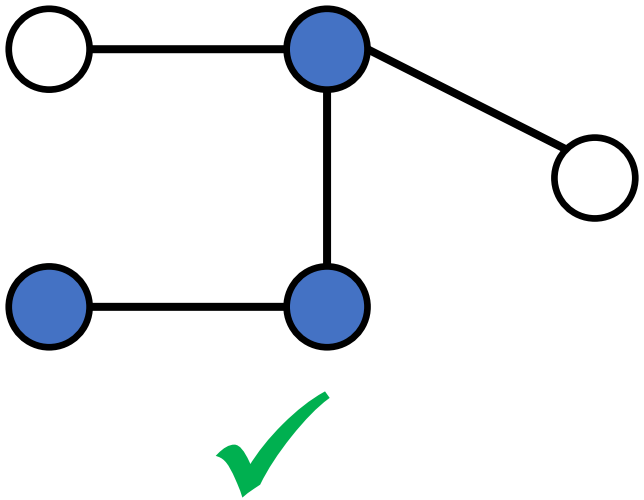
Dynamic Programming
CSCI 532

Vertex Cover

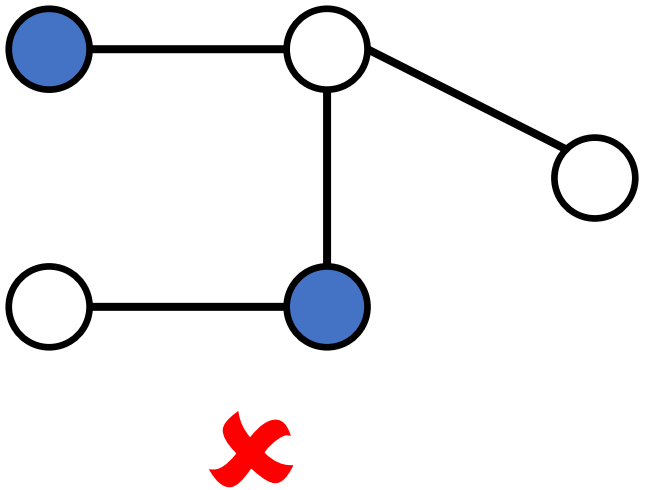
tree

Vertex Cover: Given ~~graph~~, find the smallest subset of vertices such that every edge in the ~~graph~~ has at least one vertex in the subset.

tree

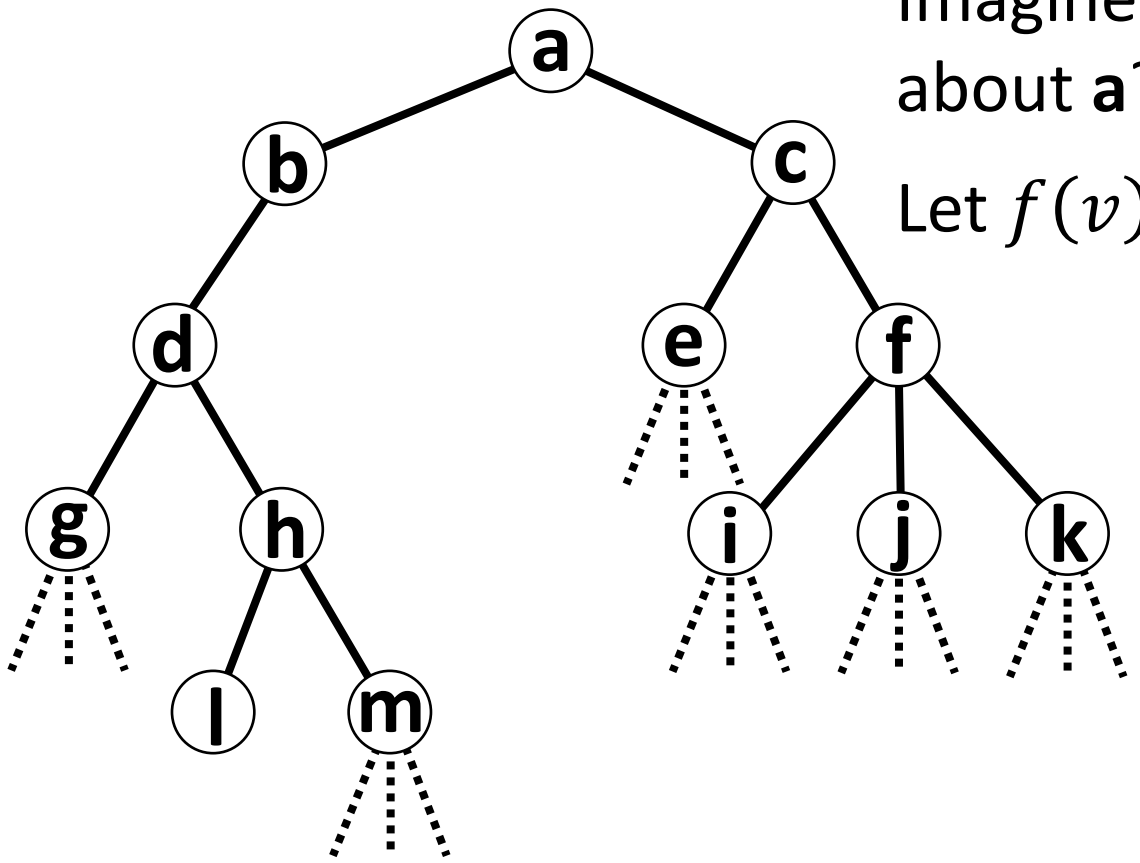


✓



✗

Vertex Cover in Trees



Imagine the minimum vertex cover. What can we say about **a**? **a** is in a minimum vertex cover, or it's not.

Let $f(v) =$ Size of minimum vertex cover rooted at v.

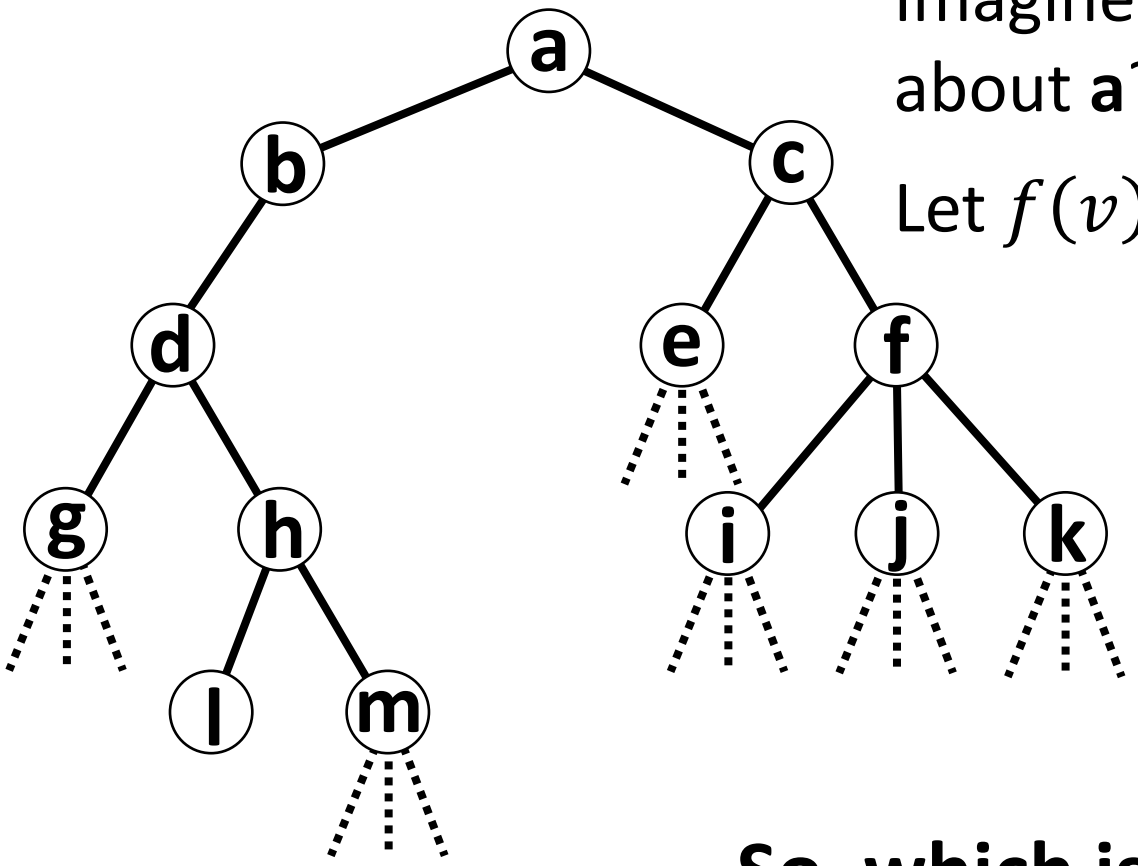
If a **is** in a minimum VC

$$f(a) = 1 + f(b) + f(c)$$

If a **is not** in a minimum VC

$$f(a) = 2 + f(d) + f(e) + f(f)$$

Vertex Cover in Trees



Imagine the minimum vertex cover. What can we say about **a**? **a** is in a minimum vertex cover, or it's not.

Let $f(v) =$ Size of minimum vertex cover rooted at v.

If a **is** in a minimum VC

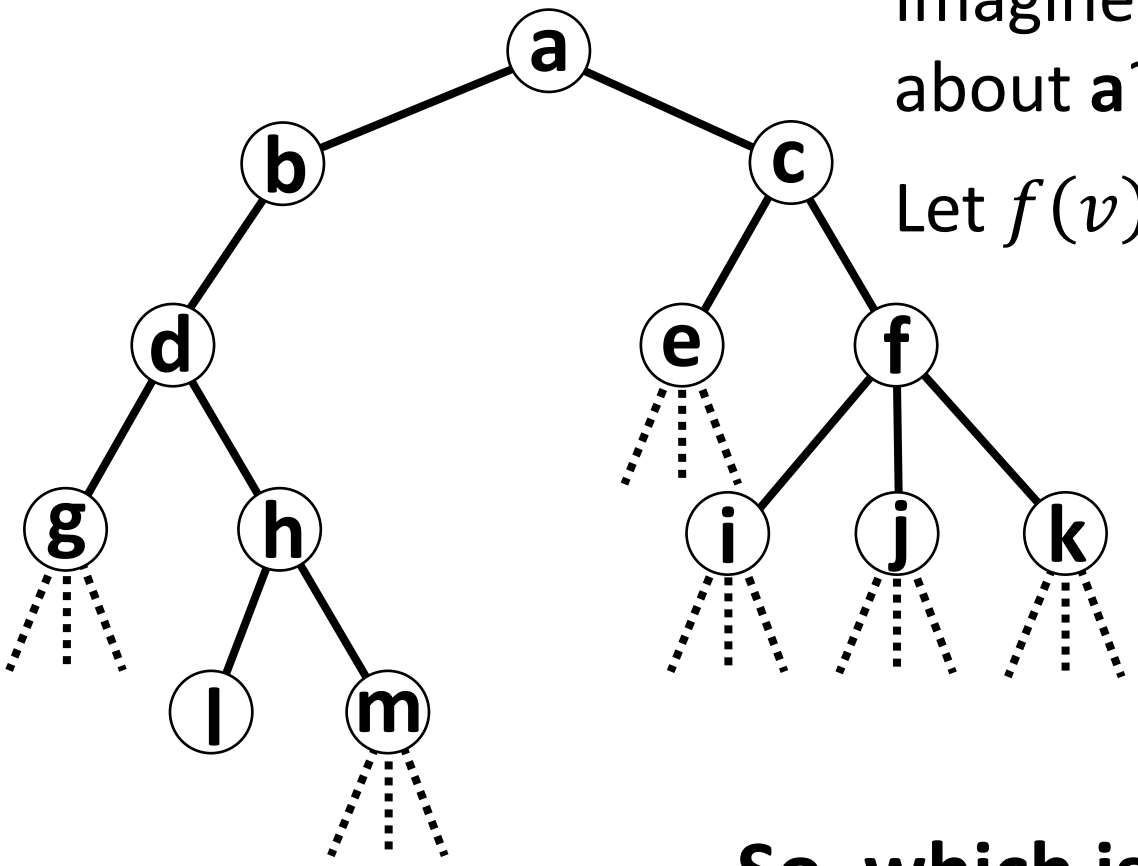
$$f(a) = 1 + f(b) + f(c)$$

If a **is not** in a minimum VC

$$f(a) = 2 + f(d) + f(e) + f(f)$$

So, which is it?

Vertex Cover in Trees



Imagine the minimum vertex cover. What can we say about **a**? **a** is in a minimum vertex cover, or it's not.

Let $f(v) =$ Size of minimum vertex cover rooted at v.

If a **is** in a minimum VC

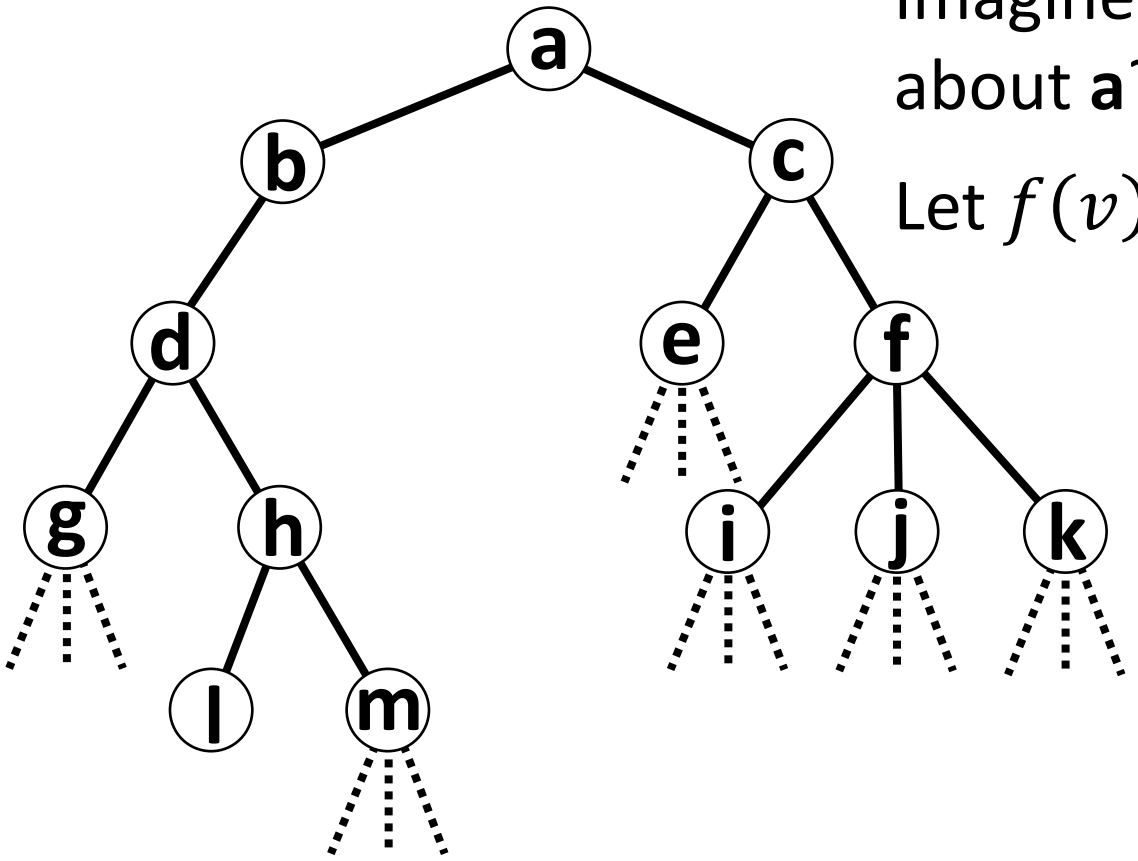
$$f(a) = 1 + f(b) + f(c)$$

If a **is not** in a minimum VC

$$f(a) = 2 + f(d) + f(e) + f(f)$$

So, which is it?
The smallest!

Vertex Cover in Trees



Imagine the minimum vertex cover. What can we say about **a**? **a** is in a minimum vertex cover, or it's not.

Let $f(v) =$ Size of minimum vertex cover rooted at v.

If a **is** in a minimum VC

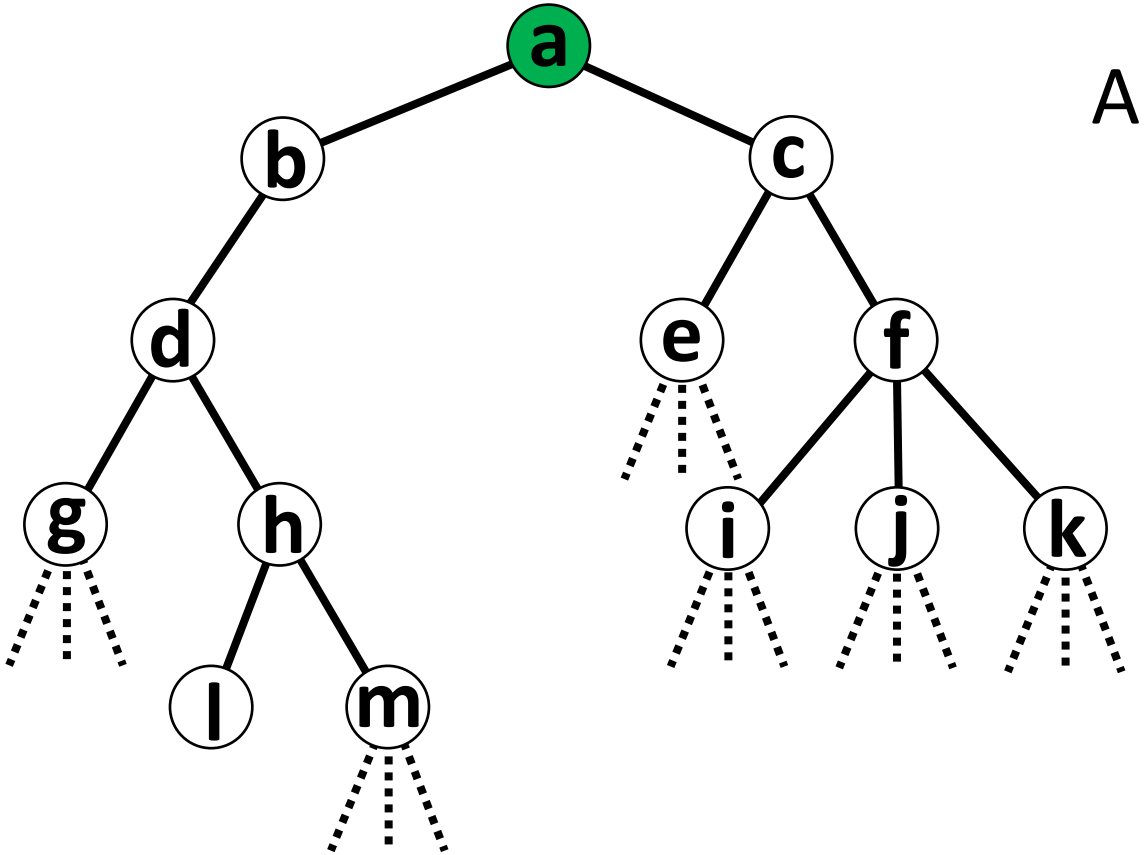
$$f(a) = 1 + f(b) + f(c)$$

If a **is not** in a minimum VC

$$f(a) = 2 + f(d) + f(e) + f(f)$$

$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u),\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

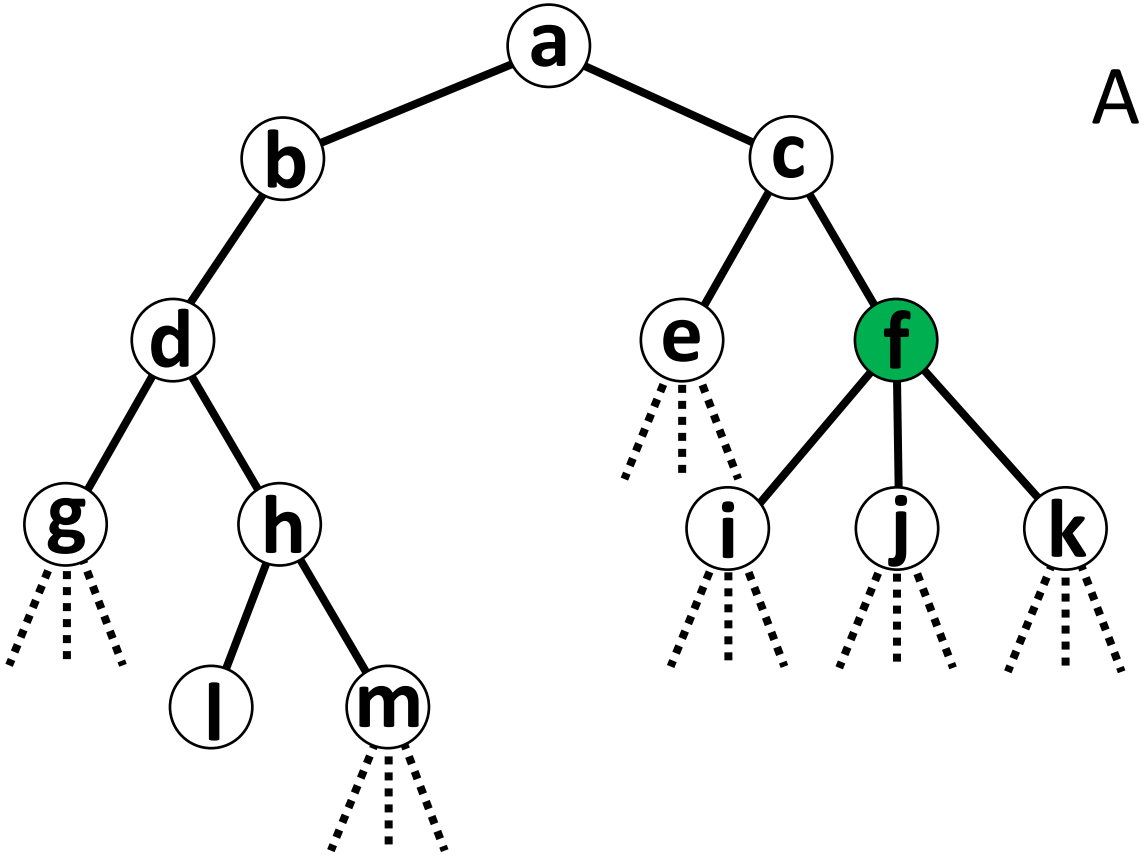
Vertex Cover in Trees



Are we ready to calculate $f(v)$ **here**?

