Area-Efficient Order-Preserving Planar Straight-line Drawings of Ordered Trees*

Ashim Garg Adrian Rusu

Department of Computer Science and Engineering
University at Buffalo
Buffalo, NY 14260
{agarg, adirusu}@cse.buffalo.edu

Abstract

Ordered trees are generally drawn using order-preserving planar straight-line grid drawings. We therefore investigate the area-requirements of such drawings, and present several results: Let T be an ordered tree with n nodes. Then:

- T admits an order-preserving planar straight-line grid drawing with $O(n \log n)$ area.
- If T is a binary tree, then T admits an order-preserving planar straight-line grid drawing with $O(n \log \log n)$ area.
- If T is a binary tree, then T admits an order-preserving *upward* planar straight-line grid drawing with *optimal* $O(n \log n)$ area.

We also study the problem of drawing binary trees with user-specified arbitrary aspect ratios. We show that an ordered binary tree T with n nodes admits an order-preserving planar straight-line grid drawing Γ with width $O(A + \log n)$, height $O((n/A) \log A)$, and area $O((A + \log n)(n/A) \log A) = O(n \log n)$, where $2 \leq A \leq n$ is any user-specified number. Also note that all the drawings mentioned above can be constructed in $O(n)$ time.

1 Introduction

An *ordered tree* T is one with a prespecified counterclockwise ordering of the edges incident on each node. Ordered trees arise commonly in practice. Examples of ordered trees include binary search trees, arithmetic

*Research supported by NSF CAREER Award No. IIS-9985136 and NSF CISE Research Infrastructure Award No. 0101244.

expression trees, BSP-trees, B-trees, and range-trees.

An *order-preserving drawing* of T is one in which the counterclockwise ordering of the edges incident on a node is the same as their prespecified ordering in T. A *planar drawing* of T is one with no edge-crossings. An *upward drawing* of T is one, where each node is placed either at the same y-coordinate as, or at a higher y-coordinate than the y-coordinates of its children. A *straight-line drawing* of T is one, where each edge is drawn as a single line-segment. A *grid drawing* of T is one, where each node is assigned integer x- and y-coordinates.

Ordered trees are generally drawn using order-preserving planar straight-line grid drawings, as any undergraduate textbook on data-structures will show. An upward drawing is desirable because it makes it easier for the user to determine the parent-child relationships between the nodes.

We investigate the area-requirement of the order-preserving planar straight-line grid drawings of ordered trees, and present several results: Let T be an ordered tree with n nodes.

Result 1: We show that T admits an order-preserving planar straight-line grid drawing with $O(n \log n)$ area, $O(n)$ height, and $O(\log n)$ width, which can be constructed in $O(n)$ time.

Result 2: If T is a binary tree, then we show stronger results:

Result 2a: T admits an order-preserving planar straight-line grid drawing with $O(n \log \log n)$ area, $O((n/ \log n) \log \log n)$ height, and $O(\log n)$ width, which can be constructed in $O(n)$ time.

Result 2b: T admits an order-preserving *upward* planar straight-line grid drawing with *optimal* $O(n \log n)$ area, $O(n)$ height, and $O(\log n)$ width, which can be constructed in $O(n)$ time.

An important issue is that of the *aspect ratio* of a drawing D. Let E be the smallest rectangle, with sides parallel to x and y-axis, respectively, enclosing D. The *aspect ratio* of D is defined as the ratio of the larger and smaller dimensions of E, i.e., if h and w are the height and width, respectively, of E, then the aspect ratio of D is equal to $\max\{h, w\}/ \min\{h, w\}$. It is important to give the user control over the aspect ratio of a drawing because this will allow her to fit the drawing in an arbitrarily-shaped window defined by her application. It also allows the drawing to fit within display-surfaces with predefined aspect ratios, such as a computer-screen and a sheet of paper. We consider the problem of drawing binary trees with arbitrary aspect ratio, and prove the following result:

Result 3: Let T be a binary tree with n nodes. Let $2 \le A \le n$ be any user-specified number. T admits an order-preserving planar straight-line grid drawing Γ with width $O(A + \log n)$, height $O((n/A) \log A)$, and area $O((A + \log n)(n/A) \log A) = O(n \log n)$, which can be constructed in $O(n)$ time.

Also note that [7] shows an n-node binary tree that requires $\Omega(n)$ height and $\Omega(\log n)$ width in any order-preserving upward planar grid drawing. Hence, the $O(n)$ height and $O(\log n)$ width achieved by Result *2b* is optimal in the worst case.

2 Previous Results

Throughout this section, n denotes the number of nodes in a tree. The *degree* of a tree is equal to the maximum number of edges incident on a node.

In spite of the natural appeal of order-preserving drawings, quite surprisingly, little work has been done on optimizing the area of such drawings. The previous best upper bound on the area-requirement of an order-preserving planar upward straight-line grid drawing of a tree was $O(n^{1+\epsilon})$, where $\epsilon > 0$ is any user-defined constant, which was shown in [2]. [10] has shown that a special class of balanced binary trees, which includes k-balanced, red-black, $BB[\alpha]$, and (a, b) trees, admits order-preserving planar upward straight-line grid drawings with area $O(n(\log\log n)^2)$. [3], [4], and [12] give order-preserving planar upward straight-line grid drawings of complete binary trees, logarithmic, and Fibonacci trees, respectively, with area $O(n)$. [7] has given an upper bound of $O(n \log n)$ on order-preserving planar upward *polyline* grid drawings. (A polyline drawing is one, where each edge is drawn as a connected sequence of one *or more* line-segments.)

As for the lower bound on the area-requirement of order-preserving drawings, [7] has shown a lower bound of $\Omega(n \log n)$ for order-preserving planar upward grid drawings. There is no known lower bound for non-upward order-preserving planar grid drawings other than the trivial $\Omega(n)$ bound.

We are not aware of any non-trivial results on order-preserving drawings of trees with user-defined arbitrary aspect-ratios. However, a few results are available on non-order-preserving drawings. [7] shows that any tree with degree d admits a non-order-preserving planar upward polyline grid drawing with height $h = O(n^{1-\alpha})$ and area $O(n + dh \log n)$, where $0 < \alpha < 1$ is any user-specified constant. This result implies that any tree with degree $O(n^{\beta})$, where $0 \le \beta < 1$ is any constant, can be drawn in this fashion in $O(n)$ area with aspect ratio $O(n^{\gamma})$, where γ is any user-defined constant, such that $\max\{0, 2\beta - 1\} < \gamma < 1$. [1] shows that any binary tree admits a non-order-preserving upward planar straight-line *orthogonal* (each edge drawn as a horizontal or vertical line-segment) grid drawing with area $O(n \log n)$, and any user-specified aspect ratio in the range $[1, n/\log n]$. They also prove that the $O(n \log n)$ bound on area is also optimal for such drawings by showing that for any n and a number $2 \le A \le n$, there exists a binary tree with n nodes that requires $\Omega(n \log n)$ area in any upward planar straight-line orthogonal grid drawing with aspect ratio in the range $[1, n/\log n]$. [1] and [10] show that any binary tree admits a non-order-preserving non-upward planar straight-line orthogonal grid drawing with height $O(n/A) \log A$, width $O(A + \log n)$, where $2 \le A \le n$ is any user-specified number. This result also implies that we can draw any binary tree in this fashion in area $O(n \log\log n)$ (by setting $A = \log n$).

