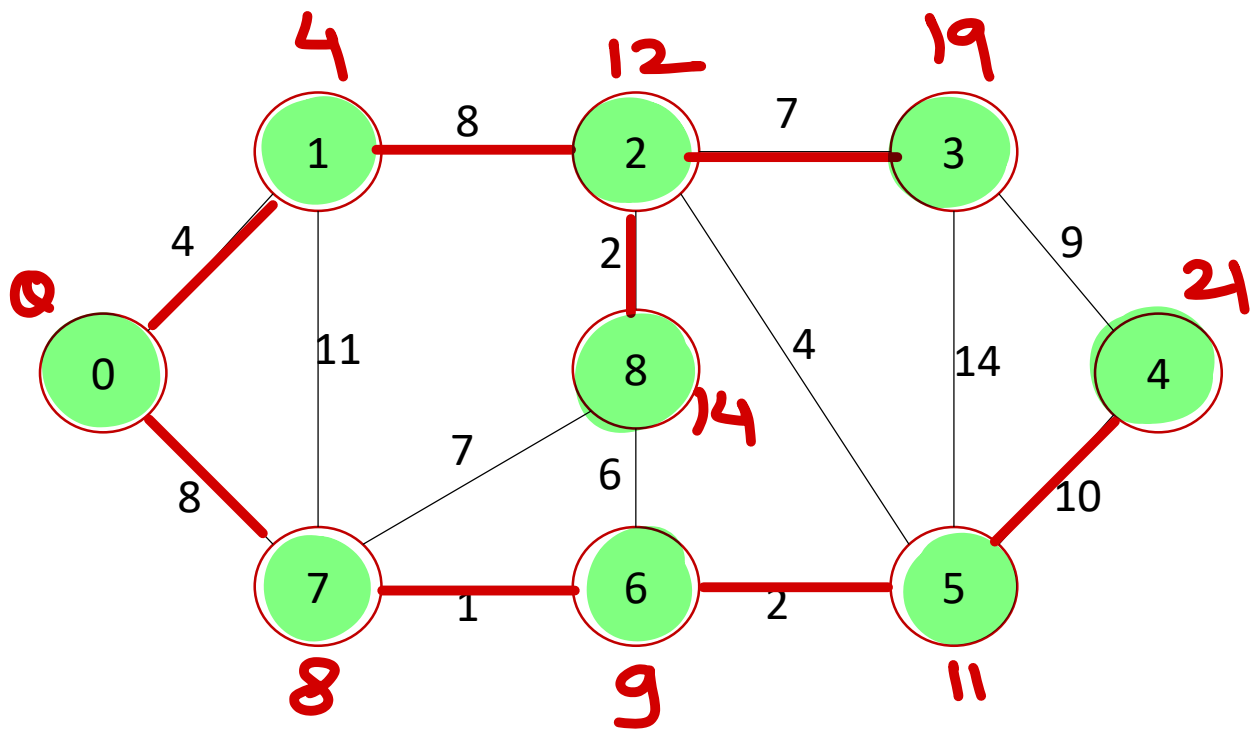
Data Structure & Algorithms

Nilesh Ghule

Dijkstra's Algorithm

1. Create a set *spt* to keep track of vertices included in shortest path tree.
2. Track distance of all vertices in the input graph. Distance for all vertices should be initialized to INF. The start vertex distance should be 0.
3. While *spt* doesn't include all vertices
   i. Pick a vertex u which is not there in *spt* and has minimum distance.
   ii. Include vertex u to *spt*.
   iii. Update distances of all adjacent vertices of u. For each adjacent vertex v, if distance of u + weight of edge u-v is less than the current distance of v, then update its distance as distance of u + weight of edge u-v.



