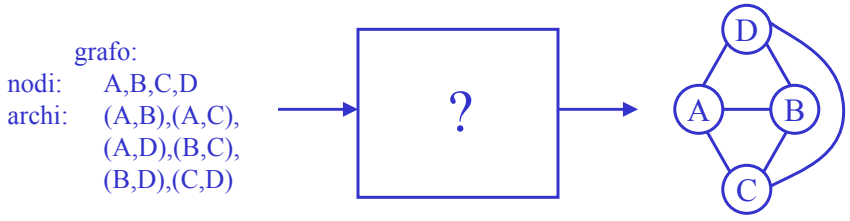
Graph Drawing e tecniche algoritmiche avanzate

Giuseppe Di Battista
Maurizio Patrignani

programma

- vengono introdotte alcune tecniche algoritmiche e viene discussa la loro applicazione al graph drawing
  - introduzione al graph drawing
  - divide and conquer
  - come trovare una struttura dove una struttura non c'è (dfs e planarità)



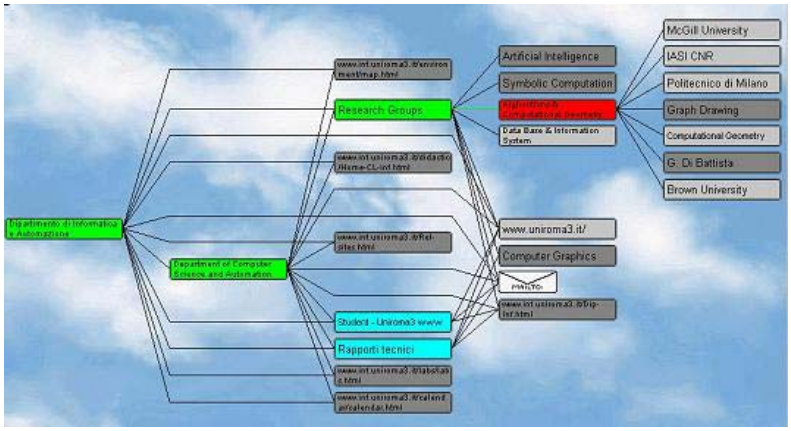
graph drawing e basi di dati

- per visualizzare schemi ER
- per visualizzare risposte ad interrogazioni
- nella integrazione di schemi
- nel reverse engineering

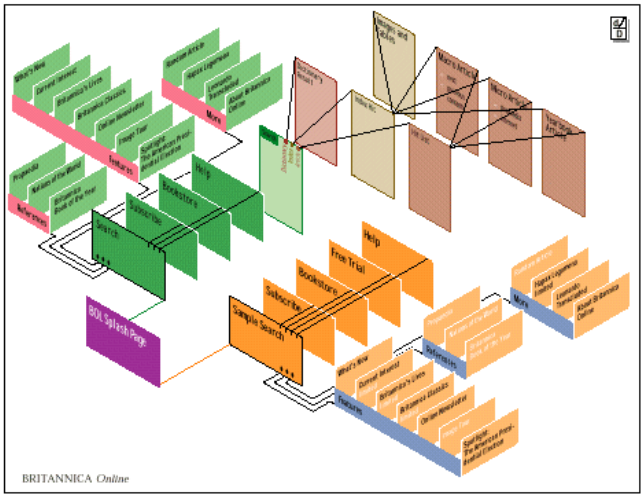
graph drawing e …

- project planning (pert)
- analisi delle funzioni (data flow diagrams)
- analisi organizzativa (organization chart)
- …

