C.S. 252
Computational Geometry

Prof. Roberto Tamassia
Sem. II, 1992–1993

Graph Drawing algorithms

Date: April 7, 1993
Scribe: Shamsi Moussavi

1 Cont. planar, straight-line, grid drawing of binary tree in $O(n \log n)$ area

.

In previous lecture, it was shown that for a given binary n-node tree; there exists a planar, upward, straight-line, grid drawing algorithm with $O(n \log n)$ area.
Claim: The precomputation of this algorithm can be done in $O(n)$ time.

Proof: The total number of nodes in left and right subtrees of each node of the tree can be found in $O(n)$ time, by a postorder traverse of the tree. A comparison between the number of nodes in the left and right subtrees of any given node determines the heavy and light paths from that node, which can also be done in $O(n)$ time. Therefore, the total preprocessing time would be $O(n)$ time.

In order to draw the graph, the X and Y coordinates of the nodes must be known.

—The Y coordinate of a node is the number of light edges preceding the node in the path from root to that node, which can be found using a Depth-First search ($O(n)$ time).

—The X coordinate of a node is the number of heavy edges preceeding the node, which is computed in the following manner:

* Visit the root

* Visit the node attached by a light edge first. The X coordinate is the X coordinate of the parent. Record the number of heavy edges.

* Visit the node attached by a heavy edge next. The X coordinate is the last number of the heavy edges plus one. Record this as the highest number of heavy edges.

Figure 1 shows an example binary tree and it's planar, straight-line, grid drawing using this algorithm.

A new research by Garg, Goodrich and Tamassia presents an algorithm that constructs a planar upward grid drawing of any rooted bounded degree tree T with N nodes which has $O(N^\alpha)$ width and $O(N^{1-\alpha})$ height, for a prespecified constant α such that $0 < \alpha < 1$. However, this $O(N)$ area drawing is a polyline drawing.

