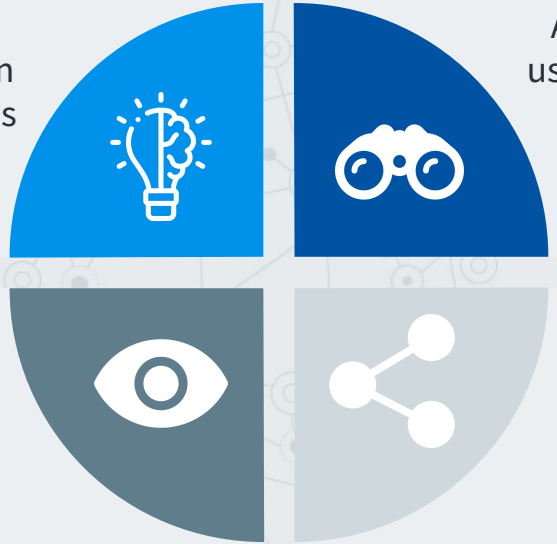
Graph Drawing

Why Layouts?

Easy to understand

Visually displaying data ensures a faster comprehension which, in the end, reduces the time it takes to make decisions and take action.



Discover more insights in your data

A study showed that people who use visual data discovery tools are 28% more likely to find information in less time

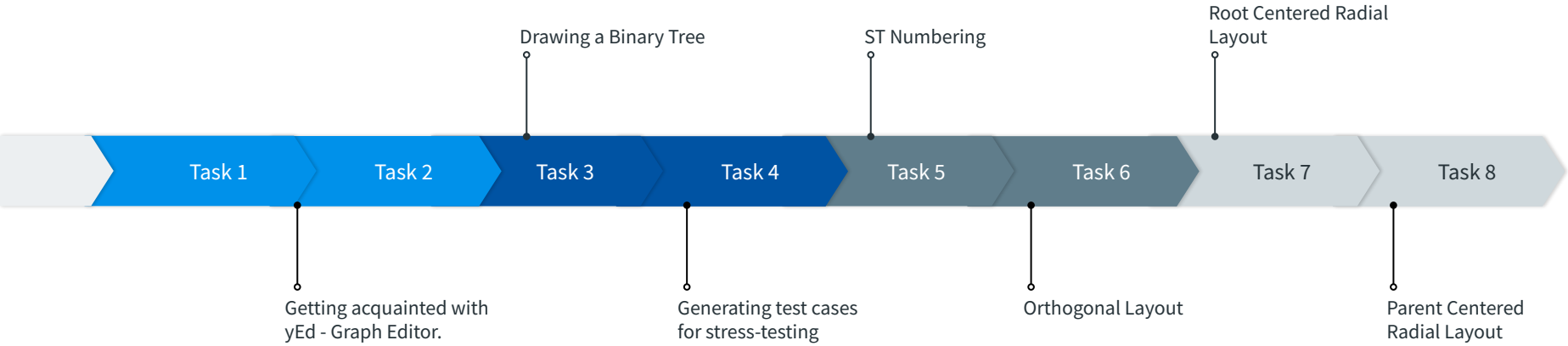
You can achieve a better understanding of a problem by visualizing patterns and context

Better Visualization

It’s an effective form of communication. Visual representations offer a more intuitive way to understand the data

Share your findings with ease

Timeline


Drawing a Binary Tree
ST Numbering
Root Centered Radial Layout
Task 1
Task 2
Task 3
Task 4
Task 5
Task 6
Task 7
Task 8
Getting acquainted with yEd - Graph Editor.
Generating test cases for stress-testing
Orthogonal Layout
Parent Centered Radial Layout

Task 1 & 2

Getting acquainted with yEd - Graph Editor.