ptolomaeus

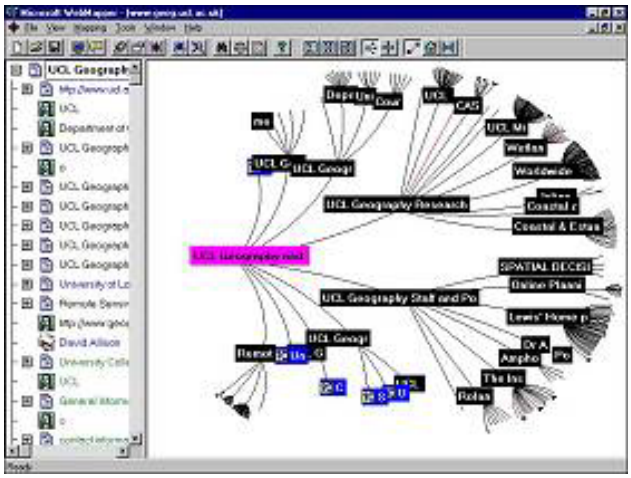


drawings of graphs



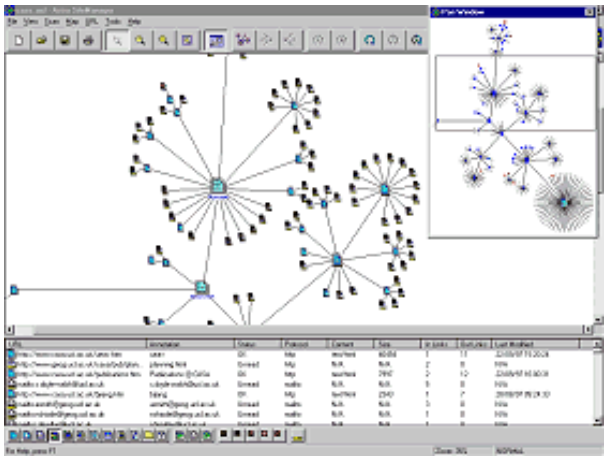
mappe di siti web prodotte da tool automatici

- microsoft, siteserver (backoffice)

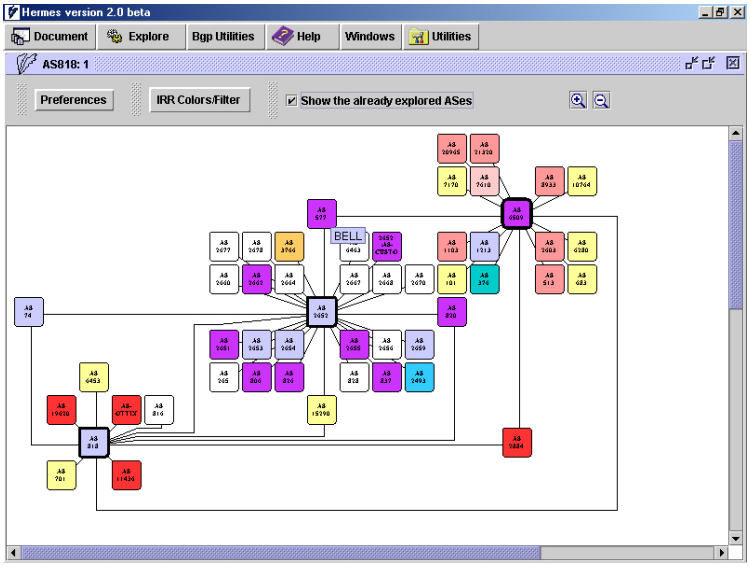


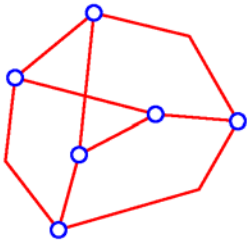
mappe di siti web
prodotte da tool automatici