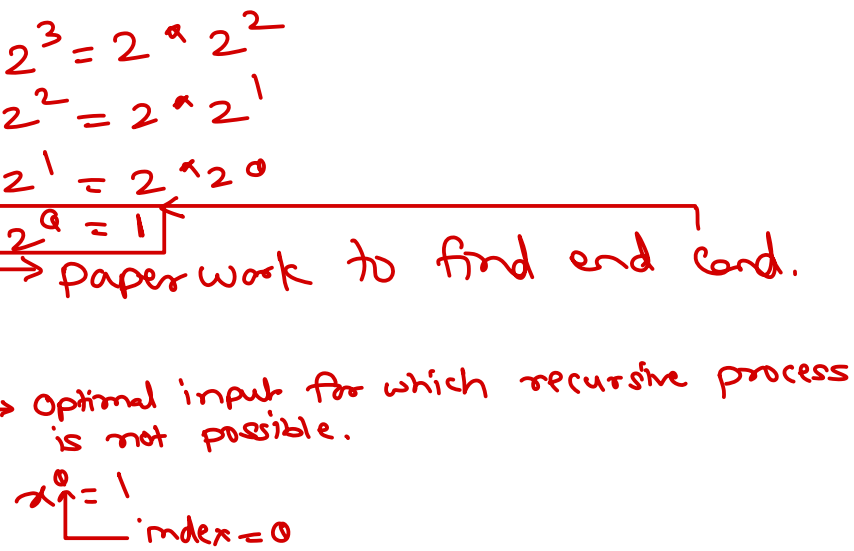
Dijkstra's SPT – Analysis

1. Create a set *spt* to keep track of vertices included in shortest path tree.
2. Track distance of all vertices in the input graph. Distance for all vertices should be initialized to INF. The start vertex distance should be 0.
3. While *spt* doesn't include all vertices
   i. Pick a vertex u which is not there in *spt* and has minimum distance.
   ii. Include vertex u to *spt*.
   iii. Update distances of all adjacent vertices of u. For each adjacent vertex v, if distance of u + weight of edge u-v is less than the current distance of v, then update its distance as distance of u + weight of edge u-v.

- Time complexity (adjacency matrix)
  - V vertices: O(V)
  - get min key vertex: O(V)
  - update adjacent: O(V)
- Time complexity (adjacency matrix)
  - $O(V^2)$
- Time complexity (adjacency list)
  - V vertices: O(V)
  - get min key vertex: O(log V)
  - update adjacent: O(E) – E edges
- Time complexity (adjacency list)
  - O(E log V)