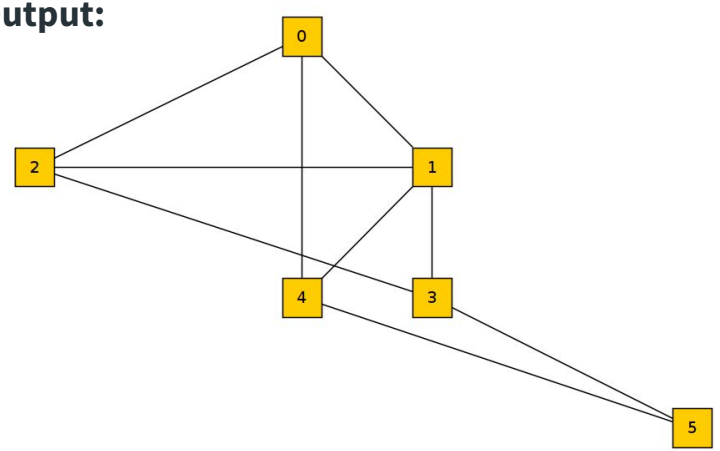
- ◎ Accepting a csv file as input with each row having 3 entries: node id, x-coordinate, y-coordinate, and create a corresponding graphml file.
- ◎ Also adding a set of edges in the graphml file by reading from a csv file.

Input:

```
"0",100,200
"1",200,300
"2",-200,300
"3",200,400
"4",100,400
"5",400,500
```

```
9
0,1
0,2
0,4
1,2
1,3
1,4
2,3
3,5
4,5
```

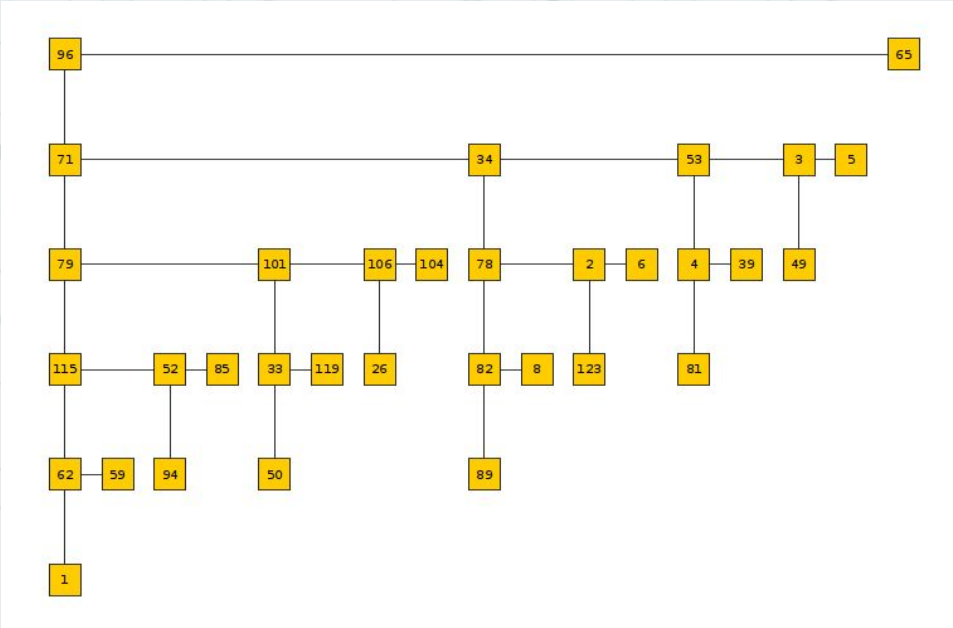
Output:



Task 3

Generating a graphml file for the binary tree while adhering to the one down and one to the right rule.

Output:



Task 4

Generating graph input files for testing purposes.

- A complete binary tree of given height h.
- A complete multi-partite graph with n partitions and k vertices in each partition.

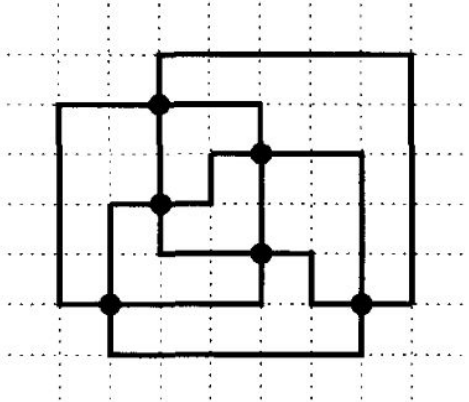
A) Input file for a complete binary tree of height 4

```
14
5,8
5,2
8,15
8,13
2,3
2,14
15,4
15,11
13,1
13,6
3,10
3,7
14,12
14,9
```

