Maximum Weight Independent Set on Trees

Rakshya Pandey

CSCE500 – Algorithm Design & Analysis – Spring 2025

01

What is a MWIS Tree?

The MWIS Tree

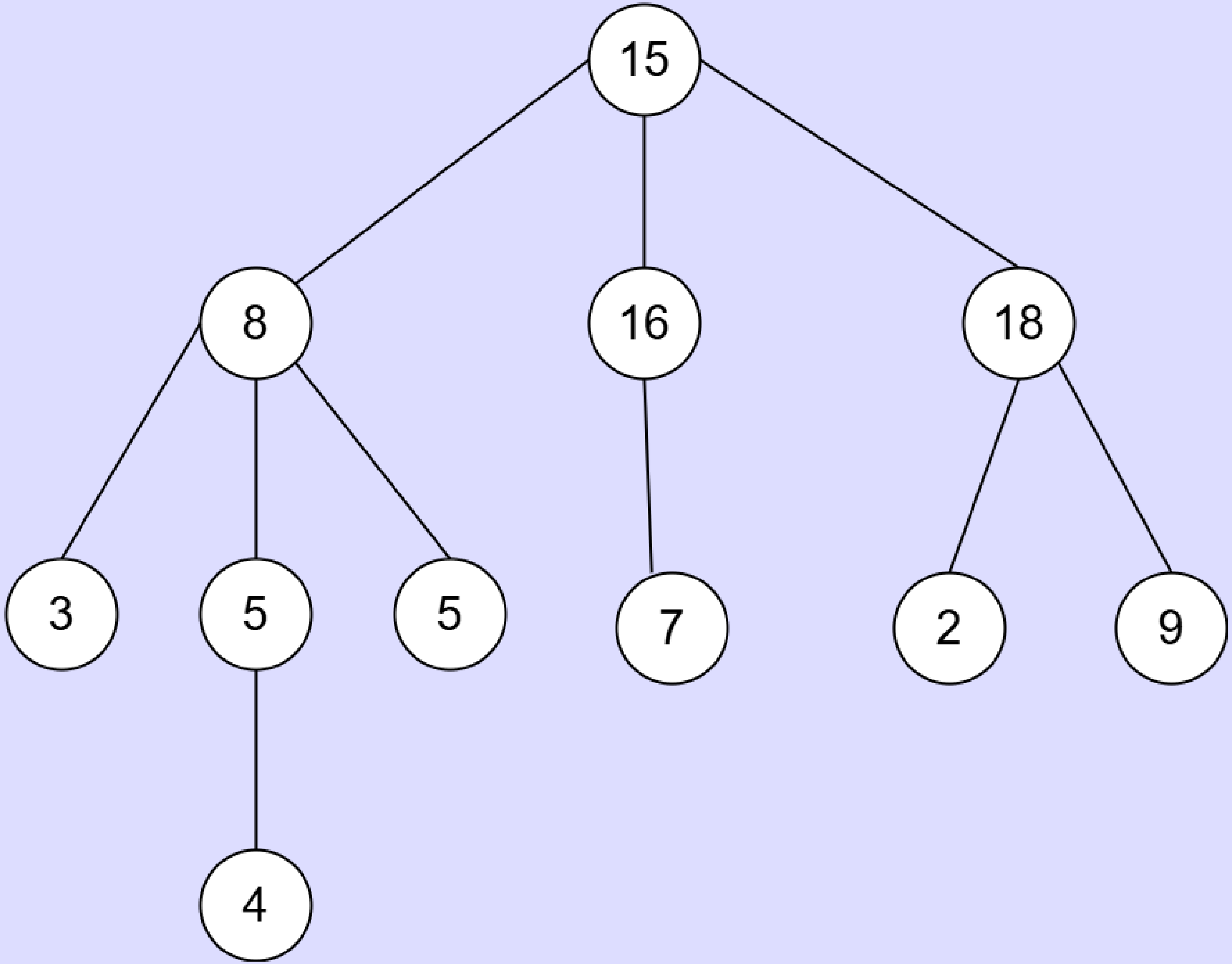
A concept from graph theory and combinatorial optimization which refers to finding an independent set in a tree (**a connected acyclic graph**) with the maximum possible weight.