$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u)\,,\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

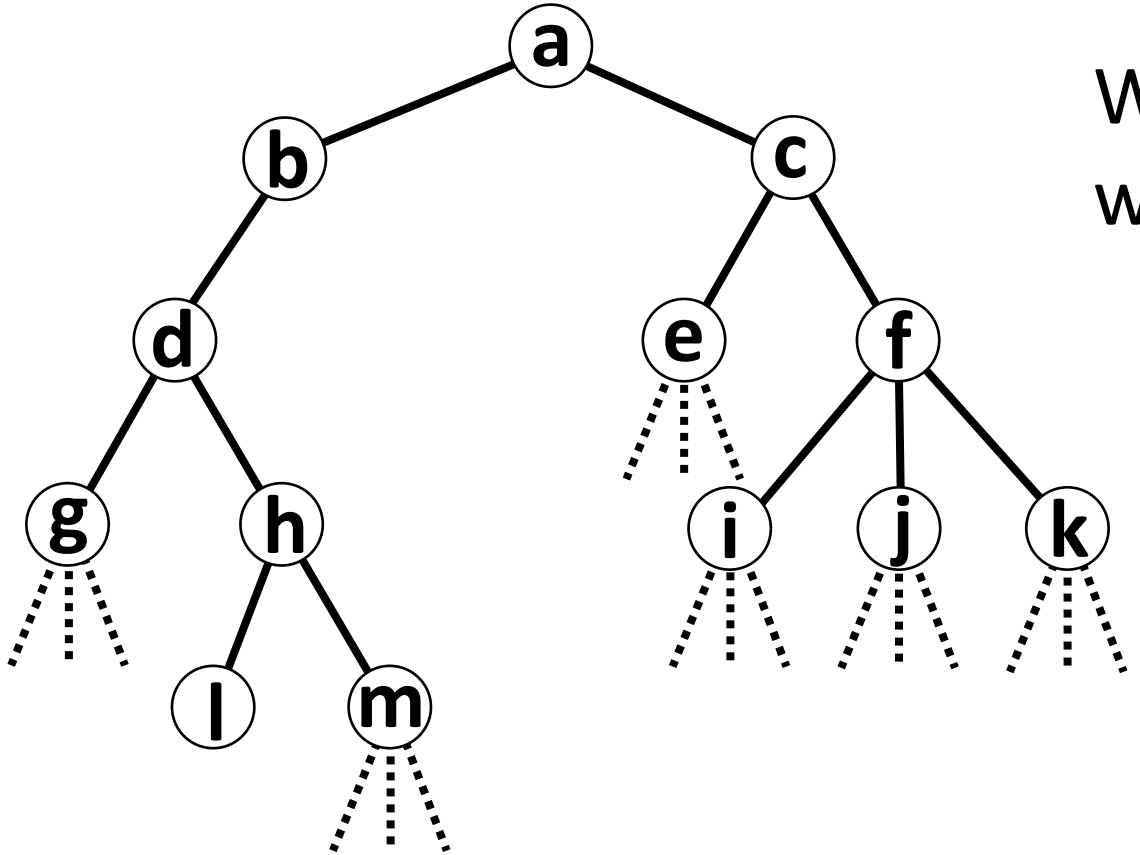
Vertex Cover in Trees



Are we ready to calculate $f(v)$ **here**?

$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u)\,,\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

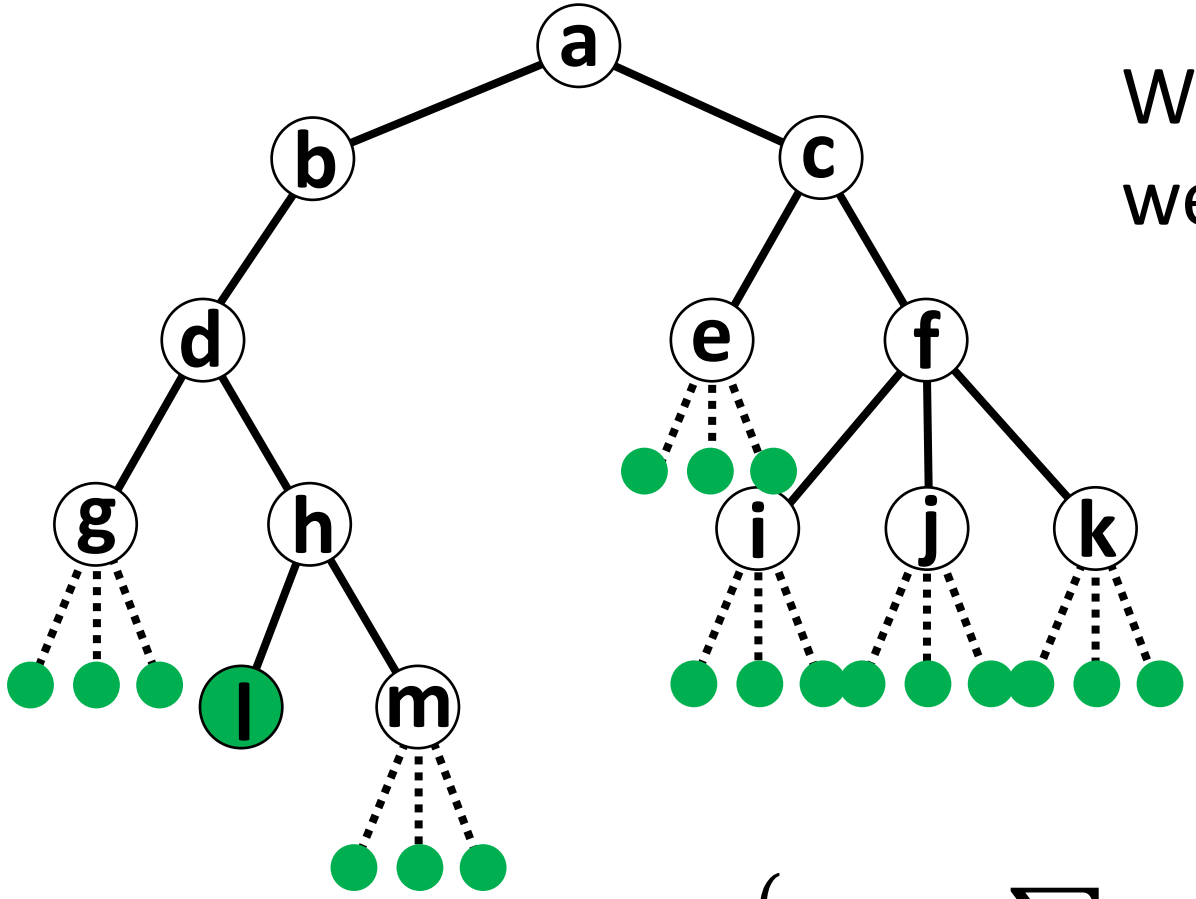
Vertex Cover in Trees



Where do we have all the information we need to calculate $f(v)$?

$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u)\,,\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

Vertex Cover in Trees



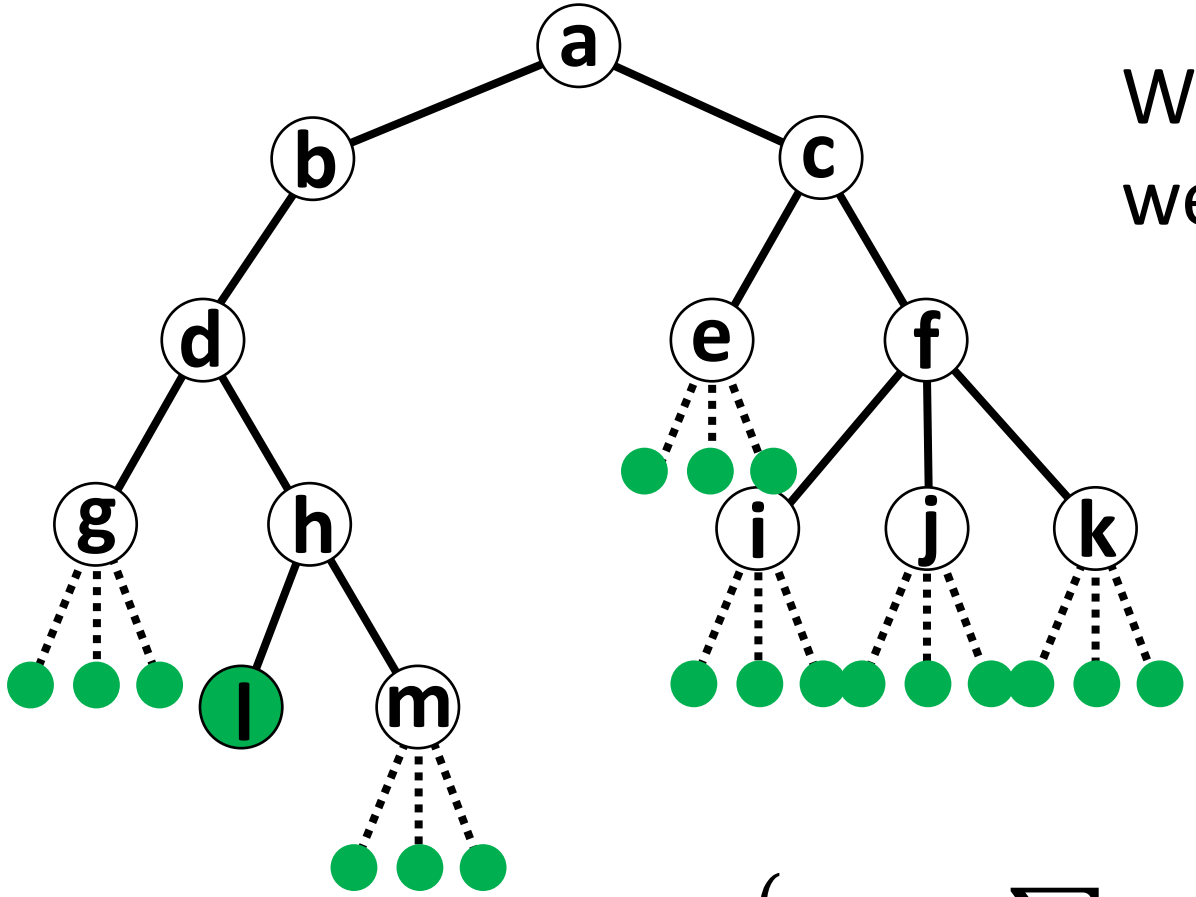
Where do we have all the information we need to calculate $f(v)$?

At the leaves!!

$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u)\,,\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

Vertex Cover in Trees

Let $f(v)$ = Size of minimum vertex cover rooted at v.



Where do we have all the information we need to calculate $f(v)$?

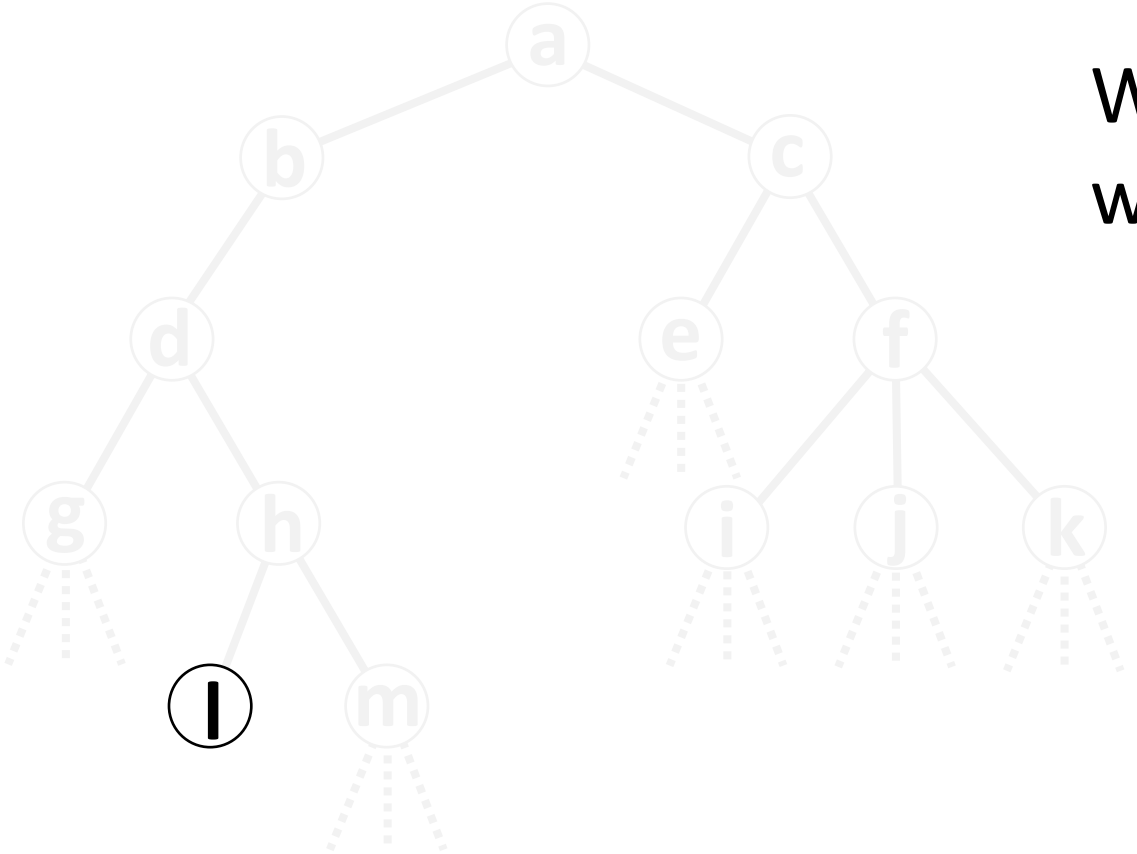
At the leaves!!

If v is a leaf, $f(v) = ??$

$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u),\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

Vertex Cover in Trees

Let $f(v)$ = Size of minimum vertex cover rooted at v.



Where do we have all the information we need to calculate $f(v)$?

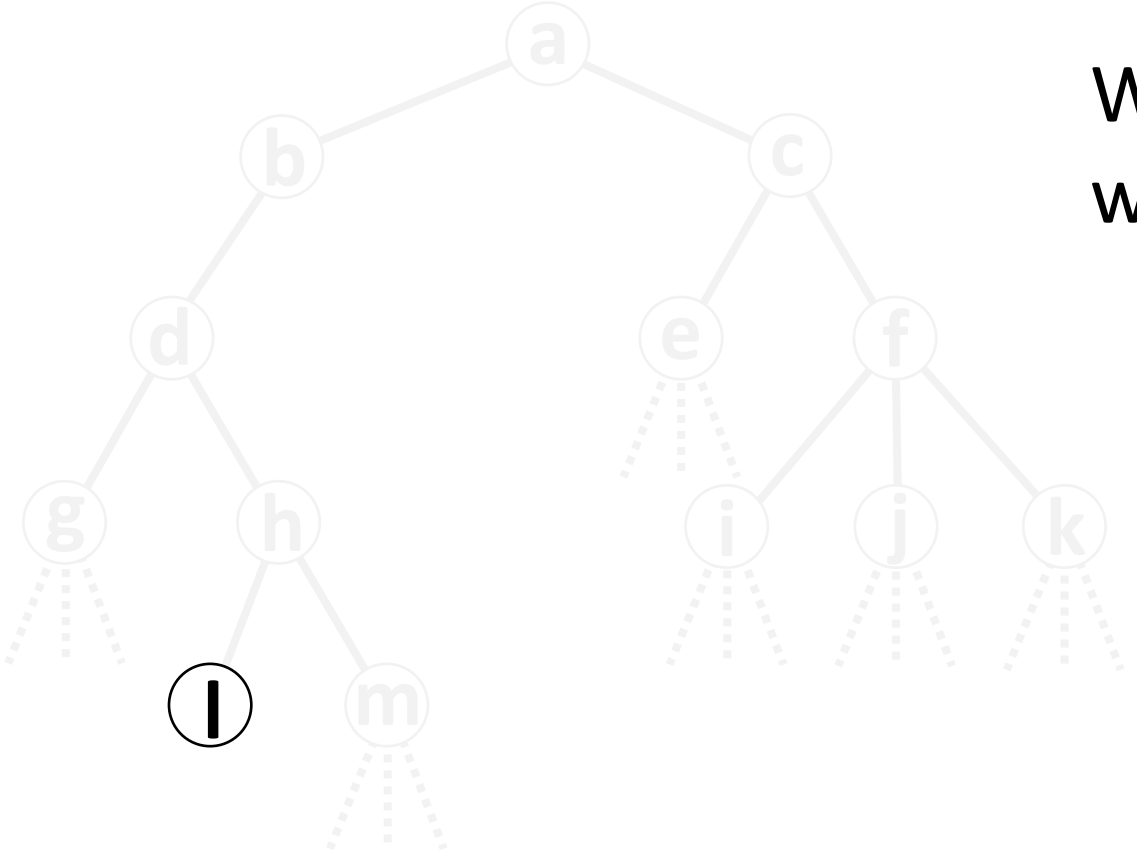
At the leaves!!

If v is a leaf, $f(v) = ??$

$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u)\,,\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

Vertex Cover in Trees

Let $f(v)$ = Size of minimum vertex cover rooted at v.



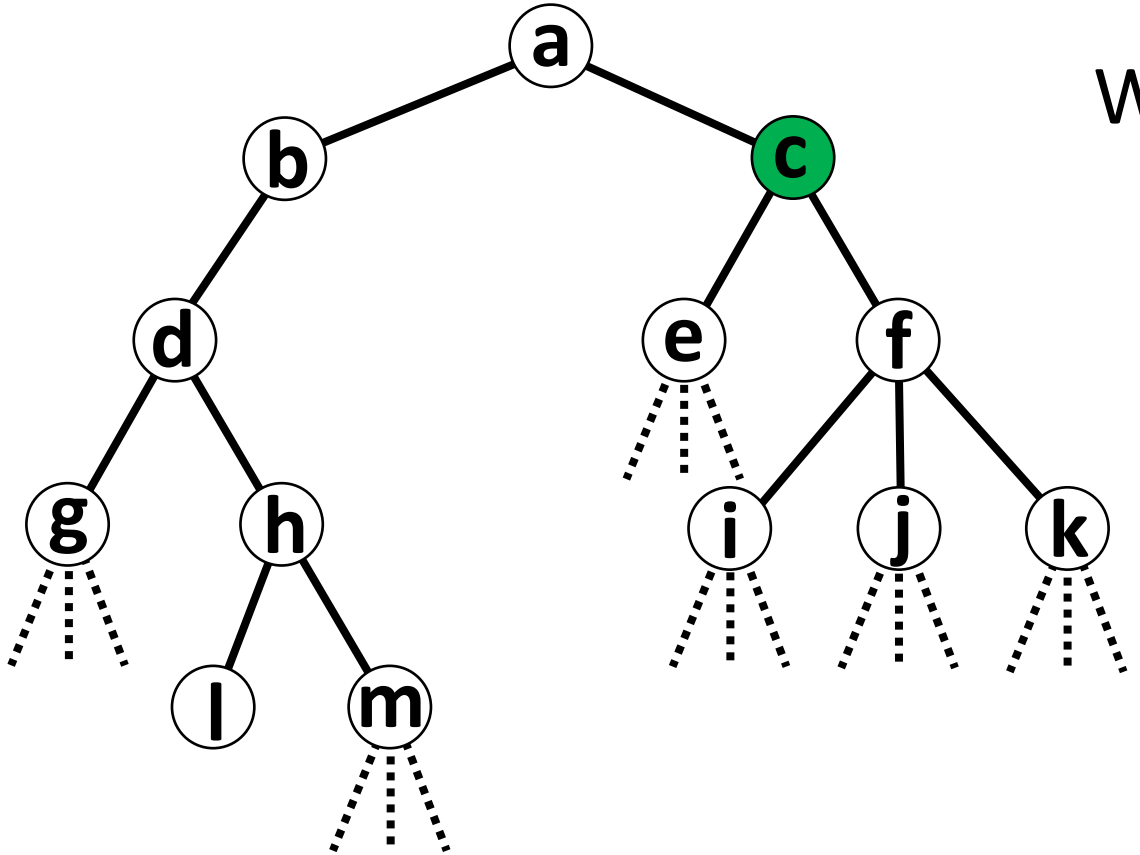
Where do we have all the information we need to calculate $f(v)$?

At the leaves!!

