Algorithms for graph visualization

Divide and Conquer - Tree Layouts
Overview

- Basic Definitions
- Level-based tree layout algorithm
- H(horizontal) V(vertical) tree layout algorithm
- Radial tree layout algorithm
- Other visualization styles
Basic Definitions

- Tree - connected graph without cycles
- Binary tree
Basic Definitions

- Tree - connected graph without cycles
- Binary tree

Tree traversals
Basic Definitions

- Tree - connected graph without cycles
- Binary tree

Tree traversals

Depth-first search
Basic Definitions

- Tree - connected graph without cycles
- Binary tree

**Tree traversals**

**Depth-first search**

- Pre-order (First parent, then subtrees)
- In-order (Left child, parent, right child)
- Post-order (First subtrees, then parent)
Basic Definitions

- Tree - connected graph without cycles
- Binary tree

Tree traversals

Depth-first search
- Pre-order (First parent, then subtrees)
- In-order (Left child, parent, right child)
- Post-order (First subtrees, then parent)

Breadth-first search
Basic Definitions

- Tree - connected graph without cycles
- Binary tree

Tree traversals

Depth-first search
- Pre-order (First parent, then subtrees)
- In-order (Left child, parent, right child)
- Post-order (First subtrees, then parent)

Breadth-first search
- Assignes vertices to levels corresponding to depth

Tree - connected graph without cycles
Binary tree

Root of the tree

Pre-order (First parent, then subtrees)
In-order (Left child, parent, right child)
Post-order (First subtrees, then parent)

Binary tree

Depth-first search

Assignes vertices to levels corresponding to depth

Breadth-first search
Basic Definitions

- Tree - connected graph without cycles
- Binary tree

Tree traversals

Depth-first search
- Pre-order (First parent, then subtrees)
- In-order (Left child, parent, right child)
- Post-order (First subtrees, then parent)

Breadth-first search
- Assignes vertices to levels corresponding to depth

Isomorphism (of ordered trees)

### Simple

3 - 7
Applications

Decision tree analysis for prediction of outcome after traumatic brain injury

*Nature Reviews Neurology*
Applications

Chart to aid students in shaping geographical questions by Gaultier, 1821
Applications

Chart to aid students in shaping geographical questions by Gaultier, 1821.

X-MEN FAMILY TREE

Applications
Level-Based Layout of a Tree

Discuss with your neighbour or in groups of three and write down

- What are the properties of the layout?
- What are the drawing conventions and the aesthetics that we have to take into account?
Level-Based Layout of a Tree

Discuss with your neighbour or in groups of three and write down

- What are the properties of the layout?
- What are the drawing conventions and the aesthetics that we have to take into account?

**Drawing Conventions**

- Vertices lie on layers
- Parent is above the children
- Edges are straight lines
- Parent is centred with respect to the children
- Isomorphic subtrees have identical drawings
Level-Based Layout of a Tree

Discuss with your neighbour or in groups of three and write down

- What the properties of the layout?
- What are the drawing conventions and the aesthetics that we have to take into account?

Drawing Conventions

- Vertices lie on layers
- Parent is above the children
- Edges are straight lines
- Parent is centred with respect to the children
- Isomorphic subtrees have identical drawings

Drawing Aesthetics

- Area

6 - 3
Level-based Layout

Algorithm Outline:
Input: A binary tree
Output: A leveled drawing of T

Base case: A single vertex
Divide: Recursively apply the algorithm to draw the left and the right subtrees of T

Conquer:
Level-based Layout

Algorithm Outline:

Input: A binary tree
Output: A leveled drawing of T

Base case: A single vertex
Divide: Recursively apply the algorithm to draw the left and the right subtrees of T

Conquer:
Level-based Layout

**Algorithm Outline:**

**Input:** A binary tree  
**Output:** A leveled drawing of $T$

**Base case:** A single vertex

**Divide:** Recursively apply the algorithm to draw the left and the right subtrees of $T$

**Conquer:**
Level-based Layout

Algorithm Outline:
Input: A binary tree
Output: A leveled drawing of $T$

Base case: A single vertex
Divide: Recursively apply the algorithm to draw the left and the right subtrees of $T$

Conquer:
Level-based Layout

Algorithm Outline:
Input: A binary tree
Output: A leveled drawing of T

Base case: A single vertex
Divide: Recursively apply the algorithm to draw the left and the right subtrees of T

Conquer:
Level-based Layout

Algorithm Outline:
Input: A binary tree
Output: A leveled drawing of T

Base case: A single vertex
Divide: Recursively apply the algorithm to draw the left and the right subtrees of T

