****

**Name:** Husnain Momin

**SAP:** 38589

**Semester:** 5-A

**Course:** Design and Analysis of Algorithms

**Dynamic Programming Comparison Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Aspect | *Dynamic Programming* | Brute Force | Greedy | Divide & Conquer |
| Time complexity | Improves brute force using memorization resulting in a better time complexity. | Is high especially in large input sizes. | Generally low as it makes locally optimal choices. | May be efficient if sub-problems are non-overlapping. |
| Space Complexity | Higher complexity | Usually lower compared to dynamic. | Low space complexity | May require additional memory |
| Suitability for Different Problems | Well-suited for problems with optimal substructure | Applicable to a wide range of problems but may be impractical for large input sizes. | Suitable where locally optimal choices lead to a global optimum. | Effective for problems that can be broken down into smaller parts. |
| Optimality | Provides globally optimal solutions by combining optimal solutions to sub-problems. | May or may not result in globally optimal solutions; depends on the problem. | Often does not guarantee globally optimal solutions due to its myopic decision-making. | Can achieve optimal solutions if sub-problems are solved optimally. |
| Examples | Knapsack problem | Traveling salesman problem | Huffman Coding | Merge sort |
| Advantages | Provides optimized solutions by avoiding redundant computations. | Simple and easy to implement | Efficient and often easy to implement; can be greedy-choice property exploitable | Efficient for problems with a clear divide-and-combine structure. |
| Disadvantages | May have higher space complexity; not always applicable to all problems. | Can be inefficient for large input | May not always yield globally optimal solutions | Can have higher time complexity for problems with significant overlap. |

**Chain-Matrix Multiplication**

It is an optimization problem that gives us the most efficient way to multiple a sequence of matrices. The problem itself is not to find the product of the matrices, but to decide the sequence of matrix multiplications involved.

**Real World Applications**

Matrix chain multiplication is used in many real-world applications, including computer graphics, robotics, and optimization problems.

Let’s talk about **3D Computer Graphics** where you have a series of transformations to be applied to a set of 3D points. Consider we have 3 matrices T1, T2 and T3 that represent the transformation of the 3D object. The order in which you apply these transformations matters, and the final transformation matrix would be the product of these matrices.

To transform a 3D point, you would typically apply these transformations in a specific order. The order of multiplication directly influences the number of multiplications needed to compute the final transformation.