A note on optimal area algorithms for upward drawings of binary trees

P. Crescenzi

Dipartimento di Matematica Pura ed Applicata, Università dell’Aquila, Via Vetoio, I-67010 Coppito, L’Aquila, Italy

G. Di Battista*

Dipartimento di Informatica e Sistemistica, Università di Roma ‘La Sapienza’, Via Salaria 113, I-00198 Roma, Italy

A. Piperno

Dipartimento di Scienze dell’Informazione, Università di Roma ‘La Sapienza’, Via Salaria 113, I-00198 Roma, Italy

Communicated by Mikhail Atallah
Submitted 9 August 1991
Accepted 1 May 1992

Abstract


The goal of this paper is to investigate the area requirements for upward grid drawings of binary trees. First, we show that there is a family of binary trees with n vertices that require $\Omega(n \log n)$ area; this bound is tight to within a constant factor, i.e. any binary tree with n vertices can be drawn in $O(n \log n)$ area. Then we present an algorithm for constructing an upward drawing of a complete binary tree with n vertices in $O(n)$ area, and, finally, we extend this result to the drawings of Fibonacci trees.

Keywords. Graph drawing; upward drawing; area requirement.

1. Introduction

Algorithms for constructing readable drawings of graphs are receiving an

* The work of this author has been partially supported by Progetto Finalizzato ‘Sistemi Informatici e Calcolo Parallelo’, and by ESPRIT II BRA ‘Algorithms and Complexity’.

0925-7721/92/$05.00 © 1992 — Elsevier Science Publishers B.V. All rights reserved
increasing theoretical and practical interest. A list of applications that need to automatically produce a drawing of a graph include algorithm animation, circuit schematics, diagrams for information systems analysis and design, user interfaces, and VLSI design.

In the graph drawing field, a special attention is given to algorithms for drawing binary trees. We refer to directed rooted unordered trees where each edge is directed from the child to its parent.

For binary trees the most readable drawing is an upward drawing, that is, a drawing such that, for any edge \((u, v)\), \(u\) is placed below \(v\), edges are represented with straight-line segments, and the drawing does not contain crossings between edges (see Fig. 1 where arrows are omitted because they are implied by the upward requirement).

The goal of this paper is to investigate the area requirements for upward drawings of binary trees and its contribution can be summarized as follows. First, we show that there is a family of binary trees with \(n\) vertices that require \(\Omega(n \log n)\) area to be upward drawn. Moreover, this bound turns to be tight to within a constant factor, i.e. any binary tree with \(n\) vertices can be drawn in \(O(n \log n)\) area.

Then we present an algorithm for constructing an upward drawing of a complete binary tree with \(n\) vertices in \(O(n)\) area, and, finally, we extend the latter result to the drawings of Fibonacci trees.

1.1. Related results

Several references on graph drawing algorithms can be found in [23, 8].

Algorithms for constructing upward drawings of binary trees have been presented in [28, 27, 17]. In particular in [17] a simple and practical bottom-up algorithm is given that allows to produce in linear time eumorphous upward drawings. Informally speaking, an eumorphous upward drawing is a drawing such that symmetries and isomorphisms of subtrees are nicely represented. Supowit and Reingold [22] have investigated the complexity of producing aesthetically
pleasing upward drawings of binary trees. Namely, they have shown that minimizing the width of eumorphous upward drawings of binary trees with vertices having integer coordinates is NP-hard. Other algorithms for constructing aesthetically pleasing drawings of trees are presented in [16].

More generally, the problem of constructing upward drawings has been investigated in the fields of the acyclic directed graphs (DAGs) and of the ordered sets (surveys on drawing techniques in the latter field are presented in [18, 19]). Combinatorial characterizations of the class of DAGs that have an upward drawing are given in [13, 6, 24]. Algorithms for testing if a DAG has an upward drawing are given in [5] (for bipartite DAGs), in [12] (for single source DAGs), and in [1] (for triconnected DAGs).

Recently, it has been tackled the problem of the area requirements for upward drawings. In this context we have surprising results for planar graphs. In [7] it was shown that there exists a family of planar acyclic DAGs with \( n \) vertices that require \( \Omega(2^n) \) area for any upward drawing. This result sharply contrasts with the fact that any planar graph with \( n \) vertices can be drawn on the grid, using straight-line segments for the edges, without edge crossings, in \( O(n^3) \) area [3, 4, 21].