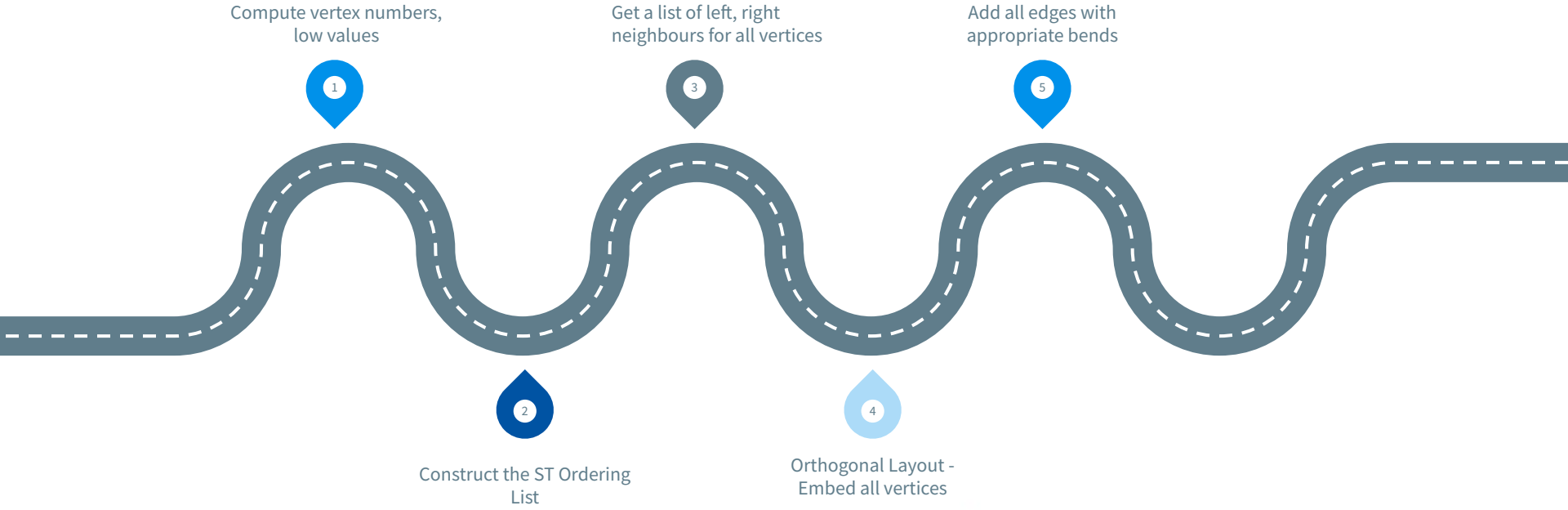
B) Snippet of the input file generated for a complete multi-partite graph with n = 5 and k = 4

```
160
10,15
10,13
10,16
10,18
10,8
10,20
10,3
10,7
10,2
10,12
10,5
```

Orthogonal Layout



Roadmap to achieve orthogonal layout



Task 5

- st-numbering is a numbering of the vertices of a biconnected graph by the integers 1 through n such that s is vertex 1, t is vertex n, and every other vertex is **adjacent both to a lower-numbered and to a higher-numbered vertex.**

- We number the vertices from 1 to n in the order they are first visited during the depth first search. This numbering is a preorder numbering.

