MIHIR
2310700025
BTECH compilers LAB 6

Date
Page

1. longest common subsequence

```
// implementing longest common subsequence to find
// longest subsequence between 2 strings
// input : string P, Q, 2 pointers m and n
// output: length of the longest common subsequence

def LCS ( P, Q, m, n)
  if  m == 0  or  n == 0
  return 0

if memo [m] [n] != -1
  return memo [m] [n]

if P[ m-1] == Q [n-1]
  memo [m] [n] = 1+ LCS ( P, Q, m-1, n-1)
  return memo [m] [n]
else :
  memo [m] [n] = max ( LCS(P, Q, m, n-1), LCS(P, Q, m-1, n))
  return memo [m] [n]
```

2. Time complexity

without memoization the normal recursive approach would have looked at all the subsequence of each character P and Q, however with memoization we avoid recalculating subproblems, instead we calculate it only once

There are m possible lengths of substring of P and n possible length of substring of Q. Therefore no of unique subproblems is m*n, since m*n subproblem and each subproblem being unique is solved in constant time of O(1)

$T(n) = O(m*n)$

1. Matrix chain multiplication

// we find efficient way to multiply n matrices to
// minimise cost
// input: list containing size of matrices
// output: fastest way to compute n matrices multiplication

```
n = P.length - 1
let m[1...n, 1...n] and s[1...n-1, 2...n] be 2 tables
for i=1 to n
    m[i,j] = 0
for l=2 to n                    // l is chain length
    for i=1 to n-l+1
        j = i+l-1               // used this in summation formula
        m[i,j] = ∞
        for K=i to j-1
            q = m[i,K] + m[K+1,j] + P_{i-1} * P_K * P_j
            if q < m[i,j]
                m[i,j] = q
                s[i,j] = K
return m, s
```

2. Time complexity

$$\sum_{i=1}^{n} 1 + \sum_{l=2}^{n} \sum_{i=1}^{n-l+1} \sum_{K=i}^{j-1} 1$$

$$(n-x+x) + \sum_{l=2}^{n} \sum_{i=1}^{n-l+1} (j-x-i+x)$$

$$n + \sum_{l=2}^{n} \sum_{i=1}^{n-l+1} (i+l-1-i) \qquad \{ j = i+l-1 \}$$

$$n + \sum_{l=2}^{n} \sum_{i=1}^{n-l+1} l-1$$

$$n + \sum_{l=2}^{n} (l-1)(n-l+x-x+1)$$

$$n + (n-l+1) \sum_{l=2}^{n} (l-1)$$

$$n + (l-1)(n-l+1) \sum_{l=2}^{n} 1$$

$$(l-1)(n-l+1) = (l-1)(n+1) - l(l-1)$$

$$n + (n+1) \sum_{l=2}^{n} (l-1) + \sum_{l=2}^{n} -l(l-1)$$

$$n + (n+1) \sum_{l=1}^{n-1} l + -\sum_{l=2}^{n} l^2 + \sum_{l=2}^{n} l \qquad \left\{ \sum_{l=2}^{n} l-1 = \sum_{l=1}^{n-1} l \right\}$$

$$n + \frac{(n+1)n(n-1)}{2} - \left( \frac{n(n+1)(2n+1)}{6} - 1 \right) + \left( \frac{n(n+1)}{2} - 1 \right)$$

$$n + \frac{n^3-n}{2} - \frac{2n^3+3n^2+n}{6} + 1 + \frac{n^2+n}{2} - 1$$

$$\approx O(n^3)$$

Date
Page

3. Test cases

positive :

1] 

| ID | grades |
|---|---|
| S1 | FF CD AA ... FF |
| S2 | AB AA AA ... AB |
| S3 | FF CD FF ... CC |
| ⋮ | ⋮ |
| S20 | AA FF CD ... AB |

output = ABB

2]

| ID | grades |
|---|---|
| S1 | BC CC FF ... AA |
| S2 | CD FF AB ... CC |
| S3 | FF AA AB ... DD |
| ⋮ | ⋮ |
| S20 | CCBBAA ... AB |

output = CCBB

3]

| ID | grades |
|---|---|
| S1 | BB BC FF ... FF |
| S2 | AB AA AB ... CD |
| S3 | BB BB AB ... CC |
| ⋮ | ⋮ |
| S20 | CCCCCD ... DD |

output = CCCCCCC

4]

| ID | grades |
|---|---|
| S1 | CCBBCC ... AA |
| S2 | CD AAAA ... BB |
| S3 | AABCBB ... AA |
| ⋮ | ⋮ |
| S20 | BBAABC ... CC |

output = BCCCC

5]

| ID | grades |
|---|---|
| S1 | CCABBC ... CD |
| S2 | BCBCDD ... FF |
| S3 | BB FF AB ... FF |
| ⋮ | ⋮ |
| S20 | BCFFDD ... FF |

output = BBBBB

negative

1] ID grades

S1

S2

⋮

S20

output = since grades are empty we cant find longest common subsequence

2] ID grades

S1 AAZZ.. KK

S2 BCAD... 00

⋮ ⋮

S20 ABCZ... KK

output = incorrect grades have been detected please rectify like KK, 00, CZ

4] ID grades

S1 AAAA... AA

S2 AAAA... AA

⋮ ⋮

S20 AAAA... AA

output = all students have same grades which cant be possible

5] ID grades

S1 AACC... B%

S2 AADD... C%

⋮ ⋮

S20 AACC... C%

output = special characters or invalid character found

3] ID grades

S1 AAAA... A1

S2 AABC... C1

⋮ ⋮

S20 AAAA... B1

output = numeric values are present but grades are in the form of char

Date
Page

3. Test cases

positive :

1] matrices_1 = [4, 7, 4, 7, 4, 7]
output = 400

2] matrices_2 = [3, 7, 3, 7, 3, 7, 3, 7]
output = 306

3] matrices_3 = [5, 7, 5, 7, 5, 7, 5, 7, 5, 7]
output = 1250

4] matrices_4 = [6, 7, 6, 7, 6, 7, 6, 7, 6, 7, 6, 7]
output = 2376

5] matrices_5 = [3, 7, 3, 7]
output = 126

negative

1] matrices_1 = [ ]
output = empty list cant find cost

2] matrices_2 = [5, 7]
output = cost is 0 for single matrix

3] matrices_3 = [5, 7, 6, 8, 7, 9]
output = incompatible matrix dimension for multiplication

4] matrices_4 = [5, 7, -6, 8, 7, 9]
output = matrix dimensions must be positive

5] matrices_s = [5, 7, 7, 5, 5, 7]
output = non square matrices heve incompatible matrix dimensions for multiplication

4. Conlusion:

Hence we have studied the solid principles of programming and also implemented programs to find largest common subsequence between multiple students and also studied and implemented the code to find minimum no of steps for matrix multiplication chain