Algorithms for drawing binary trees have been also deeply investigated (with different purposes) in the VLSI field (several results are reviewed in [25]). Valiant [26] has shown that each binary tree can be drawn with vertices at integer coordinates in linear area. This result contrasts with the lower bound we will prove when the upward condition is considered.

Ruzzo and Snyder [20] have presented several bounds on the area of drawings of binary trees with integer coordinates, allowing bends along the edges, and such that the ordering of the edges is preserved.

Brent and Kung [2] tackled the problem of constructing drawings of complete binary trees with all the leaves on the boundary of the drawing and they have shown a \( \Omega(n \log n) \) lower bound on the area of such drawings.

1.2. Preliminary definitions and notation

We refer to directed rooted unordered binary trees, in short binary trees, where each edge is directed from the child to its parent.

We denote by \( e \) the empty binary tree. Given two binary trees \( t_1 \) and \( t_2 \), we denote by \( t_1 \oplus t_2 \) the binary tree whose immediate subtrees are \( t_1 \) and \( t_2 \).

In particular, \( c_i \) denotes the complete binary tree of height \( i \) while \( F_i \) denotes the Fibonacci tree of height \( i \) [15, 9, 10]. We recall that such trees are inductively defined in the following way:

\[
c_0 = e, \quad c_i = c_{i-1} \oplus c_{i-1} \quad \text{for } i \geq 1,
\]

and

\[
F_0 = e, \quad F_1 = e \oplus e, \quad F_i = F_{i-1} \oplus F_{i-2} \quad \text{for } i > 1.
\]
Let \( n_F(i) \) be the number of nodes of \( F_i \). From the definition, it follows that:

\[
\begin{align*}
n_F(0) &= 0, \\
n_F(1) &= 1, \\
n_F(i) &= n_F(i - 1) + n_F(i - 2) + 1 \quad \text{for } i > 1.
\end{align*}
\]

Furthermore, it is easy to prove by induction on \( i \) that \( n_F(i) = f_{i+2} - 1 \), where \( f_i \) denotes the \( i \)th Fibonacci number.

An upward drawing \( \delta_i \) of a binary tree \( t \) is a drawing of \( t \) such that:

1. vertices are points with integer coordinates;
2. edges are straight-line segments;
3. a vertex has the ordinate greater than that of its parent—we are thus assuming that the y-axis is downward oriented.

We denote by \( \rho_{\delta_i} \) the grid point of \( \delta_i \) corresponding to the root of \( t \).

The height (respectively, width) of \( \delta_i \) is the height (respectively, width) of the smallest isothetic rectangle bounding \( \delta_i \), and it will be denoted by \( h_{\delta_i} \) (respectively, \( w_{\delta_i} \)). We adopt the convention that the height and width of such a rectangle are measured by the number of grid points, so that any drawing of a nonempty binary tree has both height and width greater than zero. The area of an upward drawing \( \delta_i \) is then defined as \( h_{\delta_i} w_{\delta_i} \). Fig. 1 shows an example of an upward drawing of a binary tree such that \( \rho_{\delta_1} = (0, 3) \), \( h_{\delta_1} = 7 \) and \( w_{\delta_1} = 11 \).

2. A lower bound on tree layouts

In this section we prove an \( \Omega(n \log n) \) lower bound on the area required to upward draw binary trees. In the next section we will show that such a bound is tight to within a constant factor.

**Theorem 1.** An infinite class of binary trees exists requiring \( \Omega(n \log n) \) area to be upward drawn.

**Proof.** For any integer \( h > 0 \), let \( n = 2^h - 1 \) and let \( t_n = p_n \oplus c_h \) where \( p_n \) is the list of \( n \) nodes and \( c_h \) is the complete binary tree of height \( h \) (see Fig. 2). Obviously, any upward drawing of \( p_n \) has height at least \( n \).

Now we prove that any upward drawing of \( c_h \) has width at least \( h \). Namely, let \( w_h \) be the width of the upward drawing of \( c_h \) with minimum width: we prove that \( w_h \geq w_{h-1} + 1 \); observing that \( w_1 = 1 \) we can prove the claim.