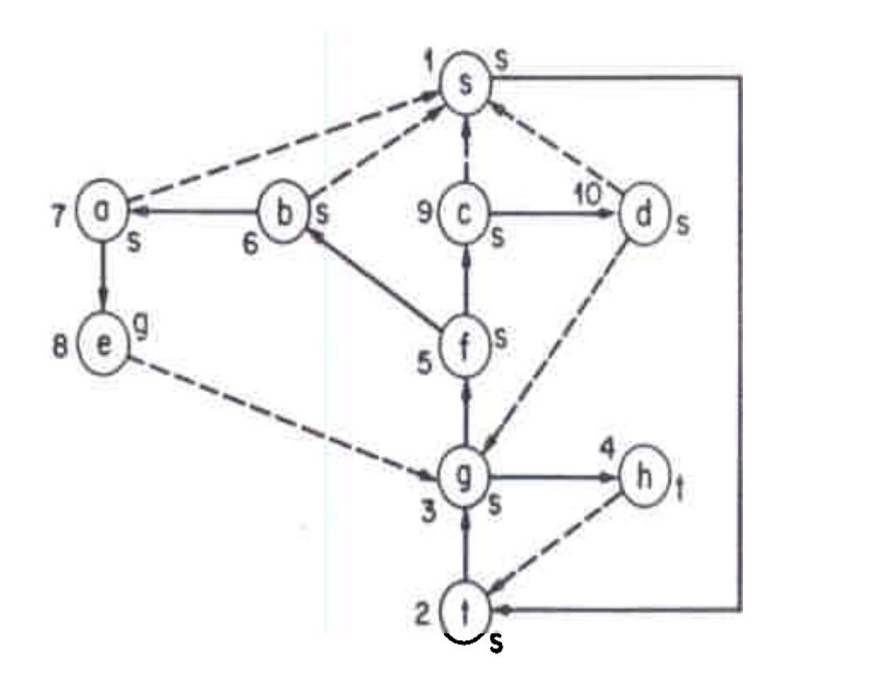
- Denote low(v) to be the vertex of smallest number reachable from v by a path consisting of zero or more tree edges followed by at most one back edge.

- The vertex low(v) is guaranteed to be an ancestor of v in the spanning tree.

computed in linear time in a single depth-first search; to compute the *low* values, we use the fact that $low(v)$ is the vertex of minimum number in the set $\{v\} \cup \{low(w) \mid (v, w) \text{ is a tree edge}\} \cup \{w \mid (v, w) \text{ is a back edge}\}$. The following lemma relates *low* values to biconnectivity:

- Initially L - [s, t] and s has sign minus. Then the following step is repeated for each vertex v in preorder.
- Add a vertex. If sign(low(v)) is plus, insert v after p(v) in L and set sign(p(v)) as minus; else insert v before p(v) in L and set sign(p(v)) as plus.

Example Biconnected Graph



Dry run of ST Numbering Algorithm

VERTEX ADDED	LIST
	s-,t
g	s-,g,t+
h	s-,g-,h,t+
f	s-,f,g+,h,t+
b	s-,b,f+,g+,h,t+
a	s-,a,b+,f+,g+,h,t+
e	s-,a-,e,b+,f+,g+,h,t+
c	s-,a,e,b,c,f+,g+,h,t+
d	s-,a,e,b,d,c+,f+,g+,h,t+

Task 6

- Given a planar biconnected 4-graph G = (V, E), obtain an st-ordering for G with s as first and t as last vertex (v1, v2 ..., vn).
- We build the drawing of G incrementally by adding vk into the drawing of {v1,..., vk-1}, k = 1, ..., n holding the invariant that at every stage every edge where exactly one endpoint is drawn is associated to a column.

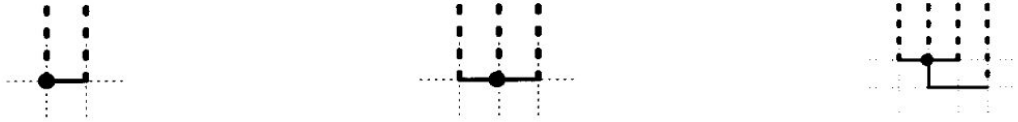


Fig. 2. Embedding of the first vertex for various degrees.

