Maximum Independent Set in a Tree

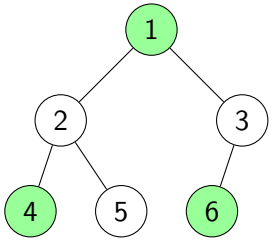
February 20, 2026

Outline

What is an Independent Set?

Definition

An **Independent Set** of a graph $G = (V, E)$ is a subset $S \subseteq V$ such that no two vertices in S are adjacent.



Green vertices = Independent Set

$S = \{1, 4, 6\}$

Key Property

If vertex v is in the independent set, then **none of its neighbors** can be in the set.

Maximum Independent Set (MIS)

Find an independent set with the **maximum number of vertices**.

Why Trees are Special?

General Graphs

- MIS is **NP-Hard**
- No known polynomial-time algorithm
- Need approximation algorithms

Trees

- Solvable in **O(n)** time!
- Uses Dynamic Programming
- Exploits tree's hierarchical structure

Key Insight

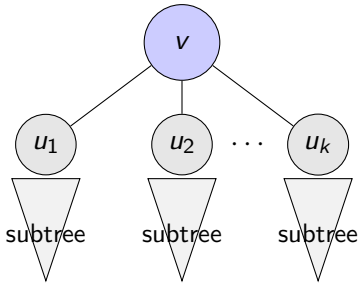
In a tree, once we decide whether to include the root, the problem decomposes into **independent subproblems** on subtrees.

DP State Definition

State Variables

For each vertex v in the tree, define:

- **dp[v][0]** = Size of MIS in subtree rooted at v, when v is **NOT included**
- **dp[v][1]** = Size of MIS in subtree rooted at v, when v is **included**



DP Recurrence Relations

Let $u_1, u_2, \ldots, u_k$ be the children of vertex v.

Case 1: Vertex v is NOT included ($dp[v][0]$)

Children can be either included or excluded (take best option):

$$dp[v][0] = \sum_{i=1}^{k} \max(dp[u_i][0],\ dp[u_i][1])$$

Case 2: Vertex v IS included ($dp[v][1]$)

Children **cannot** be included (must exclude all children):

$$dp[v][1] = 1 + \sum_{i=1}^{k} dp[u_i][0]$$

Final Answer

MIS Size $= \max(dp[root][0],\ dp[root][1])$

Base Case: Leaf Nodes

Base Case

For a leaf node v (no children):

- $dp[v][0] = 0$ (leaf not included $\Rightarrow$ empty set)
- $dp[v][1] = 1$ (leaf included $\Rightarrow$ set of size 1)

Leaf Node

$dp[v][0] = 0$

$dp[v][1] = 1$

Processing Order

Process vertices in **post-order** (children before parents), typically using DFS.

DP Algorithm for MIS in Tree

Algorithm 1 Maximum Independent Set in Tree

Require: Tree T rooted at vertex r

Ensure: Size of Maximum Independent Set

Initialize $dp[v][0] \leftarrow 0$ and $dp[v][1] \leftarrow 0$ for all v

function DFS(v, parent):
    $dp[v][1] \leftarrow 1$ {Include current vertex}
    for each child u of v (where $u \neq$ parent):
        DFS(u, v)
        $dp[v][0] \leftarrow dp[v][0] + \max(dp[u][0], dp[u][1])$
        $dp[v][1] \leftarrow dp[v][1] + dp[u][0]$

Call DFS(r, -1)

return $\max(dp[r][0], dp[r][1])$

Time and Space Complexity

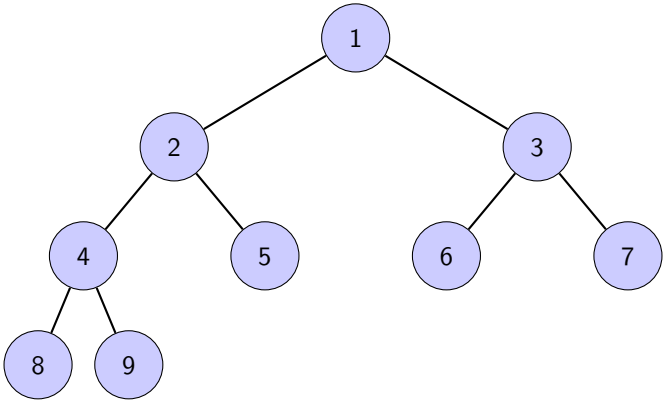
Time Complexity: $O(n)$

- Each vertex is visited exactly once during DFS
- At each vertex, we do $O(1)$ work per child
- Total: $O(n)$ where n = number of vertices

Space Complexity: $O(n)$

- DP table: $O(n)$ for storing $dp[v][0]$ and $dp[v][1]$
- Recursion stack: $O(h)$ where h = height of tree
- Worst case (skewed tree): $O(n)$