Let \( \delta_i^{\oplus} \) be the upward drawing of \( c_h \) with minimum width. By removing from \( \delta_i^{\oplus} \) the root and its incident edges we have two upward drawings of \( c_{h-1} \). Denote such drawings by \( \delta_1 \) and \( \delta_2 \). Two cases are possible.

**Case 1:** \( w_{\delta_1} < w_{\delta_2} \) (without loss of generality).

In this case the proof is trivial. In fact \( w_{h-1} \leq w_{\delta_1} < w_{\delta_1} + 1 \leq w_{\delta_2} \leq w_h \).

**Case 2:** \( w_{\delta_1} = w_{\delta_2} \).
If $w_h > w_{\delta_1}$ then the proof is again trivial. Else ($w_h = w_{\delta_1}$) suppose, for a contradiction, that $w_h = w_{h-1}$; it follows that $w_{\delta_1} = w_{\delta_2} = w_h = w_{h-1}$. Consider vertices (see Fig. 3) $x'_{\delta_1}$ and $x''_{\delta_1}$ of $\delta_1$ with minimum and maximum $x$-coordinate (respectively). Let $p$ be the path (in general unoriented) of $\delta_1$ between $x'_{\delta_1}$ and $x''_{\delta_1}$. Let $\mu'$ be the vertical line through $x'_{\delta_1}$ and let $\mu''$ be the vertical line through $x''_{\delta_1}$. Let $\nu$ be the horizontal line through the root of $\delta_1$. We have that, in order to have $w_h = w_{\delta_1}$, $\delta_2$ is placed inside the region delimited by $\mu'$ and $\mu''$. Moreover, in order to avoid crossings, $\delta_2$ has to be placed entirely above or below $p$. Suppose...
(the other case is analogous) that $\delta_2$ is placed below $p$. But, the root of $\delta''_n$, in order to be connected with a straight line segment with the roots of $\delta_1$ and $\delta_2$ without creating crossings and with the upward constraint, has to be placed inside the region to the left of $\mu'$ and above $\nu$ (exclusive) or inside the region to the right of $\mu''$ and above $\nu$ (exclusive). Thus, $w_h \geq w_{h-1} + 1$, a contradiction.

It thus comes out that, for any upward drawing $\delta_n$, $h_{\delta_n} \approx n$ and $w_{\delta_n} \approx h = \lceil \log(2^n - 1) \rceil = \lceil \log n \rceil$. Since the number of nodes of $t_n$ is $2n + 1$, then $\delta_n$ requires $\Omega(n \log n)$ area. □

3. Upward drawings and h-v drawings

We now introduce the notion of h-v drawing. Such drawings will be used extensively in the paper to prove upper bounds on the area requirements. Intuitively, an h-v drawing consists of rightward-horizontal and downward-vertical straight-line segments only, in such a way that the root of the tree is placed at the topmost leftmost corner of the grid, as shown by the example in Fig. 4(a).

More precisely, the h-v drawing of the empty binary tree is the empty drawing $\varepsilon$. Given two h-v drawings $\delta_1$ and $\delta_2$ of two binary trees $t_1$ and $t_2$, an h-v drawing $\delta_t$ of $t = t_1 \oplus t_2$ can be one of the following (see Fig. 5):

1. $\rho_{\delta_t} = (0, 0)$; if $t_1 \neq e$, then $\delta_1$ is translated in such a way that $\rho_{\delta_1} = (0, 1)$ and $\rho_{\delta_1}$ is connected to $\rho_{\delta_t}$ by a vertical segment; similarly, if $t_2 \neq e$, then $\delta_2$ is translated in such a way that $\rho_{\delta_2} = (w_{\delta_1} + 1, 0)$ and $\rho_{\delta_2}$ is connected to $\rho_{\delta_t}$ by a horizontal segment;

2. $\rho_{\delta_t} = (0, 0)$; if $t_1 \neq e$, then $\delta_1$ is translated in such a way that $\rho_{\delta_1} = (0, h_{\delta_2} + 1)$ and $\rho_{\delta_1}$ is connected to $\rho_{\delta_t}$ by a vertical segment; similarly, if $t_2 \neq e$, then $\delta_2$ is translated in such a way that $\rho_{\delta_2} = (1, 0)$ and $\rho_{\delta_2}$ is connected to $\rho_{\delta_t}$ by a horizontal segment.