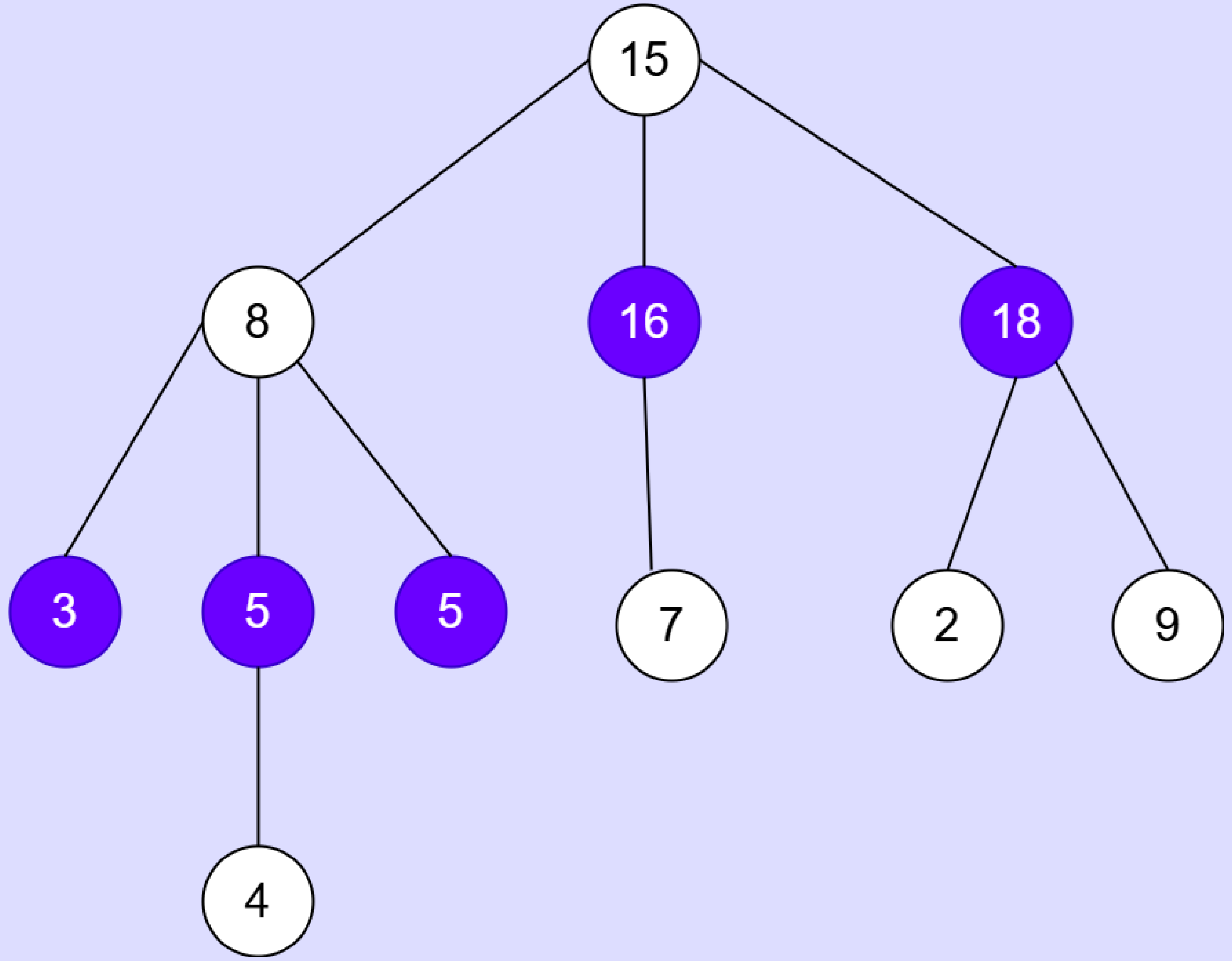
An independent set is a set of vertices with where no two vertices are adjacent.

Each vertex has a weight associated with it.

Maximum weight independent set (MWIS) is the independent set whose vertices have the highest total weight sum.

When this problem is specifically applied to trees (rather than general graphs), it's called the MWIS tree problem.


15
8
16
18
3
5
5
7
2
9
4


15
8
16
18
3
5
5
7
2
9
4

While the general MWIS problem is NP-hard for arbitrary graphs, it can be solved efficiently (in linear time) on trees using dynamic programming.

The MWIS problem on Trees

File System Backup Optimization

MWIS can be used to back up the most valuable files without duplicating folder and subfolder data as file systems naturally form trees. It helps to reduce redundancy by not selecting both a folder and its children and hence maximize backup importance.

The MWIS problem applications

Influencer Selection in Hierarchical Networks

Enterprise marketing platforms (e.g., Adobe Experience Platform, Oracle Eloqua, or Salesforce Marketing Cloud) often model influence hierarchies to: avoid sending marketing content to both a manager and their subordinate, prevent overlap of outreach within teams and maximize the reach of a campaign across a customer organization chart.

The MWIS problem applications

02

How do we solve this problem?

Algorithm

Use post-order DFS to calculate values bottom-up.