[9] shows that any binary tree admits a non-order-preserving planar non-upward straight-line drawing with area $O(n)$, and any user-specified aspect ratio in the range $[1, n^{\alpha}]$, where $0 \le \alpha < 1$ is any constant. [8] extends this result to trees with degree $O(n^{\delta})$, where $0 \le \delta < 1/2$ is any constant.

As for other kinds of drawings (non-order-preserving and with fixed aspect ratio), a variety of results are available. See [6] for a survey on these results.

Table 1 compares our results with some previously known results.

Tree Type	Drawing Type	Area	Aspect Ratio	Reference
Complete Binary	Upward Straight-line Order-preserving	$\Theta(n)$	$O(1)$	[3]
Fibonacci	Upward Straight-line Order-preserving	$\Theta(n)$	$O(1)$	[12]
Special Balanced Binary Trees such as Red-black	Upward Straight-line Order-preserving	$O(n(\log\log n)^2)$	$n/\log^2 n$	[11]
Logarithmic Tree	Upward Straight-line Order-preserving	$\Theta(n)$	$O(1)$	[4]
Binary	Upward Straight-line Orthogonal Non-order-preserving	$\Theta(n\log n)$	$[1, n/\log n]$	[1]
	Non-upward Straight-line Orthogonal Non-order-preserving	$O(n\log\log n)$	$(n\log\log n)/\log^2 n$	[1, 11]
	Upward Straight-line Non-order-preserving	$O(n\log\log n)$	$(n\log\log n)/\log^2 n$	[11]
	Upward Straight-line Order-preserving	$O(n^{1+\epsilon})$	$n^{1-\epsilon}$	[2]
		$O(n\log n)$	$n/\log n$	this paper
	Non-upward Straight-line Non-order-preserving	$\Theta(n)$	$[1, n^\alpha]$	[9]
	Non-upward Straight-line Order-preserving	$O(n^{1+\epsilon})$	$n^{1-\epsilon}$	[2]
		$O(n\log n)$	$[1, n/\log n]$	this paper
		$O(n\log\log n)$	$(n\log\log n)/\log^2 n$	this paper
Tree with degree $O(n^\delta)$, for any constant $0 \leq \delta < 1/2$	Non-upward Straight-line Non-order-preserving	$\Theta(n)$	$[1, n^\alpha]$	[8]
Tree with degree $O(n^\beta)$, for any constant $0 \leq \beta < 1$	Upward Polyline Non-order-preserving	$\Theta(n)$	$[\max\{1, n^{2\beta-1}\}, n^\gamma]$	[7]
General	Upward Polyline Order-Preserving	$\Theta(n\log n)$	$n/\log n$	[7]
	Upward Straight-line Order-Preserving	$O(n^{1+\epsilon})$	n	[2]
	Non-upward Straight-line Order-preserving	$O(n^{1+\epsilon})$	n	[2]
		$O(n\log n)$	$n/\log n$	this paper

Table 1: Bounds on the areas and aspect ratios of various kinds of planar straight-line grid drawings of an n-node tree. Here, α, γ, and ϵ, are any user-defined constants, such that $0 \leq \alpha < 1$, $0 \leq \gamma < 1$, and $0 < \epsilon < 1$. $[a, b]$ denotes the range $a \ldots b$.

3 Definitions

We assume a 2-dimensional Cartesian space. We assume that this space is covered by an infinite rectangular grid, consisting of horizontal and vertical channels.

A *left-corner* drawing of an ordered tree T is one, where no node of T is to the left of, or above the root of T. The *mirror-image* of T is the ordered tree obtained by reversing the counterclockwise order of edges around each node. Let R be a rectangle with sides parallel to the x- and y-axis, respectively. The *height* (*width*) of R is equal to the number of grid-points with the same x-coordinate (y-coordinate) contained within R. The *area* of R is equal to the number of grid-points contained within R. The *enclosing rectangle* E of a drawing D is the smallest rectangle with sides parallel to the x- and y-axis covering the entire drawing. The *height* h, *width* w, and *area* of D is equal to the height, width, and area, respectively, of E. The *aspect ratio* of D is equal to $\max\{h, w\}/\min\{h, w\}$.

A *subtree* rooted at a node v of an ordered tree T is the maximal tree consisting of v and all its descendents. A *partial* tree of T is a connected subgraph of T. A *spine* of T is a path $v_0v_1v_2\dots v_m$, where $v_0, v_1, v_2, \dots, v_m$ are nodes of T, that is defined recursively as follows (see Figure 1):

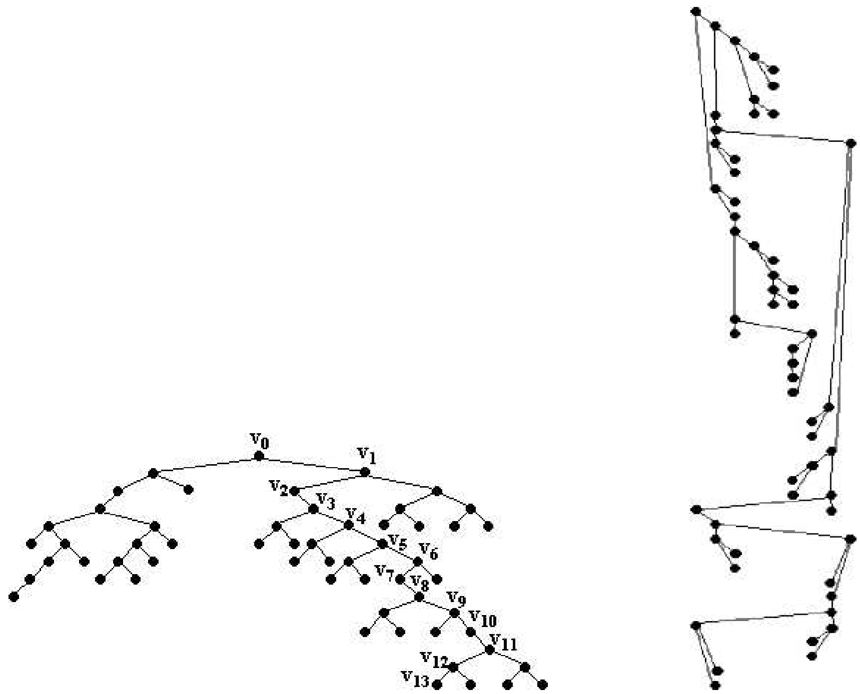
- v_0 is the same as the root of T;
- v_{i+1} is the child of v_i, such that the subtree rooted at v_{i+1} has the maximum number of nodes among all the subtrees that are rooted at the children of v_i.

The concept of a spine has been used implicitly by several tree drawing algorithms, including those of [1, 2, 7]. In particular, [2] uses it to construct order-preserving drawings. However, our algorithms typically draw the spine as a more zig-zagging path than the algorithms of [2]. (In fact, some algorithms of [2] draw the spine completely straight as a single line-segment.) This enables our algorithms to draw a tree more compactly than the algorithms of [2].