The drawing produced in the first case will be denoted by $\delta_t \ominus \delta_{t_2}$, while the drawing produced in the second case will be denoted by $\delta_t \ominus \delta_{t_2}$.

![Fig. 4. An h-v drawing and an upward drawing of the tree of Fig. 1.](image-url)
The following fact shows that h-v drawings are powerful tools to deal with upward drawings.

**Fact 1.** Let $\delta_t$ be an h-v drawing of a binary tree $t$. There exists an upward drawing $\delta'_t$ of $t$ whose area is at most twice the area of $\delta_t$.

**Proof.** The drawing $\delta'_t$ is obtained from $\delta_t$ by means of the following transformations (see Fig. 4(b)):

- for every node having coordinates $(x, y)$ in $\delta_t$, draw the node $(x, x + y)$ in $\delta'_t$;

- if a segment connects the nodes $(x, y)$ and $(x', y')$ in $\delta_t$, draw the segment connecting the nodes $(x, x + y)$ and $(x', x' + y')$ in $\delta'_t$.

The drawing $\delta'_t$ obtained with the above construction is obviously upward and isomorphic to $\delta_t$, i.e. $\delta_t$ and $\delta'_t$ are drawings of the same tree. In order to prove that $\delta'_t$ does not contain crossing edges, we first show that it contains vertical and 45° sloped straight-line segments only. In fact, any nonvertical segment $s$ in $\delta'_t$ that connects two nodes $(x_1, x_1 + y_1)$ and $(x_2, x_2 + y_2)$ corresponds to a horizontal segment in $\delta_t$ connecting the nodes $(x_1, y_1)$ and $(x_2, y_2)$; hence, $y_1 = y_2$. This, in turn, implies that the slope of $s$ is equal to 45°. Let us now suppose that $\delta'_t$ contains two crossing edges $s_1$ and $s_2$. If both $s_1$ and $s_2$ are vertical (respectively, diagonal) segments, then they overlap. This implies that the corresponding segments in $\delta_t$ overlap too, contradicting the hypothesis that $\delta_t$ is an h-v drawing.

On the other hand, let $s_1$ be a diagonal segment connecting the nodes $(x'_1, x'_1 + y'_1)$ and $(x'_2, x'_2 + y'_2)$ and $s_2$ be a vertical segment connecting the nodes $(x'_2, x'_2 + y'_2)$ and $(x'_3, x'_2 + y'_3)$. Observe that, by construction, $y'_1 = y'_2$, since $s_1$ corresponds to a horizontal segment in $\delta_t$. It also comes out that the crossing point of $s_1$ and $s_2$ has coordinates $(x_2, x_2 + y_1)$; this implies that $x_1' \leq x_2 \leq x_3'$ and $y_2' \leq y_1' \leq y_3'$, that is, $\delta_t$ contains two crossing edges.
In order to complete the proof, we are left to show that $\delta'_i$ has the required area. Since we are considering unordered trees, we can assume, without loss of generality, that $w_{\delta_i} \leq h_{\delta_i}$. It follows that $w_{\delta'_i} = w_{\delta_i}$ and $h_{\delta'_i} = h_{\delta_i} + w_{\delta_i}$, thus the area of $\delta'_i$ is at most twice the area of $\delta_i$. \qed

As a consequence of the previous fact, from here onwards we shall describe algorithms to construct h-v drawings only. The area upper bounds that we will obtain can be easily interpreted—just doubling them—in terms of area upper bounds for upward drawings.

The following algorithm constructs an h-v drawing of a binary tree.

**Algorithm BT;**

begin
  if $t = e$ then $\delta_i = e$;
  if $t = t_1 \oplus t_2$ and $w_{\delta_{t_1}} \geq w_{\delta_{t_2}}$ then $\delta_i = \delta_{t_1} \odot \delta_{t_2}$;
  if $t = t_1 \oplus t_2$ and $w_{\delta_{t_1}} < w_{\delta_{t_2}}$ then $\delta_i = \delta_{t_2} \odot \delta_{t_1}$;
end.

Fig. 4(a) shows the drawing of the tree of Fig. 1 constructed by Algorithm BT. The following theorem establishes an upper bound on the area of the drawing produced by Algorithm BT.