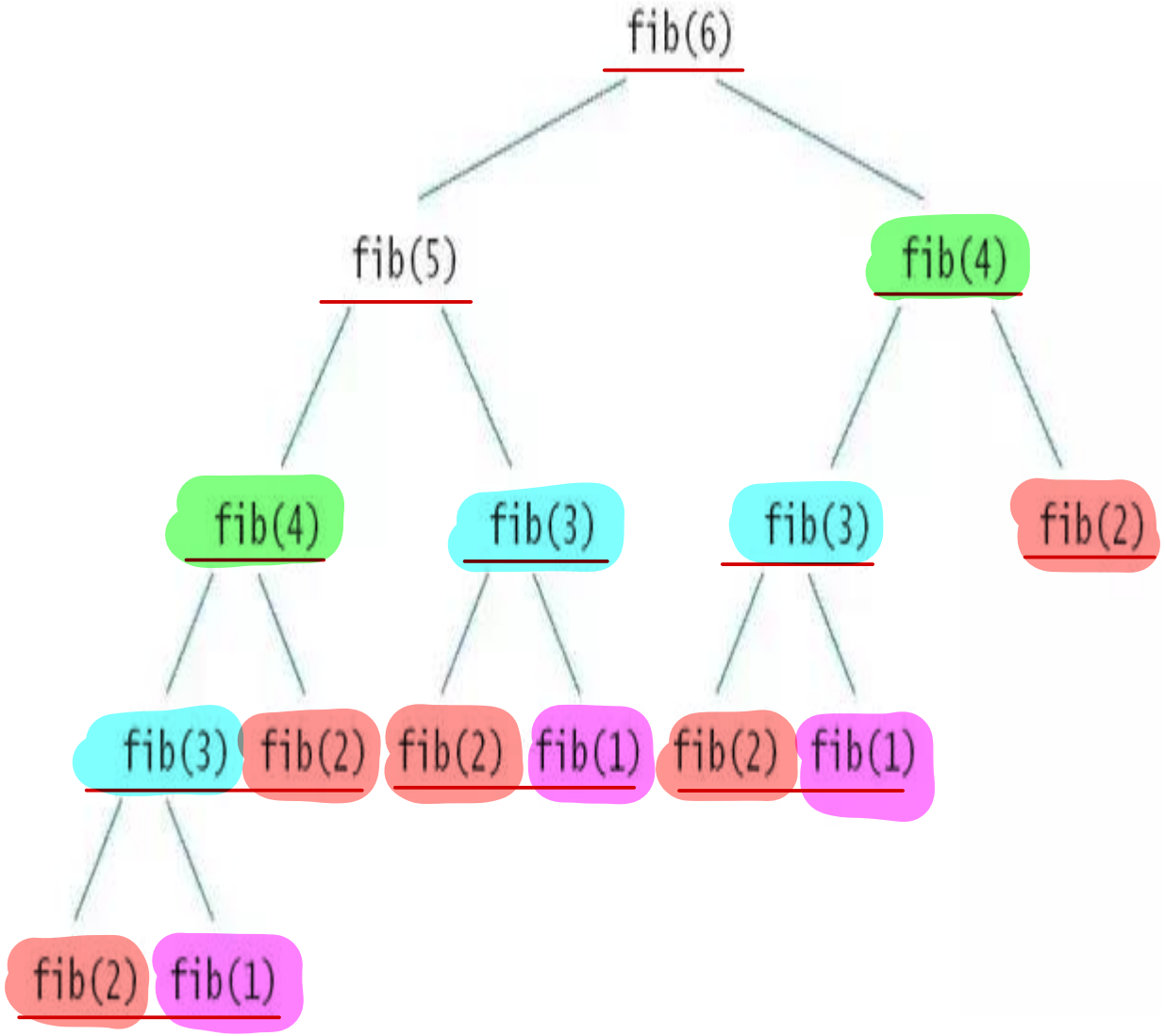
Recursion

- Function calling itself is called as recursive function.
- For each function call stack frame is created on the stack. (LIFO)
- Thus it needs more space as well as more time for execution.
- However recursive functions are easy to program.
- Typical divide and conquer problems are solved using recursion.
- For recursive functions two things are must
  - Recursive call (Explain process it terms of itself)
  - Terminating or base condition (Where to stop)