Conquer:

Some agreed distance
Level-based Layout

Algorithm Outline:
Input: A binary tree
Output: A leveled drawing of T

Base case: A single vertex
Divide: Recursively apply the algorithm to draw the left and the right subtrees of T

Conquer:
Level-based Layout

Algorithm Outline:
Input: A binary tree
Output: A leveled drawing of T

Base case: A single vertex
Divide: Recursively apply the algorithm to draw the left and the right subtrees of T

Conquer:

Parent is centered wrt to children

Some agreed distance
Implementation Details (postorder and preorder traversals)

Postorder traversal: For each vertex $v$ compute horizontal displacement of the left and the right child

\[ T_l(v) \quad T_r(v) \]
Level-based Layout

Implementation Details (postorder and preorder traversals)

**Postorder traversal:** For each vertex $v$ compute horizontal displacement of the left and the right child.

![Diagram of a graph with labels $T_l(v)$ and $T_r(v)$ and node $v$.]
Implementation Details (postorder and preorder traversals)

Postorder traversal: For each vertex $v$ compute horizontal displacement of the left and the right child

Assume at each vertex $u$ (below $v$) we have stored the left and the right boundary of the subtree $T(u)$

Level-based Layout
Implementation Details (postorder and preorder traversals)

**Postorder traversal:** For each vertex $v$ compute horizontal displacement of the left and the right child

- Assume at each vertex $u$ (below $v$) we have stored the left and the right boundary of the subtree $T(u)$
Level-based Layout

Implementation Details (postorder and preorder traversals)

**Postorder traversal:** For each vertex $v$ compute horizontal displacement of the left and the right child

- Assume at each vertex $u$ (below $v$) we have stored the left and the right boundary of the subtree $T(u)$
Level-based Layout

Implementation Details (postorder and preorder traversals)

Postorder traversal: For each vertex $v$ compute horizontal displacement of the left and the right child

- Assume at each vertex $u$ (below $v$) we have stored the left and the right boundary of the subtree $T(u)$
Level-based Layout

Implementation Details (postorder and preorder traversals)

Postorder traversal: For each vertex \( v \) compute horizontal displacement of the left and the right child

- Assume at each vertex \( u \) (below \( v \)) we have stored the left and the right boundary of the subtree \( T(u) \)
- “Summ up” the horizontal displacements of the right boundary of \( T_l(v) \) and the left boundary of \( T_r(v) \) to obtain the displ. of the children of \( v \)
Level-based Layout

Implementation Details (postorder and preorder traversals)

Postorder traversal: For each vertex \( v \) compute horizontal displacement of the left and the right child

- Assume at each vertex \( u \) (below \( v \)) we have stored the left and the right boundary of the subtree \( T(u) \)
- “Summ up” the horizontal displacements of the right boundary of \( T_l(v) \) and the left boundary of \( T_r(v) \) to obtain the displ. of the children of \( v \)
Implementation Details (postorder and preorder traversals)

Postorder traversal: For each vertex $v$ compute horizontal displacement of the left and the right child

- Assume at each vertex $u$ (below $v$) we have stored the left and the right boundary of the subtree $T(u)$
- “Summ up” the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$ to obtain the displ. of the children of $v$
Implementation Details (postorder and preorder traversals)

Postorder traversal: For each vertex $v$ compute horizontal displacement of the left and the right child

- Assume at each vertex $u$ (below $v$) we have stored the left and the right boundary of the subtree $T(u)$
- “Summ up” the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$ to obtain the displ. of the children of $v$
- Store at $v$ the left and the right boundaries of $T(v)$
Level-based Layout

Implementation Details (postorder and preorder traversals)

Postorder traversal: For each vertex \( v \) compute horizontal displacement of the left and the right child

Preorder traversal: Compute \( x \)- and \( y \)-coordinates.
Level-based Layout

Implementation Details (postorder and preorder traversals)

Postorder traversal: For each vertex $v$ compute horizontal displacement of the left and the right child

Preorder traversal: Compute $x$- and $y$-coordinates.
Level-based Layout

Implementation Details (postorder and preorder traversals)

**Postorder traversal:** For each vertex \( v \) compute horizontal displacement of the left and the right child

**Preorder traversal:** Compute x- and y-coordinates.
**Level-based Layout**

**Implementation Details (postorder and preorder traversals)**

**Postorder traversal:** For each vertex $v$ compute horizontal displacement of the left and the right child

**Preorder traversal:** Compute $x$- and $y$-coordinates.
Level-based Layout: Time Complexity

Think and write down and then discuss with your neighbour(s)

- What is the time complexity of the algorithm?
- What should we keep in mind to achieve this time complexity?