**Theorem 2.** Given a binary tree $t$ with $n$ nodes, Algorithm BT produces an h-v drawing of $t$ with at most $n(\log n + 1)$ area in linear time and space.

**Proof.** Let $\delta_i$ be the drawing obtained by Algorithm BT. By induction on $n$, we prove that $h_{\delta_i} \leq n$ and $w_{\delta_i} \leq \log n + 1$.

For $n = 1$, the proof is straightforward. Assume now that the bounds are true for any tree with less than $n$ nodes, and let $t = t_1 \oplus t_2$ be a tree with $n$ nodes. Without loss of generality, we assume that $w_{\delta_{t_1}} \geq w_{\delta_{t_2}}$. We distinguish the following cases:

**Case (a):** $t_2 = e$.

By definition of the $\odot$ operation, $h_{\delta_i} = h_{\delta_{t_1}} + 1$ and $w_{\delta_i} = w_{\delta_{t_1}}$; by induction hypothesis, it follows that $h_{\delta_i} \leq n$ and $w_{\delta_i} \leq \log n + 1$;

**Case (b):** $t_2 \neq e$ and $w_{\delta_{t_1}} > w_{\delta_{t_2}}$.

By definition of the $\odot$ operation, $h_{\delta_i} = h_{\delta_{t_1}} + h_{\delta_{t_2}}$ and $w_{\delta_i} = w_{\delta_{t_1}}$; by induction hypothesis, it follows that $h_{\delta_i} \leq n$ and $w_{\delta_i} \leq \log n + 1$;

**Case (c):** $t_2 \neq e$ and $w_{\delta_{t_1}} = w_{\delta_{t_2}} = w$.

By definition of the $\odot$ operation, $h_{\delta_i} = h_{\delta_{t_1}} + h_{\delta_{t_2}}$ and $w_{\delta_i} = w + 1$. Let $n_1$ (respectively, $n_2$) be the number of nodes in $t_1$ (respectively, $t_2$); by induction hypothesis, $n_1 \geq 2^{w-1}$ and $n_2 \geq 2^{w-1}$, and this, in turn, implies that $n \geq 2^{w+1}$, that is, $w + 1 \leq \log n + 1$.

Thus, the area $h_{\delta_i}w_{\delta_i}$ of the drawing is at most $n(\log n + 1)$. 


Finally, Algorithm BT can be trivially implemented in linear time and space.

4. Linear area layouts of complete trees

In this section we give an efficient way to h-v draw a complete binary tree.
In the algorithm of the previous section we made use of the $\Theta$ operation only. In the algorithm we are going to describe, instead, we make use of both the $\Theta$ and the $\Sigma$ operations, in order to alternatively expand the horizontal side or the vertical side of the drawing.

The following algorithm constructs an h-v drawing of $c_i$.

**Algorithm CT;**

begin
if $i = 0$ then $\delta_i = \varepsilon$;
if $i$ is even then $\delta_i = \delta_{i-1} \Theta \delta_{c_{i-1}}$;
if $i$ is odd then $\delta_i = \delta_{c_{i-1}} \Theta \delta_{c_{i-1}}$;
end.

Fig. 6 shows the h-v drawing of $c_8$ produced by Algorithm CT.

**Theorem 3.** Algorithm CT produces an h-v drawing of $c_i$ with linear area in linear time and space.

![The h-v drawing of the complete binary tree of height 8.](image)
Proof. Let \( L_i \) (respectively, \( l_i \)) be the longest (respectively, the shortest) side of \( \delta_c \). By definition of Algorithm CT, we obtain a recurrence that yields both \( L_i \) and \( l_i \) for each \( i \):

\[
\begin{align*}
    l_1 &= L_1 = 1, \\
    l_{i+1} &= L_i + 1, \\
    L_{i+1} &= 2l_i.
\end{align*}
\]

The solutions of such a recurrence are

\[
l_i = \begin{cases} 
    2\frac{i-1}{2} - 1 & \text{if } i \text{ is odd}, \\
    3 \cdot 2\frac{i-2}{2} - 1 & \text{if } i \text{ is even},
\end{cases}
\]

and

\[
L_i = \begin{cases} 
    3 \cdot 2\frac{i-3}{2} - 2 & \text{if } i \text{ is odd}, \\
    2\frac{i-2}{2} - 2 & \text{if } i \text{ is even}.
\end{cases}
\]