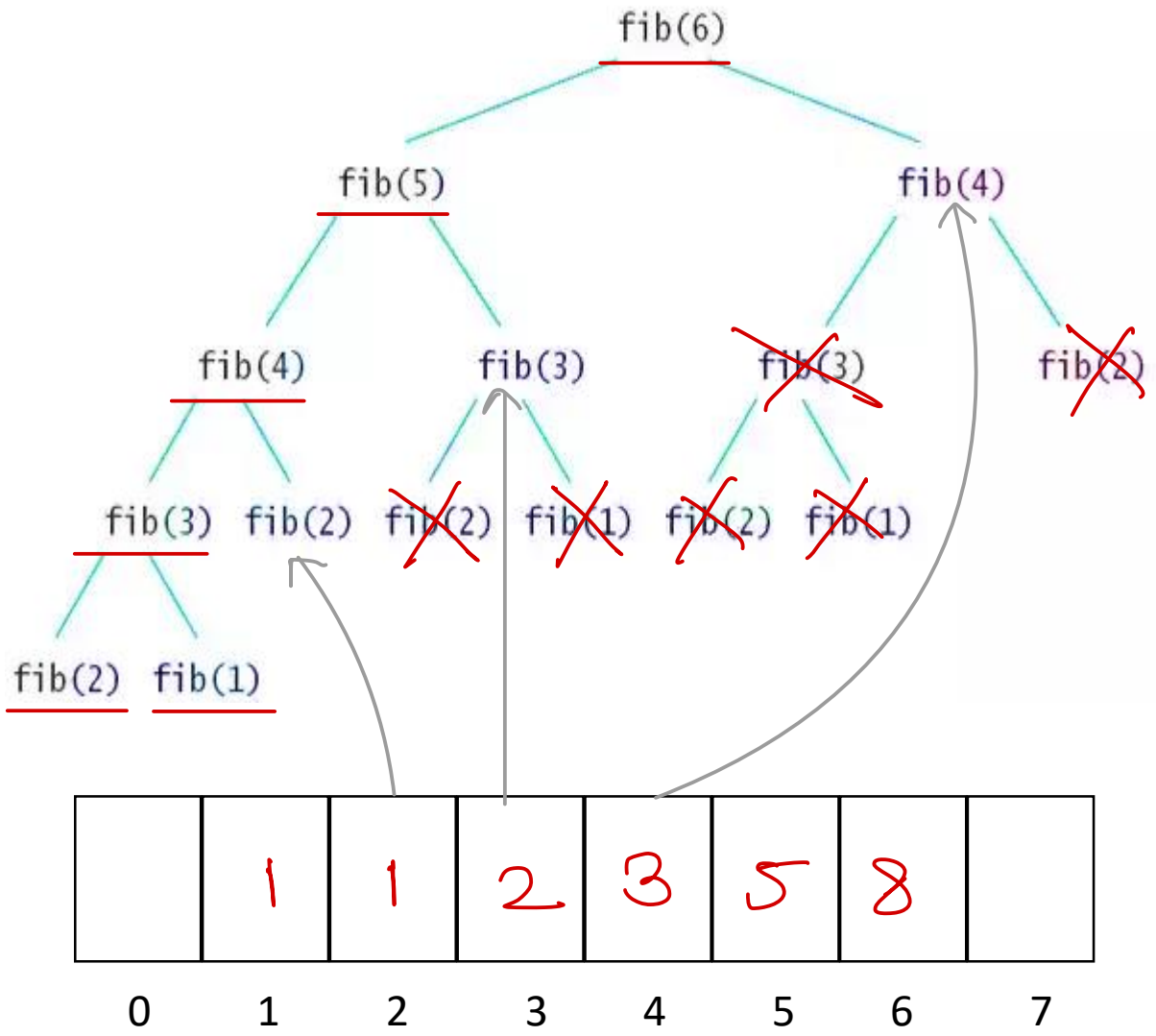
Recursion – Fibonacci Series

- Recursive formula
  - $T_n = T_{n-1} + T_{n-2}$
- Terminating condition
  - $T_1 = T_2 = 1$
- Overlapping sub-problem