- site manager, astra

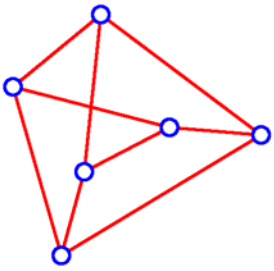


a session with hermes

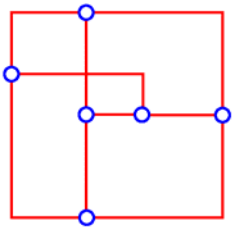




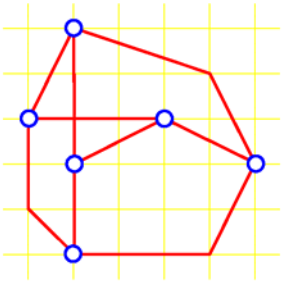
polyline



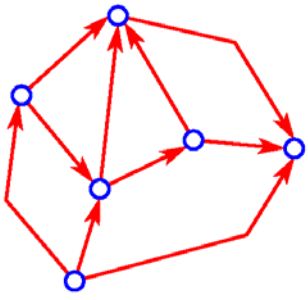
straight-line



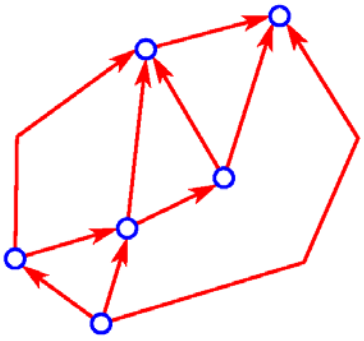
orthogonal



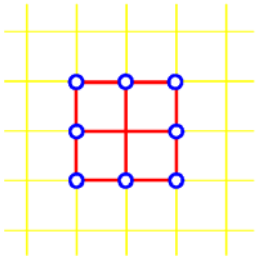
polyline grid



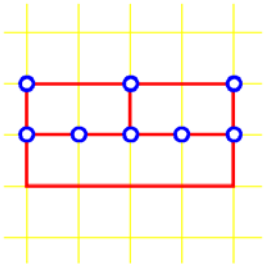
planar polyline



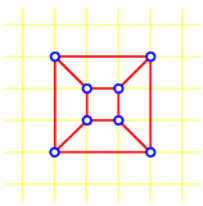
strictly upward planar polyline

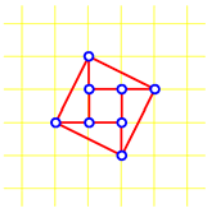


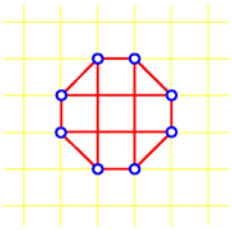
minimum number
of bends

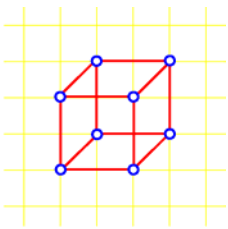


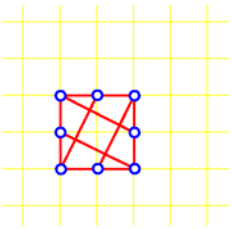
minimum number of crossings

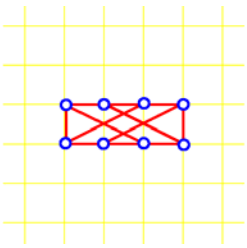


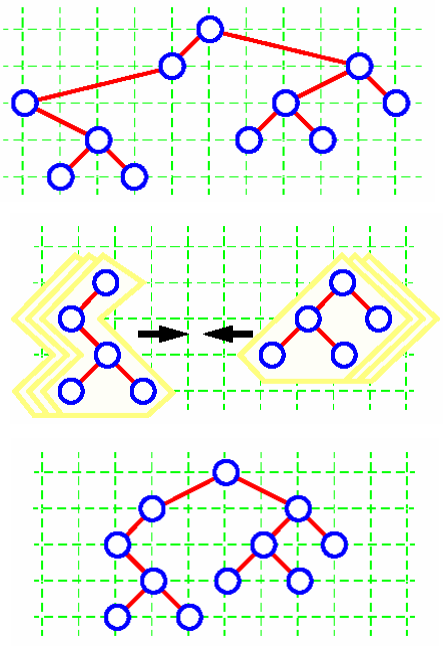












algorithm: Layered-Tree-Draw

input: a binary tree T

output: a layered drawing of T

base case: if T consists of a single vertex, its drawing is trivially defined

divide: recursively apply the algorithm to draw the left and right subtrees of T

conquer: imagine that each subtree is drawn on a separate sheet of paper; move the drawings of the subtrees towards each other until their horizontal distance becomes equal to 2; place the root r of T vertically one unit above and horizontally half way between its children; if r has only one subtree, say the left one, then place r at horizontal distance 1 to the right of its left child.