This is proved by induction on \( i \). For \( i = 1 \) the proof is straightforward. Assume that the solution is correct for any \( j < i \). If \( i \) is odd, then \( i - 1 \) is even and by the induction hypothesis

\[
l_i = L_{i-1} + 1 = 2^{(i-1+2)/2} - 2 + 1 = 2^{(i+1)/2} - 1,
\]

and

\[
L_i = 2l_{i-1} - 2(3 \cdot 2^{(i-1-2)/2} - 1) = 3 \cdot 2^{(i-1)/2} - 2.
\]

Similarly, we can deal with the case in which \( i \) is even.

Hence, both in the case \( i \) is odd and in the case \( i \) is even, the area \( L_il_i \) of the drawing is at most \( 3 \cdot 2^i \). But that is a drawing for a complete binary tree of height \( i \), which has \( 2^i - 1 \) nodes.

Finally, Algorithm CT can be trivially implemented in linear time and space. \( \Box \)

5. Linear area layouts of Fibonacci trees

In this section we give an algorithm to upward draw in linear area a Fibonacci tree. Such trees—as the complete binary ones—have the nice property that, for any height \( i \), a ‘unique’ \( F_i \) exists.

A trivial way of drawing the Fibonacci tree \( F_i \) is the following: (a) ‘complete’ \( F_i \) by adding fictitious nodes and edges in such a way to obtain \( c_i \), (b) draw \( c_i \) with Algorithm CT, and (c) delete all the fictitious nodes and edges of the drawing. Unfortunately, this algorithm yields a drawing whose area is more than linear in the number of nodes. In fact, such a drawing has area \( O(2^i) \) and it is well known that the ratio between \( 2^i \) and \( n_F(i) \) is not bounded by any constant. Thus, we have to look for a more clever approach.

The algorithms of the previous sections produced exactly one drawing for each
A note on optimal area algorithms

considered tree. The algorithm we are going to describe, instead, produces two different drawings (namely $\delta^A_F$ and $\delta^B_F$) for each Fibonacci tree $F_i$. Roughly speaking, drawings belonging to the classes $A$ and $B$ are alternatively composed in such a way to keep the area requirement linear.

The following algorithm constructs two h-v drawings of $F_i$.

**Algorithm FT;**

begin
  if $i = 0$ then $\delta^A_F = \delta^B_F = \epsilon$;
  if $i = 1$ then $\delta^A_F = \delta^B_F = \epsilon \oplus \epsilon = \epsilon \odot \epsilon$;
  if $i$ is even then
    begin
      $\delta^A_F = \delta^A_{F_{i-1}} \odot \delta^B_{F_{i-2}}$;
      $\delta^B_F = \delta^A_{F_{i-1}} \ominus \delta^A_{F_{i-2}}$;
    end;
  if $i$ is odd then
    begin
      $\delta^A_F = \delta^B_{F_{i-1}} \odot \delta^A_{F_{i-2}}$;
      $\delta^B_F = \delta^A_{F_{i-1}} \ominus \delta^A_{F_{i-2}}$;
    end;
end.

Fig. 7 shows the drawings $\delta^A_F$ (first row) and $\delta^B_F$ (second row) for $i = 0, \ldots, 7$. The following result shows that both $\delta^A_F$ and $\delta^B_F$ are linear area drawings of $F_i$.

$$
\begin{array}{cccccccc}
\epsilon & * & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\epsilon & * & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{array}
$$

Fig. 7. The h-v drawings of the first eight Fibonacci trees.
Theorem 4. Algorithm FT produces two $h$-$v$ drawings of $F_i$ both with linear area in linear time and space.

Proof. We first prove by induction on $i$ that the width and height of the two drawings $\delta^A_F$ and $\delta^B_F$ produced by the algorithm satisfy the rules indicated in Table 1.