If v is a leaf, $f(v) = 0$

$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u),\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

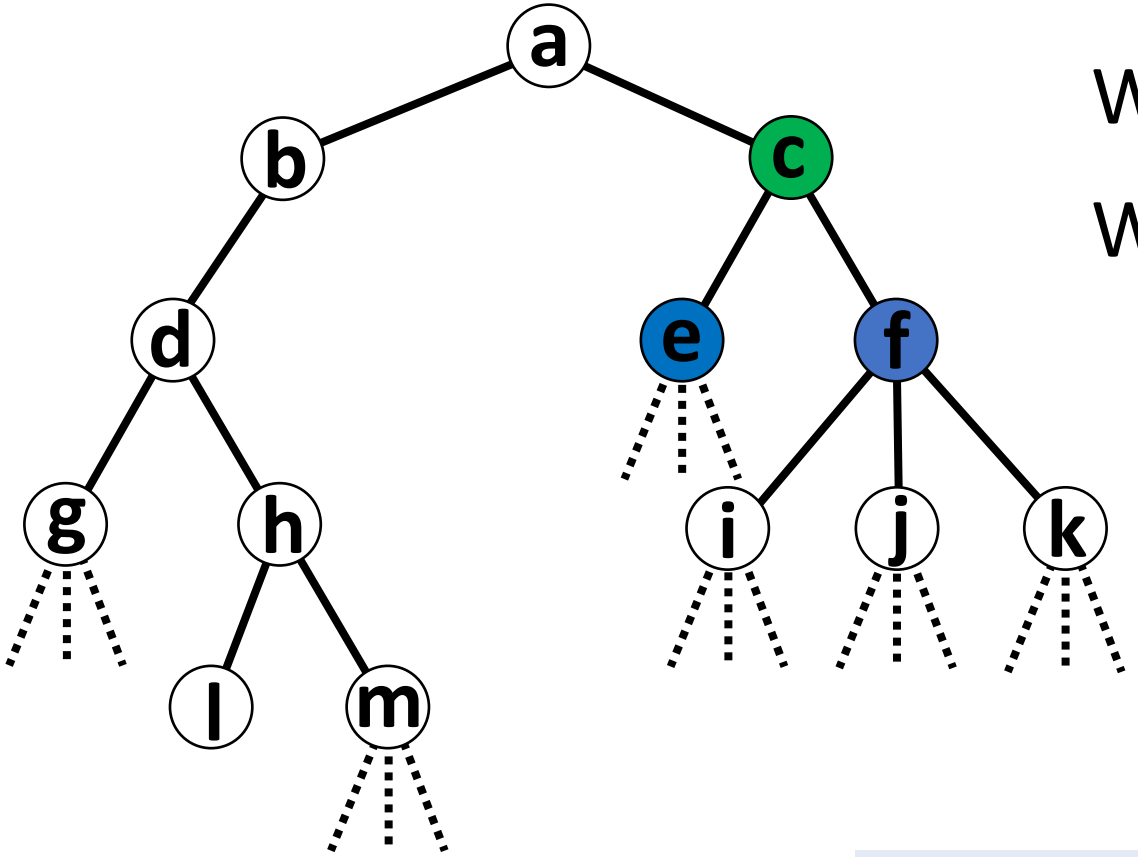
Vertex Cover in Trees



When are we ready to calculate $f(c)$?

$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u)\,,\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

Vertex Cover in Trees

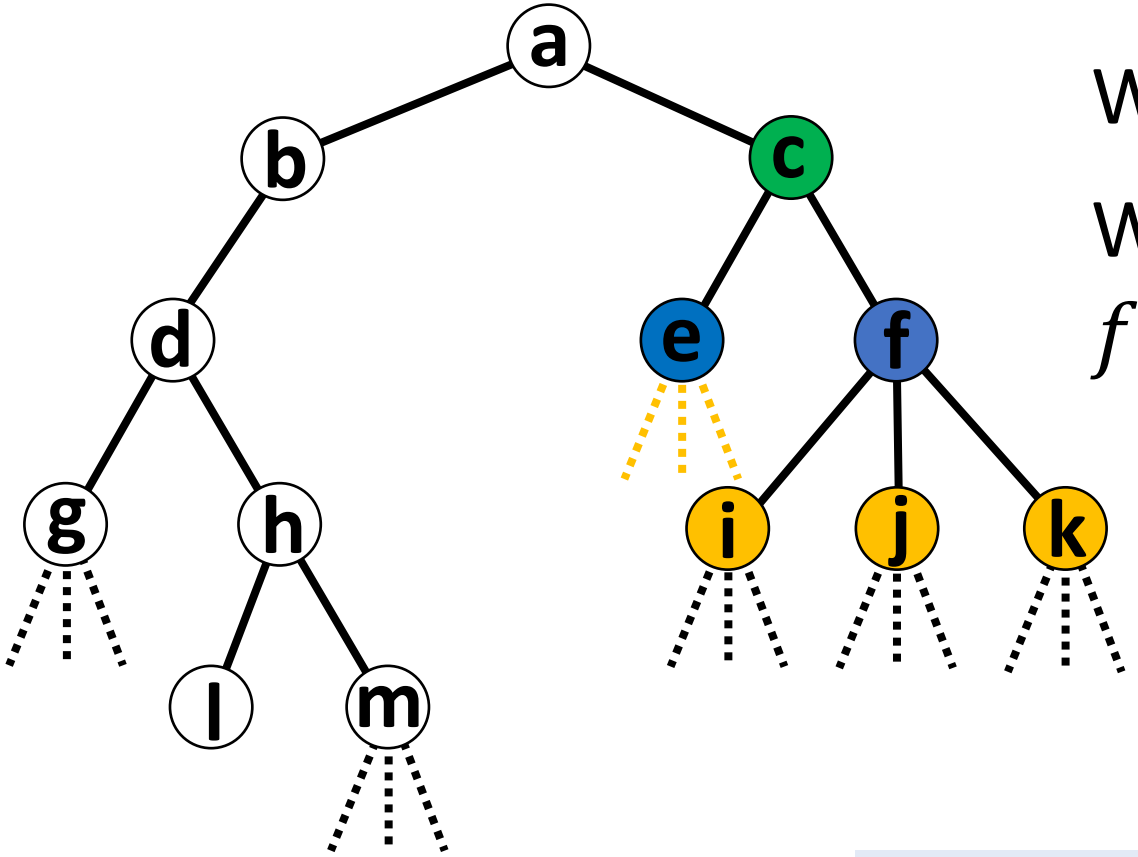


When are we ready to calculate $f(c)$?

When we have $f(e)$ and $f(f)$…

$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u),\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

Vertex Cover in Trees

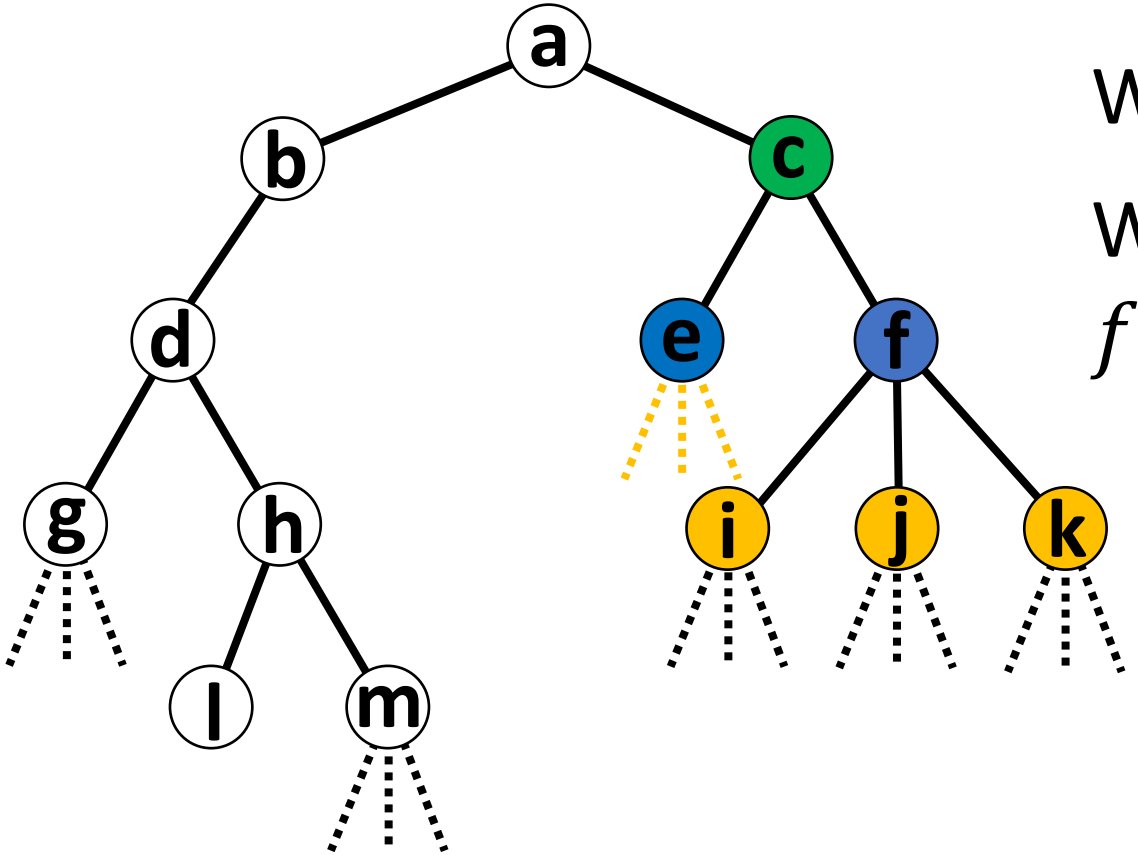


When are we ready to calculate $f(c)$?

When we have $f(e)$ and $f(f)$ and $f(v)$ for all of c's grandchildren.

$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u),\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

Vertex Cover in Trees



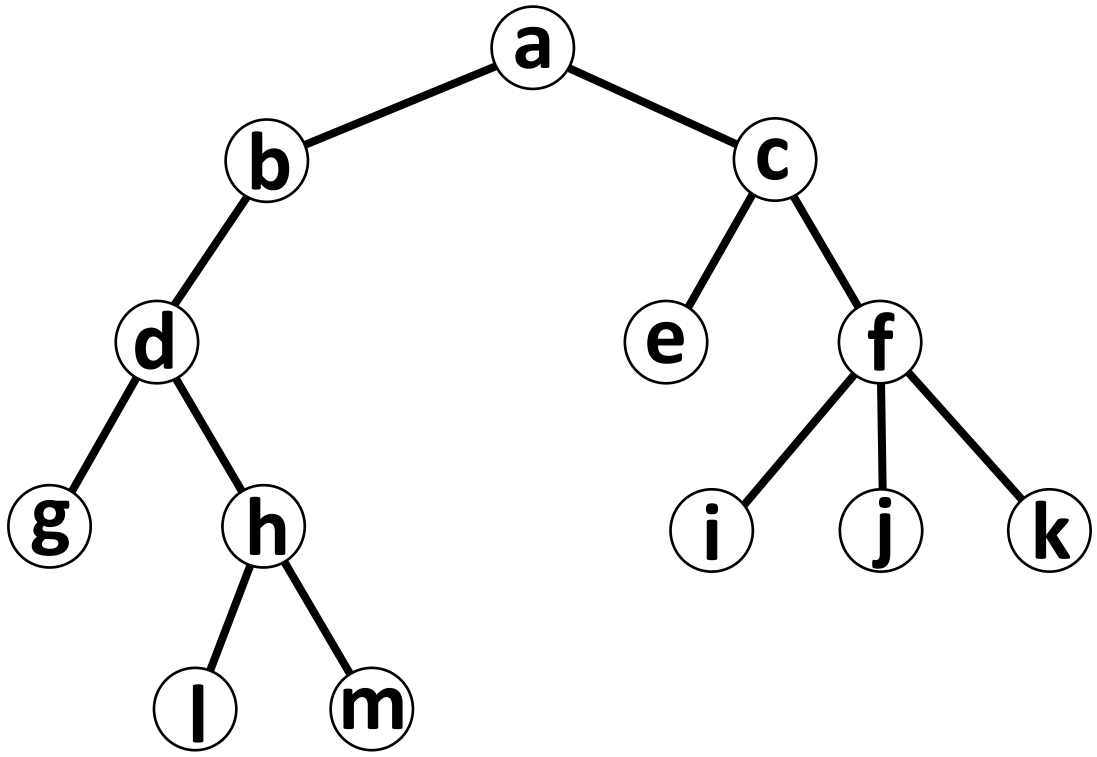
When are we ready to calculate $f(c)$?

When we have $f(e)$ and $f(f)$ and $f(v)$ for all of c's grandchildren.

How can we do this?

$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u),\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

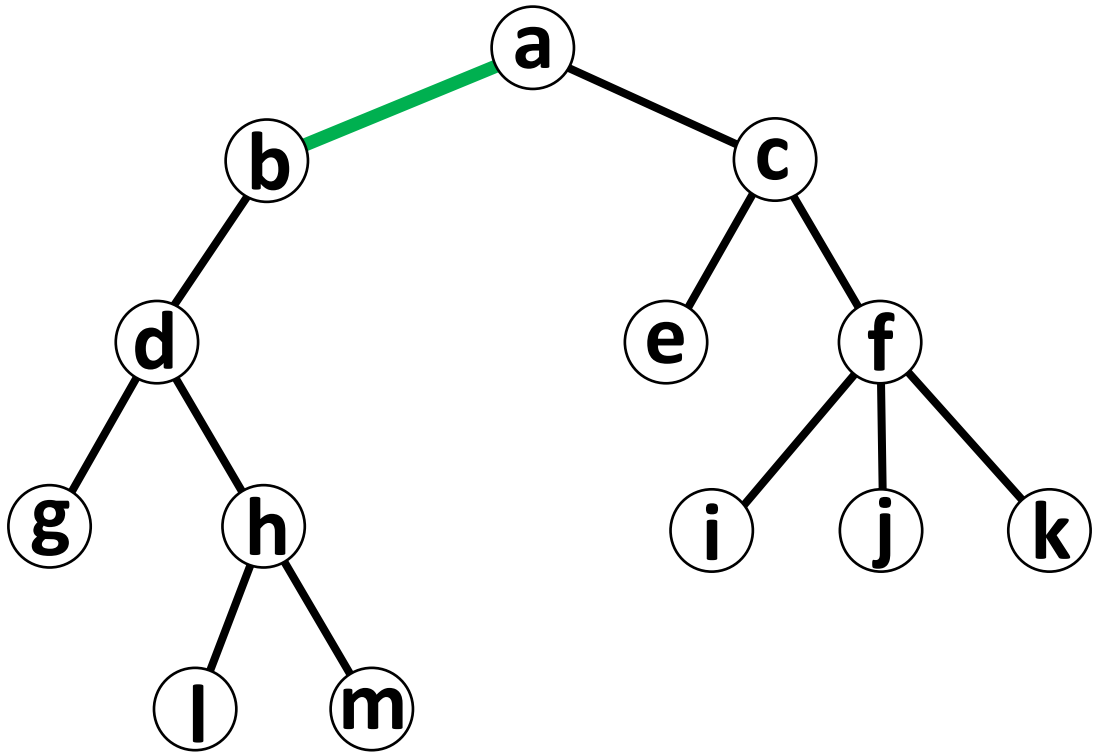
Tree Traversals



Breadth First: Start at a node. Visit all of its children, then all of its grandchildren, then great-grandchildren,…

Order visited: a, b, c, d, e, f, g, h, i, j, k, l ,m

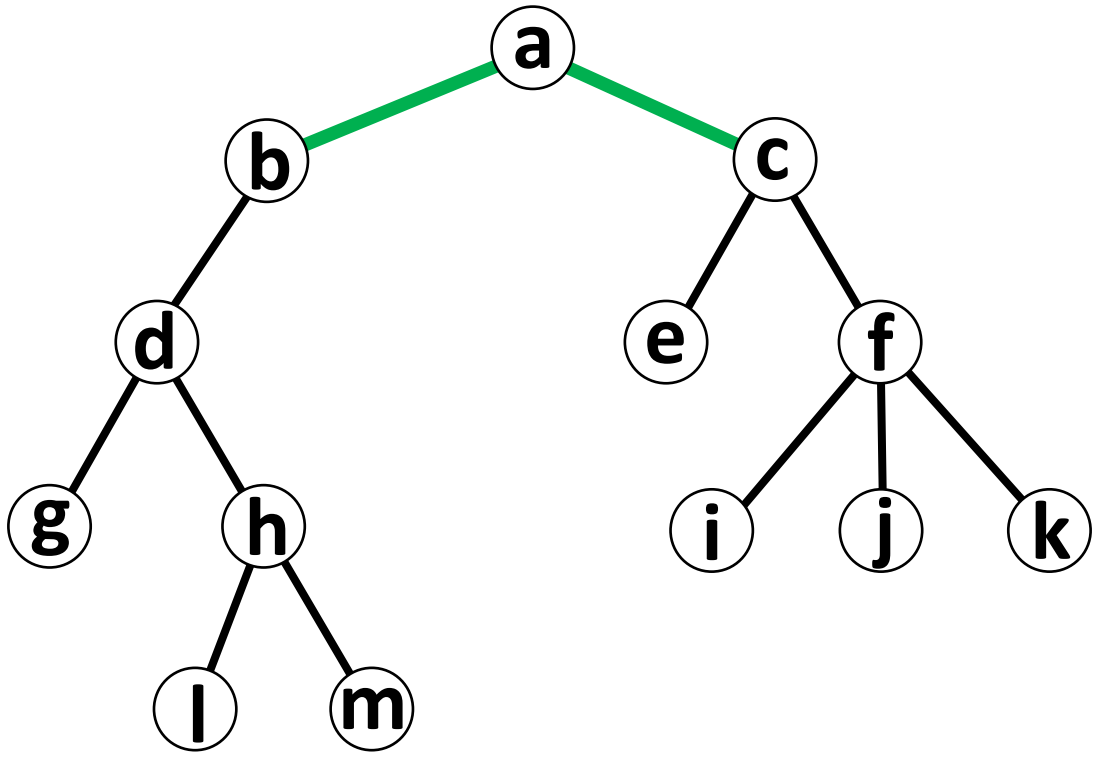
Tree Traversals



Breadth First: Start at a node. Visit all of its children, then all of its grandchildren, then great-grandchildren,…

Order visited: a, b, c, d, e, f, g, h, i, j, k, l ,m

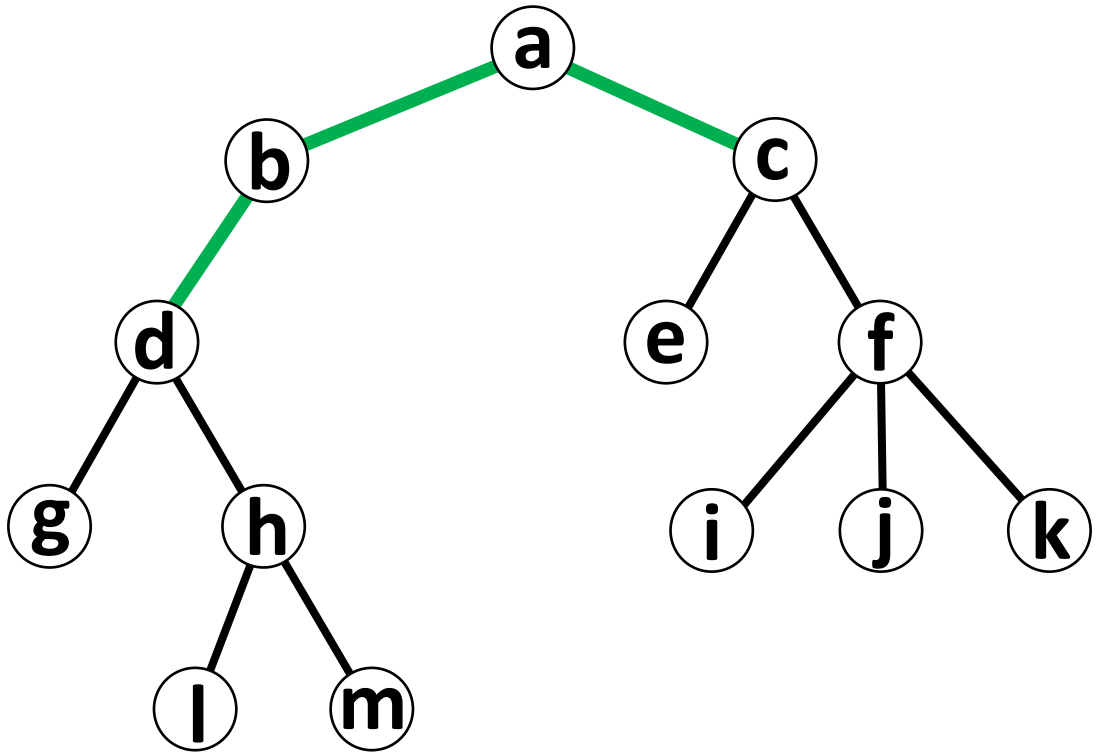
Tree Traversals



Breadth First: Start at a node. Visit all of its children, then all of its grandchildren, then great-grandchildren,…

Order visited: a, b, c, d, e, f, g, h, i, j, k, l ,m

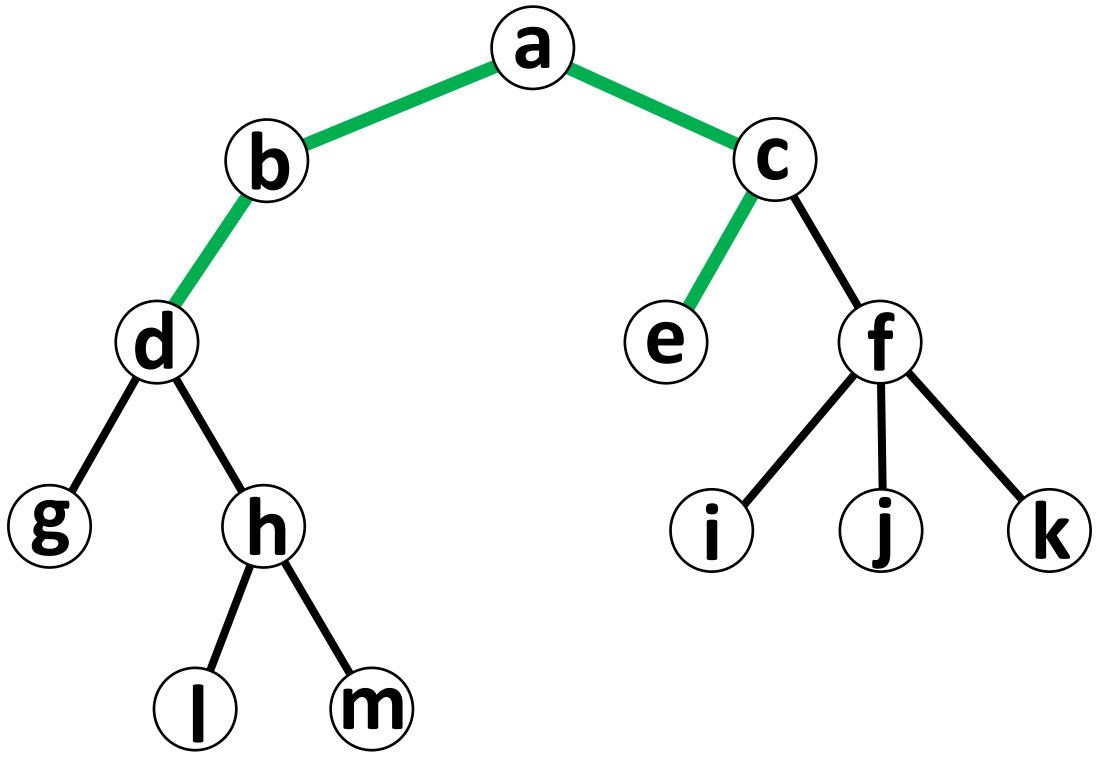
Tree Traversals



Breadth First: Start at a node. Visit all of its children, then all of its grandchildren, then great-grandchildren,…