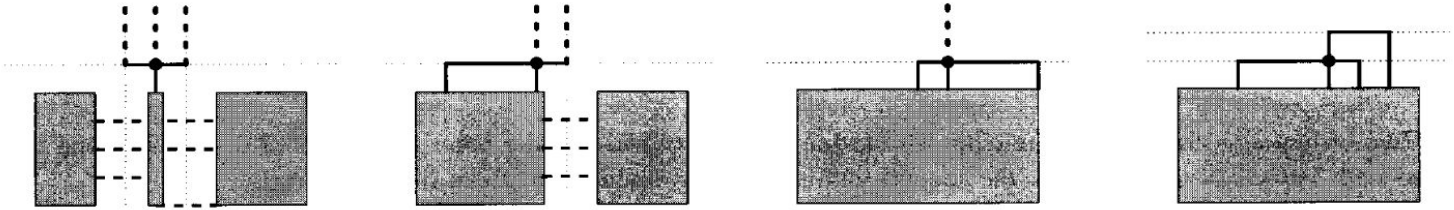
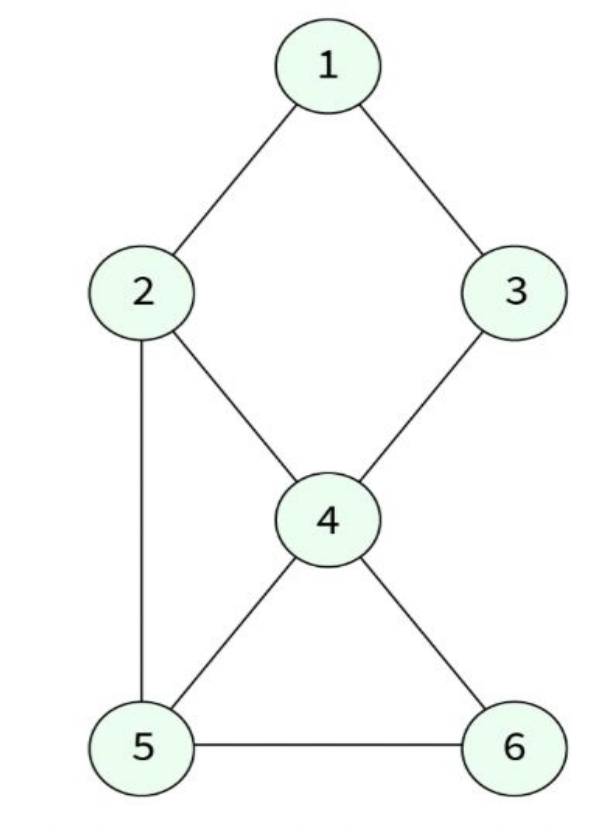
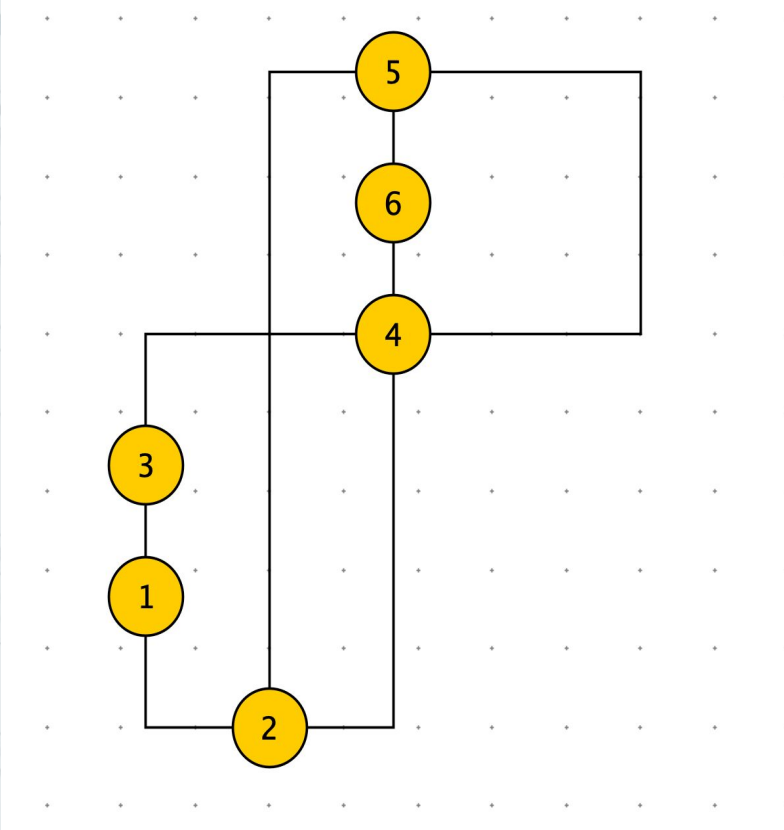


Fig. 3. Embedding of v_k, $k \geq 2$, for $\text{indeg}(v_k) = 1, 2, 3, 4$.