4 Drawing Binary Trees

We now give our drawing algorithm for constructing order-preserving planar upward straight-line grid drawings of binary trees. In an ordered binary tree, each node has at most two children, called its *left* and *right* children, respectively.

Our drawing algorithm, which we call *Algorithm BT-Ordered-Draw*, uses the divide-and-conquer paradigm to draw an ordered binary tree T. In each recursive step, it breaks T into several subtrees, draws each subtree recursively, and then combines their drawings to obtain an upward left-corner drawing $D(T)$ of T. We now give the details of the actions performed by the algorithm to construct $D(T)$. Note that during its working, the algorithm will designate some nodes of T as either *left-knee*, *right-knee*, *ordinary-left*, *ordinary-right*, *switch-left* or *switch-right* nodes (for an example, see Figure 2):



(a) (b)

Figure 1: (a) A binary tree T with spine $v_0 v_1 \dots v_{13}$. (b) The order-preserving planar upward straight-line grid drawing of T constructed by *Algorithm BT-Ordered-Draw*.

1. Let $P = v_0 v_1 v_2 \dots v_m$ be a spine of T. Define a *non-spine* node of T to be one that is not in P.

2. Designate v_0 as a *left-knee* node.

3. for $i = 0$ to m do (see Figure 2)

   Depending upon whether v_i is a *left-knee*, *right-knee*, *ordinary-left*, *ordinary-right*, *switch-left*, or *switch-right* node, do the following:

   (a) *v_i is a left-knee node:* If v_{i+1} has a left child, and this child is not v_{i+2}, then designate v_{i+1} as a *switch-right node*, otherwise designate it as an *ordinary-left node.* Recursively construct an upward left-corner drawing of the subtree of T rooted at the non-spine child of v_i.

   (b) *v_i is an ordinary-left node:* If v_{i+1} has a left child, and this child is not v_{i+2}, then designate v_{i+1} as a *switch-right node*, otherwise designate it as an *ordinary-left node.* Recursively construct an upward left-corner drawing of the subtree of T rooted at the non-spine child of v_i.

   (c) *v_i is a switch-right node:* Designate v_{i+1} as a *right-knee* node. Recursively construct an upward left-corner drawing of the subtree of T rooted at the non-spine child of v_i.

   (d) *v_i is a right-knee, ordinary-right, or switch-left node:* Do the same as in the cases, where

v_i is a left-knee, ordinary-left, or switch-right node, respectively, with "left" exchanged with "right", and instead of constructing an upward left-corner drawing of the subtree T_i of T rooted at the non-spine child of v_i, we recursively construct an upward left-corner drawing of the *mirror image* of T_i.

4. Let G be the drawing with the maximum width among the drawings constructed in Step 3. Let W be the width of G.

5. Place v_0 at the origin.

6. for $i = 0$ to m do (see Figures 2 and 3)

   Let H_i be the horizontal channel corresponding to the node placed lowest in the drawing of T constructed so far.

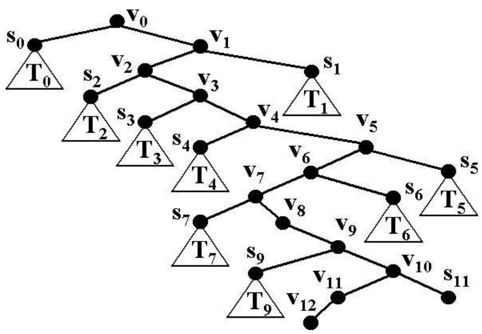
   Depending upon whether v_i is a *left-knee*, *right-knee*, *ordinary-left*, *ordinary-right*, *switch-left*, or *switch-right* node, do the following:

   (a) *v_i is a left-knee node:* If v_{i+1} is the only child of v_i, then place v_{i+1} on the horizontal channel $H_i + 1$ and one unit to the right of v_i (see Figure 3(a)). Otherwise, let s be the child of v_i different from v_{i+1}. Let D be the drawing of the subtree rooted at s constructed in Step 3. If s is the right child of v_i, then place D such that its top boundary is at the horizontal channel $H_i + 1$ and its left boundary is one unit to the right of v_i; place v_{i+1} one unit below D and one unit to the right of v_i (see Figure 3(b)). If s is the left child of v_i, then place v_{i+1} one unit below and one unit to the right of v_i (see Figure 3(a)) (the placement of D will be handled by the algorithm when it will consider a switch-right node later on).

   (b) *v_i is an ordinary-left node:* Since v_i is an ordinary-left node, either v_{i+1} will be the only child of v_i, or v_i will have a right child s, where $s \neq v_{i+1}$. If v_{i+1} is the only child of v_i, then place v_{i+1} one unit below v_i in the same vertical channel as it (see Figure 3(c)). Otherwise, let s be the right child of v_i. Let D be the drawing of the subtree rooted at s constructed in Step 3. Place D one unit below and one unit to the right of v_i; place v_{i+1} on the same horizontal channel as the bottom of D and in the same vertical channel as v_i (see Figure 3(d)).

   (c) *v_i is a switch-right node:* Note that, since v_i is a switch-right node, it will have a left child s, where $s \neq v_{i+1}$. Let v_j be the left-knee node of P closest to v_i in the subpath $v_0 v_1 \ldots v_i$ of P. v_j is called the *closest left-knee ancestor* of v_i. Place v_{i+1} one unit below and $W + 1$ units to the right of v_i.

   Let D be the drawing of the subtree rooted at s constructed in Step 3. Place D one unit below v_i such that s is in the same vertical channel as v_i (see Figure 3(e)). If v_j has a left child s', which is different from v_{j+1}, then let D' be the drawing of the subtree rooted at s'



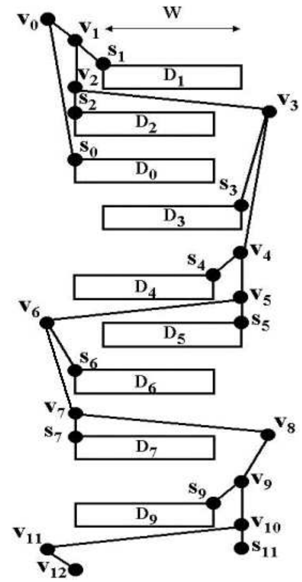


Figure 2: (a) A binary tree T with spine $v_0 v_1 \ldots v_{12}$. (b) A schematic diagram of the drawing $D(T)$ of T constructed by *Algorithm BT-Ordered-Draw*. Here, v_0 is a left-knee, v_1 is an ordinary-left, v_2 is a switch-right, v_3 is a right-knee, v_4 is an ordinary-right, v_5 is a switch-left, v_6 is a left-knee, v_7 is a switch-right, v_8 is a right-knee, v_9 is an ordinary-right, v_{10} is a switch-left, v_{11} is a left-knee, and v_{12} is an ordinary-left node. For simplicity, we have shown $D_0, D_1, \ldots, D_9$ with identically sized boxes but in actuality they may have different sizes.

constructed in Step 3. Place D' one unit below D such that s' is in the same vertical channel as v_i (see Figure 3(f)).