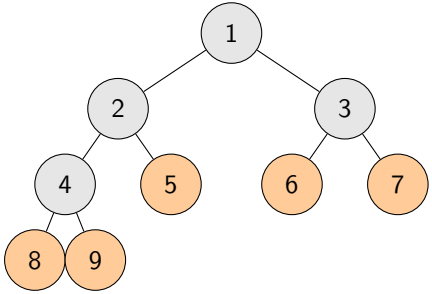
Example: The Tree



Tree Structure

- Root: Vertex 1
- 9 vertices total
- We will compute $dp[v][0]$ and $dp[v][1]$ for each vertex

Step 1: Process Leaf Nodes



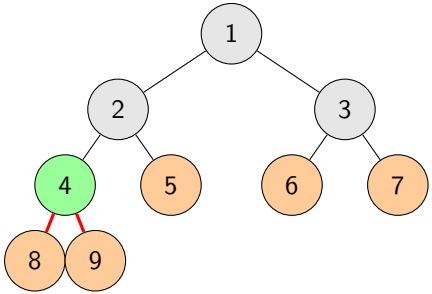
Orange = Leaf nodes

Leaf Nodes: 5, 6, 7, 8, 9

Base case applies:

$$dp[5][0] = 0, \quad dp[5][1] = 1$$
$$dp[6][0] = 0, \quad dp[6][1] = 1$$
$$dp[7][0] = 0, \quad dp[7][1] = 1$$
$$dp[8][0] = 0, \quad dp[8][1] = 1$$
$$dp[9][0] = 0, \quad dp[9][1] = 1$$

Step 2: Process Vertex 4



Processing vertex 4

Children: 8, 9

Computing $dp[4][0]$

Vertex 4 NOT included:

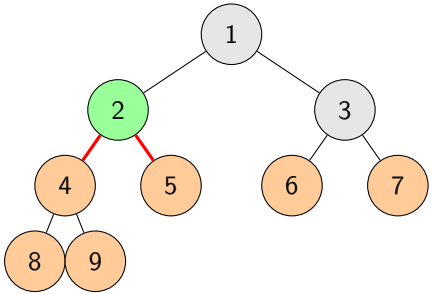
$$\begin{aligned} dp[4][0] &= \max(dp[8][0], dp[8][1]) \\ &\quad + \max(dp[9][0], dp[9][1]) \\ &= \max(0,1) + \max(0,1) \\ &= 1 + 1 = \mathbf{2} \end{aligned}$$

Computing $dp[4][1]$

Vertex 4 IS included:

$$\begin{aligned} dp[4][1] &= 1 + dp[8][0] + dp[9][0] \\ &= 1 + 0 + 0 = \mathbf{1} \end{aligned}$$

Step 3: Process Vertex 2



Processing vertex 2

Children: 4, 5

Known: $dp[4] = (2, 1)$, $dp[5] = (0, 1)$

Computing $dp[2][0]$

Vertex 2 NOT included:

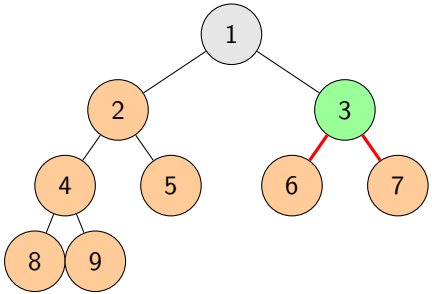
$$\begin{aligned} dp[2][0] &= \max(dp[4][0], dp[4][1]) \\ &\quad + \max(dp[5][0], dp[5][1]) \\ &= \max(2, 1) + \max(0, 1) \\ &= 2 + 1 = \mathbf{3} \end{aligned}$$

Computing $dp[2][1]$

Vertex 2 IS included:

$$\begin{aligned} dp[2][1] &= 1 + dp[4][0] + dp[5][0] \\ &= 1 + 2 + 0 = \mathbf{3} \end{aligned}$$

Step 4: Process Vertex 3



Processing vertex 3

Children: 6, 7

Known: $dp[6] = (0, 1)$, $dp[7] = (0, 1)$

Computing $dp[3][0]$

Vertex 3 NOT included:

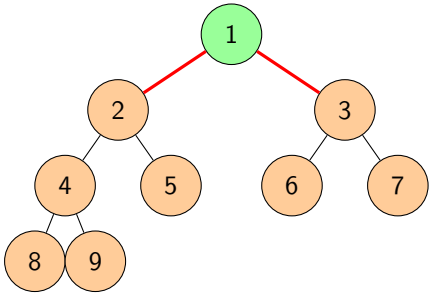
$$\begin{aligned} dp[3][0] &= \max(dp[6][0], dp[6][1]) \\ &\quad + \max(dp[7][0], dp[7][1]) \\ &= \max(0, 1) + \max(0, 1) \\ &= 1 + 1 = \mathbf{2} \end{aligned}$$