Order visited: a, b, c, d, e, f, g, h, i, j, k, l ,m

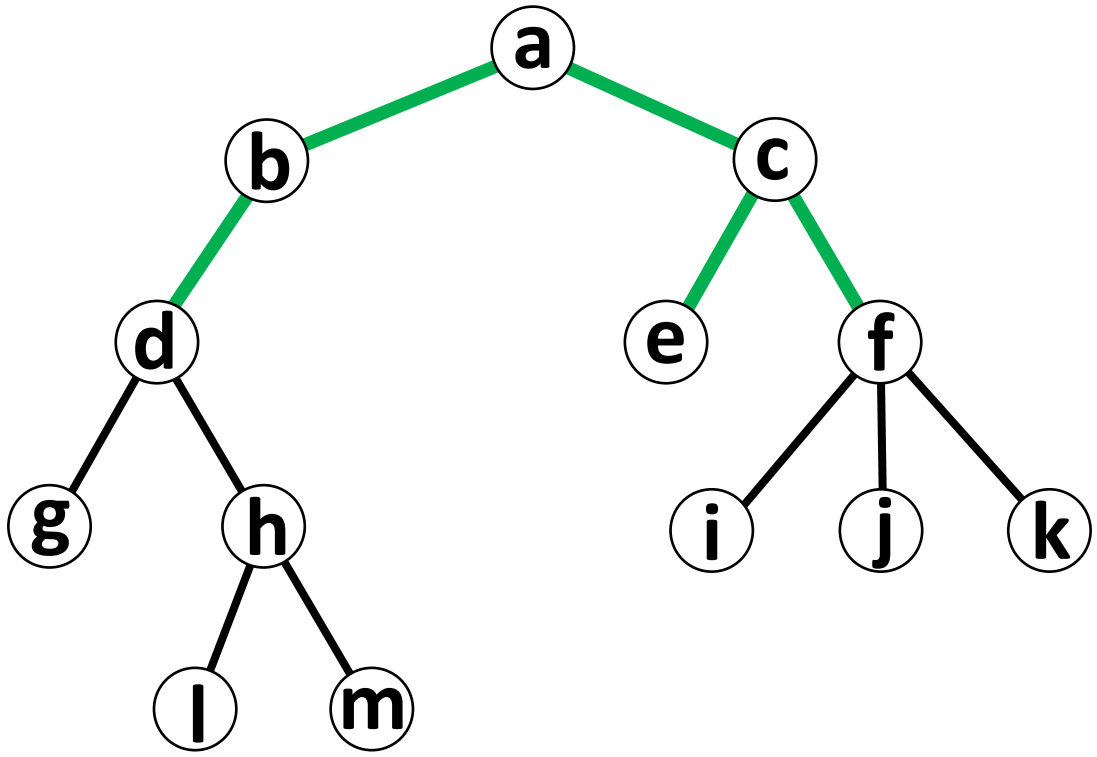
Tree Traversals



Breadth First: Start at a node. Visit all of its children, then all of its grandchildren, then great-grandchildren,…

Order visited: a, b, c, d, e, f, g, h, i, j, k, l ,m

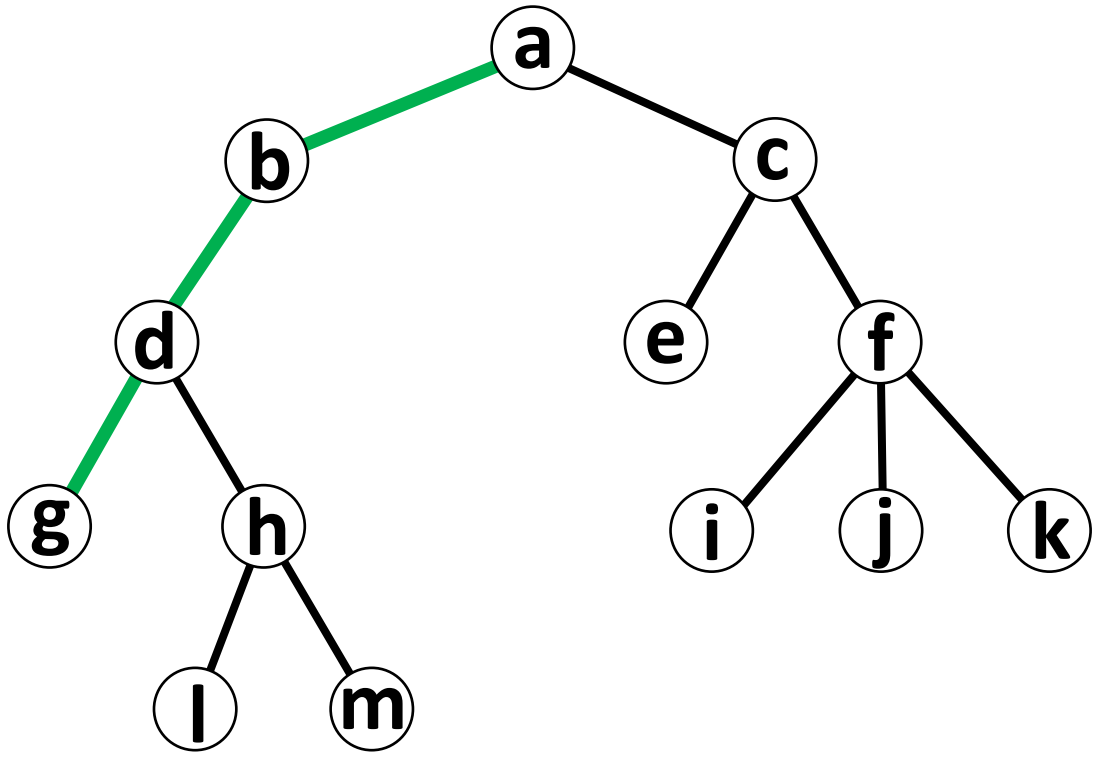
Tree Traversals



Breadth First: Start at a node. Visit all of its children, then all of its grandchildren, then great-grandchildren,…

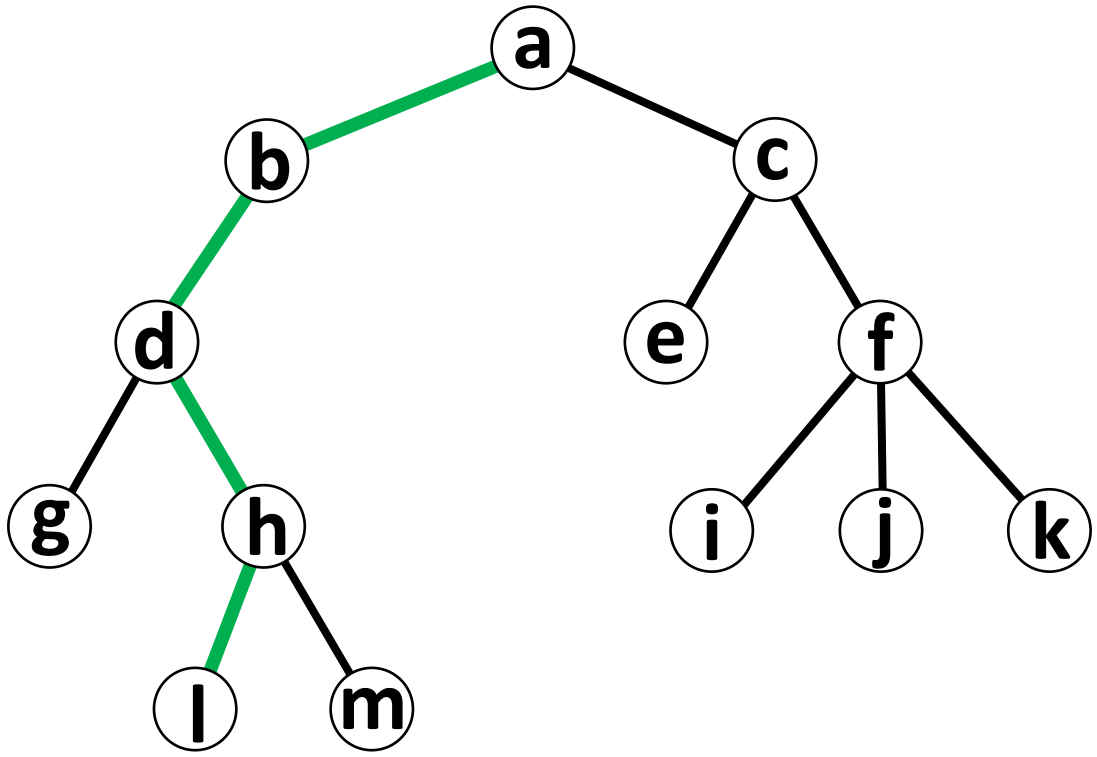
Order visited: a, b, c, d, e, f, g, h, i, j, k, l ,m

Tree Traversals



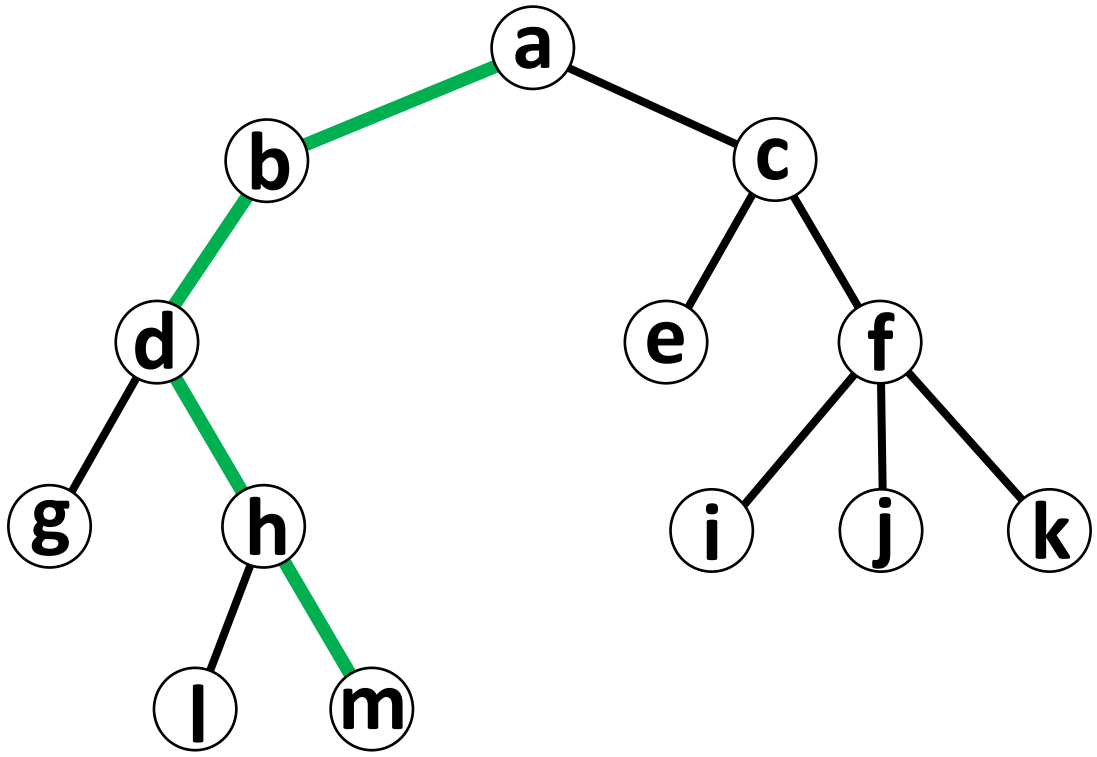
Depth First: Start at a node. Follow children until leaf is reached. Back track until unvisited child is available. Repeat…

Tree Traversals



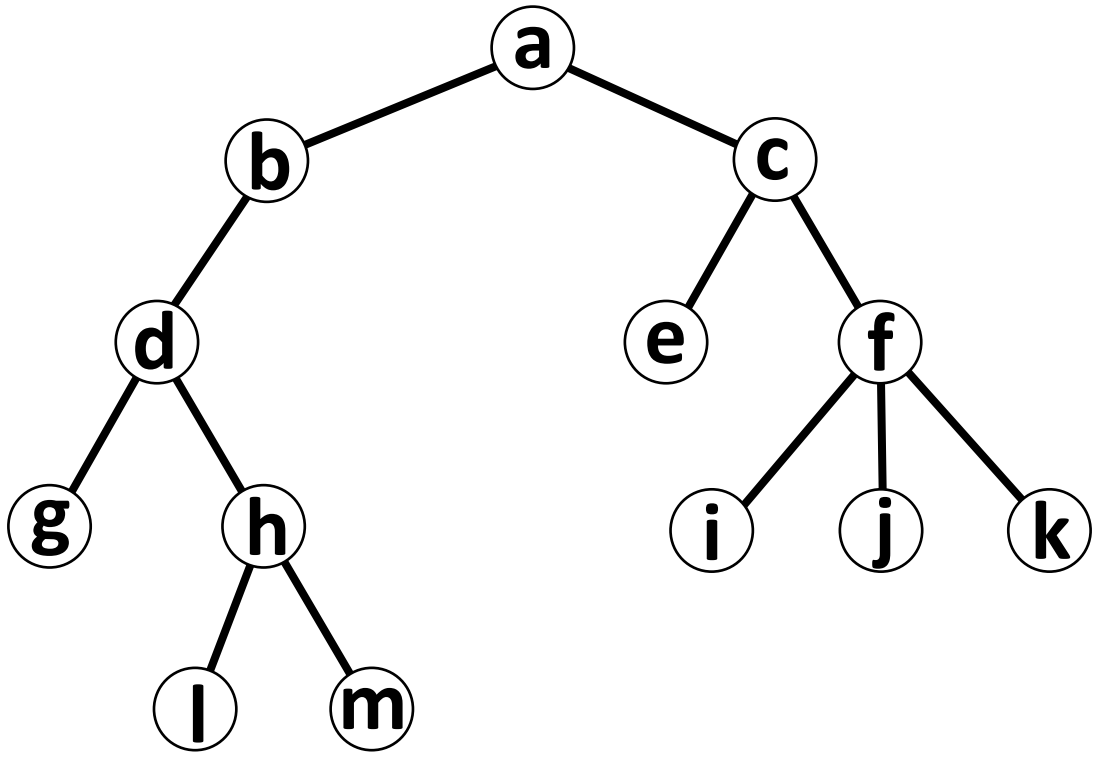
Depth First: Start at a node. Follow children until leaf is reached. Back track until unvisited child is available. Repeat…

Tree Traversals



Depth First: Start at a node. Follow children until leaf is reached. Back track until unvisited child is available. Repeat…

Tree Traversals

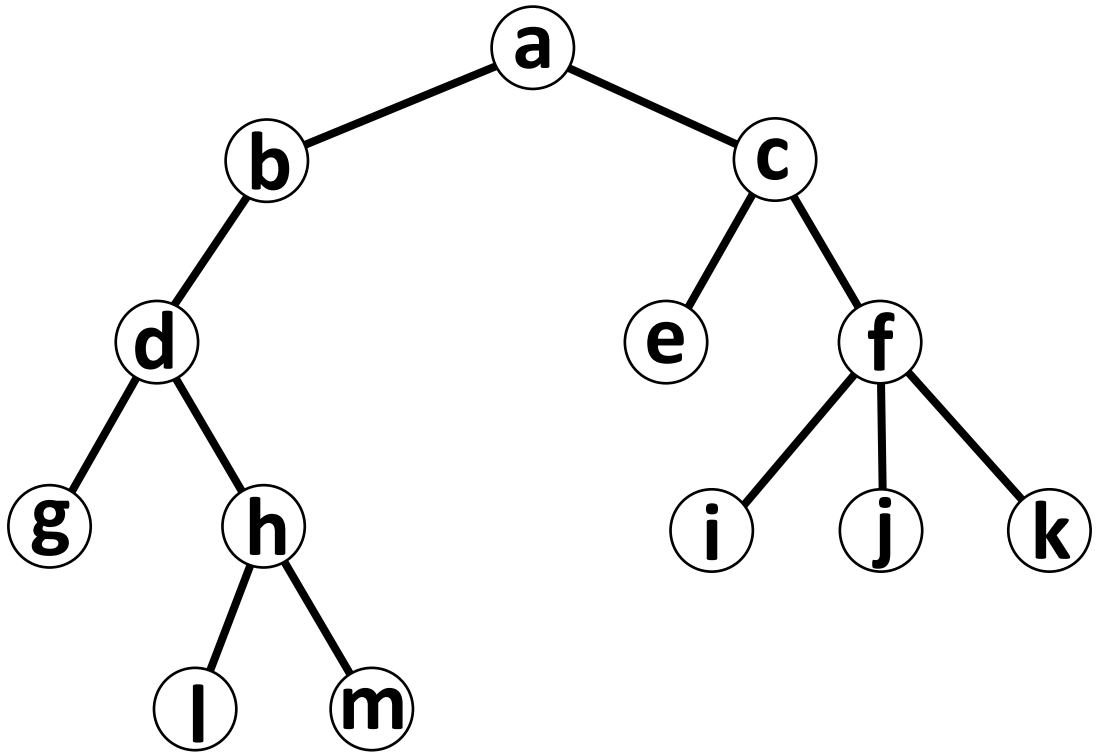


Depth First: Start at a node. Follow children until leaf is reached. Back track until unvisited child is available. Repeat…

```
preorder(Vertex v):
    if v not null:
        print(v.value())
        for u in child(v):
            preorder(u)
```

```
postorder(Vertex v):
    if v not null:
        for u in child(v):
            postorder(u)
        print(v.value())
```

Tree Traversals



Depth First: Start at a node. Follow children until leaf is reached. Back track until unvisited child is available. Repeat…

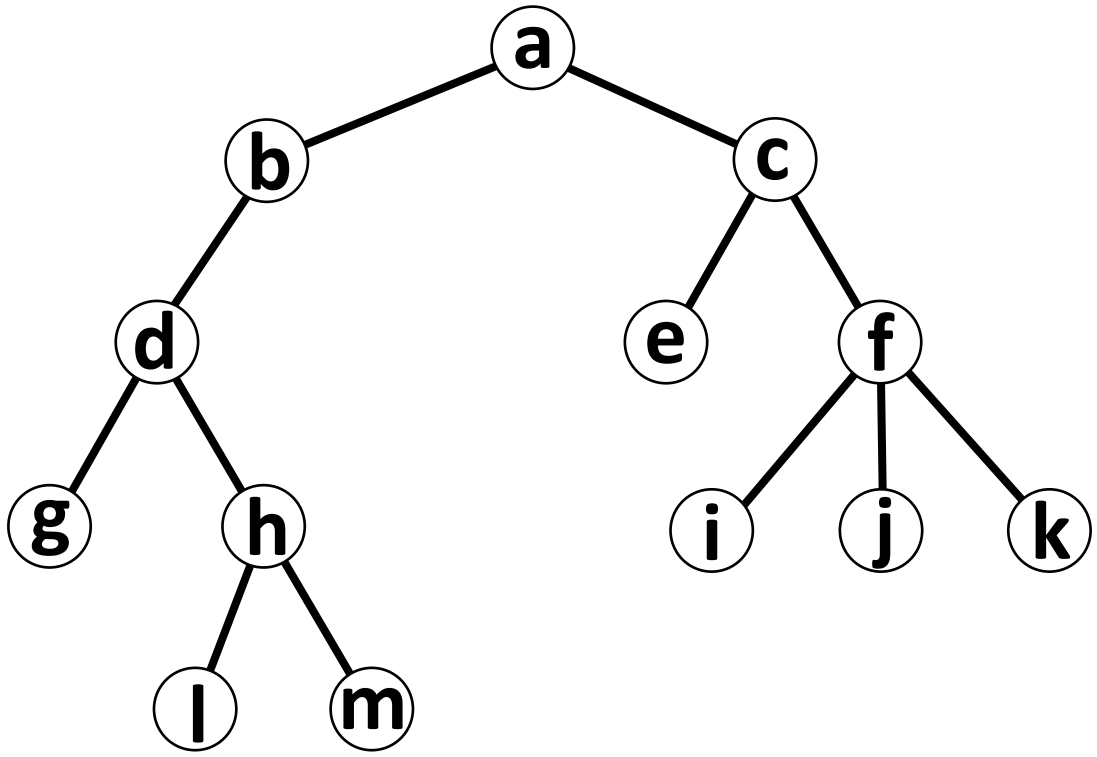
1. Process parent
2. Process children

```
preorder(Vertex v):
    if v not null:
        print(v.value())
        for u in child(v):
            preorder(u)
```