(d) *v_i is a right-knee, ordinary-right, or switch-left node:* These cases are the same as the cases, where v_i is a left-knee, ordinary-left, or switch-right node, respectively, except that "left" is exchanged with "right", and the left-corner drawing of the mirror image of the subtree rooted at the non-spine child of v_i, constructed in Step 3, is first flipped left-to-right and then is placed in $D(T)$.

To determine the area of $D(T)$, notice that the width of $D(T)$ is equal to $W + 3$ (see the definition of W given in Step 3). From the definition of a spine, it follows easily that the number of nodes in each subtree rooted at a non-spine node of T is at most $n/2$, where n is the number of nodes in T. Hence, if we denote by $w(n)$, the width of $D(T)$, then, $W \leq w(n/2)$, and so, $w(n) \leq w(n/2) + 3$. Hence, $w(n) = O(\log n)$. The height of $D(T)$ is trivially at most n. Hence, the area of $D(T)$ is $O(n \log n)$. It is easy to see that the Algorithm can be implemented such that it runs in $O(n)$ time.

[7] has shown a lower bound of $\Omega(n \log n)$ for order-preserving planar upward straight-line grid drawings of binary trees. Hence, the upper bound of $O(n \log n)$ on the area of $D(T)$ is also optimal. We therefore get

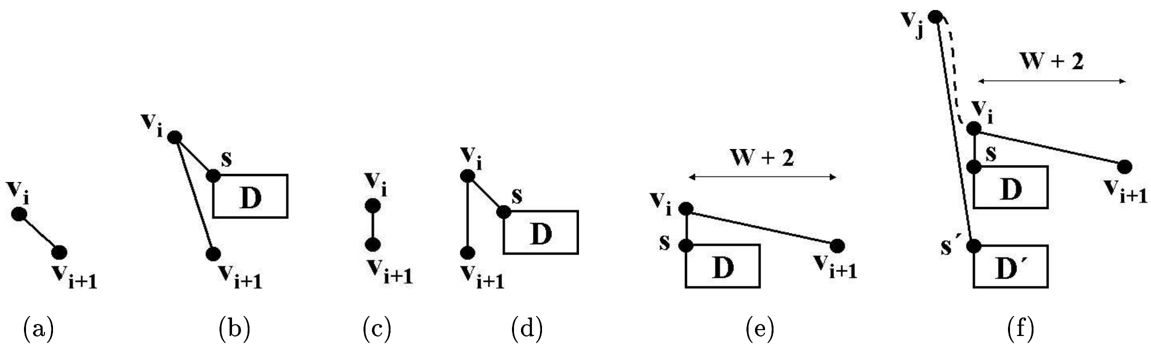


Figure 3: (a,b) Placement of v_i, v_{i+1}, and D in the case when v_i is a left-knee node: (a) v_{i+1} is the only child of v_i or s is the left child of v_i, (b) s is the right child of v_i. (c,d) Placement of v_i, v_{i+1}, and D in the case when v_i is an ordinary-left node: (c) v_{i+1} is the only child of v_i, (d) s is the right child of v_i. (e,f) Placement of v_i, v_{i+1}, D, and D in the case when v_i is a switch-right node. (e) v_j does not have a left child, (f) v_j has a left child s'. Here, D' is the drawing of the subtree rooted at s'.

the following theorem:

Theorem 1 *A binary tree with n nodes admits an order-preserving upward planar straight-line grid drawing with height at most n, width $O(\log n)$, and optimal $O(n \log n)$ area, which can be constructed in $O(n)$ time.*

We can also construct a non-upward left-corner drawing $D'(T)$ of T, such that $D'(T)$ has height $O(\log n)$ and width at most n, by first constructing a left-corner drawing of the mirror image of T using Algorithm *BT-Ordered-Draw*, then rotating it clockwise by 90°, and then flipping it right-to-left. This gives Corollary 1.

Corollary 1 *Using Algorithm* BT-Ordered-Draw*, we can construct in $O(n)$ time, a non-upward left-corner order-preserving planar straight-line grid drawing of an n-node binary with area $O(n \log n)$, height $O(\log n)$, and width at most n.*

5 Drawing General Trees

In a general tree, a node may have more than two children. This makes it more difficult to draw a general tree.

In this section, we give an algorithm, which we call *Algorithm Ordered-Draw*, for constructing (non-upward) order-preserving planar straight-line grid drawing with $O(n \log n)$ area in $O(n)$ time. Algorithm *Ordered-Draw* is a modification of the algorithm for drawing binary trees presented in Section 4.

Let T be a tree with n nodes. In each recursive step, Algorithm *Ordered-Draw* breaks T into several subtrees, draws each subtree recursively, and then combines their drawings to obtain a left-corner drawing $D(T)$ of T.

We now give the details of the actions performed at each recursive step of the algorithm to construct a a left-corner drawing $D(T)$ of T. Note that the counterclockwise ordering of edges around each node, induces a counterclockwise ordering of the children of each node. During its working, the algorithm will designate some nodes of T as either *left-knee*, *right-knee*, *switch-left*, or *switch-right* nodes (for an example, see Figure 4):

1. Let $P = v_0 v_1 v_2 \dots v_m$ be a spine of T. Define a *non-spine* node of T to be one that is not in P.
2. Designate v_0 as a left-knee node.
3. for $i = 0$ to m do (see Figure 4)

   Depending upon whether v_i is a *left-knee*, *right-knee*, *ordinary-left*, *ordinary-right*, *switch-left*, or *switch-right* node, do the following:

   (a) *v_i is a left-knee node:* Designate v_{i+1} as a *switch-right node.* Recursively construct left-corner drawings of the subtrees of T rooted at all the non-spine children of v_i.

   (b) *v_i is a switch-right node:* Designate v_{i+1} as a *right-knee* node. Recursively construct left-corner drawings of the subtrees of T rooted at all the non-spine children of v_i.

   (c) *v_i is a right-knee, or switch-left node:* These cases are the same as the cases, where v_i is a left-knee node, or switch-right node, respectively, with "left" exchanged with "right", and instead of recursively constructing left-corner drawings of the subtrees of T rooted at all the non-spine children of v_i, we recursively construct the left-corner drawings of the *mirror images* of these subtrees.

4. Let G be the drawing with the maximum width among the drawings constructed in Step 3. Let W be the width of G.
5. Place v_0 at the origin. Let Y_0 be the horizontal channel one unit below the origin.
6. for $i = 0$ to m do (see Figures 4 and 5)

   Depending upon whether v_i is a *left-knee*, *right-knee*, *ordinary-left*, *ordinary-right*, *switch-left*, or *switch-right* node, do the following:

   (a) *v_i is a left-knee node:* Let $Q = s_1 s_2 \dots s_k$ be the (possibly empty) sequence of the children of v_i that come after v_{i+1} in the counterclockwise order of the children of v_i (see Figure 5(a)). In this sequence, the s_j's, $1 \leq j \leq k$, are placed in the same order as they occur in the counterclockwise order of the children of v_i. Let D_j be the drawing of the subtree rooted at s_j constructed in Step 3. Place $D_1, D_2, \dots, D_k$ in that order one above the other at unit vertical separation from each other, such that D_1 is at the bottom and D_k is at the top, the top of D_k is at the horizontal channel Y_i, and each D_j is placed one unit to the right of v_i (see Figure 5(b)).

