CS2020 – Data Structures and Algorithms Accelerated

Recitation Week11 – Algorithms on Tree

stevenhalim@gmail.com

NUS National University of Singapore

School of Computing

Quick Review

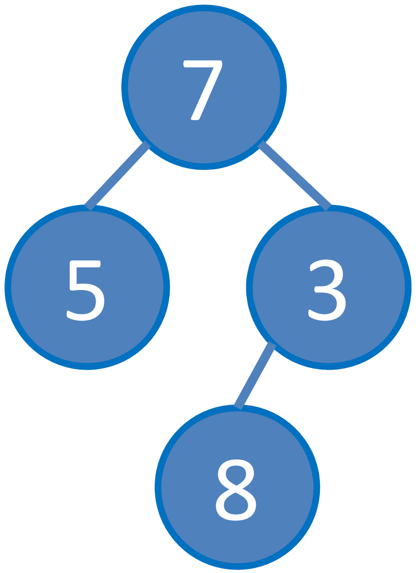
- The Concepts of Tree
  - Connected; E = V - 1; Unique path between two vertices
  - Types: Binary, n-ary, Complete, Full
  - Terminologies: Root, Internal Nodes, Leaves, Sub-trees
  - Tree Traversal Algorithms: Pre-order, In-order, Post-order, Level-order (essentially BFS)
- Two Key Ingredients for Dynamic Programming
  - Optimal Substructures
  - Overlapping Subproblems

Size of a Binary Tree

- This is a naturally recursive algorithm

```
size(T)
  if (T = NULL) return 0;
  else return 1 + size(T->left) + size(T->right);
```

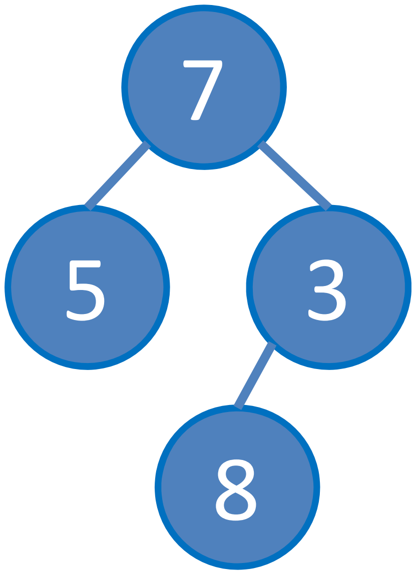
- Time Complexity: O(V)



Height of a Binary Tree

```
height(T)
  if (T = NULL) // empty tree
    return 0;
  else if (T->left = NULL and T->right = NULL) // leaf
    return 1;
  else // internal node
    return 1 + max(height(T->left), height(T->right));
```

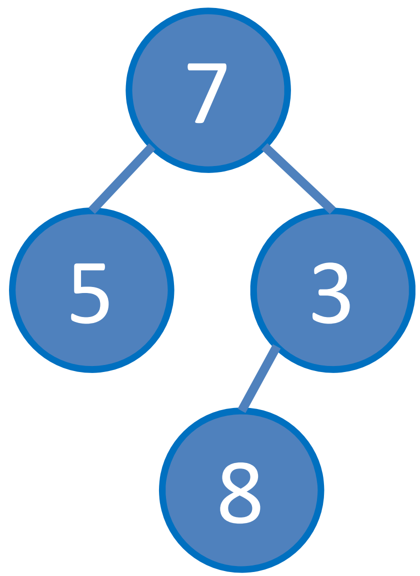
- Time Complexity: also O(V)



Max (or Min) Item of a Binary Tree

```
maxV(T) // minor adjustment for finding min
  if (T = NULL) return -INF;
  else return max(T->value,
    max(maxV(T->left), maxV(T->right));
```

- Time Complexity: again O(V)



Can You Generalize It?

- What if the tree is n-ary tree, not just binary tree?

Can we do Dynamic Programming on Tree?

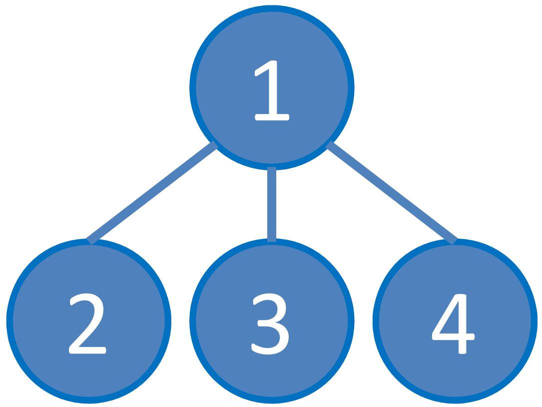
- Ingredient 1: Optimal Sub-structures → OK
  - For most recursive algorithms on Tree, we have:
    - Base cases: Leaves
    - Recursive cases: Sub-trees
- Ingredient 2: Overlapping Sub-problems → Hm?
  - Hm…, there is only one path from root to a certain vertex
  - So, we will never have overlapping subproblem on trees?
  - So… there is never any DP algorithm on tree?
  - Is it? Or is it not?