Two states per node u:

`dp[u][0]`: max weight excluding node u

`dp[u][1]`: max weight including node u

```
dp[u][0] = sum(max(dp[v][0], dp[v][1]) for v in children[u])
dp[u][1] = weight[u] + sum(dp[v][0] for v in children[u])
```

If we include u, we exclude all children.

If we exclude u, we take the best option from each child

Pseudo-code

```
def dfs(u):
    dp[u][1] = weight[u]
    for v in tree[u]:
        dfs(v)
        dp[u][0] += max(dp[v][0], dp[v][1])
        dp[u][1] += dp[v][0]
```

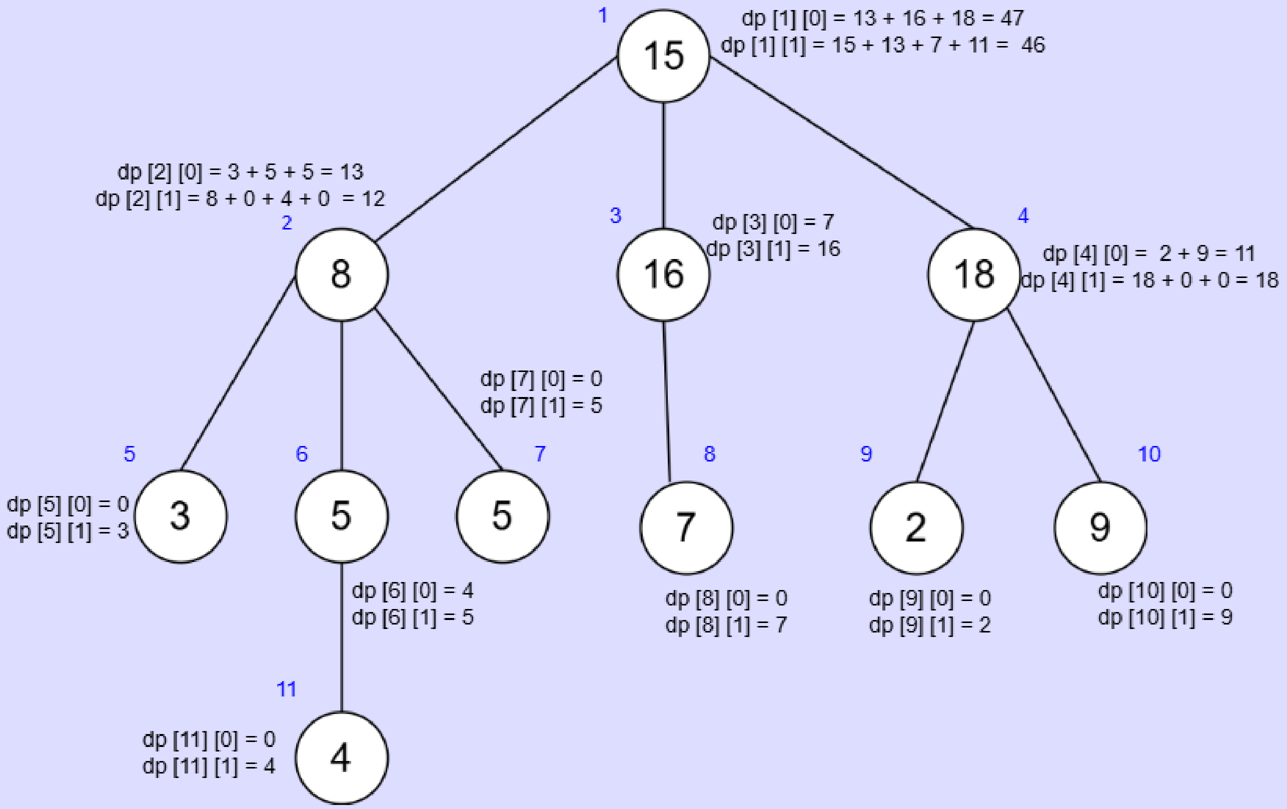
Start with `dfs(1)` from the root.

Leaf nodes are automatically handled (base case).

Final result:
`max(dp[1][0], dp[1][1])`

Link to github: https://github.com/rakshyaaa/MWISTree/blob/main/maxIndependent.py

```
dp[u][0] = sum(max(dp[v][0], dp[v][1]) for v in children[u])
dp[u][1] = weight[u] + sum(dp[v][0] for v in children[u])
```



`Finally, max(dp[1][0], dp[1][1]) = 47`

03

Complexity

Complexity

Each node is visited once during DFS.

Per node: constant-time operations over its children.

Time Complexity: `O(n)`

Space Complexity: `O(n)`

Arrays used: `dp`, `tree`, `weight`

Questions?

THANK YOU