## Introduction:

The report is written based on the analysis of three different algorithms. They are recursive top-down approach, dynamic programming extended-bottom-up approach and memorized approach. The analysis includes information about source of reference, time complexities of each algorithm, experimental results and comparison between output generated between the algorithms.

## Site/Source of Reference:

During the implementation and analysis of the sorting algorithms, the following sources were referred to:

* [GeeksforGeeks](https://www.geeksforgeeks.org/" \t "_new)
* [Rod Cutting Problem | Dynamic Programming (with code) (favtutor.com)](https://favtutor.com/blogs/rod-cutting-problem)
* [Course Modules: 2242-CSE-5311-002-DSGN & ANLY ALGORITHMS (instructure.com)](https://uta.instructure.com/courses/181469/modules)

Time Complexity:

* Recursive top-down approach:
* Best Case: O(1)
* Average Case: O(2^n)
* Worst Case: O(2^n)
* Dynamic programming extended-bottom-up approach:
* Best Case: O(1)
* Average Case: O(n)
* Worst Case: O(n)
* Memorized Approach:
* Best Case: O(1)
* Average Case: O(n)
* Worst Case: O(n)

Experimental Result:

* The experimental results for each algorithm using the provided price files show consistent performance across different rod lengths. The output matches the expected maximum revenue and optimal cuts for each rod length.
* However, the execution times vary between the algorithms, with the memoized and bottom-up approaches generally outperforming the recursive top-down approach.
* We have rods of different lengths like [5,10,20,30]. Three algorithms are executed for each rod-cutting problem of length-n. The experimental results are listed below:
* **For the rod length N=5**
* Recursive top-down approach:

Maximum revenue: 37

Optimal cuts: [2, 3]

Execution time: 0.0015826225280761719

* Memoized Approach:

Maximum revenue: 37

Optimal cuts: [2, 3]

Execution time: 0.00014328956604003906

* Dynamic Programming Extended-Bottom-Up approach:

Maximum revenue: 37

Optimal cuts: [2, 3]

Execution time: 0.0001327991485595703

* **For the rod length N=10**
* Recursive top-down approach:

Maximum revenue: 75

Optimal cuts: [1, 3, 3, 3]

Execution time: 0.00025844573974609375

* Memoized Approach:

Maximum revenue: 75

Optimal cuts: [1, 3, 3, 3]

Execution time: 0.00017213821411132812

* Dynamic Programming Extended-Bottom-Up approach:

Maximum revenue: 75

Optimal cuts: [1, 3, 3, 3]

Execution time: 0.0001289844512939453

* **For the rod length N=20**
* Recursive top-down approach:

Maximum revenue: 152

Optimal cuts: [2, 3, 3, 3, 3, 3, 3]

Execution time: 0.0004317760467529297

* Memoized Approach:

Maximum revenue: 152

Optimal cuts: [2, 3, 3, 3, 3, 3, 3]

Execution time: 0.0002753734588623047

* Dynamic Programming Extended-Bottom-Up approach:

Maximum revenue: 152

Optimal cuts: [2, 3, 3, 3, 3, 3, 3]

Execution time: 0.00023865699768066406

* **For the rod length N=30**
* Recursive top-down approach:

Maximum revenue: 230

Optimal cuts: [3, 3, 3, 3, 3, 3, 3, 3, 3, 3]

Execution time: 0.0004992485046386719

* Memoized Approach:

Maximum revenue: 230

Optimal cuts: [3, 3, 3, 3, 3, 3, 3, 3, 3, 3]

Execution time: 0.0003910064697265625

* Dynamic Programming Extended-Bottom-Up approach:

Maximum revenue: 230

Optimal cuts: [3, 3, 3, 3, 3, 3, 3, 3, 3, 3]

Execution time: 0.0002467632293701172

Difference Between Experimental And Theoretical Results:

As per theoretical understanding, the average case and worst case for the recursive algorithm is O(2^n). It takes more time to find the maximum revenue and optimal cuts. Whereas the memoize approach and dynamic approach both are producing efficient results for rod-cutting problem of different length. Theoretical time complexities provide an estimation of algorithmic efficiency, but real-world performance can be influenced by factors such as implementation details and system resources.

In our case, experimental results align with theoretical explanation. From the above results dynamic programming extends bottom-up approach producing the efficient results when comparing to other algorithms. In practical dynamic approach is showing efficient results for all the rod length cutting problems. Even though we have same time complexity for the memoized and dynamic approach, we always see the best results through dynamic approach. For rod-cutting problem of length 30. The optimal cuts and maximum revenue generated by all the algorithms are the same, but the execution varies. The recurse approach is taking maximum execution time ‘0.0004992485046386719 sec’.

Comparison Between Three Algorithms:

In general, these three algorithms use different methods for finding the maximum revenue so that they vary in time complexity, space complexity and efficiency for different types of rod length. Here we have generated the random prices files for the rods of length n. The memoized and dynamic programming extended bottom-up approaches consistently demonstrate faster execution times compared to the recursive top-down approach across all tested rod lengths.

1. **Recursive Top-Down Approach:**

* Utilizes recursive calls to break down the problem into smaller subproblems.
* Involves solving overlapping subproblems repeatedly, leading to potential inefficiencies.
* Time complexity grows exponentially with the size of the input due to redundant computations.

1. **Memoized Approach:**

* Enhances the recursive top-down approach by memorizing solutions to subproblems to avoid redundant computations.
* Utilizes a memoization table to store previously computed results, enabling constant-time retrieval.
* Improves time complexity to O(n) by ensuring each subproblem is solved only once.

1. **Dynamic Programming Extended Bottom-Up Approach:**

* Constructs the solution iteratively from the bottom up, starting from smaller subproblems and building up to the desired solution.
* Avoids recursion entirely and directly computes solutions to all subproblems in a systematic manner.
* Utilizes tabulation to store intermediate results efficiently, eliminating the need for recursive function calls.
* Achieves linear time complexity of O(n), making it highly efficient for solving large instances of the problem.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rod-Length | Time taken for  Recursive  approach | time taken for  memoized  Approach | time taken for  Dynamic  Approach | efficient  algorithm |
| 5 | 0.0015826225280761719 | 0.00014328956604003906 | 0.0001327991485595703 | Dynamic  approach |
| 10 | 0.00025844573974609375 | 0.00017213821411132812 | 0.0001289844512939453 | Dynamic  approach |
| 20 | 0.0004317760467529297 | 0.0002753734588623047 | 0.00023865699768066406 | Dynamic  approach |
| 30 | 0.0004992485046386719 | 0.0003910064697265625 | 0.0002467632293701172 | Dynamic  approach |

* The output of the memoized approach is listed in the above table. Here we have specified the rod length and time taken to execute the algorithm.

N=5, Timetaken=0.00014328956604003906 N=10,timetaken=0.00017213821411132812

N=20, Timetaken=0.0002753734588623047

N=30, Timetaken=0.0004992485046386719

Line Graph:

This graph visually illustrates the comparative performance of the three algorithms

across rod lengths of 5, 10, 20, and 30.

A graph with lines and numbers

Description automatically generated

As evident from the graph, the recursive top-down approach exhibits an increase in execution time with increasing rod length, while the memoized and dynamic programming extended bottom-up approaches maintain relatively stable and efficient performance. The anomaly at n=20 for the dynamic programming extended bottom-up approach is noticeable, indicating a potential area for improvement in the algorithm's implementation.

From the above graph we can see that with increase in the rod length there is only a few fraction of second difference in the execution of the algorithms. Among those dynamic programming executes in less time when compared to other.