Tree and DAG

- We have discussed DAG and DP in Lecture18
- Now Tree is also a good structure for DP
  - Like DAG, tree is also acyclic
  - The problem-subproblem relationship must be acyclic, otherwise we cannot write a recurrence relation…
    - We will see more in Lecture20
  - In tree, this problem-subproblem relationship is usually the parent-children relationship
- But, there is no overlapping subproblem in a standard tree… as shown in all examples earlier…
  - But, let’s look at the next problem on tree

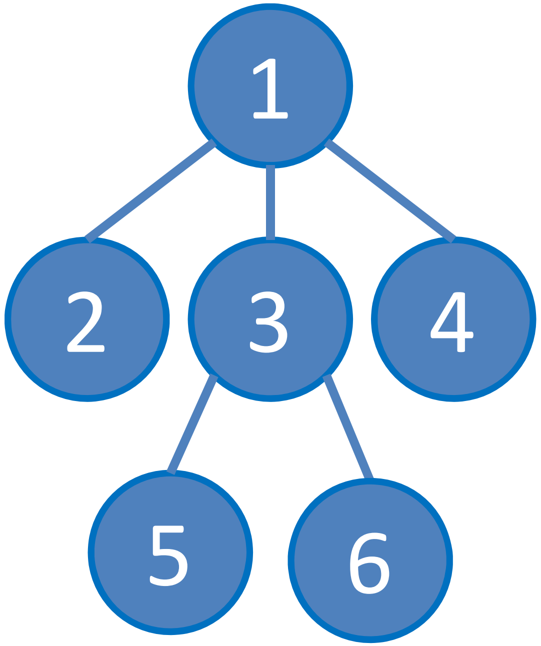
Minimum Vertex Cover on Tree (1)

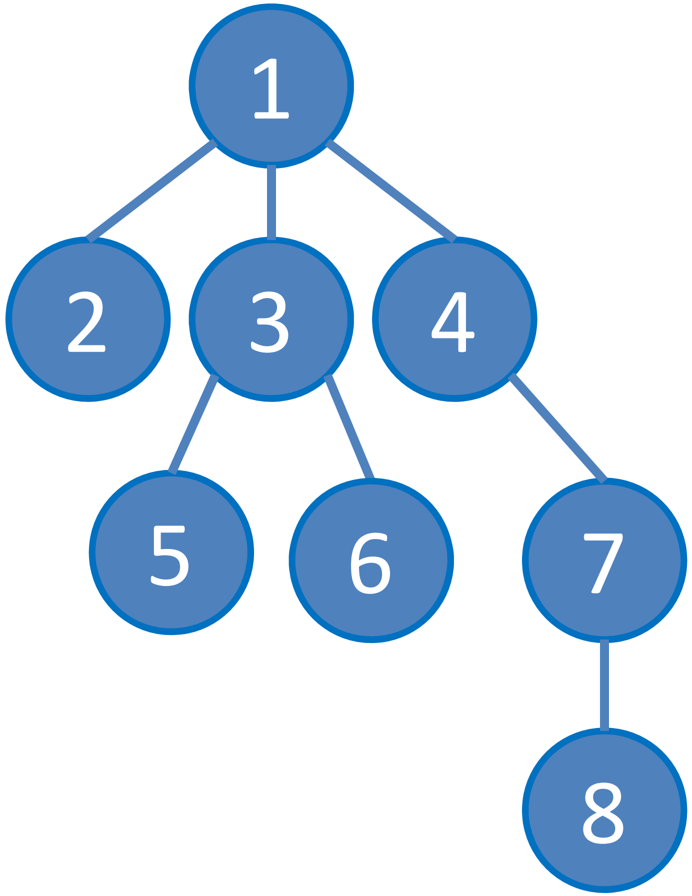
- Select a smallest possible set of vertices $S \subset V$ such that each edge of the tree is incident to at least one vertex of the set S
- For the sample tree shown here, the solution is to take vertex 1 only, because all edges 1-2, 1-3, 1-4 are all incident to vertex 1



Minimum Vertex Cover on Tree (2)

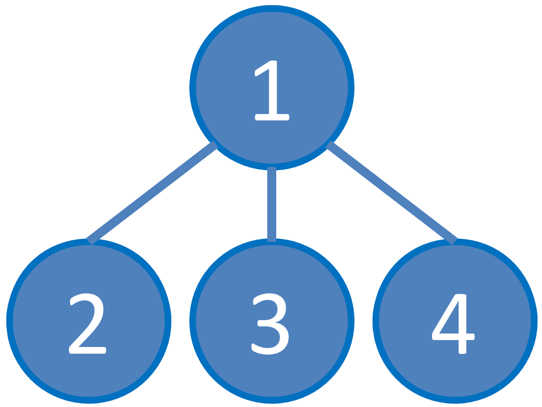
- What is the minimum vertex cover of these two trees?