Let Y_{i+1} be the horizontal channel one unit below D_1 if Q is not empty, and is the same as Y_i if Q is empty.

(b) *v_i is a switch-right node:* Note that, since v_i is a switch-right node, v_{i-1} must be a left-knee node.

Let $Q = s_1 s_2 \dots s_k$ be the (possibly empty) sequence of the children of v_i that come after v_{i+1} in the counterclockwise order of the children of v_i (see Figure 5(c)). In this sequence, the s_j's, $1 \leq j \leq k$, are placed in the same order as they occur in the counterclockwise order of the children of v_i. Let D_j be the drawing of the subtree rooted at s_j constructed in Step 3. Place $D_1, D_2, \dots, D_k$ in that order one above the other at unit vertical separation from each other, such that D_1 is at the bottom and D_k is at the top, the top of D_k is at horizontal channel Y_i, and each D_j is placed two units to the right of v_{i-1}.

Place v_i such that it is one unit to the right of v_{i-1}, and is one unit below D_1, if Q is not empty, and is at the horizontal channel Y_i if Q is empty.

Place v_{i+1} one unit below and $W+1$ units to the right of v_i (see Figure 5(d)).

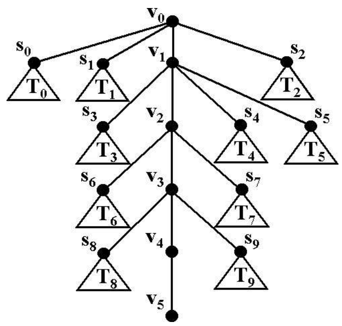
Let $Q' = s'_1 s'_2 \dots s'_r$ be the (possibly empty) sequence of the children of v_i that come before v_{i+1} in the counterclockwise order of the children of v_i (see Figure 5(c)). In this sequence, the s'_j's, $1 \leq j \leq r$, are placed in the same order as they occur in the counterclockwise order of the children of v_i. Let D'_j be the drawing of the subtree rooted at s'_j constructed in Step 3. Place $D'_1, D'_2, \dots, D'_r$ in that order one above the other at unit vertical separation from each other, such that D'_1 is at the bottom and D'_r is at the top, s'_r is placed on the same vertical channel as v_{i+1}, and each D'_j is placed two units to the right of v_{i-1} (see Figure 5(d)).

Let H be the horizontal channel which is one unit below the bottom of D'_1 if Q' is not empty, and contains v_{i+1} if Q' is empty.

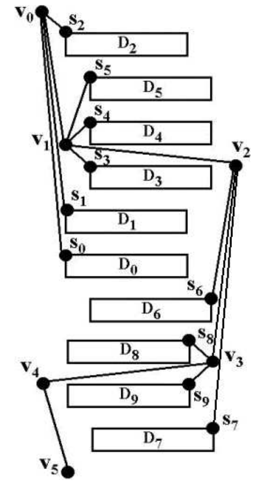
Let $Q'' = s''_1 s''_2 \dots s''_t$ be the (possibly empty) sequence of the children of v_{i-1} that come before v_i in the counterclockwise order of the children of v_{i-1} (see Figure 5(c)). In this sequence, the s''_j's, $1 \leq j \leq t$, are placed in the same order as they occur in the counterclockwise order of the children of v_i. Let D''_j be the drawing of the subtree rooted at s''_j constructed in Step 3. Place $D''_1, D''_2, \dots, D''_r$ in that order one above the other at unit vertical separation from each other, such that D''_1 is at the bottom and D''_t is at the top, the top of D''_t is at the horizontal channel H, and each D''_j is placed one unit to the right of v_{i-1} (see Figure 5(d)).

Let Y_{i+1} be the horizontal channel which is one unit below the bottom of D''_1 if Q'' is not empty, and is the same as H if Q'' is empty.

(c) *v_i is a right-knee, or switch-left node:* These cases are the same as the cases, where v_i is a left-knee node, or switch-right node, respectively, except that "left" is exchanged with "right",



(a)



(b)

Figure 4: (a) A tree T with spine $v_0 v_1 \ldots v_5$. (b) An $O(n \log n)$ area planar straight-line grid drawing of T. In this drawing, v_0 is left-knee node, v_1 is switch-right node, v_2 is right-knee node, v_3 is switch-left node, v_4 is left-knee node, v_5 is switch-right node.

> "counterclockwise" is replaced by "clockwise", and the left-corner drawings of the mirror images of the subtrees rooted at the non-spine children of v_i, constructed in Step 3, are first flipped left-to-right and then are placed in $D(T)$.

Just as for *Algorithm BT-Ordered-Draw*, we can show that the width $w(n)$ of $D(T)$ satisfies the recurrence: $w(n) \leq w(n/2) + 3$. Hence, $w(n) = O(\log n)$. The height of $D(T)$ is trivially at most n. Hence, the area of $D(T)$ is $O(n \log n)$.

Theorem 2 *A tree with n nodes admits an order-preserving planar straight-line grid drawing with $O(n \log n)$ area, $O(\log n)$ width, and height at most n, which can be constructed in $O(n)$ time.*

We can also construct a left-corner drawing $D'(T)$ of T, such that $D'(T)$ has height $O(\log n)$ and width at most n, by first constructing a left-corner drawing of the mirror image of T using Algorithm *Ordered-Draw*, then rotating it clockwise by $90°$, and then flipping it right-to-left. This gives Corollary 2.

Corollary 2 *Let T be a tree with n nodes. Using Algorithm* Ordered-Draw*, we can construct in $O(n)$ time, a left-corner order-preserving planar straight-line grid drawing D of T with $O(n \log n)$ area, such that D has height $O(\log n)$, and width at most n.*

6 Drawing Binary Trees with Arbitrary Aspect Ratio

Let T be a binary tree. We show that, for any user-defined number A, where $2 \leq A \leq n$, we can construct an order-preserving planar straight-line grid drawing of T with $O((n/A) \log A)$ height and $O(A + \log n)$ width.