Computing $dp[3][1]$

Vertex 3 IS included:

$$\begin{aligned} dp[3][1] &= 1 + dp[6][0] + dp[7][0] \\ &= 1 + 0 + 0 = \mathbf{1} \end{aligned}$$

Step 5: Process Root (Vertex 1)



Processing ROOT (vertex 1)
Children: 2, 3

Known: $dp[2] = (3, 3)$, $dp[3] = (2, 1)$

Computing $dp[1][0]$

Vertex 1 NOT included:

$$\begin{aligned} dp[1][0] &= \max(dp[2][0], dp[2][1]) \\ &\quad + \max(dp[3][0], dp[3][1]) \\ &= \max(3, 3) + \max(2, 1) \\ &= 3 + 2 = \mathbf{5} \end{aligned}$$

Computing $dp[1][1]$

Vertex 1 IS included:

$$\begin{aligned} dp[1][1] &= 1 + dp[2][0] + dp[3][0] \\ &= 1 + 3 + 2 = \mathbf{6} \end{aligned}$$

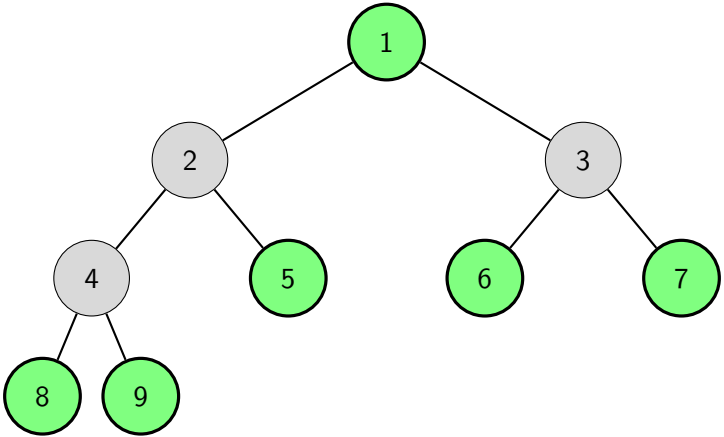
Complete DP Table

Vertex v	$dp[v][0]$	$dp[v][1]$	**Best Choice**
1 (root)	5	**6**	Include
2	**3**	3	Exclude (since 1 included)
3	**2**	1	Exclude (since 1 included)
4	**2**	1	Exclude
5	0	**1**	Include
6	0	**1**	Include
7	0	**1**	Include
8	0	**1**	Include
9	0	**1**	Include

Final Answer

Maximum Independent Set Size $= \max(dp[1][0], dp[1][1]) = \max(5, 6) = \mathbf{6}$

The Maximum Independent Set



Maximum Independent Set

$S = \{1, 5, 6, 7, 8, 9\}$ with size $|S| = 6$

Verification: No two vertices in S are adjacent! ✓

Backtracking to Find the Set

Algorithm 2 Reconstruct Maximum Independent Set

```
1:  MIS ← ∅
2:
3:  function FindMIS(v, parentIncluded):
4:      if parentIncluded:
5:          {Parent is in MIS, so v cannot be}
6:          for each child u of v: FindMIS(u, false)
7:      else:
8:          if dp[v][1] ≥ dp[v][0]:
9:              Add v to MIS
10:             for each child u of v: FindMIS(u, true)
11:         else:
12:             for each child u of v: FindMIS(u, false)
13: Call FindMIS(root, false)
14: return MIS
```

Extension: Weighted Maximum Independent Set

Problem

Each vertex v has a weight $w(v) > 0$. Find an independent set with **maximum total weight**.

Modified Recurrence

$$dp[v][0] = \sum_{u \in \text{children}(v)} \max(dp[u][0], dp[u][1])$$

$$dp[v][1] = w(v) + \sum_{u \in \text{children}(v)} dp[u][0]$$

Base Case (Leaf v)

- $dp[v][0] = 0$
- $dp[v][1] = w(v)$— Same $O(n)$ complexity!

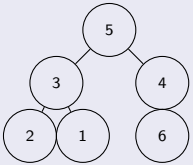
Summary

1. **Problem:** Maximum Independent Set is NP-Hard on general graphs but polynomial on trees
2. **DP States:**
   - $dp[v][0]$: MIS size when v is excluded
   - $dp[v][1]$: MIS size when v is included
3. **Key Insight:** If a vertex is included, its children must be excluded
4. **Complexity:** $O(n)$ time and space
5. **Extensions:** Easily modified for weighted version, counting all MIS, etc.

Practice Problems

Try These!

1. Find the **Minimum Vertex Cover** of a tree using similar DP
   Hint: Vertex Cover is complement of Independent Set
2. Modify the algorithm to **count** the number of maximum independent sets
3. Solve **Maximum Weighted Independent Set** for:



Thank You!

Questions?