

Minimum Camera Placement for Forest Monitoring – DP Design

Problem

We are given a forest graph $G = (V, E)$. Each vertex represents a candidate designated point (cdp) where we can place a camera, and each edge represents a shared region that can be monitored by cameras at two different cdps. The goal is to find the minimum number of cameras needed to monitor all vertices (regions).

This is the *minimum dominating set* problem. While it is NP-hard for general graphs, it can be solved in polynomial time using dynamic programming (DP) for forest structures (trees). The following DP finds the optimum by rooting each tree.

DP Design

1) Subproblems

For each node v , we maintain three states (tree rooted at r):

- $dp[v][0]$: Camera is placed at v .
- $dp[v][1]$: No camera at v , but at least one child has a camera and v is dominated.
- $dp[v][2]$: No camera at v , v is not yet dominated; domination must come from parent.

2) Recursive Formulation

For a leaf node:

$$dp[v][0] = 1, \quad dp[v][1] = \infty, \quad dp[v][2] = 0$$

For an internal node v with children set $C(v)$:

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

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

In state 1, v must be dominated by at least one child camera:

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

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

This is because at least one child must actually have a camera (state 0).

Valid answer for root: $\min(dp[r][0], dp[r][1])$ (root must be dominated).

3) Justification for DP

- Optimal substructure: The optimal solution for each subtree (child) is independent of others; combinations only interact through state labels.
- Number of subproblems: Constant number of states per node, total $O(|V|)$ subproblems.
- Overlapping subproblems: Each node's states cannot be requested by multiple parents (tree structure), so memoization with DP is efficient.

4) Pseudocode

Time Complexity (Task 2)

Let $n = |V|$ and $m = |E|$. Each edge is visited at most twice during the DFS, and per node we do $O(1)$ work for constant-state DP aggregation. Thus the time complexity is $O(n + m)$ in the worst case (for trees, $m = n - 1$, so $O(n)$). The space complexity is $O(n)$ for the DP tables and recursion stack.

Step-by-Step Example (Task 3)

Consider a sample tree with 5 cdps (nodes) where each cdp monitors 2–3 regions: edges $\{(0, 1), (1, 2), (1, 3), (3, 4)\}$ and root at 1.

- Post-order traversal: process children before parent.
- Node 0 (leaf): $dp[0] = [1, \infty, 0]$.
- Node 2 (leaf): $dp[2] = [1, \infty, 0]$.
- Node 4 (leaf): $dp[4] = [1, \infty, 0]$.
- Node 3 (child 4):
 - $dp[3][0] = 1 + \min(1, \infty, 0) = 1$

Algorithm 1 MinCamerasOnTree(G, r)

```

1: function SOLVE( $v, parent$ )
2:    $dp[v][0] \leftarrow 1$                                  $\triangleright$  Cost if camera placed at  $v$ 
3:    $dp[v][1] \leftarrow \infty$                              $\triangleright$  Initially impossible
4:    $dp[v][2] \leftarrow 0$                                  $\triangleright$  Cost if  $v$  waits for parent
5:   for child  $c$  of  $v$  where  $c \neq parent$  do
6:     SOLVE( $c, v$ )                                      $\triangleright$  Process children first
7:   end for
8:    $base \leftarrow 0, gain \leftarrow \infty$ 
9:   for child  $c$  of  $v$  where  $c \neq parent$  do
10:     $m02 \leftarrow \min(dp[c][0], dp[c][1], dp[c][2])$        $\triangleright$  Best for state 0
11:     $m01 \leftarrow \min(dp[c][0], dp[c][1])$                    $\triangleright$  Best for states 1 and 2
12:     $dp[v][0] \leftarrow dp[v][0] + m02$ 
13:     $dp[v][2] \leftarrow dp[v][2] + m01$ 
14:     $base \leftarrow base + m01$ 
15:     $gain \leftarrow \min(gain, dp[c][0] - m01)$            $\triangleright$  Extra cost to force state 0
16:  end for
17:  if  $gain < \infty$  then
18:     $dp[v][1] \leftarrow base + gain$                        $\triangleright$  At least one child has camera
19:  end if
20: end function
21: SOLVE( $r, -1$ )
22: return  $\min(dp[r][0], dp[r][1])$                        $\triangleright$  Root must be dominated

```

- $dp[3][2] = \min(1, \infty) = 1$
- $dp[3][1] = base + gain = 1 + (1 - 1) = 1$ (force child 4 to have camera)

So $dp[3] = [1, 1, 1]$.

- Node 1 (children 0,2,3):

- For child 0: $m02 = 1, m01 = 1$.
- For child 2: $m02 = 1, m01 = 1$.
- For child 3: $m02 = 1, m01 = 1$.

Aggregate:

$$\begin{aligned}
dp[1][0] &= 1 + (1 + 1 + 1) = 4 \\
dp[1][2] &= 1 + 1 + 1 = 3 \\
base &= 3, \quad gain = \min(0, 0, 0) = 0 \\
dp[1][1] &= 3 + 0 = 3
\end{aligned}$$

- Answer at root 1: $\min(dp[1][0], dp[1][1]) = \min(4, 3) = 3$ cameras.

Placement achieving 3 cameras: cameras at nodes $\{0, 3, 1\}$ (or any equivalent minimum set).