Biconnected 4-Graph

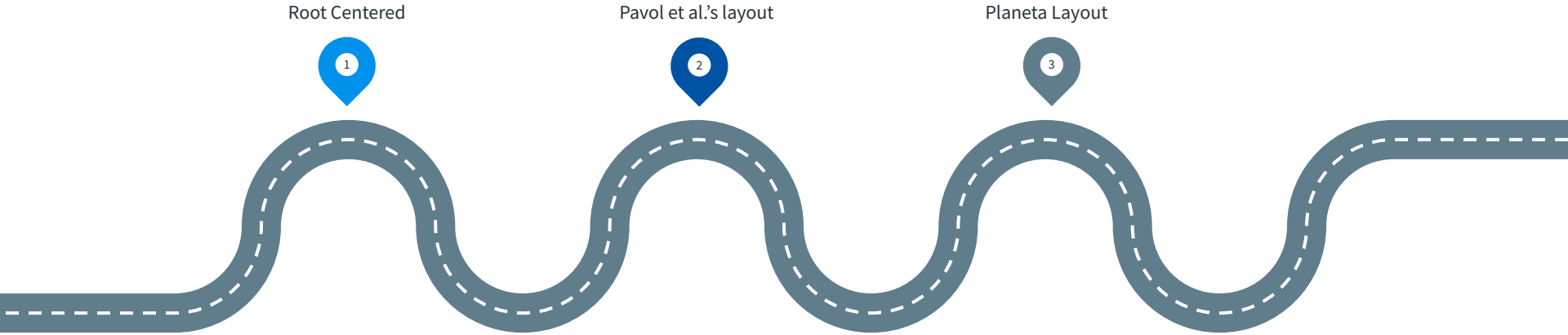


Orthogonal Layout



Radial Layout

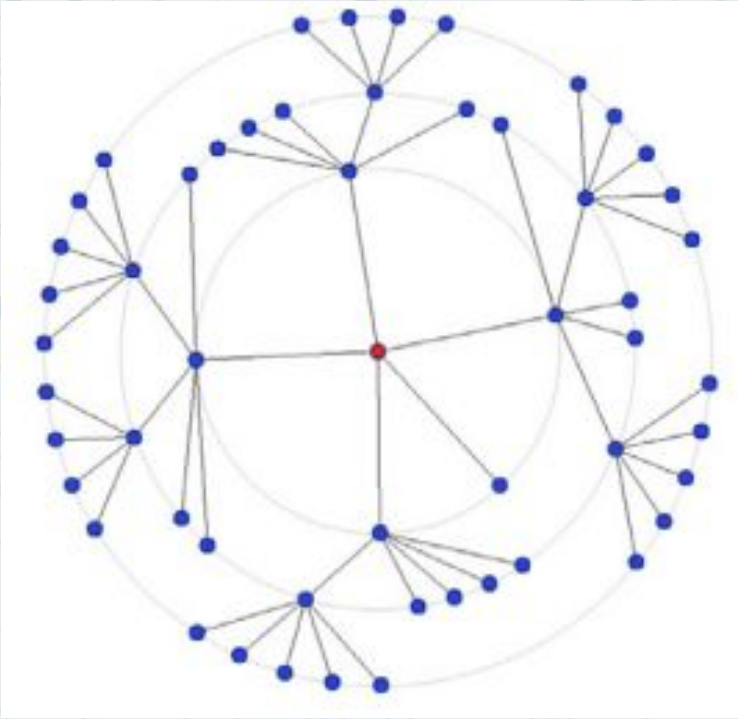
Roadmap



Task 7

Root Centric Radial Layout:

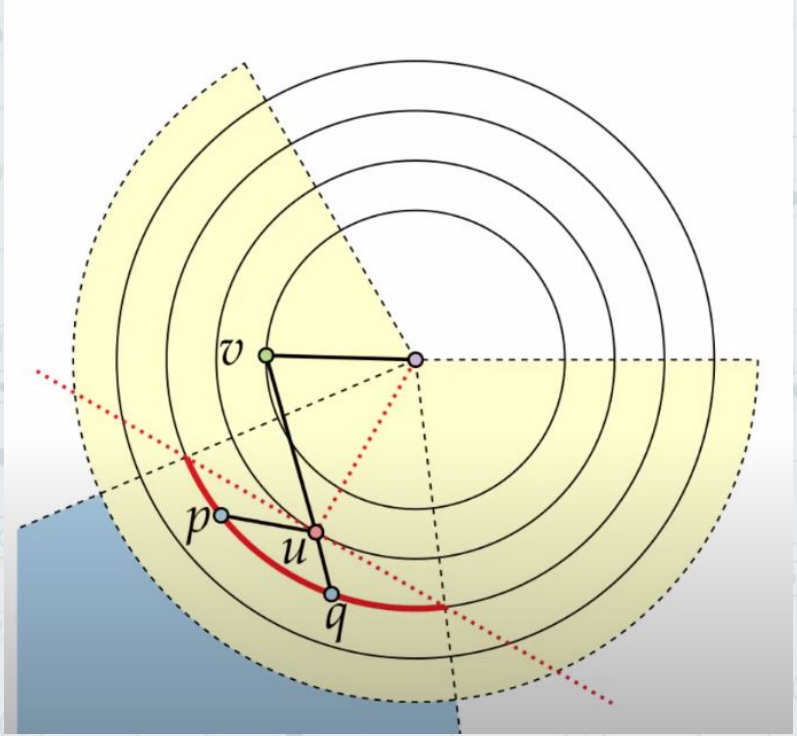
- This radial layout algorithm place the root node at the center of the coordinate system, and the other nodes then radiate outward in evenly separated concentric rings, with one ring for each set of ith order neighbors of the root node
- Area reserved for Subtree is proportional to size of the subtree



Task 7

Root Centric Radial Layout:

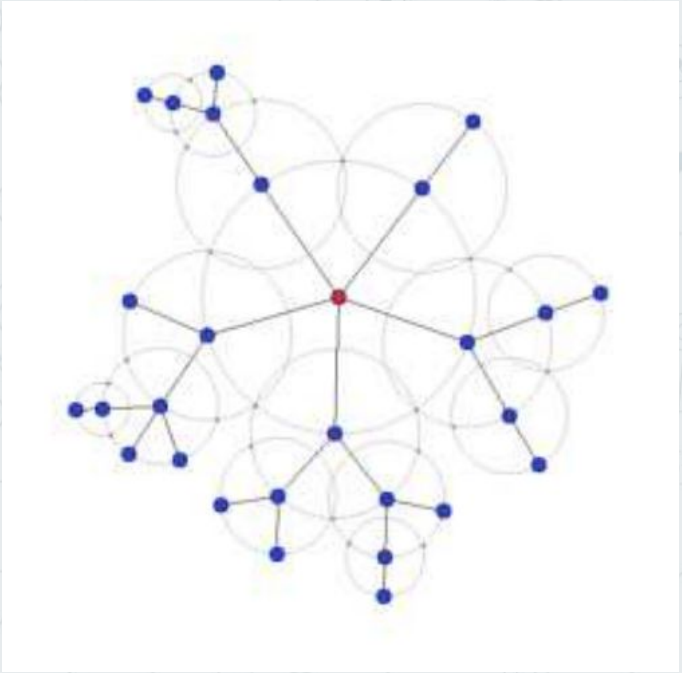
- Overlapping are avoided By Placing the Children in the Intersection arc of Tangent and Next concentric Circle.



Task 8

Pavol et al.'s Radial Layout:

- First,we place the root in the center of the display with its children evenly distributed along a containment circle centered on the root.
- Second, we draw circles around the root's children and evenly distribute their children along containment arcs that ensure that neither siblings nor cousins overlap.
- Then the second process just proceeds recursively, so that successively distant descendants of the root are positioned on successively smaller containment arcs



Task 8

As the height of the tree increases, the containment circle of remote descendants cannot be reduced any further, and the descendant nodes cannot be displayed. As shown in the dashed rectangles, the parent and child nodes are squeezed together