1. Process children
2. Process parent

```
postorder(Vertex v):
    if v not null:
        for u in child(v):
            postorder(u)
        print(v.value())
```

Tree Traversals



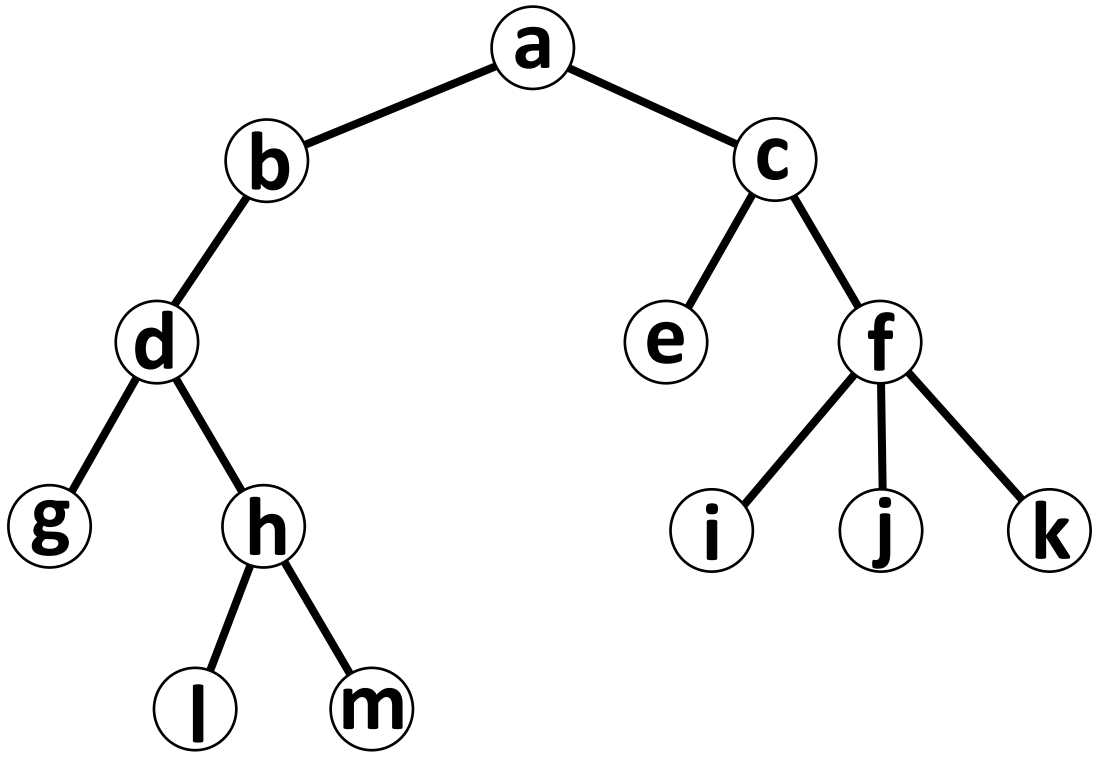
Depth First: Start at a node. Follow children until leaf is reached. Back track until unvisited child is available. Repeat…

```
preorder(Vertex v):
    if v not null:
        print(v.value())
        for u in child(v):
            preorder(u)
```

```
postorder(Vertex v):
    if v not null:
        for u in child(v):
            postorder(u)
        print(v.value())
```

Preorder visited: a, b, d, g, h, l, m, c, e, f, i, j, k

Tree Traversals



Depth First: Start at a node. Follow children until leaf is reached. Back track until unvisited child is available. Repeat…

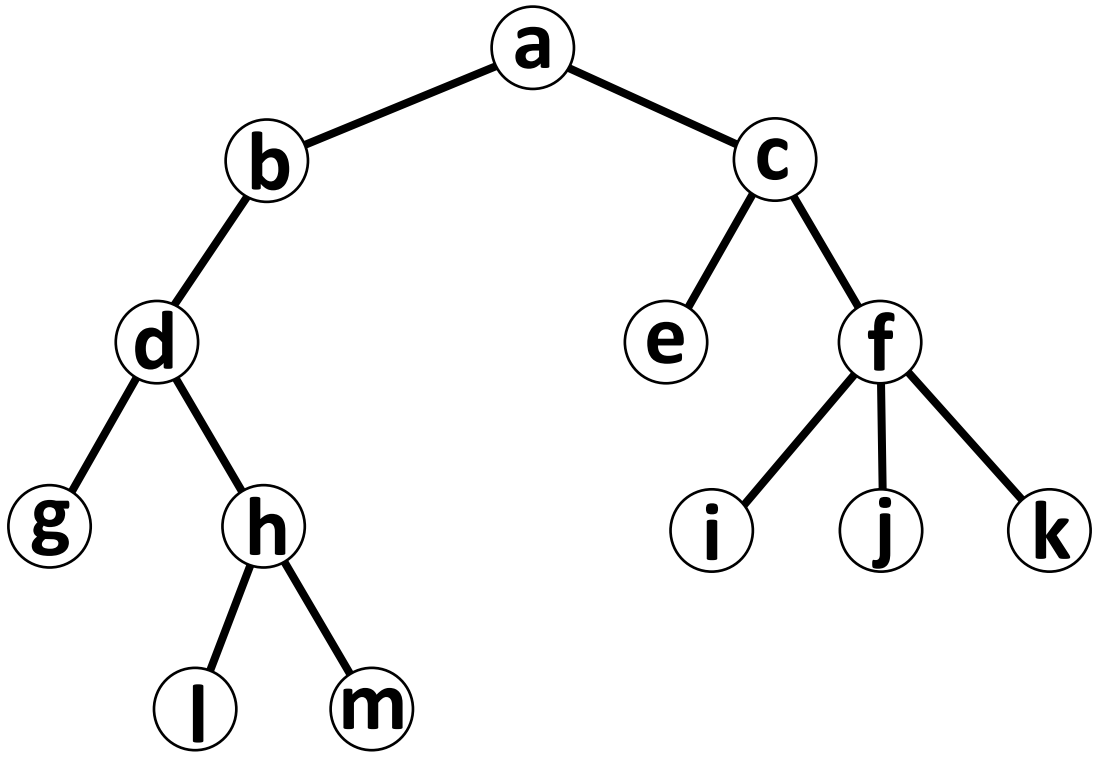
```
preorder(Vertex v):
    if v not null:
        print(v.value())
        for u in child(v):
            preorder(u)
```

```
postorder(Vertex v):
    if v not null:
        for u in child(v):
            postorder(u)
        print(v.value())
```

Preorder visited: a, b, d, g, h, l, m, c, e, f, i, j, k

Postorder visited: g, l, m, h, d, b, e, i, j, k, f, c, a

Tree Traversals



Depth First: Start at a node. Follow children until leaf is reached. Back track until unvisited child is available. Repeat…

```
preorder(Vertex v):
    if v not null:
        print(v.value())
        for u in child(v):
            preorder(u)
```

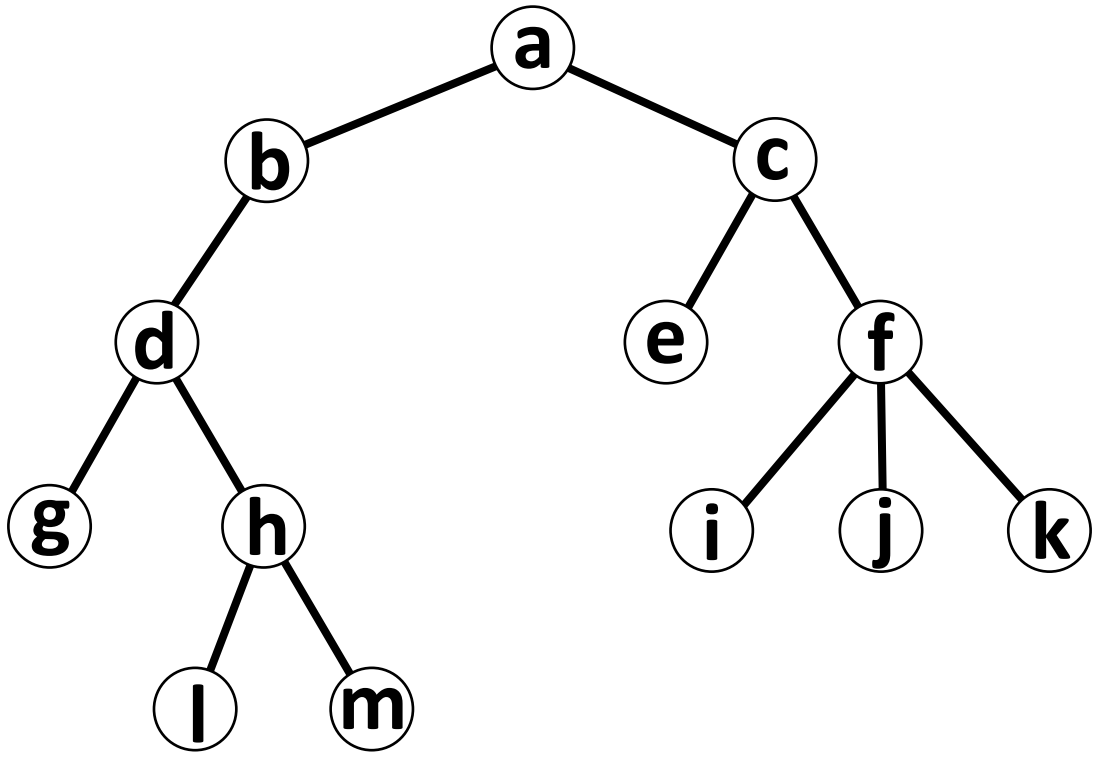
Which one do we want?

```
postorder(Vertex v):
    if v not null:
        for u in child(v):
            postorder(u)
        print(v.value())
```

Preorder visited: a, b, d, g, h, l, m, c, e, f, i, j, k

Postorder visited: g, l, m, h, d, b, e, i, j, k, f, c, a

Tree Traversals



Depth First: Start at a node. Follow children until leaf is reached. Back track until unvisited child is available. Repeat…

```
preorder(Vertex v):
    if v not null:
        print(v.value())
        for u in child(v):
            preorder(u)
```

Which one do we want?

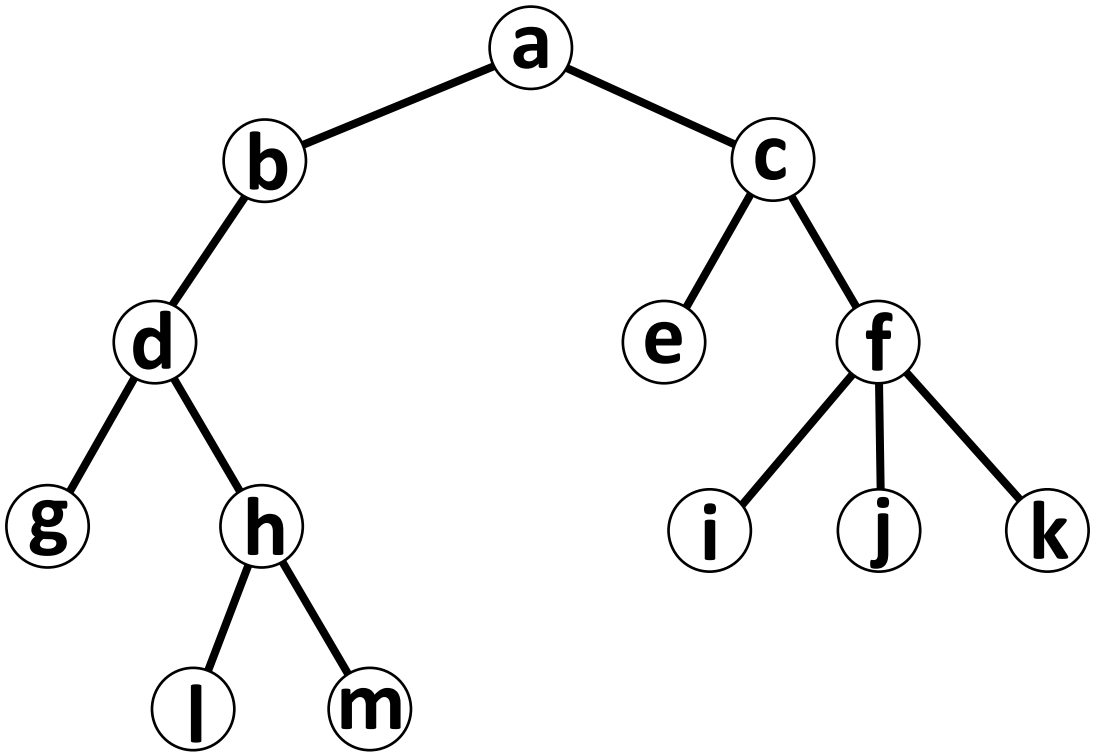
```
postorder(Vertex v):
    if v not null:
        for u in child(v):
            postorder(u)
        print(v.value())
```

Preorder visited: a, b, d, g, h, l, m, c, e, f, i, j, k

Postorder visited: g, l, m, h, d, b, e, i, j, k, f, c, a

```
min_vc(Vertex v):
```

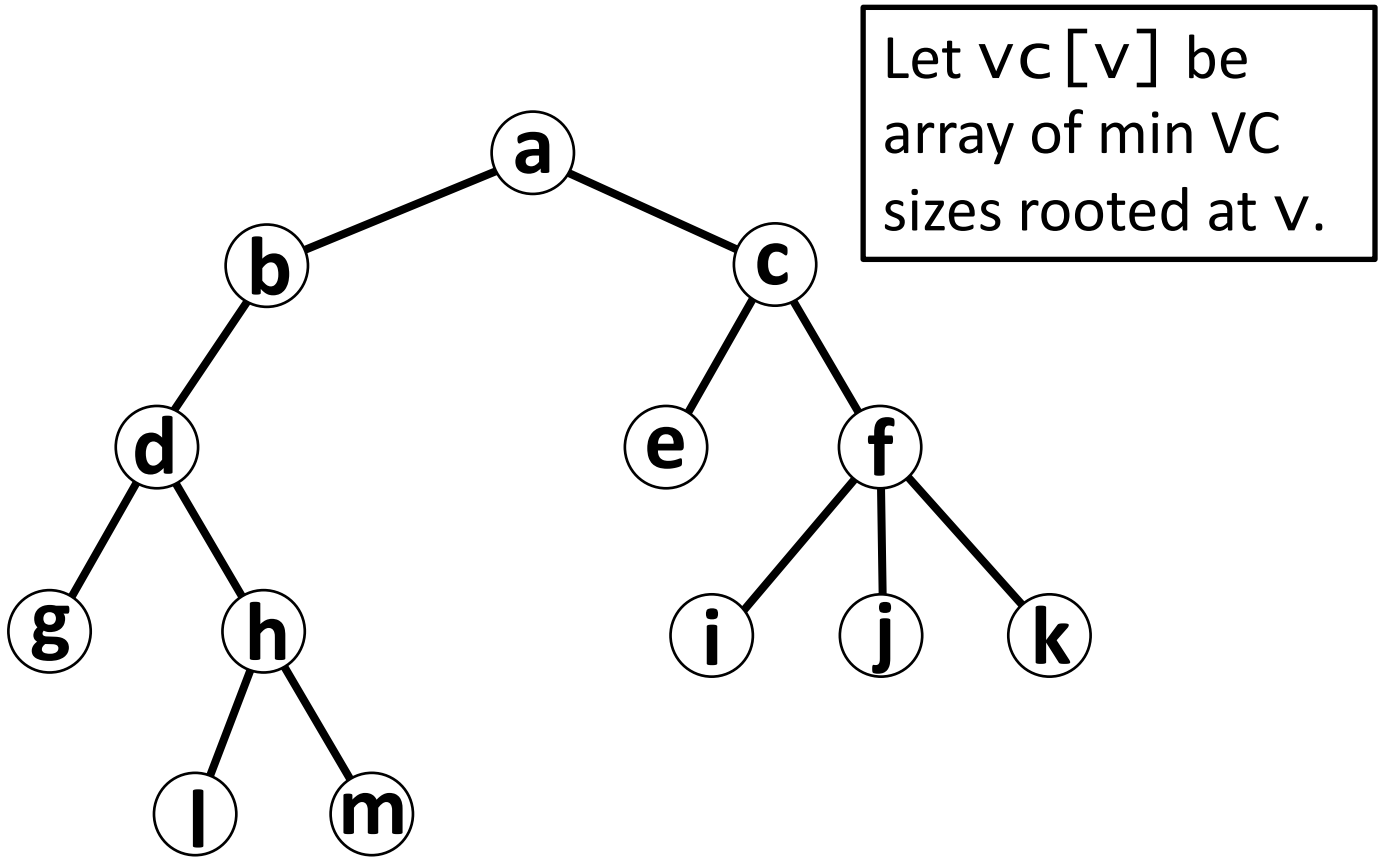
Vertex Cover in Trees



$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u),\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

```
min_vc(Vertex v):
```

Vertex Cover in Trees



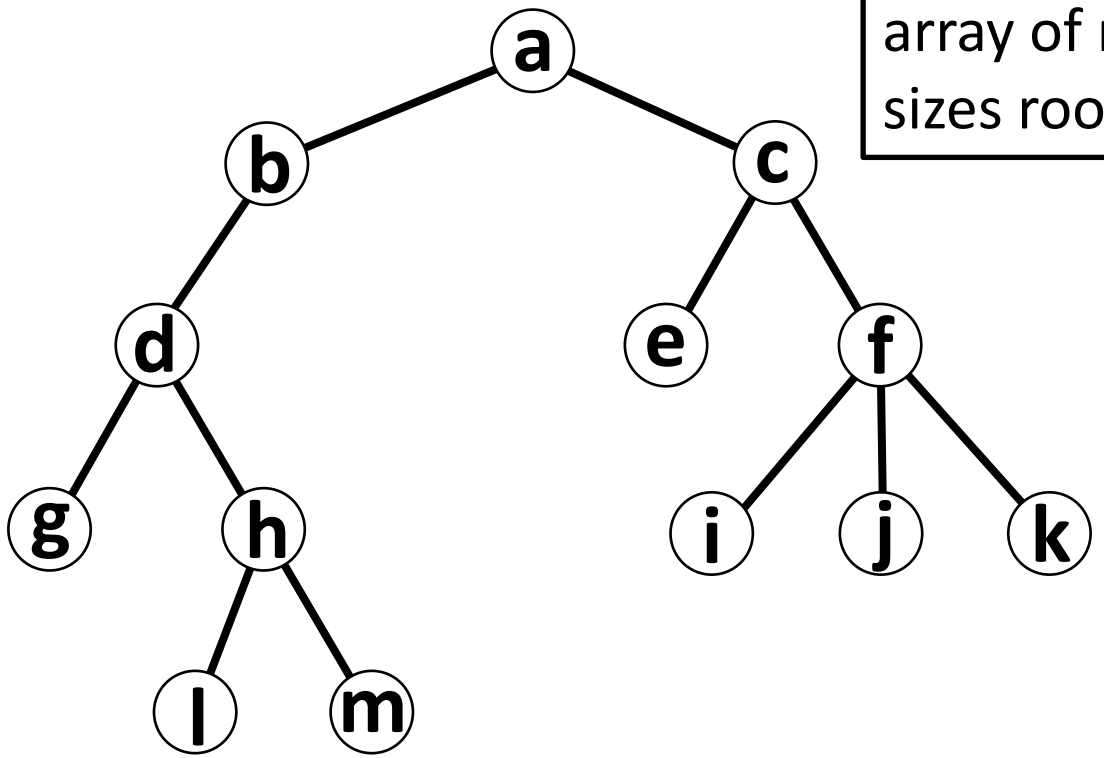
Let `vc[v]` be array of min VC sizes rooted at `v`.

$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u),\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

Vertex Cover in Trees

```
min_vc(Vertex v):
    if v is leaf:

    else:
```

Let `vc[v]` be array of min VC sizes rooted at `v`.

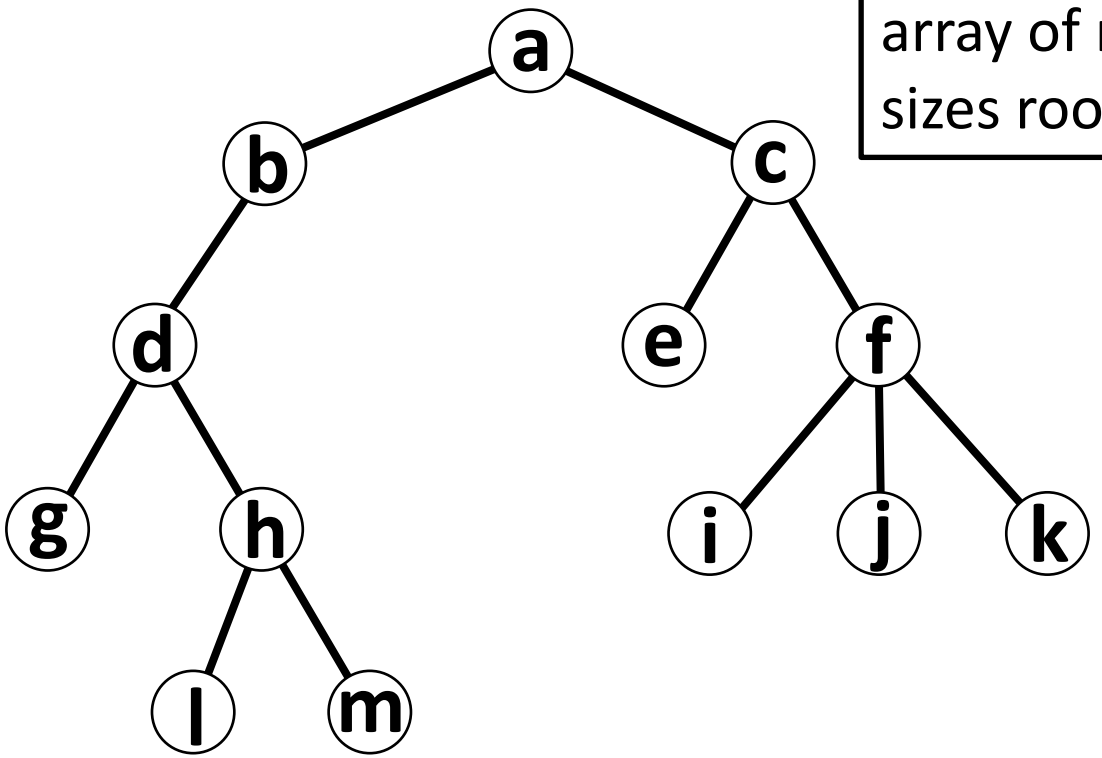


$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u),\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

Vertex Cover in Trees

```
min_vc(Vertex v):
    if v is leaf:
        vc[v] = 0
    else:
```

Let `vc[v]` be array of min VC sizes rooted at `v`.

