Info Vis comp5048 lecture August 11: HV Trees

1. ***The greedy algorithm***

HVDrawGreedy(root r)
// returns
// - a HV drawing d_r of the subtree under r
1. If r has no children then:
   a) draw it inside a box of width 1 and height 1
   b) return this drawing
2. Else if r has one child u then:
   a) Run HVDrawGreedy (u) to obtain a drawing d_u of the subtree under u,
   b) Compute two HV drawings: one where r is to the left of d_u and one where r is to the right of d_u .
   c) Choose the smaller of these two drawings (according to the size function) to be d_r
   d) Return d_r
3. Else, where r has two children u and v:
   a) Run HVDrawGreedy(u) to obtain a drawing d_u of drawings of the subtree under u.
   b) Similarly Run HVDrawGreedy (v) to obtain a drawing d_v of the subtree under v
   c) Compute the four HV drawings corresponding to the four possible HV layouts of r with the subtrees under u and v.
   d) Choose the smallest of these four drawings (according to the size function) to be d_r
   e) Return d_r

2. ***The exhaustive algorithm***

HVDrawExhaustive(root r)
// returns
// - a list $D_r = <d_1, d_2, \ldots, d_k>$ of HV drawings of the subtree under r
// - a list L_r of ordered pairs (x_i,y_i) where x_i is the width of the drawing d_i and y_i is the height of the drawing d_i.
4. If r has no children then:
   c) draw it inside a box of width 1 and height 1
   d) return this drawing and the one-element list $\{(1,1)\}$.
5. Else if r has one child u then:
   e) Run HVDrawExhaustive (u) to obtain a list D_u of drawings of the subtree under u, and a list L_u of ordered pairs (x_i,y_i) where x_i is the width of the ith drawing in D_u and y_i is the height of the ith drawing in D_u .
   f) For each drawing d_i in D_u , compute two HV drawings: one where r is to the left of d_i and one where r is to the right of d_i .
   g) Make a list D_r of these drawings, and a list L_r of ordered pairs of their widths and heights.
   h) Return D_r and L_r
6. Else, where r has two children u and v:
   f) Run HVDrawExhaustive (u) to obtain a list D_u of drawings of the subtree under u, and a list L_u of ordered pairs (x_i,y_i) where x_i is the width of the ith drawing in D_u and y_i is the height of the ith drawing in D_u .
   g) Similarly Run HVDrawExhaustive (v) to obtain a list D_v of drawings of the subtree under v, and a list L_v of widths and heights.
   h) For each pair of drawings, one in D_u and one in D_v, compute the four HV drawings corresponding to the four possible HV layouts of r with the subtrees under u and v. Make a list D_r of these drawings, and a list L_r of ordered pairs of their widths and heights.
   i) Return D_r and L_r

To construct a good layout for a tree rooted at r, call HVDrawExhaustive(r) and then linearly search the list D_r for a drawing that is good.

3. ***Domination: definitions***

If w'≥w and h'≥h then we say that (w',h') *dominates* (w,h). Suppose that L is a list of ordered pairs. Then L satisfies the *non-dominating property* if for every pair (w',h') and (w,h) of pairs in L, (w',h') *does not* dominate (w,h) and (w,h) *does not* dominate (w',h').

4. ***Size functions***

Suppose that an HV drawing d of a tree T has width w and height h. There are several possible size functions:
1. Area: *size(w,h) = wh*
2. Perimeter: *size(w,h) = 2(w+h)*
3. InfinityMetric: *size(w,h) = max(w,h)*
4. MinHeightGivenWidthBound(B): *if w>B, size(w,h) = infinity; else size(w,h) = h.*

All these size functions satisfy the following: *if (w',h') dominates (w,h) then size(w',h') ≥size(w,h).*

5. ***A more efficient algorithm***

HVDraw(root r)
// returns
// - a list $D_r = <d_1, d_2, \ldots, d_k>$ of HV drawings of the subtree under r
// - a list L_r of ordered pairs (x_i,y_i) where x_i is the width of the drawing d_i and y_i is the height of the drawing d_i.
// - such that the list L_r satisfies the non-dominating property.

1. If r has no children then:
    a. draw it inside a box of width 1 and height 1
    b. return this drawing and the one-element list {(1,1)}.
2. Else if r has one child u then:
    a. Run HVDraw(u) to obtain a non-dominating list D_u of drawings of the subtree under u, and a list L_u of ordered pairs (x_i,y_i) where x_i is the width of the ith drawing in D_u and y_i is the height of the ith drawing in D_u .
    b. For each drawing d_i in D_u , compute two HV drawings: one where r is to the left of d_i and one where r is to the right of d_i .
    c. Make a ***non-dominating*** list D_r of these drawings, and a list L_r of ordered pairs of their widths and heights.
    d. Return D_r and L_r
3. Else, where r has two children u and v:
    a. Run HVDraw (u) to obtain a non-dominating list D_u of drawings of the subtree under u, and a list non-dominating L_u of ordered pairs (x_i,y_i) where x_i is the width of the ith drawing in D_u and y_i is the height of the ith drawing in D_u .
    b. Similarly Run HVDraw (v) to obtain a non-dominating list D_v of drawings of the subtree under v, and a non-dominating list L_v of widths and heights.
    c. *Using the sStockmeyer merge (see below),* choose *some* of the pairs of drawings, one in D_u and one in D_v , and compute the four HV drawings corresponding to the four possible HV layouts of r with the subtrees under u and v. Make a non-dominating list D_r of these drawings, and a list L_r of ordered pairs of their widths and heights.
    d. Return D_r and L_r

To construct a good layout for a tree rooted at r, call HVDraw (r) and then linearly search the list D_r for a drawing that satisfies the appropitate constraints on the size function.

6. ***Stockmeyer merge algorithm***

Note: There are 4 possible HV arrangements of two subtrees and a root, but dominance reduces this to two possibilities:

a. Horizontal arrangement:
    - $w_r = w_u+w_v+1$
    - $h_r = \min(\max(h_u+1,h_v),\max(h_u,h_v+1))$
b. Vertical arrangement:
    - $w_r = \min(\max(w_u+1,w_v),\max(w_u,w_v+1))$,
    - $h_r = h_u+h_v+1$

The Stockmeyer merge has three steps:

1. vertical merge (computes all horizontal arrangements) to produce a list V_r
2. horizontal merge (computes all horizontal arrangements) to produce a list H_r
3. merge H_r and V_r to produce the list L_r

Each step takes linear time, because the size of the non-dominating list is bounded by the total possible width of the drawing.

Step 1: VerticalMerge

Input: Lists $L_u = < (a_1,b_1), (a_2,b_2),\ldots, (a_k,b_k) >$ and $L_v = < (c_1,d_1), (c_2,d_2),\ldots, (c_m,b_m) >$, sorted on decreasing order of first coordinate
Output: List $V_r = < (x_1,y_1), (x_2,y_2),\ldots, (x_t,y_t) >$

1. i=1; j=1;
2. while i<=k and j<=m
    a. $x = \min(\max(a_i+1,c_j),\max(a_i,c_j+1))$
    b. $y = b_i+d_j+1$
    c. If (x,y) does not dominate the last element added to V_r then add (x,y) to V_r
    d. If $a_i>=c_j$ then i++ else j++

Notes:

- The Horizontal merge (Step 2) is similar to the vertical merge.
- The final merge of H_r and V_r to produce the list L_r (Step 3) is also similar, but a little bit simpler.
- Each merge runs in linear time.
- The total time of HVDraw, using the Stockmeyer merge, is $O(n^2)$.