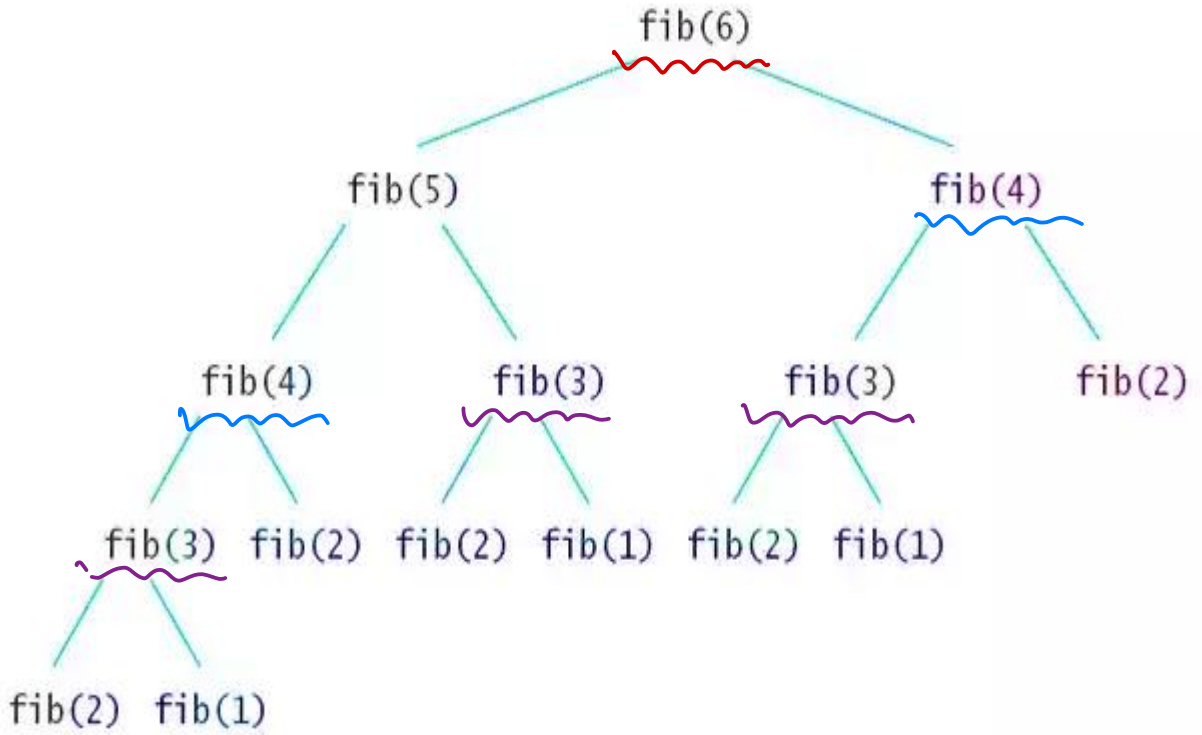
Memoization – Fibonacci Series

- It's based on the Latin word memorandum, meaning "to be remembered".
- Memoization is a technique used in computing to speed up programs.
- This is accomplished by memorizing the calculation results of processed input such as the results of function calls.
- If the same input or a function call with the same parameters is used, the previously stored results can be used again and unnecessary calculation are avoided.
- Need to rewrite recursive algorithm. Using simple arrays or map/dictionary.