For $i = 0, 1$ the proof is straightforward (see Fig. 7). Assume that the rules hold for any Fibonacci tree of height less than $i$. We shall consider only one of the four possible cases introduced by the algorithm (the remaining ones are omitted here, since they can be treated in a similar way). If $\delta^A_F = \delta^A_{F_{i-1}} \odot \delta^B_{F_{i-2}}$ and $i$ is even, then, by the inductive assumption:

$$h_{\delta^A_{F_{i-1}}} = n_F\left(\frac{i}{2}\right), \quad w_{\delta^A_{F_{i-1}}} = n_F\left(\frac{i}{2}\right),$$

and

$$h_{\delta^B_{F_{i-2}}} = n_F\left(\frac{i}{2} - 1\right) + 1, \quad w_{\delta^B_{F_{i-2}}} = n_F\left(\frac{i}{2} - 1\right).$$

Since $\delta^A_{F_{i-1}}$ and $\delta^B_{F_{i-2}}$ are composed by means of the $\odot$ operator, then:

$$h_{\delta^A_F} = n_F\left(\frac{i}{2}\right) + n_F\left(\frac{i}{2} - 1\right) + 1 = n_F\left(\frac{i}{2} + 1\right),$$

and

$$w_{\delta^A_F} = n_F\left(\frac{i}{2}\right).$$

We will now prove that, for any $i$, the area of $\delta^A_F$, that is, $h_{\delta^A_F}w_{\delta^A_F}$ is no more than $2n_F(i)$. If $i$ is odd, that is $i = 2k - 1$, then

$$h_{\delta^A_F}w_{\delta^A_F} = (n_F(k))^2 = (f_{k+2} - 1)^2 = (f_{k+2})^2 - 2f_{k+2} + 1.$$

Table 1
Heights and widths of drawings for Fibonacci trees

<table>
<thead>
<tr>
<th>$i$</th>
<th>$h_{\delta^A_F}$</th>
<th>$w_{\delta^A_F}$</th>
<th>$h_{\delta^B_F}$</th>
<th>$w_{\delta^B_F}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>odd</td>
<td>$n_F\left(\frac{i+1}{2}\right)$</td>
<td>$n_F\left(\frac{i+1}{2}\right)$</td>
<td>$n_F\left(\frac{i+1}{2} + 1\right) - 1$</td>
<td>$n_F\left(\frac{i-1}{2}\right) + 1$</td>
</tr>
<tr>
<td>even</td>
<td>$n_F\left(\frac{i}{2} + 1\right)$</td>
<td>$n_F\left(\frac{i}{2}\right)$</td>
<td>$n_F\left(\frac{i}{2}\right) + 1$</td>
<td>$n_F\left(\frac{i}{2} + 1\right) - 1$</td>
</tr>
</tbody>
</table>
Since \(2n_r(i) = 2f_{2k+1} - 2\) and, for any \(k \geq 1\), \(2f_{k+2} - 3 > 0\), it follows that we are left to prove that \((f_{k+2})^2 \leq 2f_{2k+1}\), i.e.: \((f_{k+1})^2 + (f_k)^2 + 2f_{k+1}f_k \leq 2f_{2k+1}\). It is well known that \(f_{2k+1} = (f_{k+1})^2 + (f_k)^2\) [11]. Thus the previous inequality reduces to \(0 \leq (f_{k+1})^2 + (f_k)^2 - 2f_{k+1}f_k = (f_{k+1} - f_k)^2\) that is trivially satisfied. The case where \(i\) is even can be treated in a similar way.

Finally, Algorithm FT can be trivially implemented in linear time and space.

6. Conclusions

In this paper we have investigated the area requirements for upward drawing of binary trees. First we have shown an \(\Omega(n \log n)\) lower bound. Then by simply combining Theorems 2, 3 and 4 with Fact 1 we have obtained the following results.

**Theorem 5.** The following hold:

1. any binary tree can be upward drawn in \(O(n \log n)\) area, that is, the lower bound is tight;
2. any complete binary tree can be upward drawn in \(O(n)\) area;
3. any Fibonacci tree can be upward drawn in \(O(n)\) area.

As a concluding remark, we observe that the Fibonacci trees are the balanced binary trees with the least number of vertices while the complete binary trees are the balanced binary trees with the most number of vertices. A natural question left open by this paper is whether linear area suffices to upward draw any balanced tree.

Acknowledgements

We are grateful to Roberto Tamassia for many useful discussions and suggestions.

References

200 P. Crescenzi et al.


