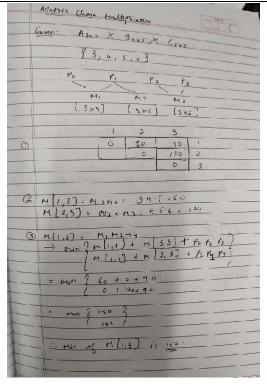


Bharatiya Vidya Bhavan's SARDAR PATEL INSTITUTE OF TECHNOLOGY

(Autonomous Institute Affiliated to University of Mumbai) Munshi Nagar, Andheri (W), Mumbai – 400 058. Department of Master of Computer Application

Experiment	3
Aim	To understand and implement Dynamic Programming Approach
Objective	1) Write Pseudocode for given problems and understanding the
	implementation of Dynamic Programming
	2) Solve Matrix Multiplication Problem using Dynamic
	Programming
	3) Calculating time complexity of the given problems
Name	Vivek Tiwari
UCID	2023510059
Class	FYMCA
Batch	C
Date of	28-02-24
Submission	

Algorithm and	1. Craft a recursive method accepting start and end indices to
Explanation of	delineate a matrix group's bounds.
the technique	2. Within this method, loop through intermediate indices, splitting
used	the given range into two subgroups.
	3. Invoke the recursive method recursively on these two
	subgroups.
	4. Compute the minimal scalar multiplication cost across all
	possible splits by amalgamating the costs of the two subgroups
	and the multiplication cost of the current split. Return this
	minimum value.
	5. The ultimate answer is the minimum value yielded by the
	recursive method when considering the entire range from the
	first to the last matrix.



```
The Complexity

(crisider the Yeurrena relation of Computer.

Misty in is K. e. j. Mist. I will think the state of the stricture of matrices being considered in special true countries of matrices being considered in represent the index at which the considered in represent use of columns

The complexity to complete cach cell of Misty is of in the care of the property of the countries of suppose the strice of the countries of complexity in the Countries of countries of the cach cell of the countries of countries of the countries
```

```
+ p[i - 1] * p[k] * p[j];
    mini = Math.min(count, mini);
}
return mini;
}

public static void main(String[] args) {
    int arr[] = {3, 4, 5, 6};
    int N = arr.length;
    System.out.println("Minimum number of
multiplications is " + matrixChainOrder(arr, 1, N - 1));
}
```

Output

"C:\Program Files\Java\jdk-21\bin\java.exe"
Minimum number of multiplications is 150

Justification of the complexity calculated

The time complexity analysis correctly examines the number of subtasks.

the effort required for each sub-task, and their combined impact. It notes that the algorithm tackles a quadratic number of sub-tasks, as it populates

an n x n matrix, with each cell representing a sub-problem. For every sub-

problem, it iterates through a linear number of possibilities, performing constant-time operations for each iteration. Consequently, the overall effort

to populate the entire matrix is cubic in the number of matrices. Thus, the

algorithm's time complexity when leveraging dynamic programming for matrix chain multiplication is $O(n^3)$, where n denotes the number of matrices.

Conclusion

Dynamic programming is a powerful technique that offers benefits like optimal substructure identification, memoization capabilities, time-efficient

solutions, versatility across domains, and deterministic outcomes. It finds

applications in diverse areas such as recursive function evaluation, path optimization, combinatorial problems, sequence analysis, matrix operations, and currency exchange optimization. By methodically decomposing problems into overlapping subproblems and reusing solved

subproblems, dynamic programming effectively tackles complex optimization challenges with enhanced efficiency.