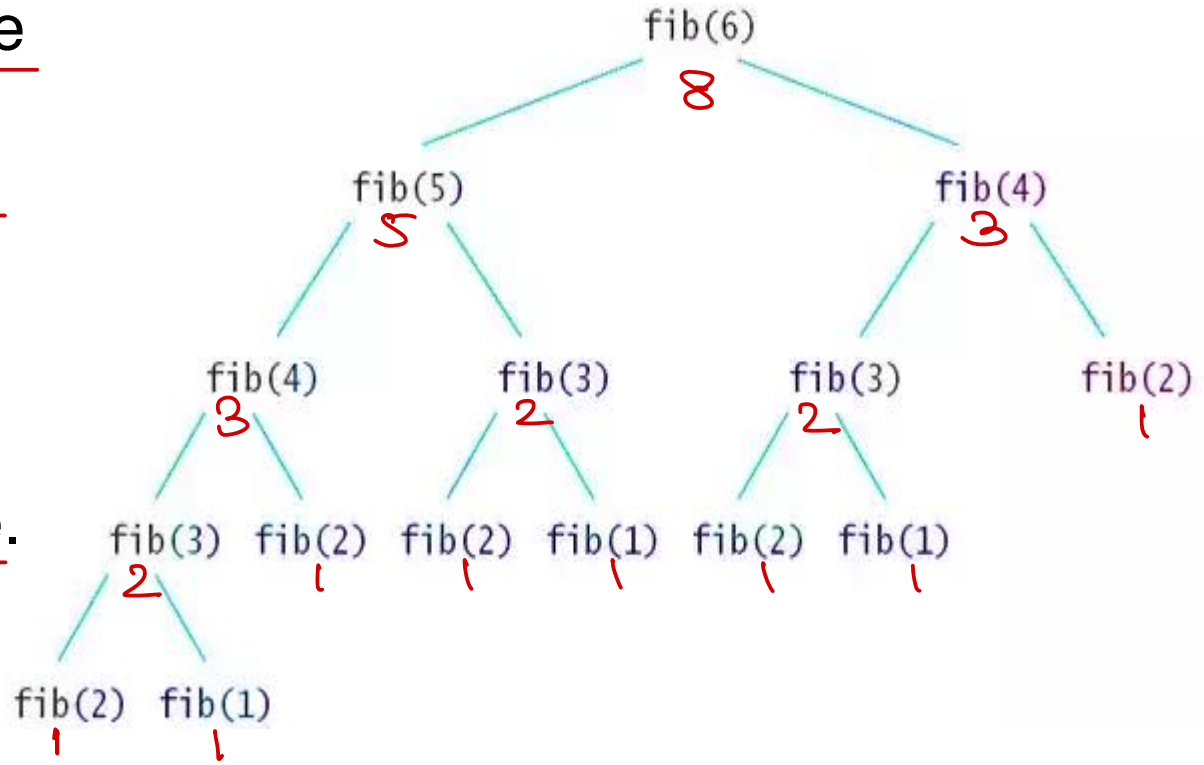
Dynamic Programming

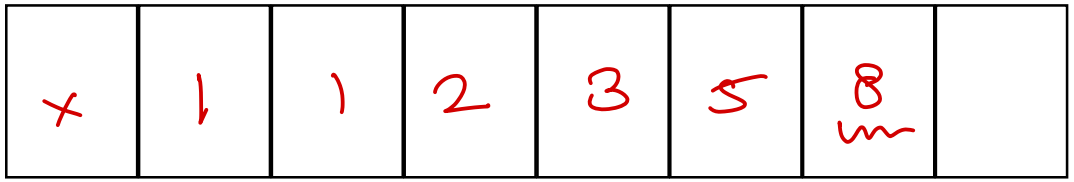
- Dynamic programming is another optimization over recursion.
- Typical DP problem give choices (to select from) and ask for optimal result (maximum or minimum).
- Technically it can be used for the problems having two properties
  - Overlapping sub-problems
  - Optimal sub-structure
- To solve problem, we need to solve its sub-problems multiple times.
- Optimal solution of problem can be obtained using optimal solutions of its sub-problems.



Dynamic Programming – Fibonacci Series

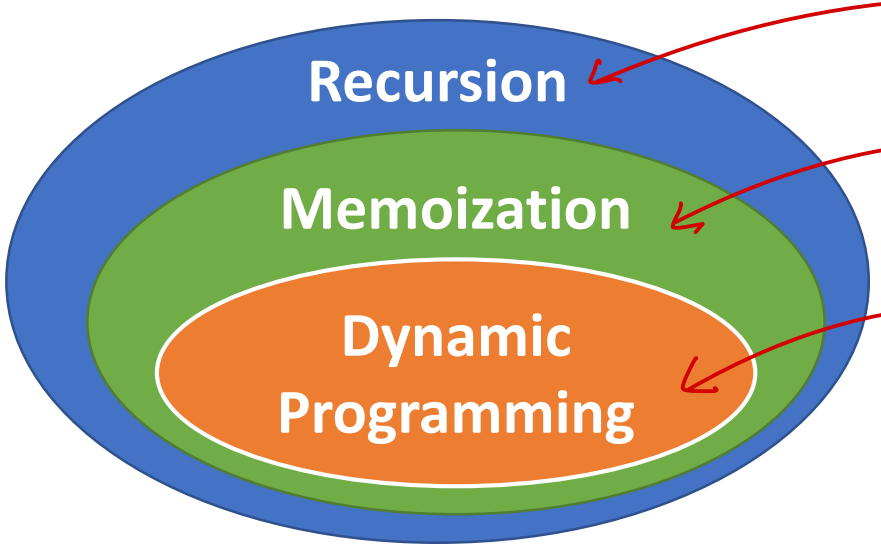
- Alternative solution to DP is memoizing the recursive calls. This solution needs more stack space, but similar in time complexity.
- Memoization is also referred as top-down approach.
- DP solution is bottom-up approach.
- DP use 1-d array or 2-d array to save state.
- Greedy algorithms pick optimal solution to local problem and never reconsider the choice done.
- DP algorithms solve the sub-problem in a iteration and improves upon it in subsequent iterations.