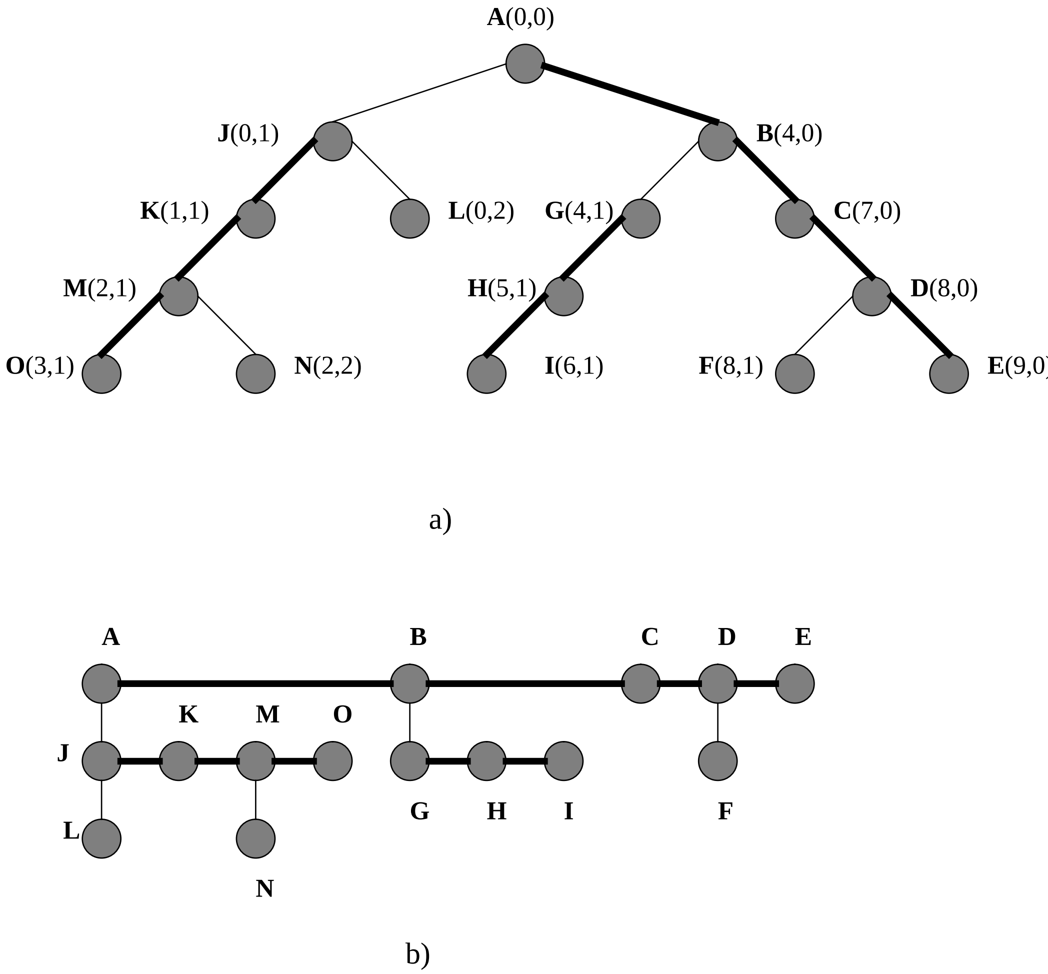

Figure 1: a) The tree with it's light and heavy edges. b) The $O(n \log n)$ planar,straight-line, grid drawing of the tree .

2 H-Tree Layout of a complete Binary Tree

A complete Binary tree can be drawn in area $O(n)$ using a planar, straight-line drawing algorithm called H-Tree. However, this drawing is not upward and the root is always in the middle.

It can be shown that the area of this drawing is in fact $O(n)$. the drawing alternates up-down composition and left-right composition at each step. Figure 2 shows different compositions. So the recurrences for the width and height of two models of compositions are: the up down compositoin