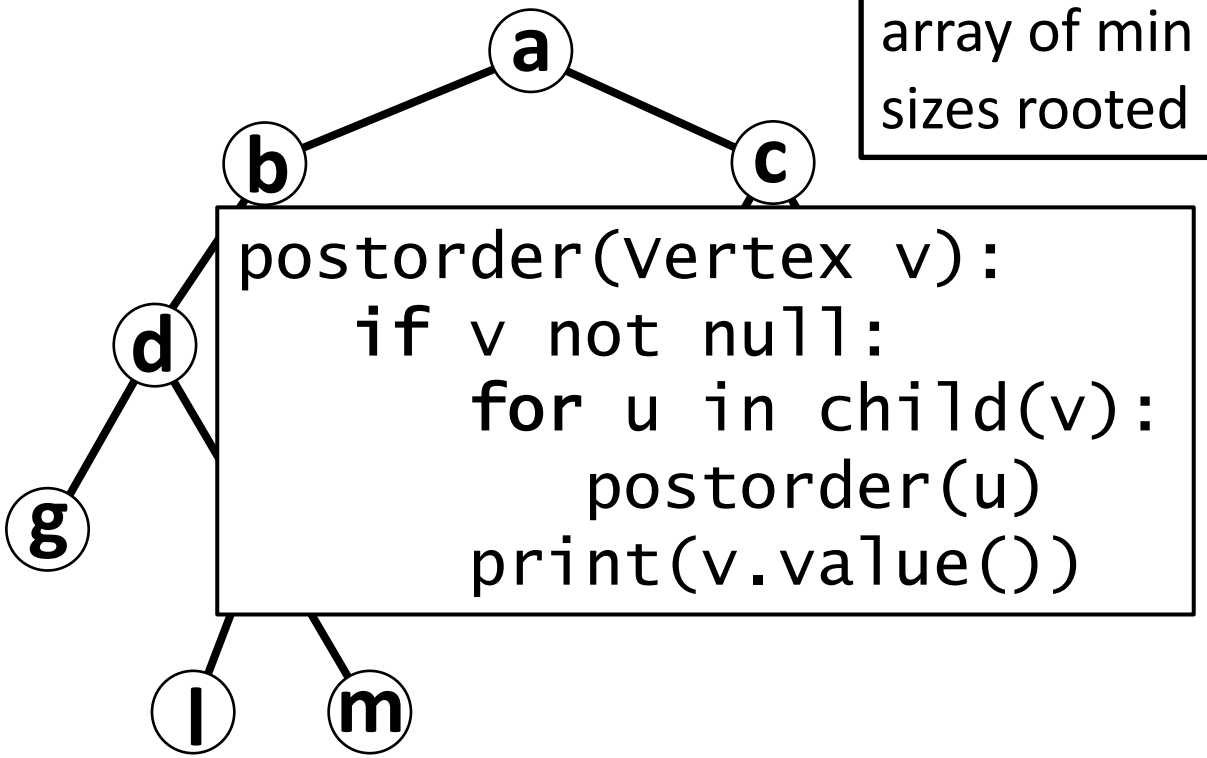


$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u),\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

Vertex Cover in Trees

```
min_vc(Vertex v):
   if v is leaf:
      vc[v] = 0
else:
      // calculate descendants
```

Let `vc[v]` be array of min VC sizes rooted at `v`.

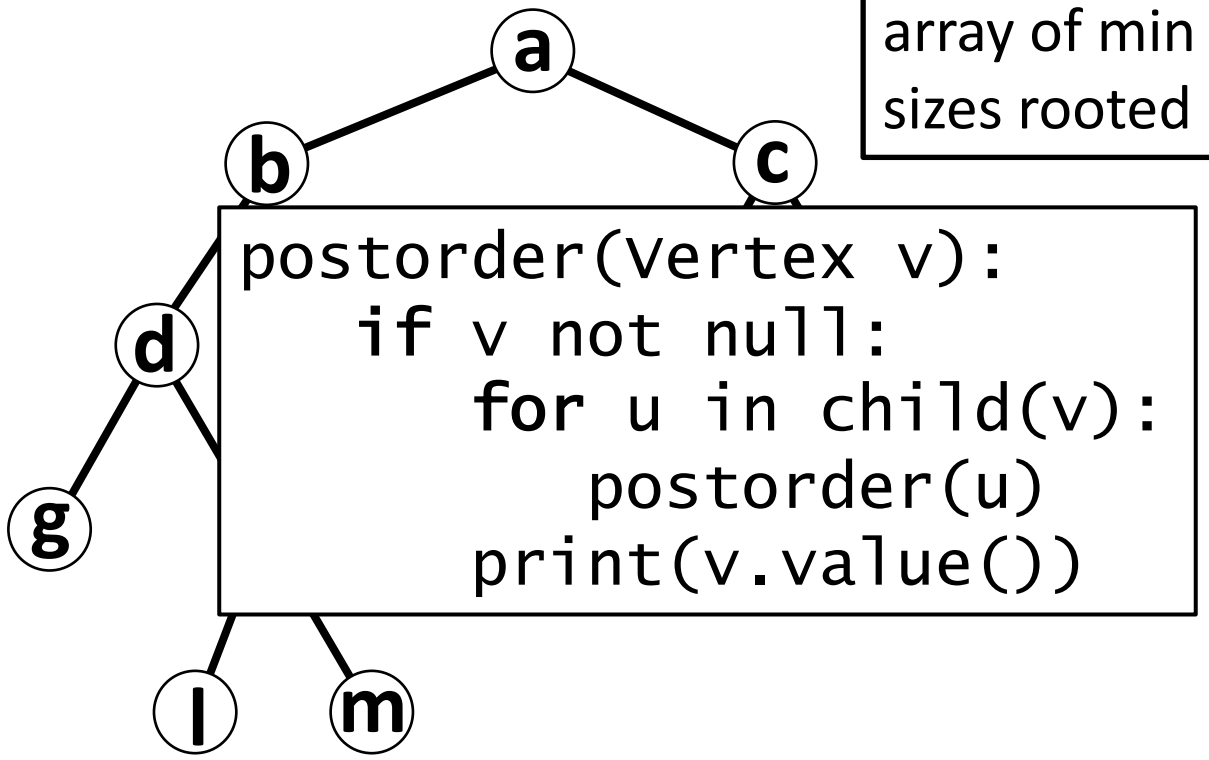



```
postorder(Vertex v):
   if v not null:
      for u in child(v):
         postorder(u)
      print(v.value())
```

$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u),\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

Vertex Cover in Trees

Let `vc[v]` be array of min VC sizes rooted at `v`.



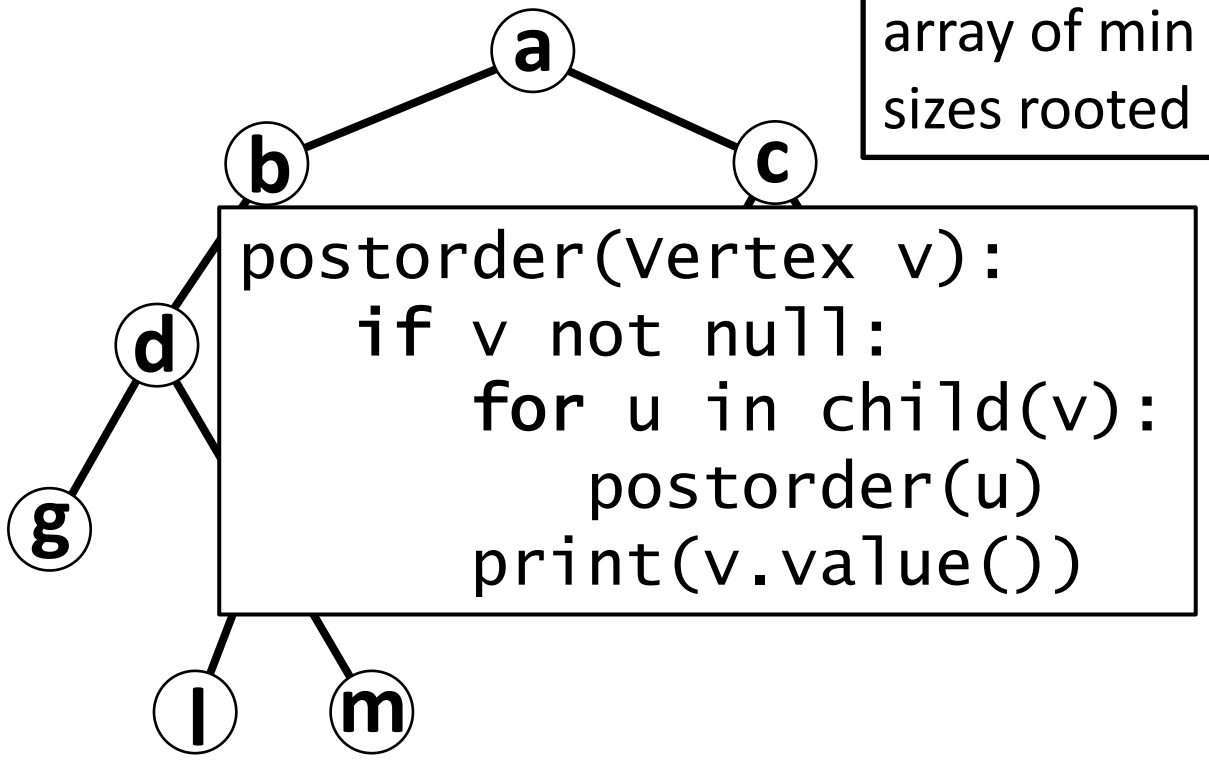

```
postorder(Vertex v):
   if v not null:
      for u in child(v):
         postorder(u)
      print(v.value())
```

$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u),\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

```
min_vc(Vertex v):
   if v is leaf:
      vc[v] = 0
   else:
      // calculate descendants

      // min VC includes v

      // min VC excludes v

      vc[v] = min(???, ???)
```

Vertex Cover in Trees

Let `vc[v]` be array of min VC sizes rooted at v.



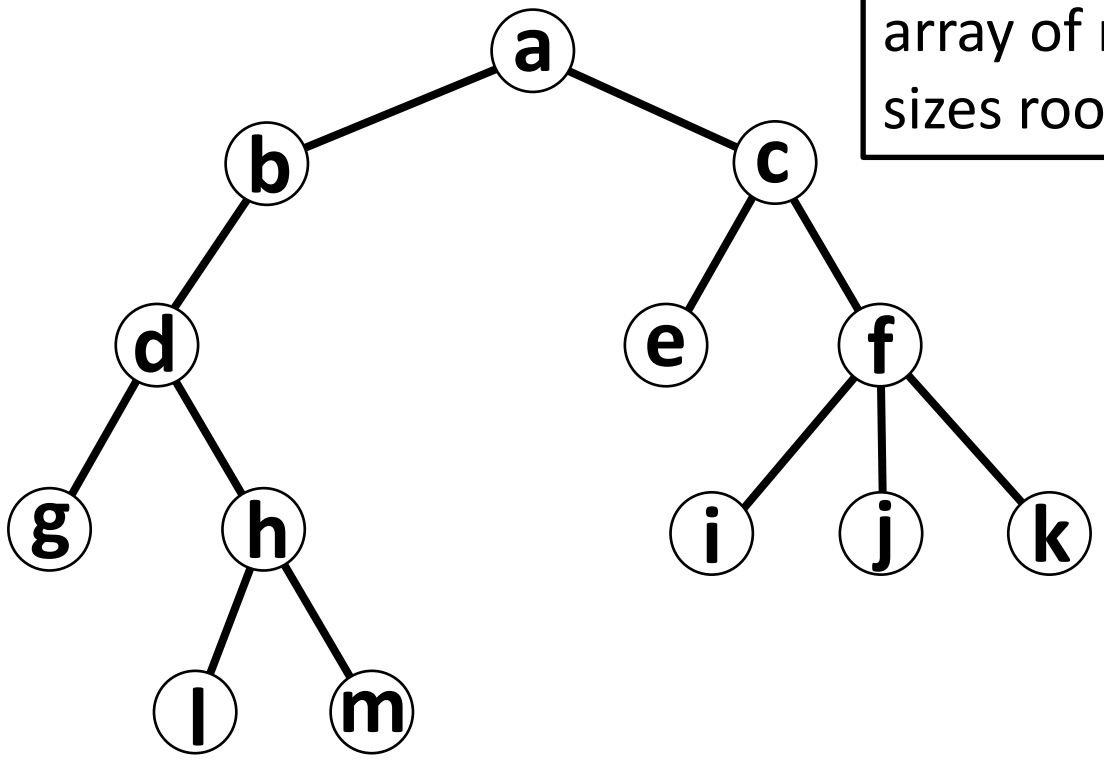

```
postorder(Vertex v):
   if v not null:
      for u in child(v):
         postorder(u)
      print(v.value())
```

```
min_vc(Vertex v):
   if v is leaf:
      vc[v] = 0
else:
      // calculate descendants
      for u in child(v):
         min_vc(u)
      // min VC includes v
      // min VC excludes v
      vc[v] = min(???, ???)
```

$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u),\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

Vertex Cover in Trees

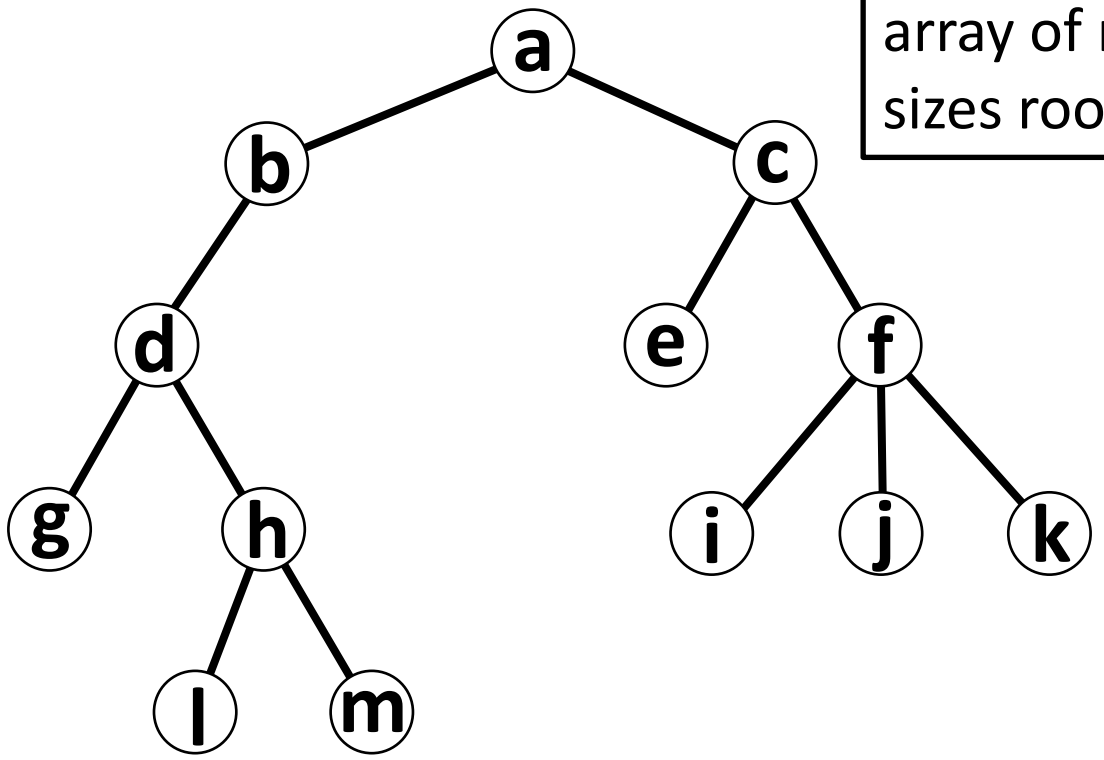
Let `vc[v]` be array of min VC sizes rooted at v.



$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u),\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

```
min_vc(Vertex v):
    if v is leaf:
        vc[v] = 0
    else:
        // calculate descendants
        for u in child(v):
            min_vc(u)

        // min VC includes v
        inc = 1



        // min VC excludes v



        vc[v] = min(???, ???)
```

Vertex Cover in Trees

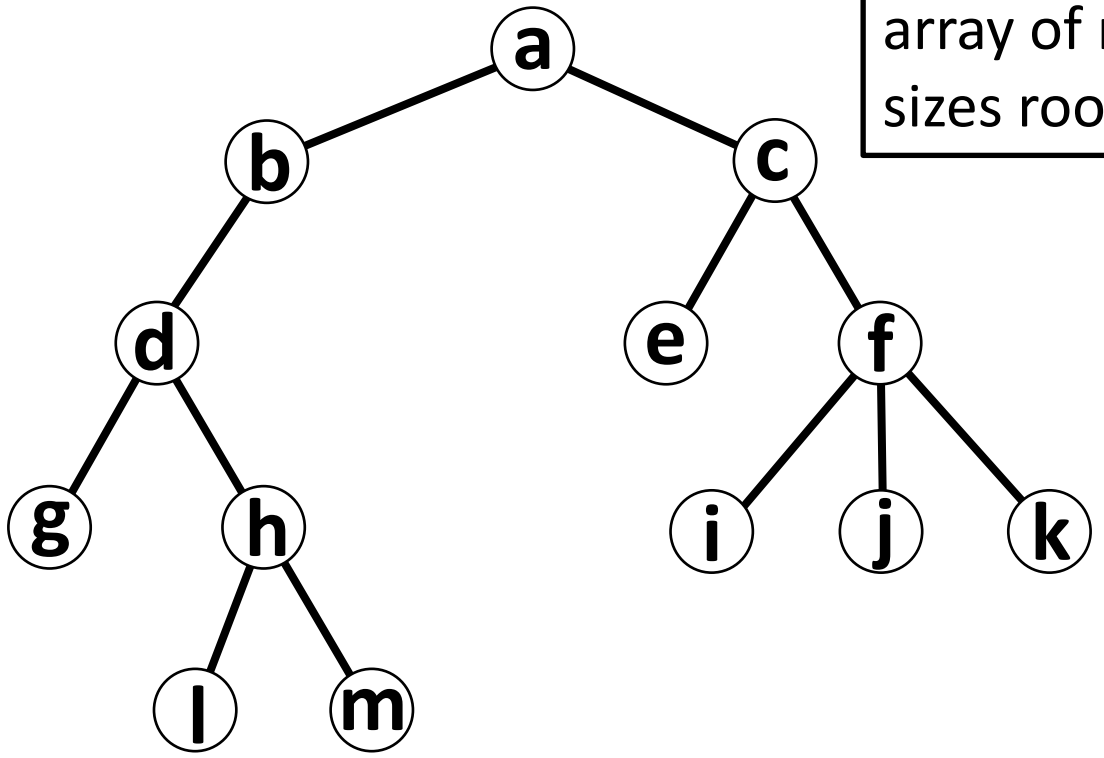


Let `vc[v]` be array of min VC sizes rooted at v.

```
min_vc(Vertex v):
    if v is leaf:
        vc[v] = 0
    else:
        // calculate descendants
        for u in child(v):
            min_vc(u)

        // min VC includes v
        inc = 1
        for u in child(v):
            inc += vc[u]

        // min VC excludes v


        vc[v] = min(???, ???)
```

$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u),\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

Vertex Cover in Trees



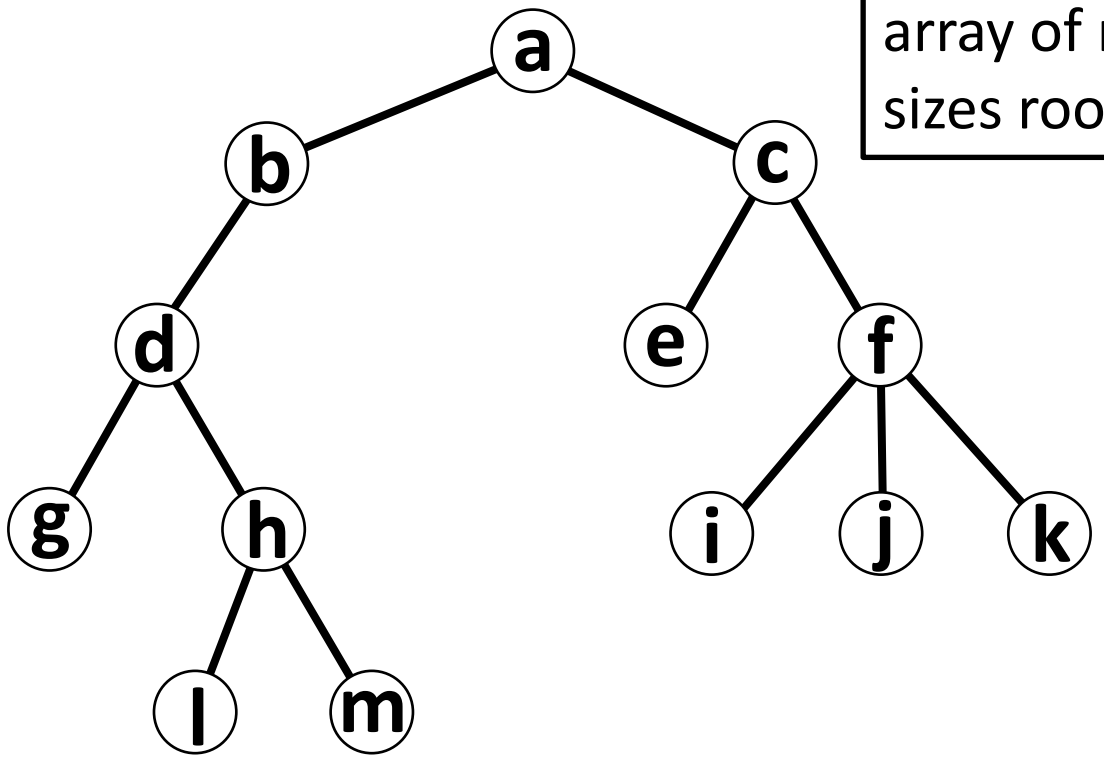
Let `vc[v]` be array of min VC sizes rooted at v.