5+5 min
Level-based Layout

**Time Complexity**

**Postorder traversal:** For each vertex \( v \) compute horizontal displacement of the left and the right child

- Assume at each vertex \( u \) (below \( v \)) we have stored the left and the right boundary of the subtree \( T(u) \)
- Summ up the horizontal displacements of the right boundary of \( T_l(v) \) and the left boundary of \( T_r(v) \)
- Store at \( v \) the left and the right boundaries of \( T(v) \)

**Preorder traversal:** Compute x- and y-coordinates.
Level-based Layout

Time Complexity

**Postorder traversal:** For each vertex $v$ compute horizontal displacement of the left and the right child

- Assume at each vertex $u$ (below $v$) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$
- Store at $v$ the left and the right boundaries of $T(v)$

**Preorder traversal:** Compute x- and y-coordinates.
Level-based Layout

**Time Complexity**

**Postorder traversal:** For each vertex $v$ compute horizontal displacement of the left and the right child

- Assume at each vertex $u$ (below $v$) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_L(v)$ and the left boundary of $T_R(v)$
- Store at $v$ the left and the right boundaries of $T(v)$

**Preorder traversal:** Compute $x$- and $y$-coordinates.
Level-based Layout

Time Complexity

**Postorder traversal:** For each vertex $v$ compute horizontal displacement of the left and the right child

- Assume at each vertex $u$ (below $v$) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$
- Store at $v$ the left and the right boundaries of $T(v)$

**Preorder traversal:** Compute x- and y-coordinates.
Level-based Layout

**Time Complexity**

**Postorder traversal:** For each vertex $v$ compute horizontal displacement of the left and the right child

- Assume at each vertex $u$ (below $v$) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$
- Store at $v$ the left and the right boundaries of $T(v)$

**Preorder traversal:** Compute x- and y-coordinates.
Level-based Layout

**Time Complexity**

**Postorder traversal:** For each vertex $v$ compute horizontal displacement of the left and the right child

- Assume at each vertex $u$ (below $v$) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$
- Store at $v$ the left and the right boundaries of $T(v)$

**Preorder traversal:** Compute x- and y-coordinates.
**Level-based Layout**

**Time Complexity**

**Postorder traversal:** For each vertex $v$ compute horizontal displacement of the left and the right child

- Assume at each vertex $u$ (below $v$) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$
- Store at $v$ the left and the right boundaries of $T(v)$

**Preorder traversal:** Compute $x$- and $y$-coordinates.
**Time Complexity**

**Postorder traversal:** For each vertex $v$ compute horizontal displacement of the left and the right child

- Assume at each vertex $u$ (below $v$) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$
- Store at $v$ the left and the right boundaries of $T(v)$

**Preorder traversal:** Compute x- and y-coordinates.
Level-based Layout

Time Complexity

**Postorder traversal:** For each vertex $v$ compute horizontal displacement of the left and the right child

- Assume at each vertex $u$ (below $v$) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$
- Store at $v$ the left and the right boundaries of $T(v)$

**Preorder traversal:** Compute x- and y-coordinates.
Level-based Layout

Time Complexity

**Postorder traversal:** For each vertex \( v \) compute horizontal displacement of the left and the right child

1. Assume at each vertex \( u \) (below \( v \)) we have stored the left and the right boundary of the subtree \( T(u) \)
2. Summ up the horizontal displacements of the right boundary of \( T_l(v) \) and the left boundary of \( T_r(v) \)
3. Store at \( v \) the left and the right boundaries of \( T(v) \)

**Preorder traversal:** Compute x- and y-coordinates.
Level-based Layout

Time Complexity

Postorder traversal: For each vertex $v$ compute horizontal displacement of the left and the right child

- Assume at each vertex $u$ (below $v$) we have stored the left and the right boundary of the subtree $T(u)$
- Summ up the horizontal displacements of the right boundary of $T_l(v)$ and the left boundary of $T_r(v)$
- Store at $v$ the left and the right boundaries of $T(v)$

Preorder traversal: Compute x- and y-coordinates.