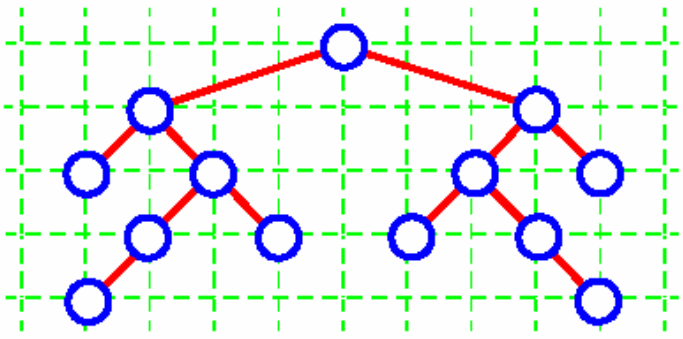
implementing Layered-Tree-Draw

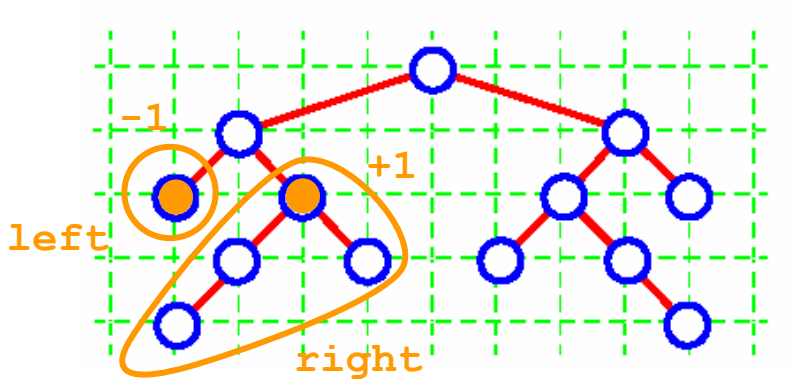
two traversals of T

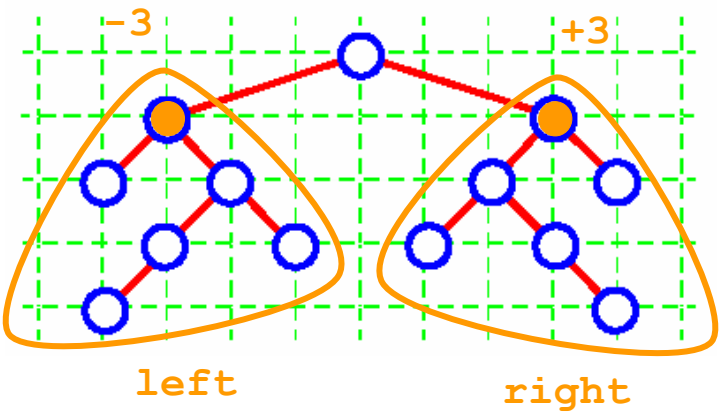
1. a postorder traversal recursively computes for each vertex v, the horizontal displacement of the left and right children of v with respect to v

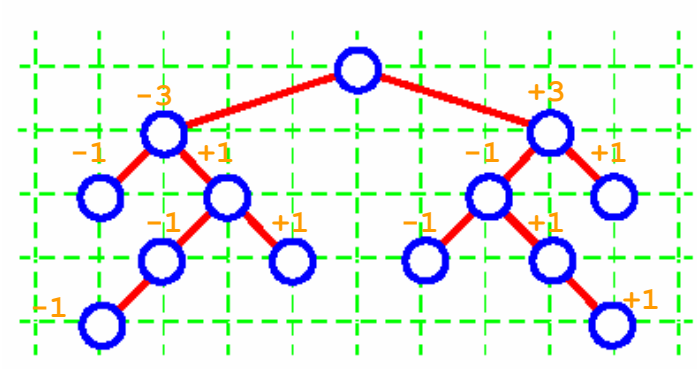
2. a preorder traversal computes the x-coordinates of the vertices by accumulating the displacements on the path from each vertex to the root, and the y-coordinates of the vertices by determining the depth of each vertex

special care is needed in order to implement the postorder traversal so that it runs in linear time



