  2  3  4
 l  6  5  4  3  3  3  3  4
 e  7  6  5  4  4  4  4  4
(base)

payalchavan@Payals-MacBook-Air /Users/payalchavan/Documents/Algorithms/Assignment6
⚡ /usr/bin/python3
/Users/payalchavan/Documents/Algorithms/Assignment6/editDistance.py gloomy glowy
Edit distance between 'gloomy' and 'glowy'
Distance: 2
Table:
    _  g  l  o  w  y
 _  0  1  2  3  4  5
 g  1  0  1  2  3  4
 l  2  1  0  1  2  3
 o  3  2  1  0  1  2
 o  4  3  2  1  1  2
 m  5  4  3  2  2  2
 y  6  5  4  3  3  2
(base)
```

```
payalchavan@Payals-MacBook-Air /Users/payalchavan/Documents/Algorithms/Assignment6
⚡ /usr/bin/python3
/Users/payalchavan/Documents/Algorithms/Assignment6/editDistance.py saturday sunday
Edit distance between 'saturday' and 'sunday'
Distance: 3
Table:
    _  s  u  n  d  a  y
 _  0  1  2  3  4  5  6
 s  1  0  1  2  3  4  5
 a  2  1  1  2  3  3  4
 t  3  2  2  2  3  4  4
 u  4  3  2  3  3  4  5
 r  5  4  3  3  4  4  5
 d  6  5  4  4  3  4  5
 a  7  6  5  5  4  3  4
 y  8  7  6  6  5  4  3
(base)

payalchavan@Payals-MacBook-Air /Users/payalchavan/Documents/Algorithms/Assignment6
⚡ /usr/bin/python3
/Users/payalchavan/Documents/Algorithms/Assignment6/editDistance.py lunch brunch
Edit distance between 'lunch' and 'brunch'
Distance: 2
Table:
    _  b  r  u  n  c  h
 _  0  1  2  3  4  5  6
 l  1  1  2  3  4  5  6
 u  2  2  2  2  3  4  5
 n  3  3  3  3  2  3  4
 c  4  4  4  4  3  2  3
 h  5  5  5  5  4  3  2
(base)

payalchavan@Payals-MacBook-Air /Users/payalchavan/Documents/Algorithms/Assignment6
⚡ /usr/bin/python3
/Users/payalchavan/Documents/Algorithms/Assignment6/editDistance.py amazing algorithm
Edit distance between 'amazing' and 'algorithm'
Distance: 7
Table:
    _  a  l  g  o  r  i  t  h  m
 _  0  1  2  3  4  5  6  7  8  9
 a  1  0  1  2  3  4  5  6  7  8
 m  2  1  1  2  3  4  5  6  7  7
 a  3  2  2  2  3  4  5  6  7  8
 z  4  3  3  3  3  4  5  6  7  8
 i  5  4  4  4  4  4  4  5  6  7
 n  6  5  5  5  5  5  5  5  6  7
 g  7  6  6  5  6  6  6  6  6  7
(base)
```

Question 2: Implementation of the maximum value sub-sequence algorithm.

Output:

```
⚡ /usr/bin/python3
/Users/payalchavan/Documents/Algorithms/Assignment6/subsequence.py
Sequence: [5, 15, -30, 10, -5, 40, 10]
Max_Sum: 55 for Max Subsequence: [10, -5, 40, 10]
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Sequence: [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11]
Max_Sum: 98 for Max Subsequence: [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11]
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Sequence: [3, -5, 1, 2, -1, 4, -3, 1, -2, 7, -6]
Max_Sum: 9 for Max Subsequence: [1, 2, -1, 4, -3, 1, -2, 7]
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Sequence: [10, 15, -4, 5, -11, -3, 31, -25, 1, -23, 8, 31, -50]
Max_Sum: 43 for Max Subsequence: [10, 15, -4, 5, -11, -3, 31]
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(base)
```



The algorithm processes each element exactly once, resulting in linear time complexity. Specifically, it runs in O(n) time, where n is the length of the input sequence. By comparing the current element with the sum ending at the previous position, it efficiently computes the maximum subarray sum.

From the above graph, we can see that, as the number of elements increases, the run time also increases linearly. If we see the processing time is increasing in multiples of 10 for number of elements 1000 and 10000, and subsequently for number of elements

10000 and 100000. Hence, we arrive at the conclusion that the time complexity is O(n) for finding maximum sum in a contiguous subsequence.

Reflection:

During this coding exercise, I gained practical experience in implementing the edit distance algorithm for comparing two words. Additionally, I explored the implementation of the maximum value subsequence algorithm.

Acknowledgements:

I would like to express my sincere gratitude to the following individuals and resources for their invaluable contributions to this assignment:
1) Prof. Bruce Maxwell: Thank you, Professor Bruce Maxwell for your guidance in this assignment. Your expertise in algorithms greatly influenced my work.
2) Classmates and TA's: I appreciate the discussions of my classmates, and TA's who clarified my doubts.
3) DPV Algorithms Textbook: This textbook was a useful resource for me to understand the basics of algorithms.
4) https://www.geeksforgeeks.org/edit-distance-dp-5/: This website helped me to clearly understand how the edit-Distance works through an illustration.
5) https://www.geeksforgeeks.org/largest-sum-contiguous-subarray/: This was a useful website to understand how to solve maximum sum sub-sequence problem.