$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u),\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

```
min_vc(Vertex v):
    if v is leaf:
        vc[v] = 0
    else:
        // calculate descendants
        for u in child(v):
            min_vc(u)

        // min VC includes v
        inc = 1
        for u in child(v):
            inc += vc[u]

        // min VC excludes v
        exc = |child(v)|

        vc[v] = min(???, ???)
```

Vertex Cover in Trees

Let vc[v] be array of min VC sizes rooted at v.

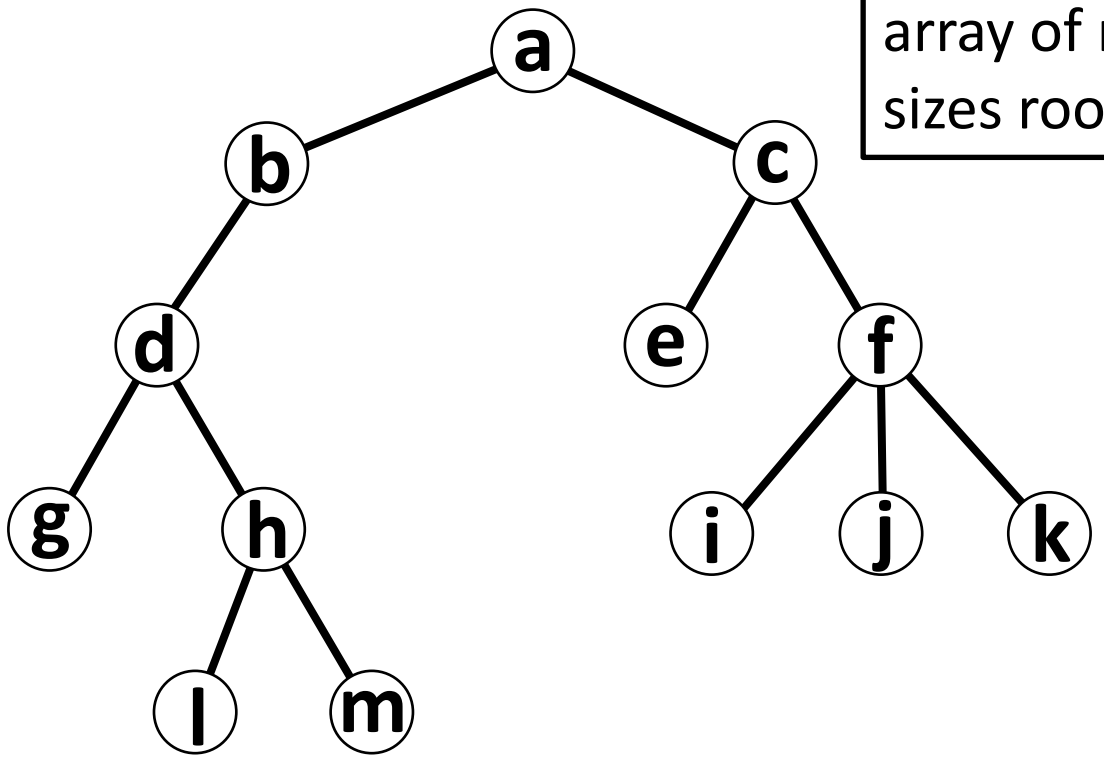


$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u),\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

```
min_vc(Vertex v):
    if v is leaf:
        vc[v] = 0
    else:
        // calculate descendants
        for u in child(v):
            min_vc(u)

        // min VC includes v
        inc = 1
        for u in child(v):
            inc += vc[u]

        // min VC excludes v
        exc = |child(v)|
        for u in child(v):
            for w in child(u):
                exc += vc[w]

        vc[v] = min(???, ???)
```

Vertex Cover in Trees

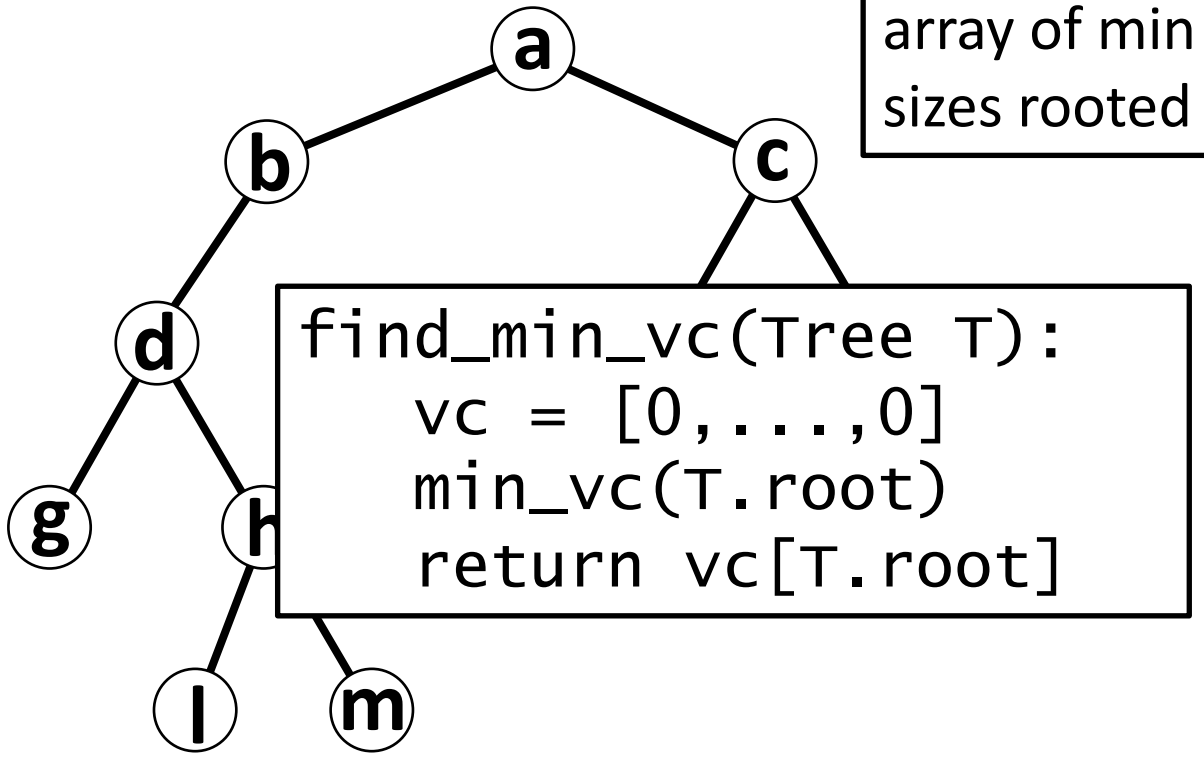
Let `vc[v]` be array of min VC sizes rooted at v.



$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u),\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

```
min_vc(Vertex v):
    if v is leaf:
        vc[v] = 0
    else:
        // calculate descendants
        for u in child(v):
            min_vc(u)

        // min VC includes v
        inc = 1
        for u in child(v):
            inc += vc[u]

        // min VC excludes v
        exc = |child(v)|
        for u in child(v):
            for w in child(u):
                exc += vc[w]

        vc[v] = min(inc, exc)
```

Vertex Cover in Trees



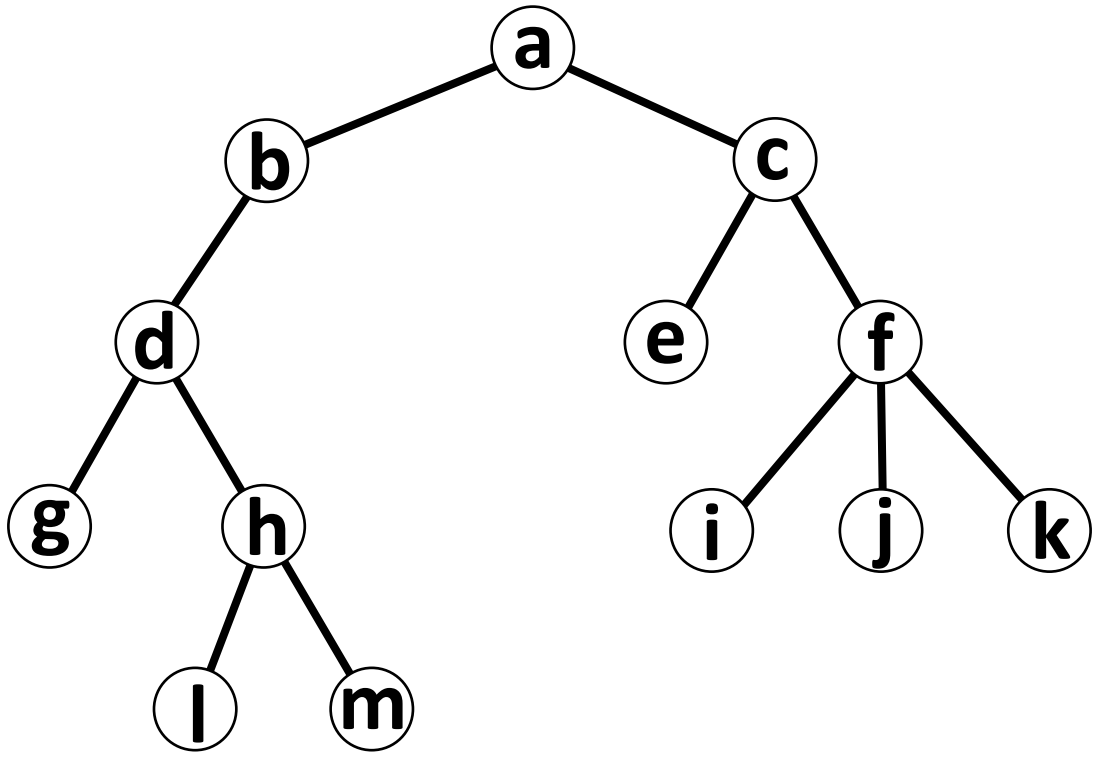
Let `vc[v]` be array of min VC sizes rooted at v.

```
find_min_vc(Tree T):
    vc = [0,...,0]
    min_vc(T.root)
    return vc[T.root]
```

$$f(v) = \min\left\{1 + \sum_{u \in \text{child}(v)} f(u),\ |\text{child}(v)| + \sum_{w \in \text{grandchild}(v)} f(w)\right\}$$

```
min_vc(Vertex v):
    if v is leaf:
        vc[v] = 0
    else:
        // calculate descendants
        for u in child(v):
            min_vc(u)

        // min VC includes v
        inc = 1
        for u in child(v):
            inc += vc[u]

        // min VC excludes v
        exc = |child(v)|
        for u in child(v):
            for w in child(u):
                exc += vc[w]

        vc[v] = min(inc, exc)
```

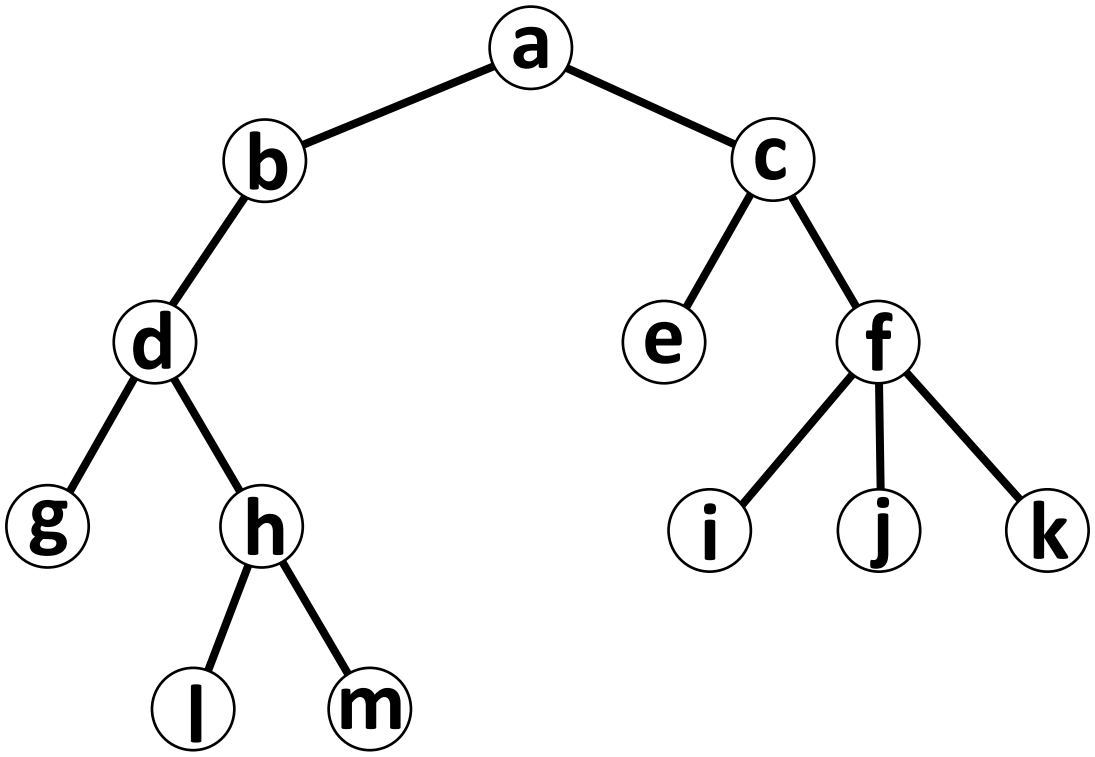
Vertex Cover in Trees



Running Time: ??

```
min_vc(Vertex v):
    if v is leaf:
        vc[v] = 0
    else:
        // calculate descendants
        for u in child(v):
            min_vc(u)

        // min VC includes v
        inc = 1
        for u in child(v):
            inc += vc[u]

        // min VC excludes v
        exc = |child(v)|
        for u in child(v):
            for w in child(u):
                exc += vc[w]

        vc[v] = min(inc, exc)
```

Vertex Cover in Trees

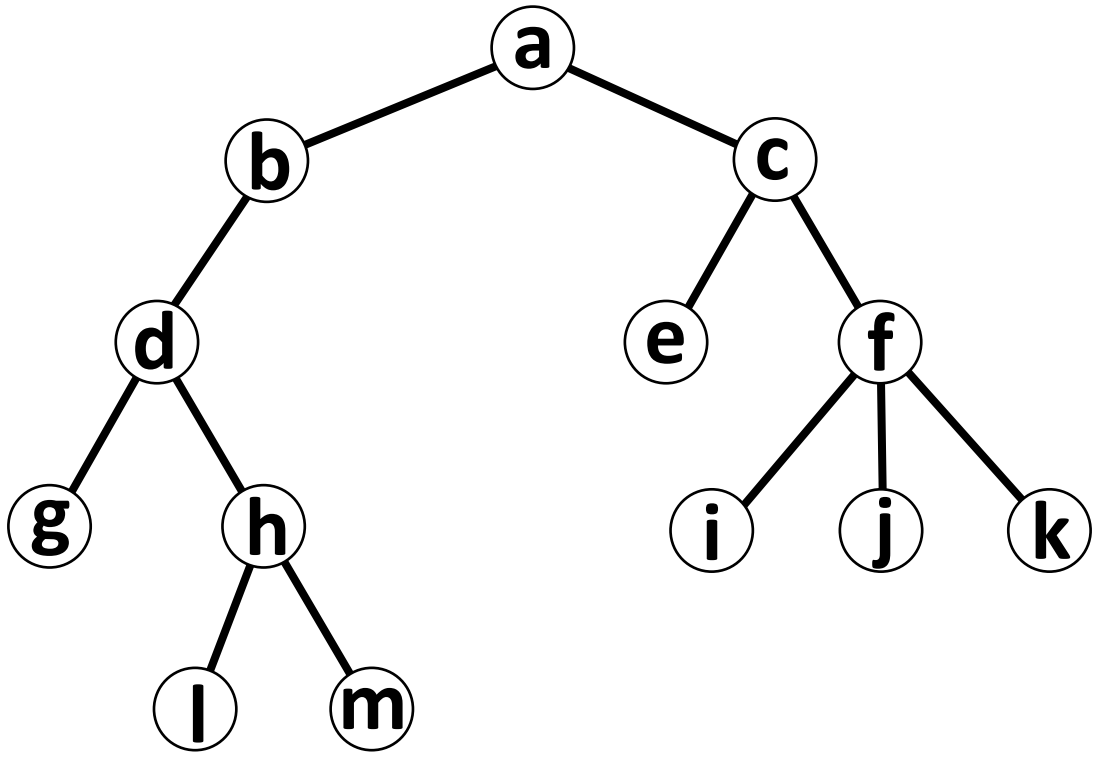


Running Time: ??

```
min_vc(Vertex v):
   if v is leaf:
      vc[v] = 0
   else:
      // calculate descendants
      for u in child(v):
         min_vc(u)

      // min VC includes v
      inc = 1
      for u in child(v):
         inc += vc[u]

      // min VC excludes v
      exc = |child(v)|
      for u in child(v):
         for w in child(u):
            exc += vc[w]

      vc[v] = min(inc, exc)
```

??

Vertex Cover in Trees

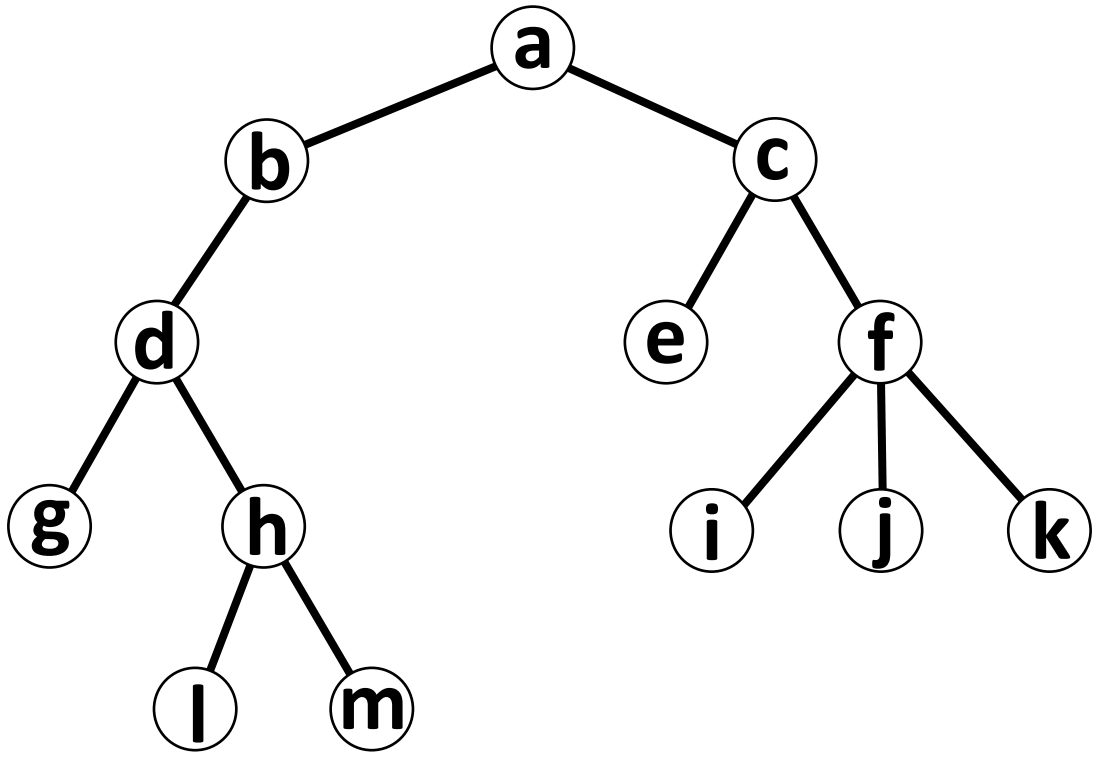


Running Time: ??

```
min_vc(Vertex v):
    if v is leaf:
        vc[v] = 0
    else:
        // calculate descendants
        for u in child(v):
            min_vc(u)

        // min VC includes v
        inc = 1
        for u in child(v):
            inc += vc[u]

        // min VC excludes v
        exc = |child(v)|
        for u in child(v):
            for w in child(u):
                exc += vc[w]

        vc[v] = min(inc, exc)
```

Visit each node once

Vertex Cover in Trees



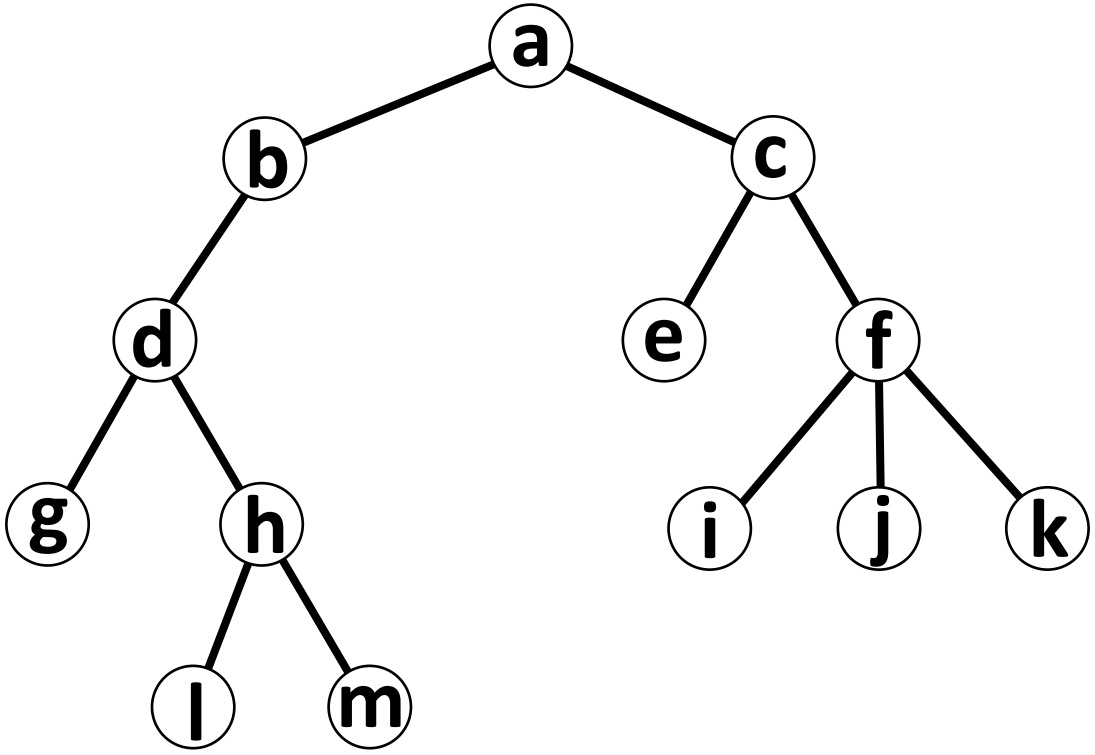
Running Time: ??

```
min_vc(Vertex v):
    if v is leaf:
        vc[v] = 0
    else:
        // calculate descendants
        for u in child(v):
            min_vc(u)

        // min VC includes v
        inc = 1
        for u in child(v):
            inc += vc[u]

        // min VC excludes v
        exc = |child(v)|
        for u in child(v):
            for w in child(u):
                exc += vc[w]

        vc[v] = min(inc, exc)
```

Visit each node once

??