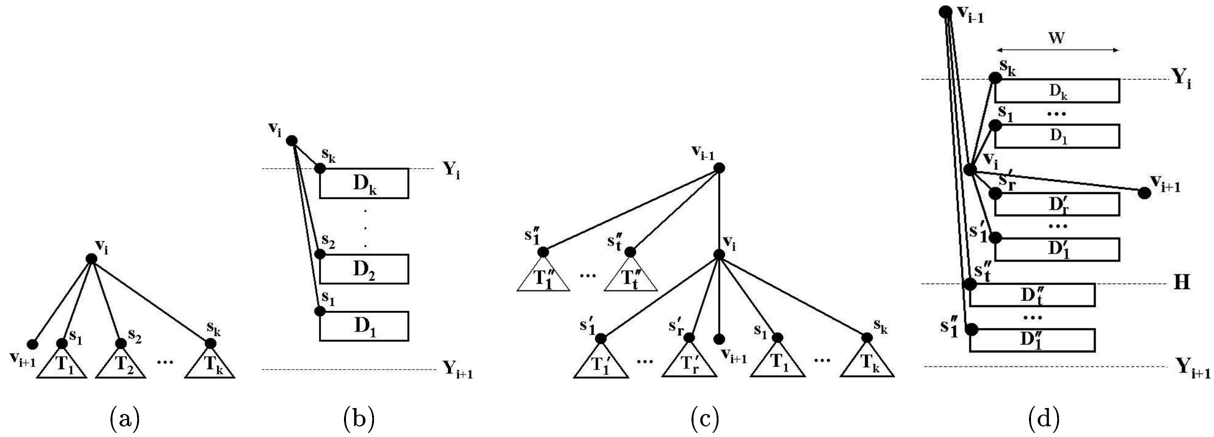


Figure 5: (a) $s_1, s_2, \ldots, s_k$ is the sequence of the children of v_i that come after v_{i+1} in the counterclockwise order of the children of v_i. (b) Placement of v_i, $s_1, s_2, \ldots, s_k$, and $D_1, D_2, \ldots, D_k$ in the case when v_i is a left-knee node. (c) $s_1, \ldots, s_k$ is the sequence of the children of v_i that come after v_{i+1} in the counterclockwise order of the children of v_i. $s'_1, \ldots, s'_k$ is the sequence of the children of v_i that come before v_{i+1} in the counterclockwise order of the children of v_i. $s''_1, \ldots, s''_k$ is the sequence of the children of v_{i-1} that come before v_i in the counterclockwise order of the children of v_{i-1}. (d) Placement of v_i, v_{i+1}, $s_1, \ldots, s_k$, $D_1, \ldots, D_k$, $s'_1, \ldots, s'_k$, $D'_1, \ldots, D'_k$, $s''_1, \ldots, s''_k$, $D''_1, \ldots, D''_k$ in the case when v_i is a switch-right node.

Thus, by setting the value of A, users can control the aspect ratio of the drawing. This result also implies that we can construct such a drawing with area $O(n \log\log n)$ by setting $A = \log n$.

Our algorithm combines the approach of [1] for constructing non-upward non-order-preserving drawings of binary trees with arbitrary aspect ratio with our approach for constructing order-preserving drawings given in Sections 4 and 5. We will also use the following generalization of Lemma 3 of [1]:

Lemma 1 *Suppose $A > 1$, and f is a function such that:*

- *if $n \leq A$, then $f(n) \leq 1$; and*
- *if $n > A$, then $f(n) \leq f(n^*) + f(n^+) + f(n'') + 1$ for some $n^*, n^+, n'' \leq n - A$ with $n^* + n^+ + n'' \leq n$.*

Then, $f(n) < 6n/A - 2$ for all $n > A$.

Proof: The proof is by induction over n, with the base case being $n = A + 1$.

If $n = A + 1$, then $n^*, n^+, n'' \leq A$. Hence, $f(n^*), f(n^+), f(n'') \leq 1$. Hence, $f(n) \leq 1 + 1 + 1 + 1 = 4 < 6n/A - 2$.

Now we prove the induction. Suppose $f(m) < 6m/A - 2$ for all $m \leq n - 1$. Consider $f(n)$. We have four cases:

- $n^*, n^+, n'' \leq A$: Then, $f(n^*), f(n^+), f(n'') \leq 1$. Hence, $f(n) \leq 1 + 1 + 1 + 1 = 4 < 6n/A - 2$.

- Exactly two of n^*, n^+, and n'' have values less than or equal to A: Assume without loss of generality that $n^*, n^+ \le A$ and $n'' > A$. Then, $f(n^*), f(n^+) \le 1$, and $f(n'') < 6n''/A - 2 \le 6(n-A)/A - 2 = 6n/A - 6 - 2 = 6n/A - 8$. Hence, $f(n) \le 1 + 1 + 6n/A - 8 + 1 = 6n/A - 5 < 6n/A - 2$.

- Exactly one of n^*, n^+, and n'' has value less than or equal to A: Assume without loss of generality that $n^* \le A$, and $n^+, n'' > A$. Then, $f(n^*) \le 1$, $f(n^+) + f(n'') < 6n^+/A - 2 + 6n''/A - 2 = 6(n^+ + n'')/A - 4 < 6n/A - 4$. Hence, $f(n) < 1 + 6n/A - 4 + 1 = 6n/A - 2$.

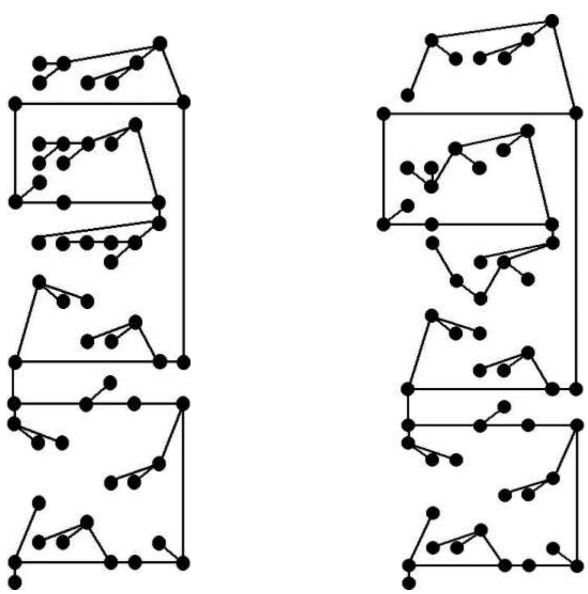
- $n^*, n^+, n'' > A$: $f(n) = f(n^*) + f(n^+) + f(n'') + 1 < 6n^*/A - 2 + 6n^+/A - 2 + 6n''/A - 2 + 1 = 6(n^* + n^+ + n'')/A - 5 \le 6n/A - 5 < 6n/A - 2$.

□

An order-preserving planar straight-line grid drawing of a binary tree T is called a *feasible drawing*, if the root of T is placed on the left boundary and no node of T is placed between the root and the upper-left corner of the enclosing rectangle of the drawing. Note that a left-corner drawing is also a feasible drawing.

We now describe our algorithm, which we call *Algorithm BDAAR*, for drawing a binary tree T with arbitrary aspect ratio. Let m be the number of nodes in T. Let $2 \le A \le m$ be any number given as a parameter to *Algorithm BDAAR*.

Figure 6(a) and Figure 6(b) show the drawings of the tree of Figure 1(a) constructed by Algorithm *BDAAR* with $A = \sqrt{m}$ and using Corollary 1, and Corollary 2, respectively.



(a) (b)

Figure 6: Drawings of the tree with $n = 57$ nodes of Figure 1(a) constructed by Algorithm *BDAAR* with $A = \sqrt{m} = \sqrt{57} = 7.55$ and using: (a) Corollary 1, and (b) Corollary 2, respectively.