base/end condition of recursive solution

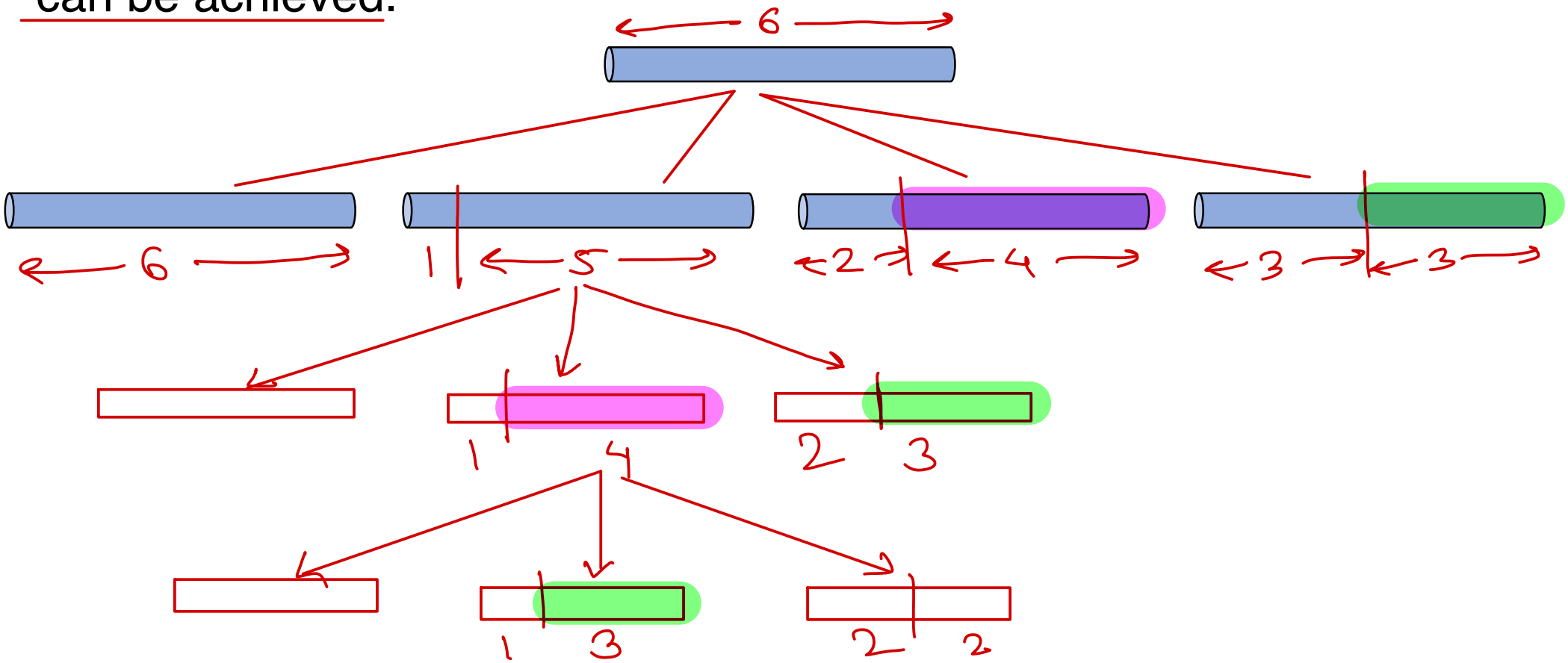
Dynamic Programming



- Top down
- Optimized Top down
- Bottom up

Dynamic Programming

- Rod cutting problem: Cut the rod of given ~~price~~ length so that maximum price can be achieved.



Length	Price
1	1
2	5
3	8
4	9
5	10
6	14
7	17
8	20
9	24
10	30

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>