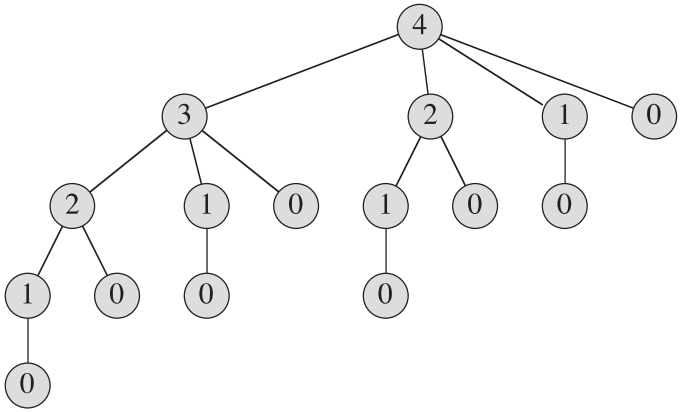
Dynamic Programming Summary

Module 4

Dr Bijoy A Jose

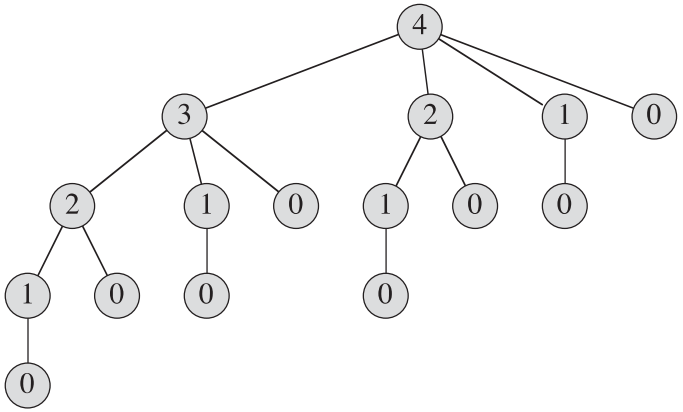
Dynamic Programming

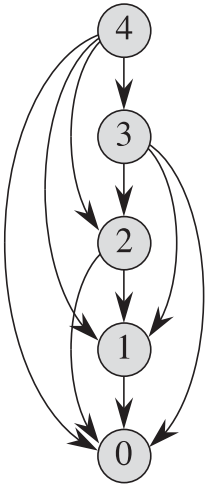


- **Rod cutting problem** – recursive solution is inefficient because it solves the same subproblems repeatedly,
- **Dynamic Programming**: each subproblem to be solved only *once*
- Dynamic programming thus uses additional memory to save computation time (it is a ***time-memory trade-off****)*
- *Rod cutting problem exhibits* ***optimal substructure***: optimal solutions to a problem incorporate optimal solutions to related subproblems, which we may solve independently.

Memoization

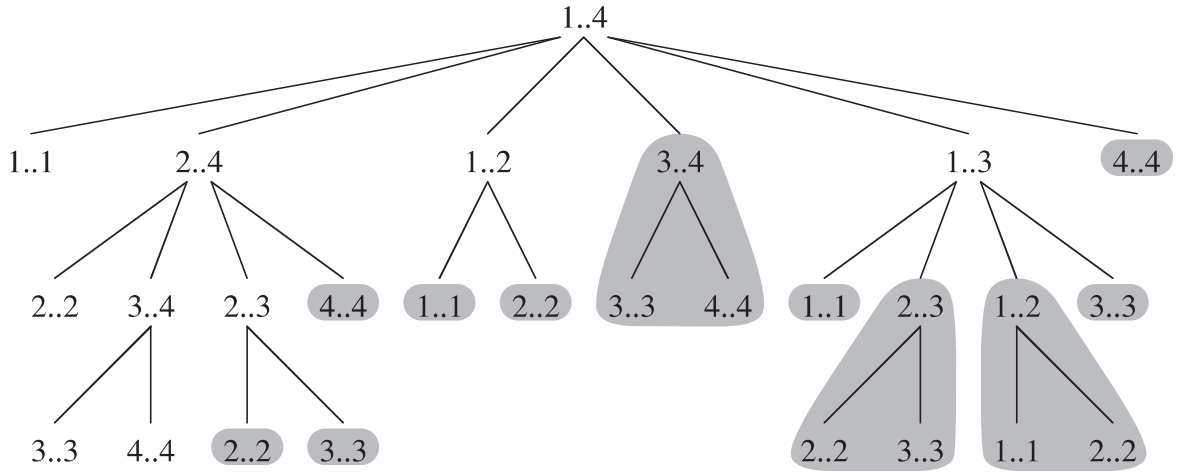
- Memoization: procedure recursively in a natural manner, but modified to save the result of each subproblem
  - Top-down or bottom-up
  - Hash table or arrays are used
- We maintain a table with subproblem solutions, but the control structure for filling in the table is more like the recursive algorithm.





Matrix multiplication optimization

- Memoized matrix multiplication
- Read topic in Algorithms - Cormen text book chapter 15



Matrix-chain multiplication

- Can we reduce the number of multiplications
- See notes uploaded in Moodle
- Read topic in Algorithms - Cormen text book chapter 15

Elements of Dynamic Programming

- Does the problem exhibit optimal substructure?
- Recursion exists: The space of subproblems must be “small” such that a recursive algorithm for the problem solves the same subproblems over and over, rather than always generating new subproblems.
  - When a recursive algorithm revisits the same problem repeatedly, we say that the optimization problem has ***overlapping subproblems***
  - In contrast, a problem for which a divide-and- conquer approach is suitable usually generates brand-new problems at each step of the recursion

Applying dynamic programming

1. Characterize the structure of an optimal solution.
2. Recursively define the value of an optimal solution.
3. Compute the value of an optimal solution.
4. Construct an optimal solution from computed information.