Like Algorithm *BT-Ordered-Draw* of Section 4, *Algorithm BDAAR* is also a recursive algorithm. In each recursive step, it also constructs a feasible drawing of a subtree T' of T. If T' has at most A nodes in it, then it constructs a left-corner drawing of T' using Corollary 1 or Corollary 2, such that the drawing has width at most n and height $O(\log n)$, where n is the number of nodes in T'. Otherwise, i.e., if T' has more than A nodes in it, then it constructs a feasible drawing of T' as follows:

1. Let $P = v_0 v_1 v_2 \dots v_q$ be a spine of T'.
2. Let n_i be the number of nodes in the subtree of T' rooted at v_i. Let v_k be the vertex of P with the smallest value for k such that $n_k > n - A$ and $n_{k+1} \leq n - A$ (since T' has more than A nodes in it and $n_0, n_1, \dots, n_q$ is a strictly decreasing sequence of numbers, such a k exists).
3. for each i, where $0 \leq i \leq k-1$, denote by T_i, the subtree rooted at the non-spine child of v_i (if v_i does not have any non-spine child, then T_i is the empty tree, i.e., the tree with no nodes in it). Denote by T^* and T^+, the subtrees rooted at the non-spine children of v_k and v_{k+1}, respectively, denote by T'', the subtree rooted at v_{k+1}, and denote by T''', the subtree rooted at v_{k+2} (if v_k and v_{k+1} do not have non-spine children, and $k+1 = q$, then T^*, T^+, and T''' are empty trees). For simplicity, in the rest of the algorithm, we assume that T^*, T^+, T''', and each T_i are non-empty. (The algorithm can be easily modified to handle the cases, when T^*, T^+, T''', or some T_i's are empty).
4. Place v_0 at origin.
5. We have two cases:
   - $k = 0$: Recursively construct a feasible drawing D^* of T^*. Recursively construct a feasible drawing D^+ of the mirror image of T^+. Recursively construct a feasible drawing D''' of the mirror image of T'''. Let s_0 be the root of T^* and s_1 be the root of T^+.

     T' is drawn as shown in Figure 7(a,b,c,d). If s_0 is the left child of v_0, then place D^* one unit below v_0 with its left boundary aligned with v_0 (see Figure 7(a,c)). If s_0 is the right child of v_0, then place D^* one unit above and one unit to the right of v_0 (see Figure 7(b,d)). Let W^*, W^+, and W''' be the widths of D^*, D^+, and D''', respectively. v_1 is placed in the same horizontal channel as v_0 to its right at distance $\max\{W^* + 1, W^+ + 1, W''' - 1\}$ from it. Let B_0 and C_0 be the lowest and highest horizontal channels, respectively, occupied by the subdrawing consisting of v_0 and D^*. If s_1 is the left child of v_1, then flip D^+ left-to-right and place it one unit below B_0 and one unit to the left of v_1 (see Figure 7(a,b)). If s_1 is the right child of v_1, then flip D^+ left-to-right, and place it one unit above C_0 and one unit to the left of v_1 (see Figure 7(c,d)). Let B_1 be the lowest horizontal channel occupied by the subdrawing consisting of v_0, D^*, v_1 and D^+. Flip D''' left-to-right and place it one unit below B_1 such that its right boundary is aligned with v_1 (see Figure 7(a,b,c,d)).

- $k > 0$: For each T_i, where $0 \le i \le k-1$, construct a left-corner drawing D_i of T_i using Corollary 1 or Corollary 2.

  Recursively construct feasible drawings D^* and D'' of the mirror images of T^* and T'', respectively. T' is drawn as shown in Figure 8(a,b,c,d). If T_0 is rooted at the left child of v_0, then D_0 is placed one unit below and with the left boundary aligned with v_0. If T_0 is rooted at the right child of v_0, then D_0 is placed one unit above and one unit to the right of v_0. Each D_i and v_i, where $1 \le i \le k-1$, are placed such that:

  - v_i is in the same horizontal channel as v_{i-1}, and is one unit to the right of D_{i-1}, and
  - if T_i is rooted at the left child of v_i, then D_i is placed one unit below v_i with its left boundary aligned with v_i, otherwise (i.e., if T_i is rooted at the right child of v_i) D_i is placed one unit above and one unit to the right of v_i.

  Let B_{k-1} and C_{k-1} be the lowest and highest horizontal channels, respectively, occupied by the subdrawing consisting of $v_0, v_1, v_2, \ldots, v_{k-1}$ and $D_0, D_1, D_2, \ldots, D_{k-1}$. Let d be the horizontal distance between v_0 and the right boundary of the subdrawing consisting of $v_0, v_1, v_2, \ldots, v_{k-1}$ and $D_0, D_1, D_2, \ldots, D_{k-1}$. Let W^* and W'' be the widths of D^* and D'', respectively.

  v_k is placed to the right of v_{k-1} in the same horizontal channel as it, such that the horizontal distance between v_k and v_0 is equal to $\max\{W'' - 1, W^* + 1, d + 1\}$. If T^* is rooted at the left-child of v_k, then D^* is flipped left-to-right and placed one unit below B_{k-1} and one unit left of v_k (see Figure 8(a,b)). If T^* is rooted at the right-child of v_k, then D^* is flipped left-to-right and placed one unit above C_{k-1} and one unit to the left of v_k (see Figure 8(c,d)) . Let B_k be the lowest horizontal channel occupied by the subdrawing consisting of $v_1, v_2, \ldots, v_k$, and $D_1, D_2, \ldots, D_{k-1}, D^*$. D'' is flipped left-to-right and placed one unit below B_k, such that its right boundary is aligned with v_k (see Figure 8(b,d)).

Let m_i be the number of nodes in T_i, where $0 \le i \le k-1$. From Corollaries 1 and 2, the height of each D_i is $O(\log m_i)$ and width at most m_i. Total number of nodes in the partial tree consisting of $T_0, T_1, \ldots, T_{k-1}$ and $v_0, v_1, \ldots, v_{k-1}$ is at most $A - 1$. Hence, the height of the subdrawing consisting of $D_0, D_1, \ldots, D_{k-1}$ and $v_0, v_1, \ldots, v_{k-1}$ is $O(\log A)$ and width is at most $A - 1$ (see Figure 8).

