Name: Vishal Ramane
Roll.No: 2019430010

Experiment No.3

Aim: Implementation of Dynamic programming: matrix chain multiplication and Cutting rod example.

Matrix chain multiplication

Problem Statement:
Given a sequence of matrices, find the most efficient way to multiply these matrices together.

Objective:
- To find the optimal substructure and overlapping sub-problems property for given problem.
- To Implement the Dynamic-programming solution for the problem.

Methodology:
I. A simple solution is to place parenthesis at all possible places, calculate the cost for each placement and return the minimum value.
II. In a chain of matrices of size n, we can place the first set of parenthesis in n-1 ways.So when we place a set of parenthesis, we divide the problem into sub-problems of smaller size.
III. Therefore, the problem has optimal substructure property and can be easily solved using recursion.
IV. Minimum number of multiplication needed to multiply a chain of size n = Minimum of all n-1 placements.
V. q = m[i][k] + m[k+1][j] + p[i-1]*p[k]*p[j], where,
q = cost/scalar multiplications,
m[i,j] = Minimum number of scalar multiplications needed to compute the matrix A[i] A[i+1] … A[j] = A[i..j] , where dimension of A[i] is p[i-1] x p[i].
VI. **Time Complexity**: O(n^3).
VII. **Auxiliary Space**: O(n^2).

Implementation:
- Implemented method *MatrixChainOrder(p,n)* where Matrix A_i has dimension p[i-1] x p[i] for i = 1..n.
- q = m[i][k] + m[k+1][j] + p[i-1]*p[k]*p[j] where q = cost/scalar multiplications.

Name: Vishal Ramane
Roll.No: 2019430010

Experiment No.3

Problem Statement:
Given a rod of length n inches and an array of prices that contains prices of all pieces of size smaller than n. Determine the maximum value obtainable by cutting up the rod and selling the pieces.

Objective:
- To find the optimal substructure and overlapping sub-problems property for given problem.
- To Implement the Dynamic-programming solution for the problem.

Methodology:
I. We can get the maximum value by making a cut to the rod at different positions and comparing the values obtained after a cut.
II. We can recursively call the same function for a piece obtained after a cut.
III. Let cutRod(n) be the required (best possible price) value for a rod of length n. cutRod(n) can be written as following:

cutRod(n) = max(price[i] + cutRod(n-i-1)) for all i in {0, 1 .. n-1}

Implementation:
- We define function *cutRod(price, n)*, where price is an array containing prices of all pieces of size smaller than n. n: length of rod.
- We define *val[ ]* array to store value of sub-problems in bottom up manner.

Results:

Matrix chain multiplication

Input:
```
# Driver program to test above function
arr = [30, 35, 15, 5, 10, 20, 25]
size = len(arr)

print("Minimum number of multiplications is " +
        str(MatrixChainOrder(arr, size)))

```

Name: Vishal Ramane
Roll.No: 2019430010

Experiment No.3

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

(base) C:\Users\Vishal Ramane\OneDrive\College\AAC Lab\Code\Expt3>python
Minimum number of multiplications is 15125
```

Cutting the Rod

Input :

```python
# Driver program to test above functions
arr = [3, 5, 8, 9, 10, 17, 17, 20]
# arr = [1, 5, 8, 9, 10, 17, 17, 20]
size = len(arr)
print("Maximum Obtainable Value is " + str(cutRod(arr, size)))

```

Output:

```
(base) C:\Users\Vishal Ramane\OneDrive\College\AAC Lab\Code\Expt3>python -u "c:\Users\Vishal Ramane\OneDrive\College\AAC Lab\Code\Expt3\DP-MCP_cuttingRod.py"
Maximum Obtainable Value is 24
```

Conclusions: Thus we have successfully implemented dynamic-programming solution for matrix chain multiplication and cutting-rod problem.