$$h(d) = 1 + 2h(d-1)$$

and

$$w(d) = w(d-1)$$

and right left composition

$$h(d) = h(d-1)$$

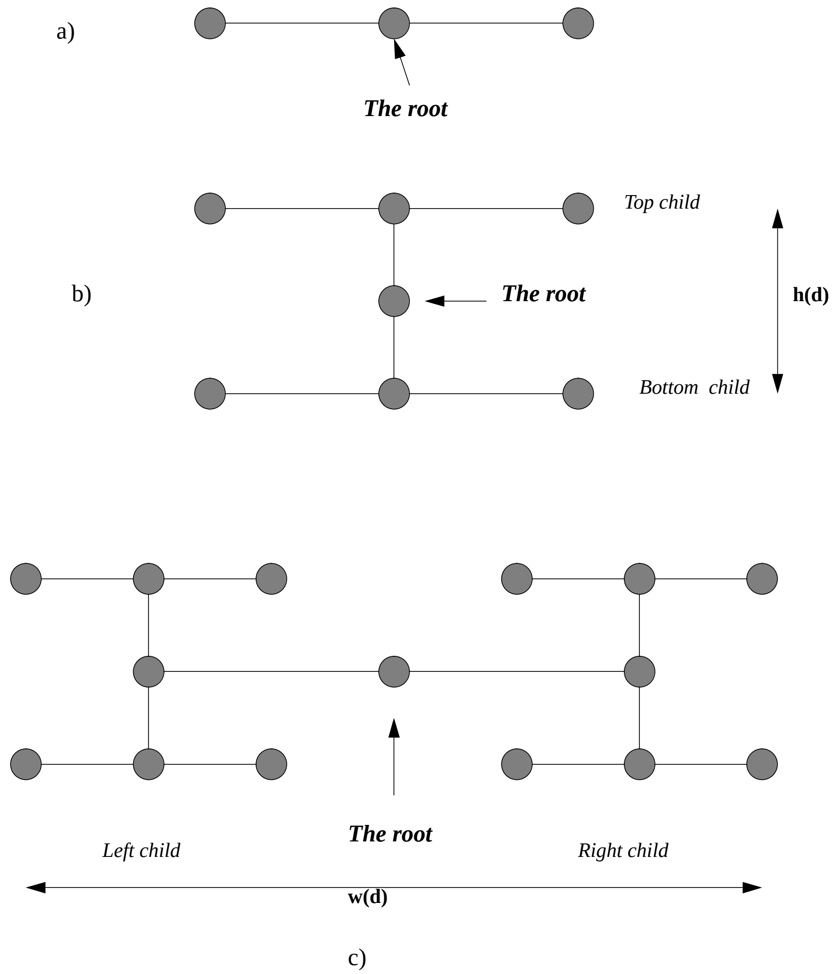

Figure 2: H-tree a)having two childern, b),c) and it's next developments

and

$$w(d) = 1 + 2w(d-1)$$

where d is the depth of the tree. Since the tree is a complete binary tree and by combining the two form of the composition the height and width for the drawing would be:

$$h(d) = 1 + 2h(d-2)$$

and

$$w(d) = 1 + 2w(d-2)$$

. The area complexity of h(d) can be found by expanding the recurrence as follows:

$$\begin{aligned} h(d) &= 1 + 2h(d-2) \\ &= 1 + 2(1 + 2h(d-4)) \\ &= 1 + 2 + 4h(d-4) \\ &= 1 + 2 + 4 + 8h(d-6) \\ &\vdots \\ &= 1 + 2 + \cdots + 2^{i-1} + 2^i h(d-2^i) \end{aligned}$$

when $i = d/2$ the expression can not be expanded anymore, so the result would be:

$$\begin{aligned} h(d) &= 1 + 2 + \cdots + 2^{d/2-1} + 2^{d/2}h(0) \\ &= 2^{d/2} - 1 + 2^{d/2}h(0) \end{aligned}$$

since $2^{d/2} = \sqrt{n}$ and $h(0) = 0 or 1$ then

$$h(d) = O(\sqrt{n})$$

By similar expansion one can get $w(d) = O(\sqrt{n})$, therefore the area of this drawing would be $O(n)$.

Moreover, this algorithm can be modified to get an upward tree. To do this the left child can be put one level below parent at the same X coordinate, and right child one unit to right on the same Y coordinate. this can be done recursively, alternating the left right composition and top down composition at each step. So the graph will grow approximately even in size at both X and Y coordinates. Figure 3 shows the first three steps of this algorithm.

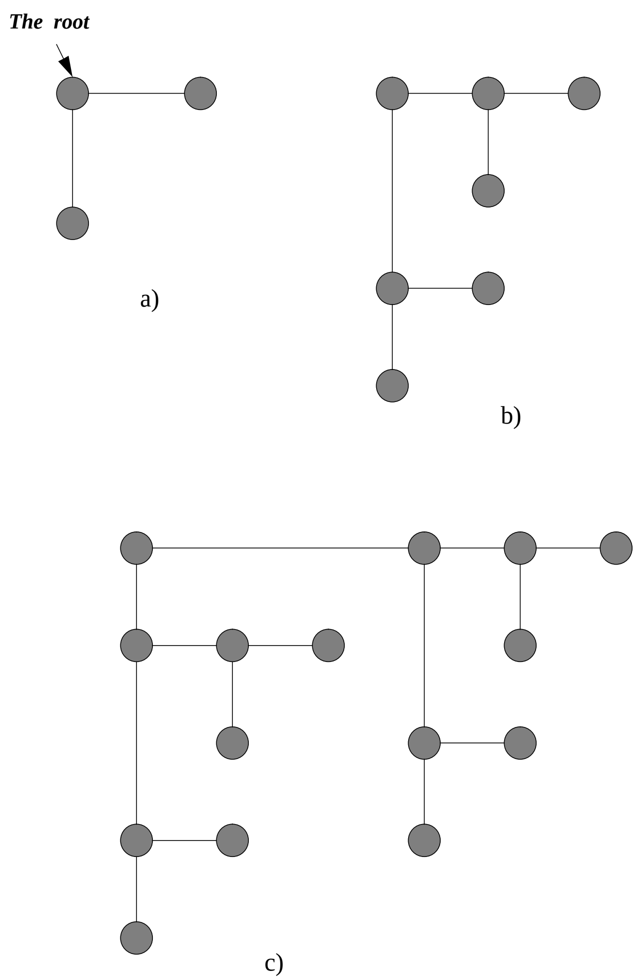

Figure 3: a),b),c) The upward drawing of the tree in figure 2 .