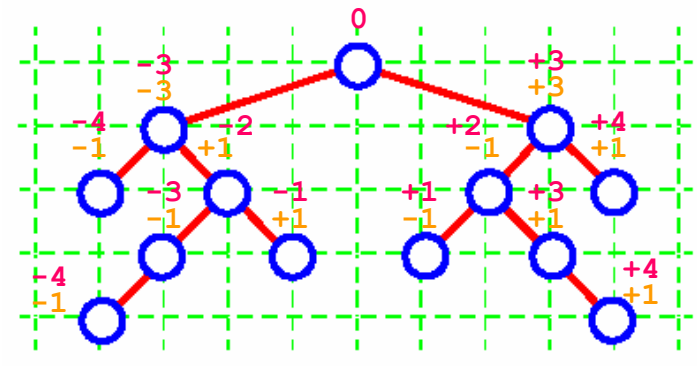

-1
+1
left
right


-3
+3
left
right



dopo la visita in postordine

dopo la visita in preordine



dopo la visita in postordine

implementing Layered-Tree-Draw the postorder traversal

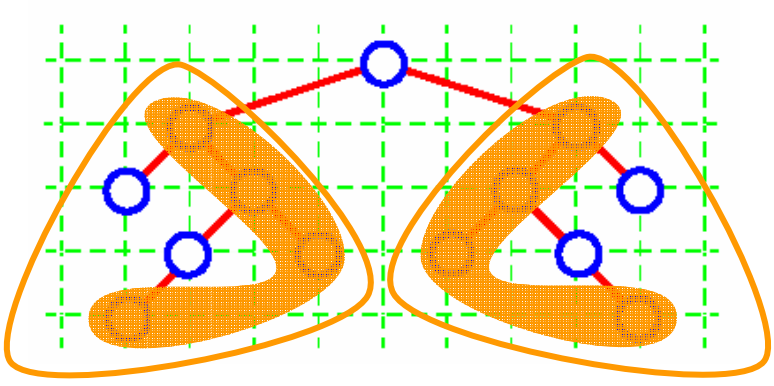
the left contour of a binary tree T with height h is the sequence of vertices $v_0 \ldots v_h$, such that v_i is the leftmost vertex of T with depth i

the right contour is defined similarly

in the conquer step, we follow the right contour of the left subtree and the left contour of the right subtree

in the postorder traversal, we maintain the invariant that after completing the processing of a vertex v, the left and right contours of the subtree rooted at v are stored in linked lists

left and right contours



implementing Layered-Tree-Draw the postorder traversal

processing v in the postorder traversal is done by scanning the right contour of the left subtree of v (following the recursively computed right contour list) and the left contour of the right subtree of v (following the recursively computed left contour list)

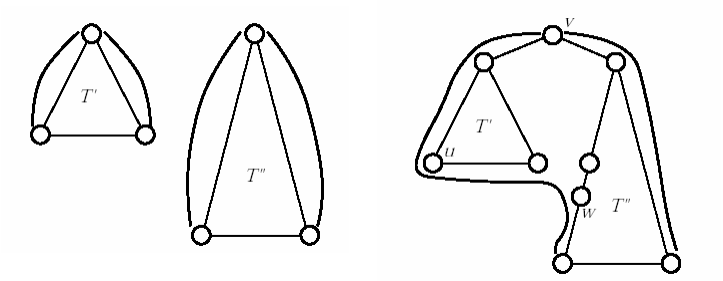
during the scan we accumulate the displacements of the vertices encountered on the left and right contours and we keep track of the maximum cumulative displacement at any depth

implementing Layered-Tree-Draw the postorder traversal

let T(v) be the subtree rooted at v, and T' and T'' be the left and right subtrees of v

the left and right contour list of T(v) can be constructed:

- if T' and T'' have the same height, then the left contour list of T(v) is the same as the left contour list of T' plus vertex v, and the right contour list of T(v) is the same as the right contour list of T'' plus v
- if the height of T' is less than the height of T'', the right contour list of T(v) is the same as the right contour list of T''; let h' be the height of T' and let u be the bottommost vertex of the left contour of T'; let w be the vertex of the left contour of T'' such that w has depth h'+1 in T''; the left contour list of T(v) consists of the concatenation of vertex v, the left contour list of T', and the portion of the left contour list of T'' beginning at vertex w



implementing Layered-Tree-Draw efficiency

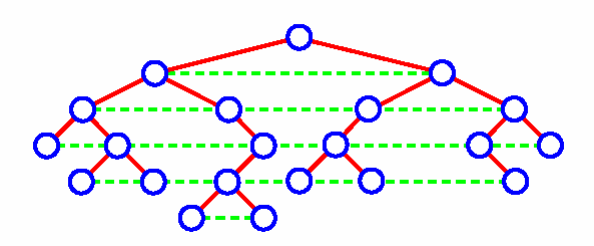
a crucial observation to prove the efficiency of the algorithm:

it is necessary to travel down the contours of T' and T'' only as far as the height of the subtree of lesser height

the time spent processing vertex v in the postorder traversal is proportional to the minimum of the heights of T' and T''

the running time of the postorder traversal of tree T is given by the following formula, where for a vertex v of T, we denote the height of the left subtree of v by h'(v) and the height of the right subtree of v by h''(v):

$$\Sigma_{v \in T} (1+\min\{h'(v), h''(v)\}) =$$
$$= n + \Sigma_{v \in T} \min\{h'(v),h''(v)\}$$



implementing Layered-Tree-Draw efficiency

we can visualize the sum $\Sigma_{v \in T} \min\{h'(v),h''(v)\}$ by connecting with new edges pairs of consecutive vertices with the same depth

the sum over all vertices v of the minimum height of the subtrees of v is equal to the number of new edges added to the tree

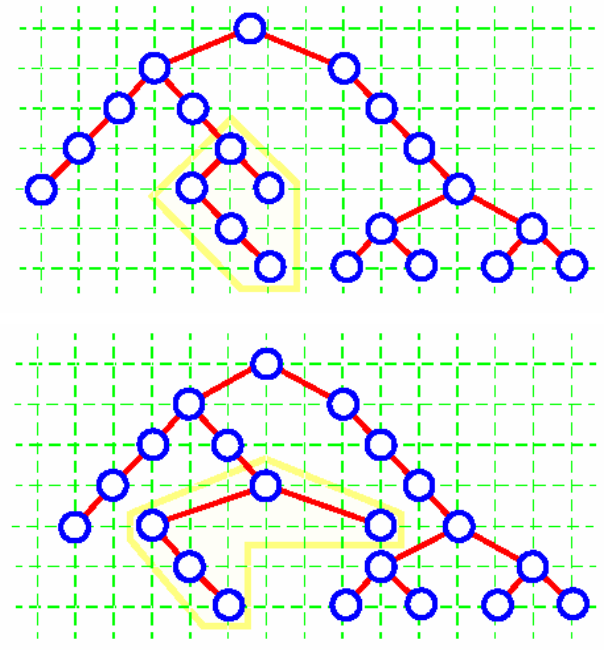
each vertex is incident to at most one new edge on its right