Vertex Cover in Trees



Running Time: ??

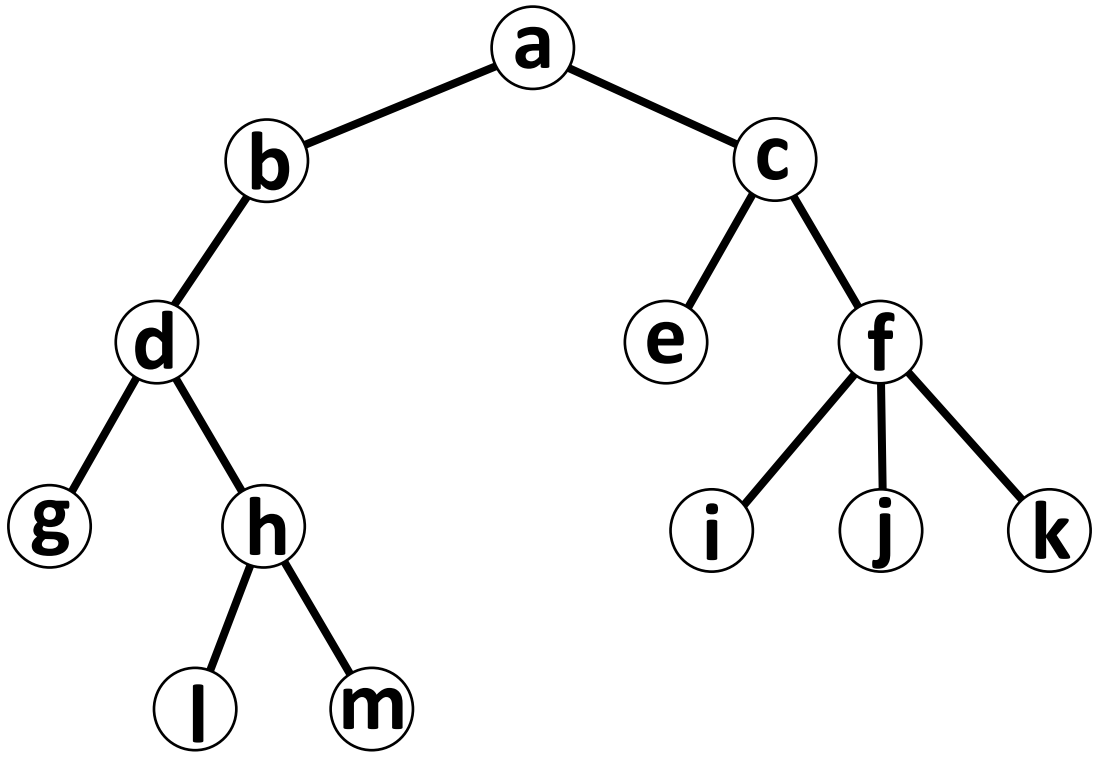
```
min_vc(Vertex v):
    if v is leaf:
        vc[v] = 0
    else:
        // calculate descendants
        for u in child(v):
            min_vc(u)

        // min VC includes v
        inc = 1
        for u in child(v):
            inc += vc[u]

        // min VC excludes v
        exc = |child(v)|
        for u in child(v):
            for w in child(u):
                exc += vc[w]

        vc[v] = min(inc, exc)
```

Visit each node once

Visit each node once

??

Vertex Cover in Trees



Running Time: ??

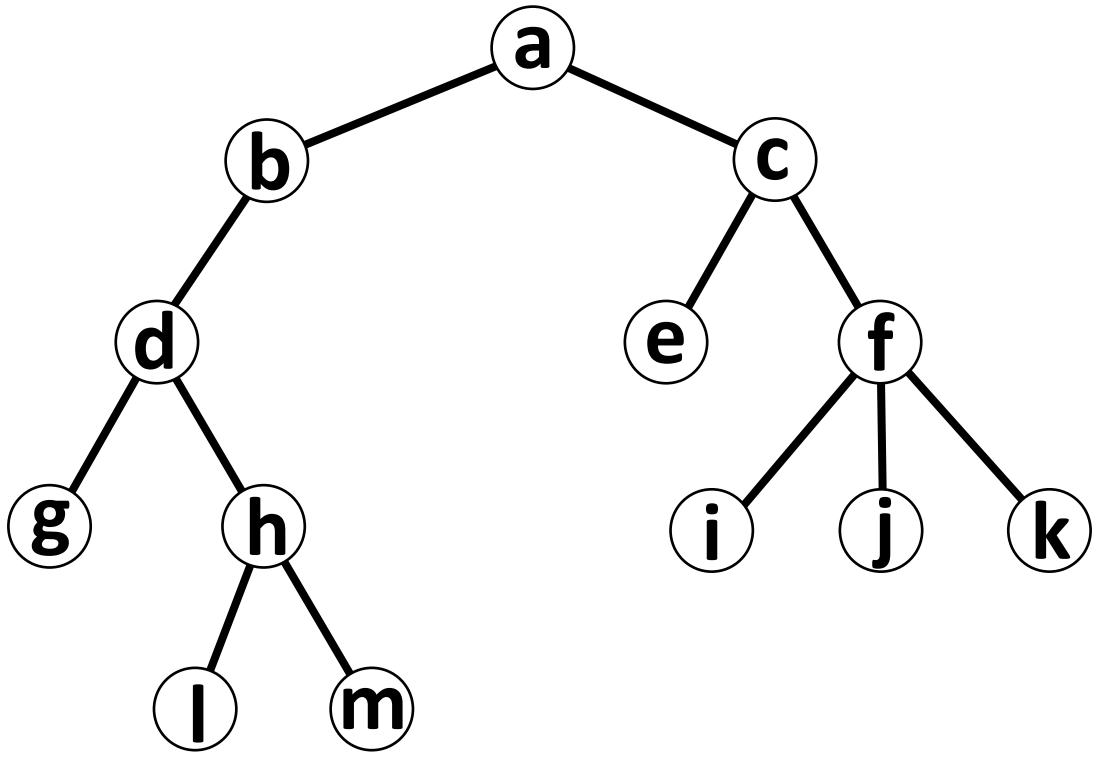
```
min_vc(Vertex v):
    if v is leaf:
        vc[v] = 0
    else:
        // calculate descendants
        for u in child(v):
            min_vc(u)

        // min VC includes v
        inc = 1
        for u in child(v):
            inc += vc[u]

        // min VC excludes v
        exc = |child(v)|
        for u in child(v):
            for w in child(u):
                exc += vc[w]

        vc[v] = min(inc, exc)
```

Visit each node once

Visit each node once

Visit each node once

Vertex Cover in Trees



Running Time: $O(n)$

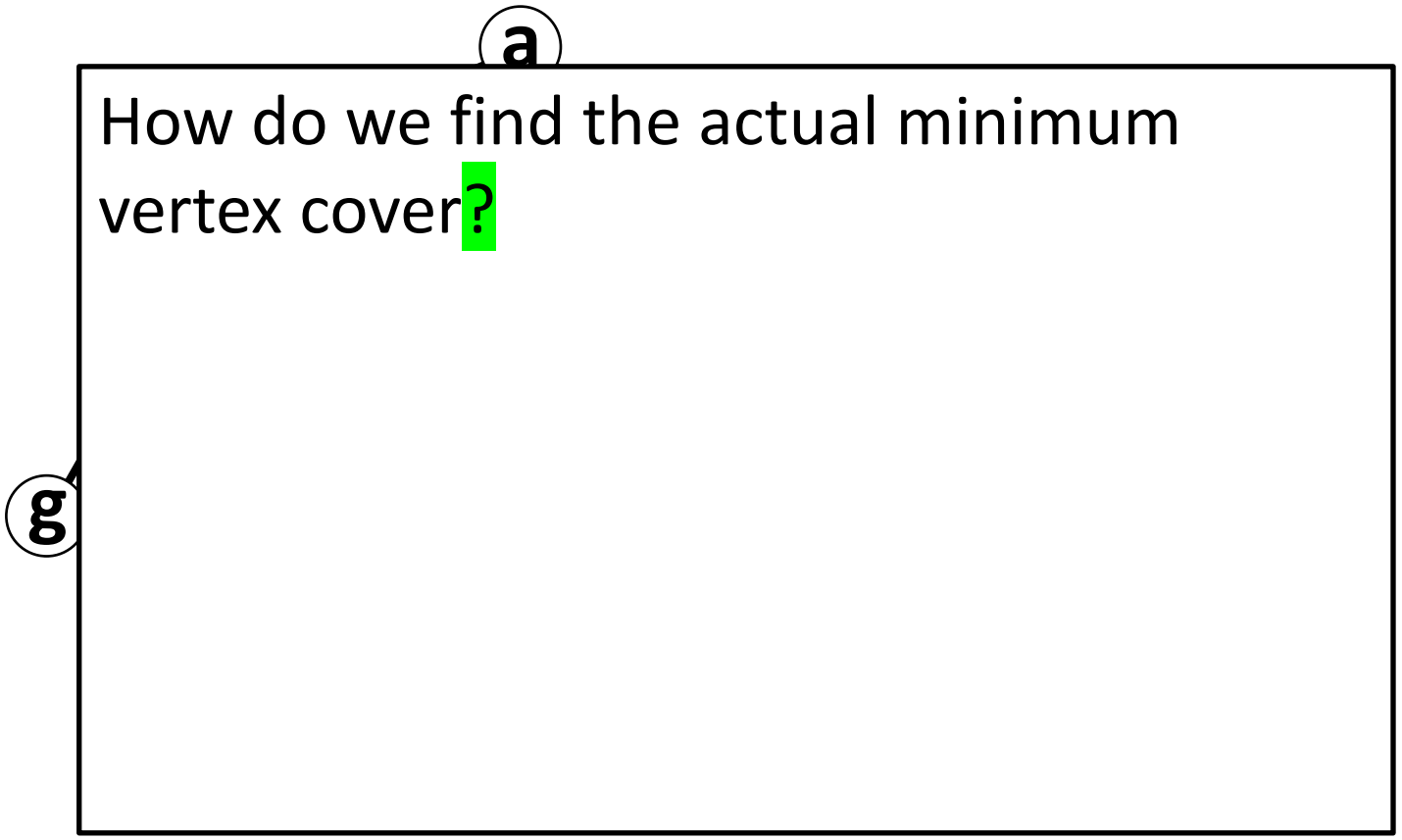
```
min_vc(Vertex v):
    if v is leaf:
        vc[v] = 0
    else:
        // calculate descendants
        for u in child(v):
            min_vc(u)

        // min VC includes v
        inc = 1
        for u in child(v):
            inc += vc[u]

        // min VC excludes v
        exc = |child(v)|
        for u in child(v):
            for w in child(u):
                exc += vc[w]

        vc[v] = min(inc, exc)
```

Visit each node once

Visit each node once

Visit each node once

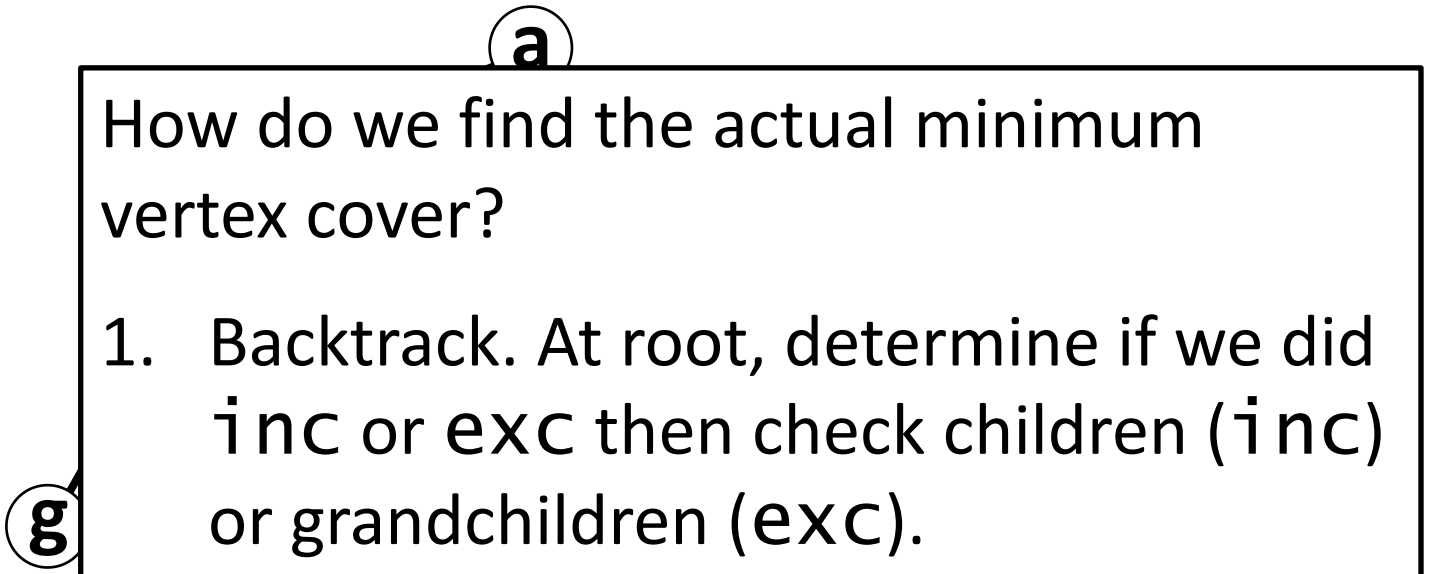
Vertex Cover in Trees



Running Time: $O(n)$

```
min_vc(Vertex v):
   if v is leaf:
      vc[v] = 0
   else:
      // calculate descendants
      for u in child(v):
         min_vc(u)

      // min VC includes v
      inc = 1
      for u in child(v):
         inc += vc[u]

      // min VC excludes v
      exc = |child(v)|
      for u in child(v):
         for w in child(u):
            exc += vc[w]

      vc[v] = min(inc, exc)
```

Vertex Cover in Trees



How do we find the actual minimum vertex cover?

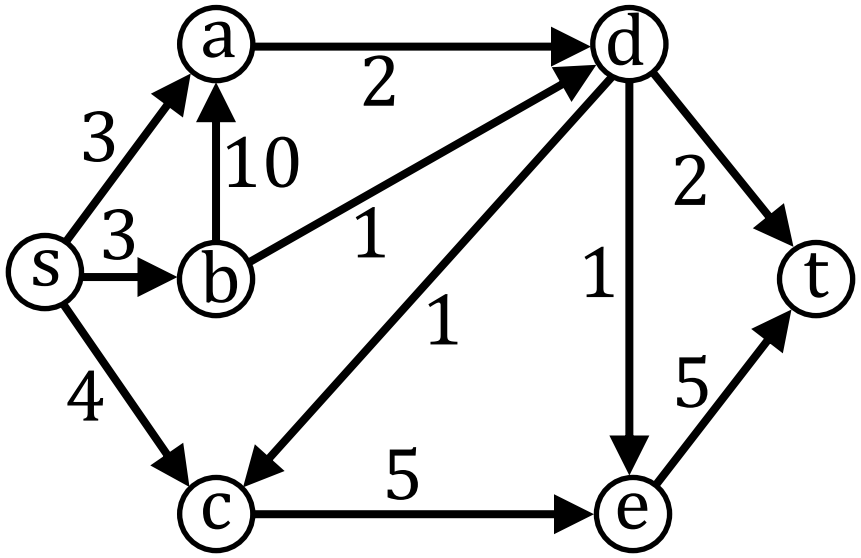
1. Backtrack. At root, determine if we did `inc` or `exc` then check children (`inc`) or grandchildren (`exc`).
2. Save Results. When the minimum of (`inc`, `exc`) is selected, save whether `v` or children of `v` are part of the cover.

Running Time: $O(n)$

```
min_vc(Vertex v):
    if v is leaf:
        vc[v] = 0
    else:
        // calculate descendants
        for u in child(v):
            min_vc(u)

        // min VC includes v
        inc = 1
        for u in child(v):
            inc += vc[u]

        // min VC excludes v
        exc = |child(v)|
        for u in child(v):
            for w in child(u):
                exc += vc[w]

        vc[v] = min(inc, exc)
```

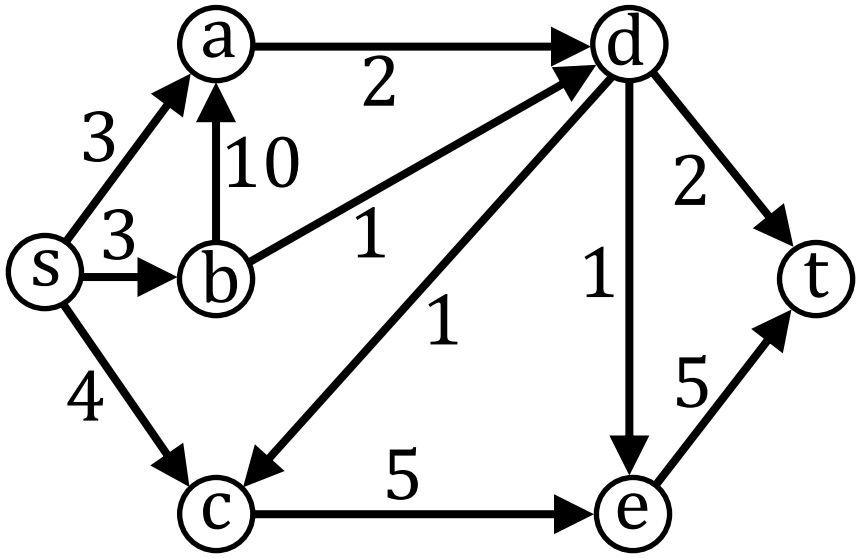
Motivation

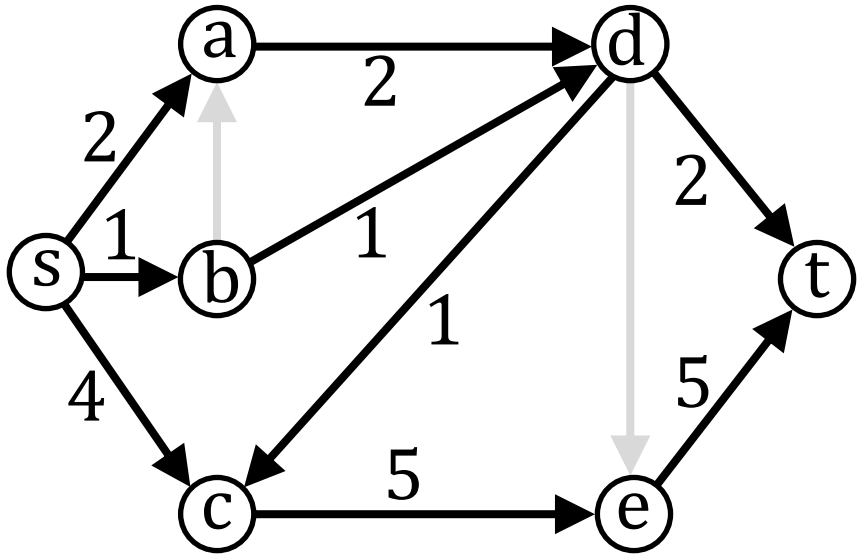
Suppose we have a directed graph that represent an oil pipeline network. Edge weight represent pipe capacity. How much oil can we transfer from source s to sink t?



Motivation

Suppose we have a directed graph that represent an oil pipeline network. Edge weight represent pipe capacity. How much oil can we transfer from source s to sink t?





Total flow = 7