Now the recurrences for the Horizontal and vertical compositions would be:
* Horizontal composition

$$h(l) = 1 + h(l/2)$$

and

$$w(l) = 2w(l/2)$$

* Vertical composition

$$h(l) = 2h(l/2)$$

and

$$w(l) = 1 + w(l/2)$$

So the combination of the two recurrences would give:

$$h(l) = 1 + 2h(l/4)$$

and

$$w(l) = 2 + 2w(l/4)$$

Note that the length of the longest edge is $\sqrt{n}$, since the area is $O(n)$.

An alternative way to show that the area is indeed $O(n)$, is to consider all the grid points on the edges that do not contain a node. When computing the area, the cost of each such a grid point is added to a neigbour node. This way each node will at most pay for 4 grid points including the node itself, since at least one of the neigbours of the node is a node.Therefore, all the grid points of the drawing are being considered and the total cost is $O(n)$, so the area would be $O(n)$. Figure 4 is a demonstration of possible cost assignment.

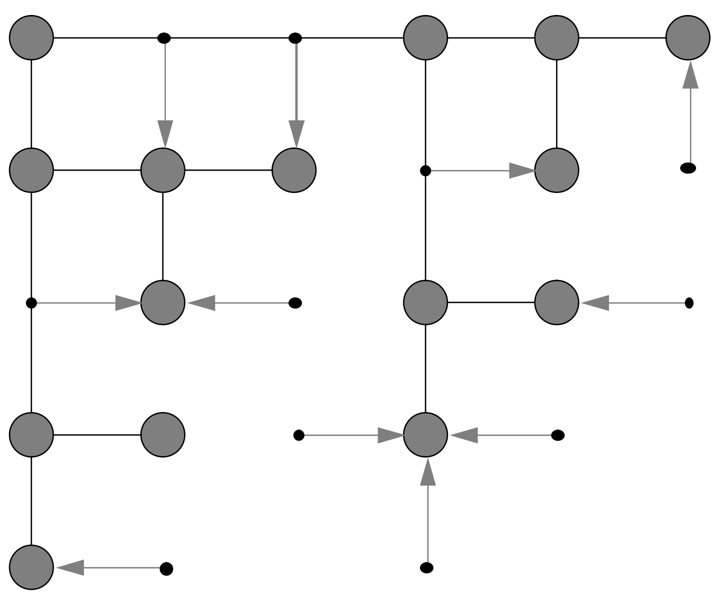

Figure 4: The assignment of costs of empty gird points.

3 Non-binary trees

For non-binary trees, the $O(n)$ area algorithm has to have a fat node to represent each node having more than two children; since the node can be drawn as a spine of some number of nodes equal to the number of children. Then the spine can be represented as a big node, which is shown in figure 5. Since the number of edges only increases by a constant factor, so the area would remain $O(n)$.

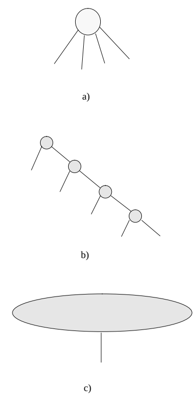

Figure 5: a) the non-binary tree, b) spine representation of the root, c) representation of the node in $O(n)$ drawing.

The other possible way to reperesent the tree would be to draw the graph using the $O(nlogn)$ area algorithm. the procedure would be to recursively draw the children while packing them together so that the area does not increase much, and then add the root. Figure 6 shows the tree and three possible drawings. this method is secially usefull when displaying the tree of directories,etc.

