KATHMANDU UNIVERSITY

Dhulikhel, Kavre

Lab No: #3

Subject: COMP 314

SUBMITTED BY:

Pratiksha Shrestha

Roll No: 53

Group: C.E.

Level: 3rd yr/ 2nd sem

SUBMITTED TO:

Dr. Bal Krishna Bal

Department of Computer Science

and Engineering

Date: 13 July, 2017

Matrix Chain Multiplication:

CODE:

matrixchain.py × LCS.py ×

```python
import sys, time

def gk(i, j):
    return str(i) + ',' + str(j)

def matrix_chain_order(p):
    n = len(p)
    m, s = {}, {}
    for i in xrange(1, n):
        m[gk(i, i)] = 0
    for l in xrange(2, n + 1):
        for i in xrange(1, n - l + 1):
            j = i + l - 1
            m[gk(i, j)] = sys.maxsize
            for k in xrange(i, j):
                q = m[gk(i, k)] + m[gk(k + 1, j)] + p[i - 1] * p[k] * p[j]
                if q < m[gk(i, j)]:
                    m[gk(i, j)] = q
                    s[gk(i, j)] = k
    return m, s

def optimal_parens(s, i, j):
    res = ''
    if i == j:
        return "A" + str(j)
    else:
        res += "("
        res += optimal_parens(s, i, s[gk(i, j)])
        res += optimal_parens(s, s[gk(i, j)] + 1, j)
        res += ")"
        return res

def main():
    print("Enter the array:")
    p = raw_input().split()
    p = [int(x) for x in p]
    m, s = matrix_chain_order(p)
    print 'Minimum cost of multiplication:', m[gk(1, len(p) - 1)]
    print 'Optimal Split value', optimal_parens(s, 1, len(p) - 1)

if __name__ == '__main__':
    a = time.time()
    main()
    print 'total run time is:', time.time() - a
```

OUTPUT:

```
C:\Python27\python.exe C:/Users/Prati/PycharmProjects/untitled/matrixchain.py
Enter the array:
1 2
Minimum cost of multiplication: 0
Optimal Split value A1
total run time is: 4.08100008965
```

```
C:\Python27\python.exe C:/Users/Prati/PycharmProjects/untitled/matrixchain.py
Enter the array:
23 21 1
Minimum cost of multiplication: 483
Optimal Split value (A1A2)
total run time is: 5.48199987411
```

```
C:\Python27\python.exe C:/Users/Prati/PycharmProjects/untitled/matrixchain.py
Enter the array:
14 8 7 1
Minimum cost of multiplication: 168
Optimal Split value (A1(A2A3))
total run time is: 5.38099980354
```

```
C:\Python27\python.exe C:/Users/Prati/PycharmProjects/untitled/matrixchain.py
Enter the array:
12 7 5 99 8
Minimum cost of multiplication: 4860
Optimal Split value ((A1A2)(A3A4))
total run time is: 6.50699996948
```

```
C:\Python27\python.exe C:/Users/Prati/PycharmProjects/untitled/matrixchain.py
Enter the array:
76 65 43 55 45 43
Minimum cost of multiplication: 522235
Optimal Split value (A1(A2((A3A4)A5)))
total run time is: 6.49900007248
```

```
C:\Python27\python.exe C:/Users/Prati/PycharmProjects/untitled/matrixchain.py
Enter the array:
98 56 34 24 75 45 34
Minimum cost of multiplication: 375096
Optimal Split value ((A1(A2A3))((A4A5)A6))
total run time is: 7.25199985504
```

```
Enter the array:
12 44 55 43 23 11 56 43
Minimum cost of multiplication: 101486
Optimal Split value ((A1(A2(A3(A4A5))))(A6A7))
total run time is: 10.0559999943
```