To compute the displacement: constant number of operations at each vertex
Level-based Layout

**Time Complexity**

**Postorder traversal:** For each vertex \( v \) compute horizontal displacement of the left and the right child

- Assume at each vertex \( u \) (below \( v \)) we have stored the left and the right boundary of the subtree \( T(u) \)
- Summ up the horizontal displacements of the right boundary of \( T_l(u) \) and the left boundary of \( T_r(u) \)
- Store at \( v \) the left and the right boundaries of \( T(v) \)

**Preorder traversal:** Compute x- and y-coordinates.

To compute the displacement: constant number of operations at each vertex
Level-based Layout

Time Complexity

**Postorder traversal:** For each vertex \( v \) compute horizontal displacement of the left and the right child

- Assume at each vertex \( u \) (below \( v \)) we have stored the left and the right boundary of the subtree \( T(u) \)
- Summ up the horizontal displacements of the right boundary of \( T_l(v) \) and the left boundary of \( T_r(v) \)
- Store at \( v \) the left and the right boundaries of \( T(v) \)

**Preorder traversal:** Compute x- and y-coordinates.

To compute the displacement: constant number of operations at each vertex
Level-based Layout

Time Complexity

**Postorder traversal:** For each vertex \( v \) compute horizontal displacement of the left and the right child

- Assume at each vertex \( u \) (below \( v \)) we have stored the left and the right boundary of the subtree \( T(u) \)
- Summ up the horizontal displacements of the right boundary of \( T_l(v) \) and the left boundary of \( T_r(v) \)
- Store at \( v \) the left and the right boundaries of \( T(v) \)

**Preorder traversal:** Compute x- and y-coordinates.

To compute the displacement: constant number of operations at each vertex

\( O(n) \)
Level-based Layout

Theorem (Reingold & Tilford)

Let $T$ be a binary tree with $n$ vertices. Algorithm (R & T) constructs a drawing $\Gamma$ of $T$ in $O(n)$ time, such that:

- $\Gamma$ is planar and straight-line
- $\forall v \in T$ y-coordinate of $v$ is $-\text{depth}(v)$
- Vertical and horizontal distance is at least 1
- Area of $\Gamma$ is
Level-based Layout

**Theorem (Reingold & Tilford)**

Let $T$ be a binary tree with $n$ vertices. Algorithm (R & T) constructs a drawing $\Gamma$ of $T$ in $O(n)$ time, such that:

- $\Gamma$ is planar and straight-line
- $\forall v \in T$ y-coordinate of $v$ is $-depth(v)$
- Vertical and horizontal distance is at least 1
- Area of $\Gamma$ is $O(n^2)$
Level-based Layout

Theorem (Reingold & Tilford)

Let $T$ be a binary tree with $n$ vertices. Algorithm (R & T) constructs a drawing $\Gamma$ of $T$ in $O(n)$ time, such that:

- $\Gamma$ is planar and straight-line
- $\forall v \in T$ y-coordinate of $v$ is $-\text{depth}(v)$
- Vertical and horizontal distance is at least 1
- Area of $\Gamma$ is $O(n^2)$
- Each vertex is centered with respect to its children
Level-based Layout

Theorem (Reingold & Tilford)

Let $T$ be a binary tree with $n$ vertices. Algorithm (R & T) constructs a drawing $\Gamma$ of $T$ in $O(n)$ time, such that:

- $\Gamma$ is planar and straight-line
- $\forall v \in T$ y-coordinate of $v$ is $-\text{depth}(v)$
- Vertical and horizontal distance is at least 1
- Area of $\Gamma$ is $O(n^2)$
- Each vertex is centered with respect to its children
- Simply isomorphic subtrees have congruent (coincident) drawing, up to translation
- Axially isomorphic trees have congruent drawing, up to translation and reflection around y-axis
Level-based Layout

The presented algorithm tries to minimize width
Level-based Layout

- The presented algorithm tries to minimize width
Level-based Layout

- The presented algorithm tries to minimize width
- Does not achieve that!
Level-based Layout

- The presented algorithm tries to minimize width
- Does not achieve that!
- Divide-and-conquer strategy cannot achieve optimal width
Level-based Layout

- The presented algorithm tries to minimize width.
- Does not achieve that!
- Divide-and-conquer strategy cannot achieve optimal width.

Drawing with min width and properties of our algorithm can be constructed by an LP.
Level-based Layout

- The presented algorithm tries to minimize width
- Does not achieve that!
- Divide-and-conquer strategy cannot achieve optimal width

Drawing with min width and properties of our algorithm can be constructed by an LP
- If integer coordinates are required, then it is NP-hard
Level-based Layout for Trees

- Book Di Battista et al: Chapter 3.1.2
- Skript: Chapter 6.1