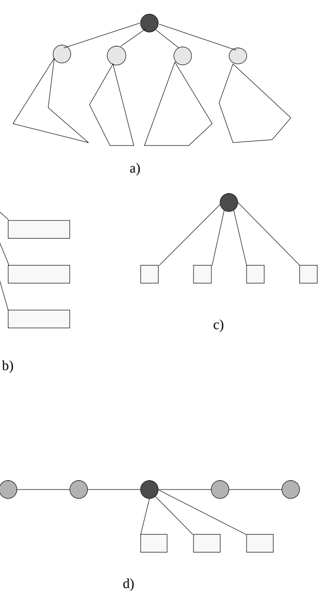

Figure 6: a) The non-binary tree, b), c), d) the possible ways of drawing in $O(nlogn)$ area .

4 UnRooted trees

For the unrooted trees, a node has to be picked as the root. the best node would be the one that minimizes the height of the tree, so it is reasonable to find the center of the graph. To do this, one possible way would be to kill the leaves first, since those are not definitly the center of the graph. Next kill those nodes which are adjacent to the leaves, and continue in this fashion until the center is found. Now the tree can be represented by a rooted tree with the center node as the root. Each node has $2\pi(K/(n-1))$ as it's X coordinate and $-l$ as it's Y coordinate, where K is the rank of the node in an inorder traverse, n is the number of nodes in the tree and l is the level in which the node is located. So, now that we have the coordinates of each node, it is pretty easy to transform the tree reperesentation to the original tree and draw it. Figure 7 shows the original tree and the rooted tree representation of it.

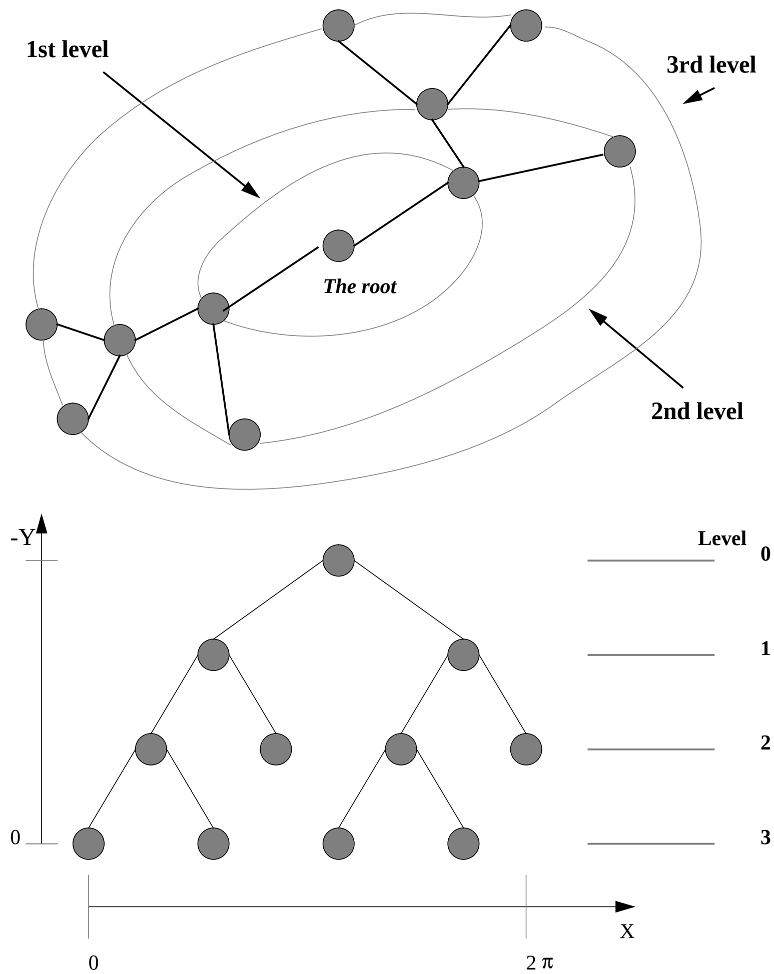

Figure 7: The unrooted tree and it's tree representation .