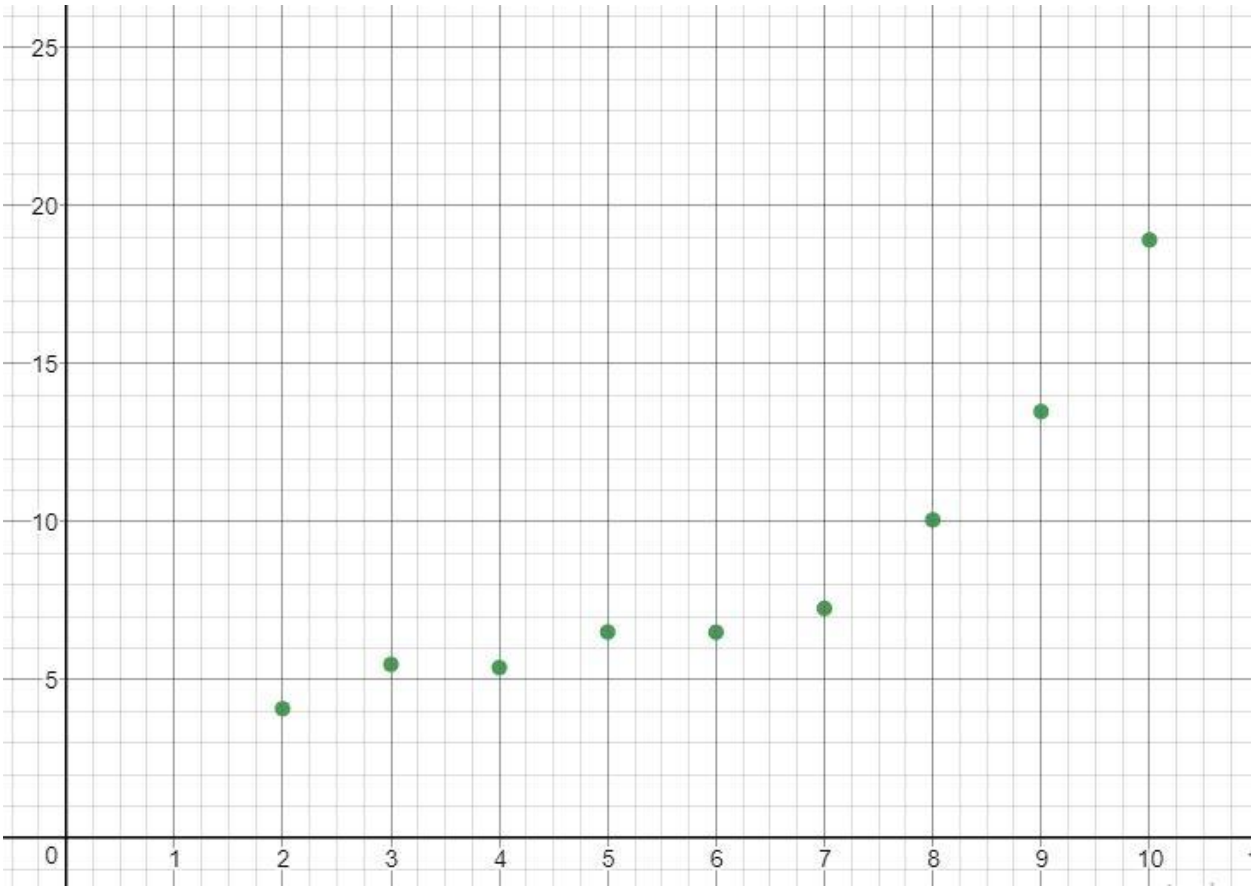
```
C:\Python27\python.exe C:/Users/Prati/PycharmProjects/untitled/matrixchain.p
Enter the array:
12 78 21 2 90 34 23 12 70
Minimum cost of multiplication: 16744
Optimal Split value ((A1(A2A3))((((A4A5)A6)A7)A8))
total run time is: 13.4839999676
C:\Python27\python.exe C:/Users/Prati/PycharmProjects/untitled/matrixchain.py
Enter the array:
12 89 78 65 34 27 10 32 17 19
Minimum cost of multiplication: 173030
Optimal Split value ((A1(A2(A3(A4(A5A6)))))((A7A8)A9))
total run time is: 18.9160001278
```

GRAPH:

x_1	y_1
2	4.08100008965
3	5.48199987411
4	5.38099980354
5	6.50699996948
6	6.49900007248
7	7.25199985504
8	10.0559999943
9	13.4839999676
10	18.9160001278

x-axis: Dimension of array

y-axis: Run time



Longest Common Subsequence

CODE:

matrixchain.py × LCS.py ×

```python
import time

def LCSLength(xstr, ystr):
  if len(xstr)==0 or len(ystr)==0:
    return 0
  if xstr[-1] == ystr[-1]:
    return LCSLength(xstr[:-1], ystr[:-1])+1
  else:
    return max(LCSLength(xstr, ystr[:-1]), LCSLength(xstr[:-1], ystr))

def LCS(xstr, ystr):
  if len(xstr)==0 or len(ystr)==0:
    return ''
  if xstr[-1] == ystr[-1]:
    return ''.join([LCS(xstr[:-1], ystr[:-1]), xstr[-1]])
  else:
    lcs1 = LCS(xstr[:-1], ystr)
    lcs2 = LCS(xstr, ystr[:-1])
    if len(lcs1) >= len(lcs2):
      return lcs1
    else:
      return lcs2

if __name__=='__main__':
  strA = raw_input('Input first string: ')
  strB = raw_input('Input second string: ')
  a = time.time()
  lcs = LCS(strA, strB)
  print 'Longest common subsequence: '
  print lcs
  print 'Time taken: ', time.time()- a

  assert len(lcs) == LCSLength(strA, strB)
  print 'Length of Longest common subsequence: %d' %(len(lcs),)
```

