Minimum Camera Placement for Forest Monitoring – DP Algorithm

1. Recursive Formulation (Task 1)

For a forest graph $G = (V, E)$, we find the minimum cameras to monitor all vertices. For trees, we use DP with three states per node v:

- $dp[v][0]$: Camera at v (cost 1)
- $dp[v][1]$: No camera at v, dominated by child (cost computed)
- $dp[v][2]$: No camera at v, waiting for parent (cost 0 for v)

Base case (leaf): $dp[v][0] = 1$, $dp[v][1] = \infty$, $dp[v][2] = 0$
Recurrence (internal node v with children $C(v)$):

$$dp[v][0] = 1 + \sum_{c \in C(v)} \min(dp[c][0], dp[c][1], dp[c][2])$$

$$dp[v][2] = \sum_{c \in C(v)} \min(dp[c][0], dp[c][1])$$

$$base = \sum_{c \in C(v)} \min(dp[c][0], dp[c][1])$$

$$dp[v][1] = \begin{cases} \infty, & C(v) = \emptyset \\ base + \min_{c \in C(v)} \big(dp[c][0] - \min(dp[c][0], dp[c][1])\big), & \text{otherwise} \end{cases}$$

Answer: $\min(dp[r][0], dp[r][1])$ for root r.

2. Pseudocode (Task 1)

Algorithm 1 MinCamerasOnTree(G, r)

function SOLVE(v, $parent$)
  $dp[v][0] \leftarrow 1$; $dp[v][1] \leftarrow \infty$; $dp[v][2] \leftarrow 0$
  for child c of v where $c \neq parent$ **do**
    SOLVE(c, v)
  end for
  $base \leftarrow 0$, $gain \leftarrow \infty$
  for child c of v where $c \neq parent$ **do**
    $m02 \leftarrow \min(dp[c][0], dp[c][1], dp[c][2])$
    $m01 \leftarrow \min(dp[c][0], dp[c][1])$
    $dp[v][0] \leftarrow dp[v][0] + m02$; $dp[v][2] \leftarrow dp[v][2] + m01$
    $base \leftarrow base + m01$; $gain \leftarrow \min(gain, dp[c][0] - m01)$
  end for
  if $gain < \infty$ **then**
    $dp[v][1] \leftarrow base + gain$
  end if
end function
SOLVE($r, -1$)
return $\min(dp[r][0], dp[r][1])$

3. Asymptotic Time Complexity (Task 2)

Let $n = |V|$ and $m = |E|$. The algorithm performs a single DFS traversal visiting each edge at most twice. Per node, we do $O(1)$ work for DP state aggregation. **Time complexity:** $O(n + m)$. For trees, $m = n - 1$, so $O(n)$. **Space complexity:** $O(n)$ for DP tables and recursion stack.

4. Example (Task 3)

Tree with 5 nodes: edges $\{(0, 1), (1, 2), (1, 3), (3, 4)\}$, root at 1.

Post-order traversal:

- Leaf nodes: $dp[0] = [1, \infty, 0]$, $dp[2] = [1, \infty, 0]$, $dp[4] = [1, \infty, 0]$
- Node 3 (child 4): $dp[3][0] = 1 + \min(1, \infty, 0) = 1$; $dp[3][2] = \min(1, \infty) = 1$; $dp[3][1] = 1 + (1 - 1) = 1 \Rightarrow dp[3] = [1, 1, 1]$
- Node 1 (children 0,2,3): $m02 = m01 = 1$ for all children. Computing: $dp[1][0] = 1 + (1 + 1 + 1) = 4$, $dp[1][2] = 1 + 1 + 1 = 3$, $dp[1][1] = 3 + \min(0, 0, 0) = 3$
- Answer: $\min(4, 3) = 3$ cameras (e.g., at nodes $\{0, 3, 1\}$)

5. Functional Testing (Task 5)

We designed 7 test instances covering base cases, edge cases, and various tree structures. All tests passed. Results:

Table 1: Functional Testing Results

Instance	Expected	Actual	Status
Single node	1	1	✓
Two nodes	1	1	✓
Path (3 nodes)	1	1	✓
Star graph	1	1	✓
Binary tree	2	2	✓
Forest (2 components)	2	2	✓
Complex tree	2	2	✓

6. Computational Performance (Task 6)

Benchmark instances: 1,406 instances across 20 input sizes (10 to 10,000 nodes), with 10-64 instances per size. Structures include paths, stars, binary trees, random trees, and forests.

Table 2: Performance Results (Sample)

Input Size	Instances	Avg CPU Time (s)
10	50	0.000011
100	64	0.000241
1000	24	0.002704
5000	10	0.010035
8500	10	0.016378

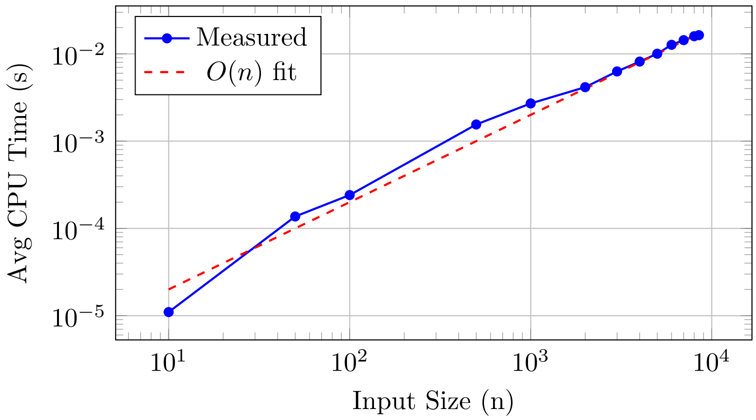

Figure 1: CPU Time vs Input Size (Log-Log Scale)

Discussion: The plot shows linear scaling ($O(n)$), confirmed by the log-log slope ≈ 1. For $n = 1000$: ~ 0.0027s; $n = 5000$: ~ 0.010s (5x input $\rightarrow$ 5x time); $n = 8500$: ~ 0.016s (8.5x input $\rightarrow$ 8.5x time). Results validate the theoretical $O(n)$ complexity.