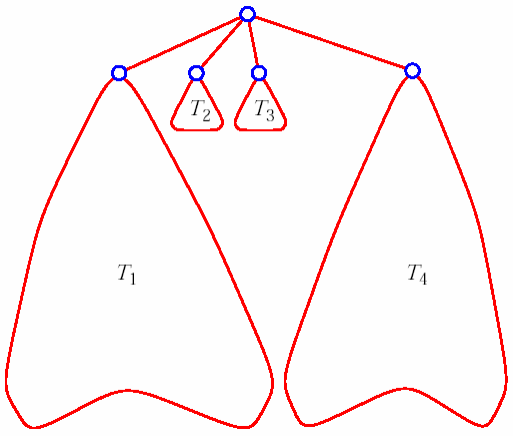
the number of new edges, and therefore the above sum, is no more than the number of vertices of the tree

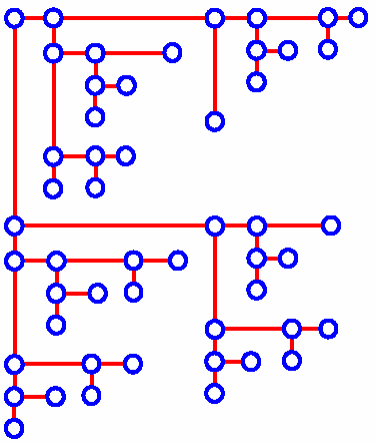
implementing Layered-Tree-Draw conclusions

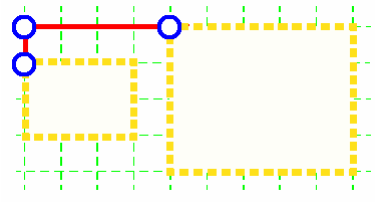
if T is a binary tree with n vertices, Layered-Tree-Draw constructs in O(n) time a drawing Γ of T such that:

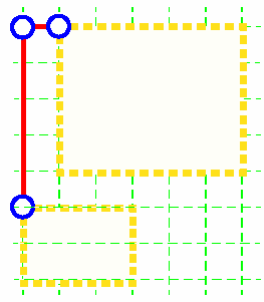
- Γ is layered, planar, straight-line, strictly downward
- Γ is embedding-preserving (the left-to-right order of the children of each vertex is preserved)
- any two vertices of Γ are at horizontal and vertical distance at least 1
- the area of Γ is $O(n^2)$
- the x-coordinate of a parent with two children is the average of the x-coordinates of its children
- simply isomorphic subtrees have congruent drawings, up to a translation
- axially isomorphic subtrees have congruent drawings, up to a translation and a reflection around the y-axis

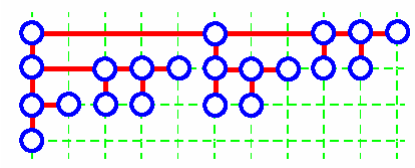


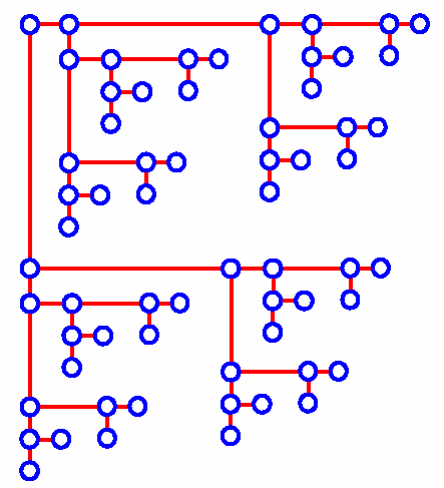

T_1
T_2
T_3
T_4

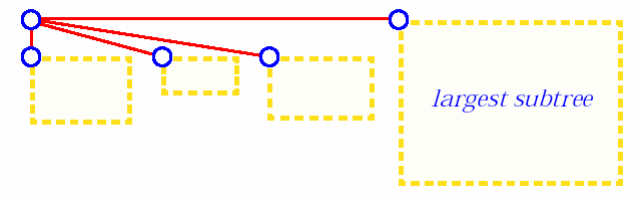











largest subtree