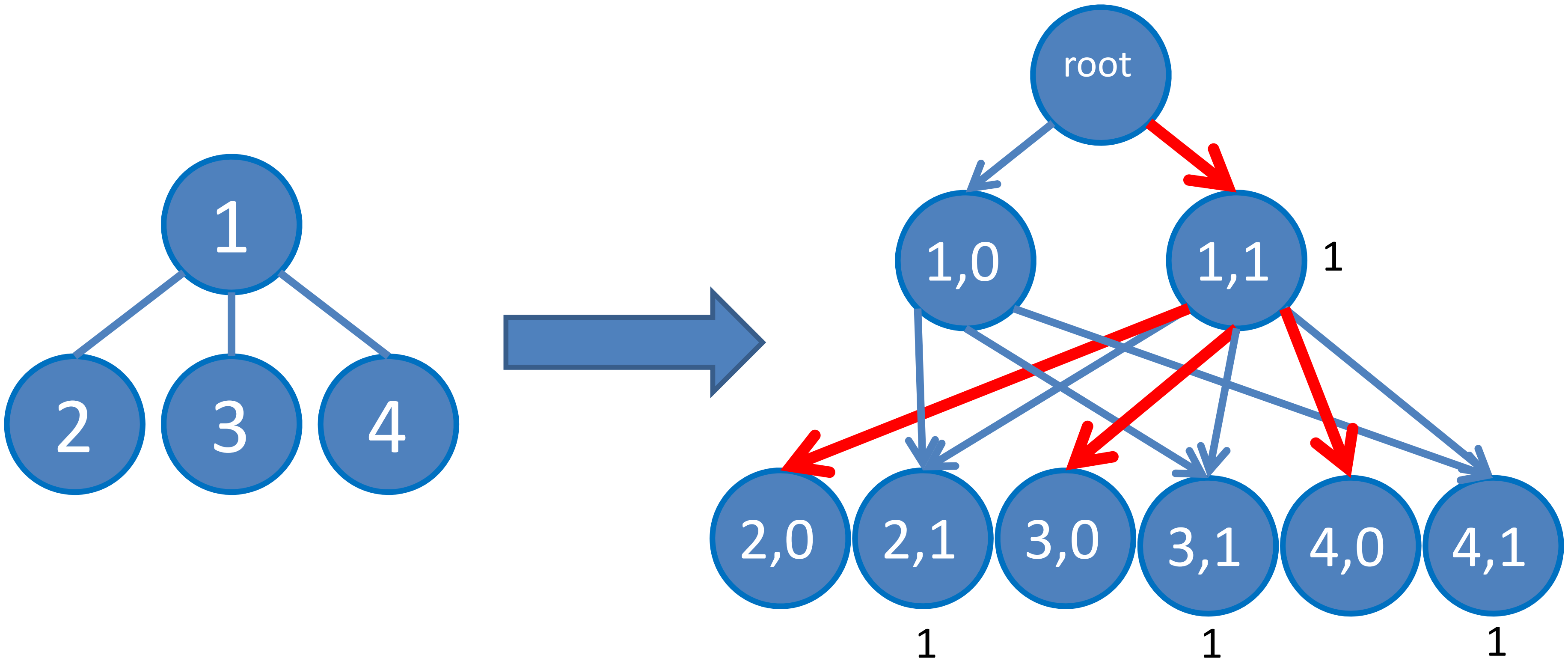


Minimum Vertex Cover on Tree (3)

- This is a hard problem in general graph
  - But on tree, we have an easier solution
- There are only two options for each vertex v
  - Take v as part of set S or ignore v
- If we attach this information to a vertex
  - This tree will turn into a DAG :O
  - Look at the next slide…
  - We will see more of this in Lecture20



Conversion to a DAG



Minimum Vertex Cover on Tree (4)

- The recurrence (with overlapping subproblems):

```
mVC(v, flag) // min Vertex Cover
  if (v = NULL) return flag ? 1 : 0; // 1/0=taken/not
  else if !flag // if v is not taken, take its children
    return 1 + mVC(c, true) for all c ∈ children(v)
  else if flag
    return min(mVC(c, true), mVC(c, false))
      for all c ∈ children(v)
```

- Time Complexity: O(kV)
  - Discussed in DG… (after Lecture20)