OUTPUT:

```
C:\Python27\python.exe C:/Users/Prati/PycharmProjects/untitled/LCS.py
Input first string: eaac
Input second string: ace
Length of first string:  4
Length of second string:  3
Longest common subsequence:
ac
Time taken to calculate LCS:  0.000089787600098
Length of Longest common subsequence: 2
Time taken to calculate the length of LCS:  0.000016794321
.................
Input first string: eacaeb
Input second string: aacebb
Length of first string:  6
Length of second string:  6
Longest common subsequence:
aceb
Time taken to calculate LCS:  0.0014678987654
Length of Longest common subsequence: 4
Time taken to calculate the length of LCS:  0.003678909991
.................
Input first string: abcdea
Input second string: abced
Length of first string:  6
Length of second string:  5
Longest common subsequence:
```

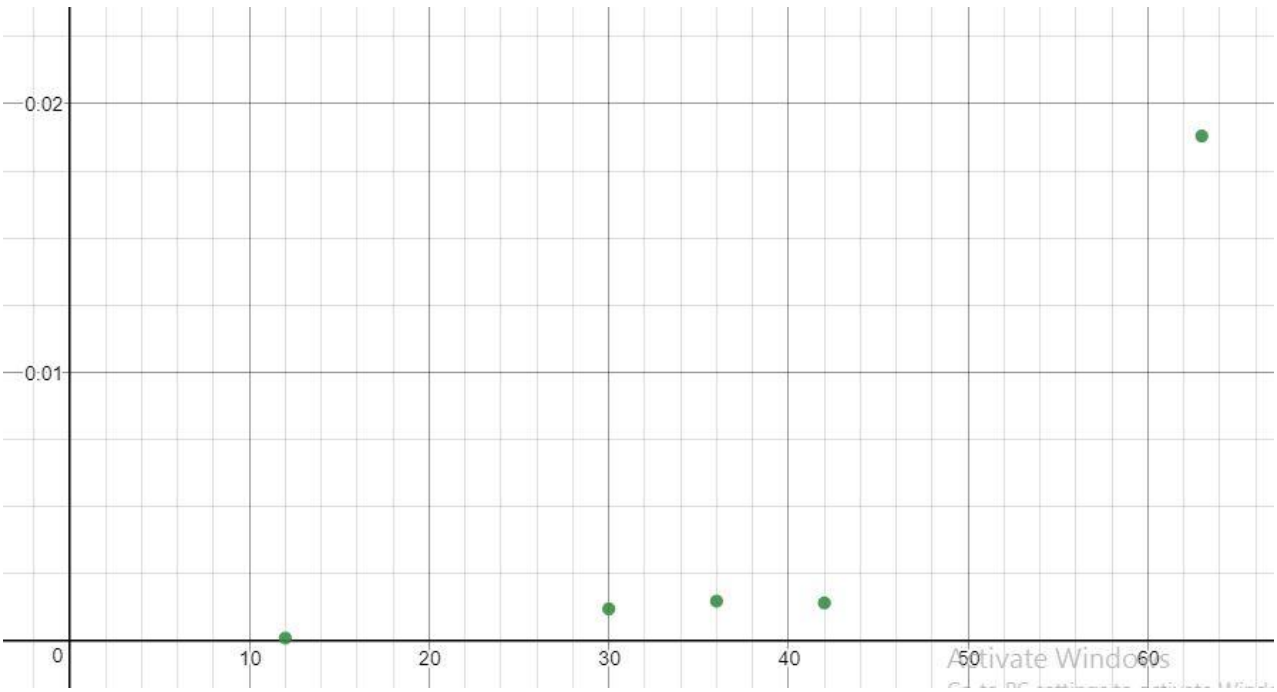
```
abcd
Time taken to calculate LCS:  0.0011789773001
Length of Longest common subsequence: 4
Time taken to calculate the length of LCS:  0.00370016593933
.................
Input first string: abcdaeb
Input second string: aeddaa
Length of first string:  7
Length of second string:  6
Longest common subsequence:
ada
Time taken to calculate LCS:  0.001399800876
Length of Longest common subsequence: 3
Time taken to calculate the length of LCS:  0.002999927520752
.................
Input first string: aabbecabd
Input second string: aaecabd
Length of first string:  9
Length of second string:  7
Longest common subsequence:
aaecabd
Time taken to calculate LCS:  0.01879870023
Length of Longest common subsequence: 7
Time taken to calculate the length of LCS:  0.027698098701
.................

Process finished with exit code 0
```

GRAPH:

For the time taken to calculate LCS:

x_1	y_1
12	0.000089787600098
30	0.0011789773001
36	0.0014678987654
42	0.001399800876
63	0.01879870023

y-axis: Run time of LCS



For time taken to calculate length of LCS:

x_1	y_1
12	0.000016794321
30	0.00370016593933
36	0.003678909991
42	0.002999927520752
63	0.027698098701

y-axis: Time taken to calculate length of LCS