Trees

T. M. Murali

Apr 5, 2004
Trees in Computer Science

- Abstract representation of a hierarchy.
- Tree consists of nodes with a parent-child relationship.
- Applications in biology:
  - Taxonomies
  - Phylogenetic trees
  - Functional ontologies (Gene ontology).
Terminology

- **Root**: node without a parent.
- **Internal node**: node with at least one child.
- **Leaf (external node)**: none with no children.
- **Ancestors**: parent and parent’s ancestors.
- **Depth of a node**: number of ancestors.
- **Height of a tree**: maximum depth over all nodes.
- **Descendants**: children and children’s descendants.
- **Subtree of a node**: Tree consisting of a node and its descendants.
Tree ADT

- Positions abstract nodes
- Generic methods:
  - integer size()
  - boolean isEmpty()
  - Iterator elements()
  - Iterator positions()
- Accessor methods:
  - position root()
  - position parent(p)
  - positionIterator children(p)
- Query methods:
  - boolean isInternal(p)
  - boolean isExternal(p)
  - boolean isRoot(p)
- Update method: object replace (p, o)
- Implementations of the Tree ADT may provide other update methods.
Computing the Depth of a Node

Algorithm depth(T, v):
  if T.isRoot(v) then
    return 0
  else
    return 1 + depth(T, T.parent(v))

What is the running time?

If $d_v$ is the depth of node $v$, the running time is $O(d_v)$. 

T. M. Murali: Trees
Computing the Depth of a Node

Algorithm depth($T, v$):

- If $T$.isRoot($v$) then return 0
- Else return $1 + \text{depth}(T, T.\text{parent}(v))$
Computing the Depth of a Node

Algorithm depth($T, v$):

\textbf{if} $T.\text{isRoot}(v)$ \textbf{then}

else
Computing the Depth of a Node

Algorithm depth($T, v$):

\[
\text{if } T.\text{isRoot}(v) \text{ then }
\]

\[
\text{return } 0
\]

\[
\text{else}
\]
Computing the Depth of a Node

Algorithm depth( T, v):
    if T.isRoot(v) then
        return 0
    else
        return 1 + depth( T, T.parent(v))
Computing the Depth of a Node

Algorithm depth($T, v$):

\[
\begin{align*}
\text{if} & \ T.\text{isRoot}(v) \ \text{then} \\
& \text{return} \ 0 \\
\text{else} & \\
& \text{return} \ 1 + \text{depth}(T, T.\text{parent}(v))
\end{align*}
\]

What is the running time?
Computing the Depth of a Node

Algorithm depth($T, v$):

- if $T$.isRoot($v$) then
  - return 0
- else
  - return $1 + \text{depth}(T, T\.parent(v))$

What is the running time?

- If $d_v$ is the depth of node $v$, the running time is $O(d_v)$. 
Computing the Height of a Tree

- The height of a tree is the maximum depth over all nodes.

\[
\text{Algorithm height1}(T):
\]

\[
h = 0
\]

for each \(v \in T.\text{positions()}\) do

if \(T.\text{isExternal}(v)\) then

\[h = \text{max}(h, \text{depth}(T, v))\]

return \(h\)

- The running time is \(O(n + \sum_{v \in E} (1 + d_v))\), where \(E\) is the set of external nodes.

- In the worst case, this sum is \(O(n^2)\).
Computing the Height of a Tree

- The height of a tree is the maximum depth over all nodes.

- A leaf realises the maximum depth.

\[\text{Algorithm height}(T):\]
\[
h = 0
\]
\[
\text{for each } v \in T.\text{positions() do}
\]
\[
\text{if } T.\text{isExternal}(v) \text{ then}
\]
\[
h = \max(h, \text{depth}(T, v))
\]
\[
\text{return } h
\]

The running time is \(O(n + \sum_{v \in E}(1 + d_v))\), where \(E\) is the set of external nodes. In the worst case, this sum is \(O(n^2)\).
Computing the Height of a Tree

- The height of a tree is the maximum depth over all nodes.
- A leaf realises the maximum depth.

Algorithm \text{height1}(T):

\begin{verbatim}
\text{h} = 0
\text{for each } v \in T\text{.positions()} \text{ do}
  \text{if } T\text{.isExternal}(v) \text{ then}
    \text{h} = \text{max}(\text{h}, \text{depth}(T, v))
\text{return } h
\end{verbatim

The running time is $O(n + \sum_{v \in E} (1 + d_v))$, where $E$ is the set of external nodes. In the worst case, this sum is $O(n^2)$. 

T. M. Murali: Trees
Computing the Height of a Tree

- The height of a tree is the maximum depth over all nodes.

- A leaf realises the maximum depth.

Algorithm height1(T):

\[
h = 0
\]

\[
\text{for each } v \in T.\text{positions()} \text{ do}
\]

\[
\text{if } T.\text{isExternal}(v) \text{ then}
\]

\[
h = \max(h, \text{depth}(T, v))
\]

\[
\text{return } h
\]

- The running time is \( O(n + \sum_{v \in E}(1 + d_v)) \), where \( E \) is the set of external nodes.
Computing the Height of a Tree

- The height of a tree is the maximum depth over all nodes.