Suppose T', T^*, T^+, T'', and T''' have n, n^*, n^+, n'', and n''' nodes, respectively. If we denote by $H(n)$ and $W(n)$, the height and width of the drawing of T' constructed by Algorithm $BDAAR$, then:

$$\begin{aligned}
H(n) &= H(n^*) + H(n^+) + H(n''') + 1 \text{ if n} > \text{A and k} = 0 \\
&= H(n^*) + H(n^+) + H(n''') + O(\log A) \\
H(n) &= H(n^*) + H(n'') + O(\log A) \text{ if n} > \text{A and k} > 0 \\
H(n) &= O(\log A) \text{ if n} \le \text{A}
\end{aligned}$$

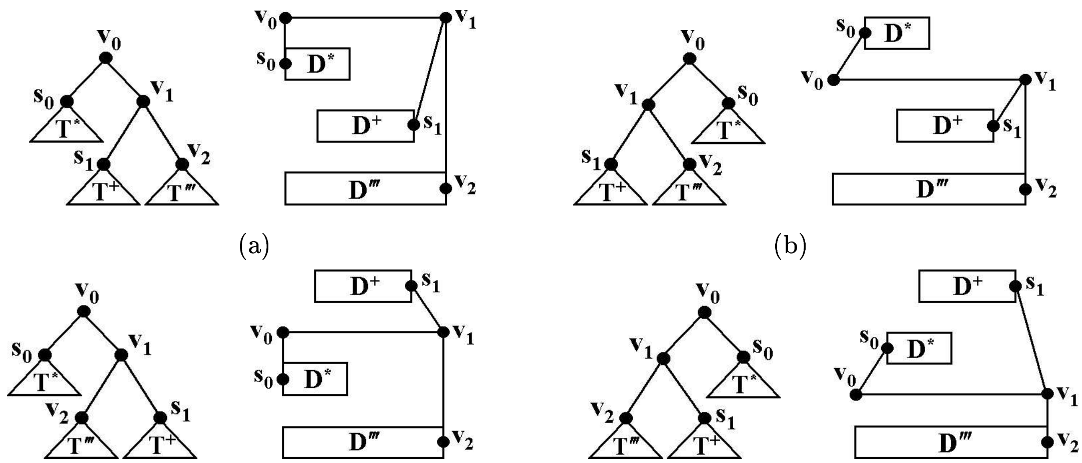


Figure 7: Case $k = 0$: (a) s_0 is the left child of v_0 and s_1 is the left child of v_1. (b) s_0 is the right child of v_0 and s_1 is the left child of v_1. (c) s_0 is the left child of v_0 and s_1 is the right child of v_1. (d) s_0 is the right child of v_0 and s_1 is the right child of v_1.

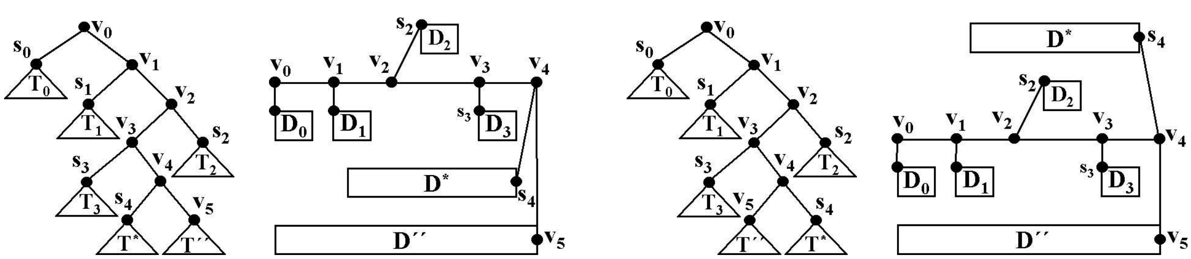


Figure 8: Case $k > 0$: Here $k = 4$, s_0, s_1, and s_3 are the left children of v_0, v_1, and v_3 respectively, s_2 is the right child of v_2, T_0, T_1, T_2, and T_3 are the subtrees rooted at v_0, v_1, v_2, and v_3 respectively. Let s_4 be the root of T^*. (a) s_4 is left child of v_4. (b) s_4 is the right child of v_4.

Since $n^*, n^+, n'', n''' \leq n - A$, from Lemma 1, it follows that $H(n) = O(\log A)(6n/A - 2) = O((n/A)\log A)$. Also we have that:

$$\begin{aligned} W(n) &= \max\{W(n^*)+2, W(n^+)+2, W(n''')\} \text{ if n > A and k = 0} \\ W(n) &= \max\{A, W(n^*)+2, W(n'')\} \text{ if n > A and k > 0} \\ W(n) &\leq A \text{ if n} \leq \text{A} \end{aligned}$$

Since, $n^*, n^+, n^* \leq n/2$, and $n'', n''' \leq n - A < n - 1$, we get that $W(n) \leq \max\{A, W(n/2)+2, W(n-1)\}$. Therefore, $W(n) = O(A + \log n)$. We therefore get the following theorem:

Theorem 3 *Let T be a binary tree with n nodes. Let $2 \leq A \leq n$ be any number. T admits an order-*

preserving planar straight-line grid drawing with width $O(A+\log n)$, height $O((n/A)\log A)$, and area $O((A+\log n)(n/A)\log A) = O(n \log n)$, which can be constructed in $O(n)$ time.

Setting $A = \log n$, we get that:

Corollary 3 *An n-node binary tree admits an order-preserving planar straight-line grid drawing with area $O(n \log\log n)$, which can be constructed in $O(n)$ time.*

References

[1] T. Chan, M. Goodrich, S. Rao Kosaraju, and R. Tamassia. Optimizing area and aspect ratio in straight-line orthogonal tree drawings. *Computational Geometry: Theory and Applications*, 23:153–162, 2002.

[2] T. M. Chan. A near-linear area bound for drawing binary trees. In *Proc. 10th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 161–168, 1999.

[3] P. Crescenzi, G. Di Battista, and A. Piperno. A note on optimal area algorithms for upward drawings of binary trees. *Comput. Geom. Theory Appl.*, 2:187–200, 1992.

[4] P. Crescenzi and P. Penna. Strictly-upward drawings of ordered search trees. *Theoretical Computer Science*, 203(1):51–67, 1998.

[5] P. Crescenzi, P. Penna, and A. Piperno. Linear-area upward drawings of AVL trees. *Comput. Geom. Theory Appl.*, 9:25–42, 1998. (special issue on Graph Drawing, edited by G. Di Battista and R. Tamassia).

[6] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing.* Prentice Hall, Upper Saddle River, NJ, 1999.

[7] A. Garg, M. T. Goodrich, and R. Tamassia. Planar upward tree drawings with optimal area. *Internat. J. Comput. Geom. Appl.*, 6:333–356, 1996.

[8] A. Garg and A. Rusu. Straight-line drawings of general trees with linear area and arbitrary aspect ratio. In *Proceedings 2003 International Conference on Computational Science and Its Applications (ICCSA 2003).* To appear.

[9] A. Garg and A. Rusu. Straight-line drawings of binary trees with linear area and arbitrary aspect ratio. In *Graph Drawing (GD'02)*, volume 2528 of *Lecture Notes in Computer Science*, pages 320–331. Springer-Verlag, 2002.

[10] C.-S. Shin, S. K. Kim, and K.-Y. Chwa. Area-efficient algorithms for upward straight-line tree drawings. In *Proc. 2nd Ann. Internat. Conf. Computing and Combinatorics*, volume 1090 of *Lecture Notes Comput. Sci.*, pages 106–116. Springer-Verlag, 1996.

[11] C.-S. Shin, S.K. Kim, S.-H. Kim, and K.-Y. Chwa. Area-efficient algorithms for straight-line tree drawings. *Comput. Geom. Theory Appl.*, 15:175–202, 2000.

[12] L. Trevisan. A note on minimum-area upward drawing of complete and Fibonacci trees. *Inform. Process. Lett.*, 57(5):231–236, 1996.