- A leaf realises the maximum depth.

Algorithm $height1(T)$:

\[ h = 0 \]

\[
\text{for each } v \in T.\text{positions()} \text{ do} \\
\quad \text{if } T.\text{isExternal}(v) \text{ then} \\
\quad \quad h = \max(h, \text{depth}(T, v))
\]

\[ \text{return } h \]

- The running time is \( O(n + \sum_{v \in E}(1 + d_v)) \), where \( E \) is the set of external nodes.

- In the worst case, this sum is \( O(n^2) \).
A Better Algorithm for Computing the Height

- Can we compute the height recursively?
- Invoke with the root of the tree.

Algorithm height2($T, v$):

```plaintext
if $T$.isExternal($v$) then
    return 0
for each $w \in T$.children($v$) do
    $h = \max(h, T$.height2($T, w$))
return $1 + h$
```
A Better Algorithm for Computing the Height

- Can we compute the height recursively?
- Invoke with the root of the tree.

Algorithm height2\((T, v)\):

\[
\text{if } T.\text{isExternal}(v) \text{ then}
\]
A Better Algorithm for Computing the Height

- Can we compute the height recursively?
- Invoke with the root of the tree.

Algorithm $\text{height2}(T, v)$:

```python
if $T.\text{isExternal}(v)$ then
    return 0
```

How many times is each node visited?

Once.

Running time is $O(n + \sum_{v \in T} (1 + c_v))$, where $c_v$ is the number of children of $v$.

This time is $O(n)$. 

T. M. Murali: Trees
A Better Algorithm for Computing the Height

- Can we compute the height recursively?
- Invoke with the root of the tree.

Algorithm height2\( (T, v) \):
  
  \[
  \text{if } T.\text{isExternal}(v) \text{ then} \\
  \quad \text{return 0} \\
  \text{for each } w \in T.\text{children}(v) \text{ do}
  \]
A Better Algorithm for Computing the Height

- Can we compute the height recursively?
- Invoke with the root of the tree.

Algorithm height2(T, v):

```
if T.isExternal(v) then
    return 0
for each w ∈ T.children(v) do
    h = max(h, T.height2(T, w))
```

How many times is each node visited?

Once.

Running time is $O(n + \sum_{v \in T} (1 + \text{c}_v))$, where $\text{c}_v$ is the number of children of $v$.

This time is $O(n)$. 

T. M. Murali: Trees
A Better Algorithm for Computing the Height

- Can we compute the height recursively?
- Invoke with the root of the tree.

Algorithm height2($T, v$):

```python
if $T$.isExternal($v$) then
    return 0
for each $w \in T$.children($v$) do
    $h = \max(h, T$.height2($T, w$))
return $1 + h$
```

How many times is each node visited? Once.

Running time is $O(n + \sum_{v \in T} (1 + c_v))$, where $c_v$ is the number of children of $v$.

This time is $O(n)$. 

---

T. M. Murali: Trees
A Better Algorithm for Computing the Height

- Can we compute the height recursively?
- Invoke with the root of the tree.

Algorithm height2$(T, v)$:
\[
\text{if } T.\text{isExternal}(v) \text{ then } \\
\quad \text{return } 0 \\
\text{for each } w \in T.\text{children}(v) \text{ do } \\
\quad h = \max(h, T.\text{height2}(T, w)) \\
\text{return } 1 + h
\]

- How many times is each node visited?

Running time is $O(n + \sum_{v \in T} (1 + c_v))$, where $c_v$ is the number of children of $v$. This time is $O(n)$. 
A Better Algorithm for Computing the Height

- Can we compute the height recursively?
- Invoke with the root of the tree.

Algorithm height2\((T, \nu)\):

1. if \(T.\text{isExternal}(\nu)\) then
   1. return 0
2. for each \(w \in T.\text{children}(\nu)\) do
   1. \(h = \max(h, T.\text{height2}(T, w))\)
3. return \(1 + h\)

- How many times is each node visited? Once.

Running time is \(O(n + \sum_{\nu \in T}(1 + c_{\nu}))\), where \(c_{\nu}\) is the number of children of \(\nu\). This time is \(O(n)\).
A Better Algorithm for Computing the Height

- Can we compute the height recursively?
- Invoke with the root of the tree.

**Algorithm height2**(*T*, *v*):

```plaintext
if *T*.isExternal(*v*) then
    return 0
for each *w* ∈ *T*.children(*v*) do
    h = max(h, *T*.height2(*T*, *w*))
return 1 + h
```

- How many times is each node visited? Once.
- Running time is \(O(n + \sum_{v \in T} (1 + c_v))\), where \(c_v\) is the number of children of \(v\).
A Better Algorithm for Computing the Height

- Can we compute the height recursively?
- Invoke with the root of the tree.

Algorithm height2($T, v$):

```
if $T$.isExternal($v$) then
    return 0
for each $w \in T$.children($v$) do
    $h = \max(h, T$.height2($T, w))$
return $1 + h$
```

- How many times is each node visited? Once.

- Running time is $O(n + \sum_{v \in T}(1 + c_v))$, where $c_v$ is the number of children of $v$.

- This time is $O(n)$. 