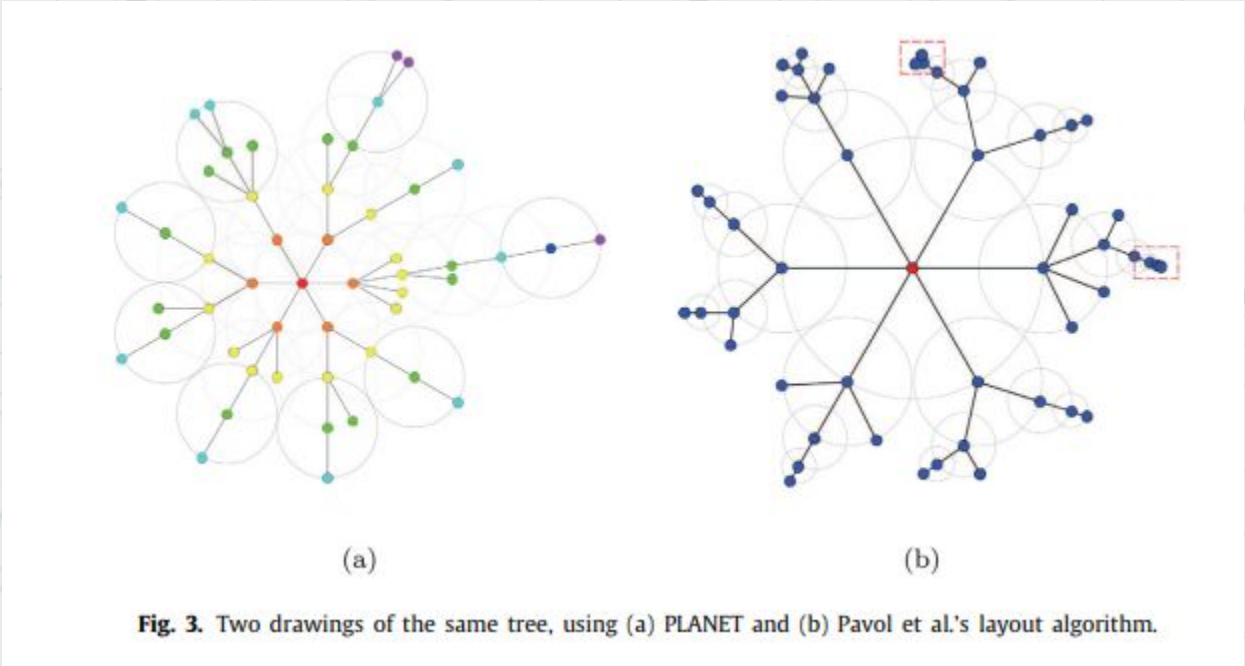
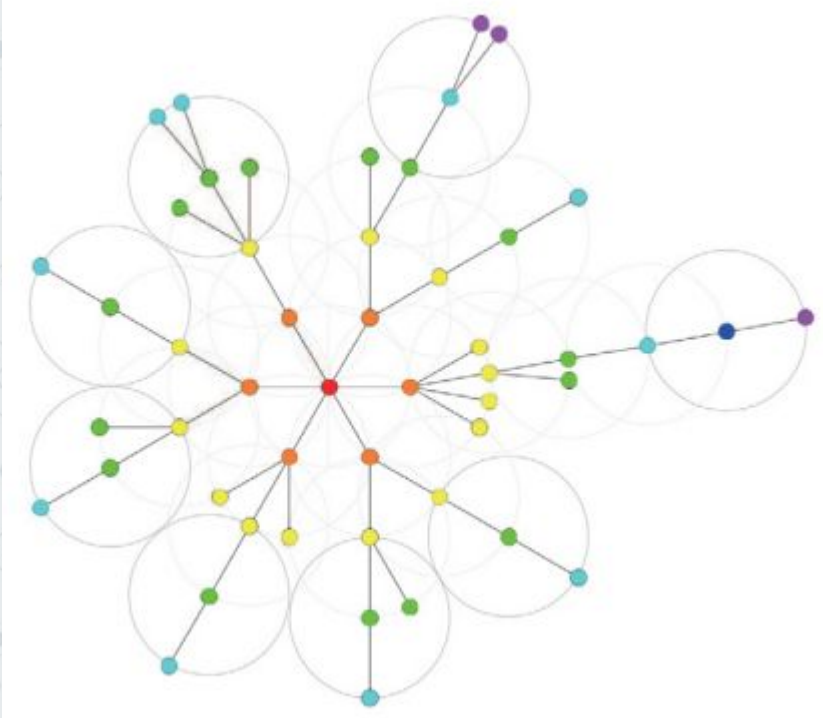


Fig. 3. Two drawings of the same tree, using (a) PLANET and (b) Pavol et al.'s layout algorithm.

Task 8

Planet Radial Layout:

- Root Node is placed at the origin. Then, we place the n_0 child nodes of the root node uniformly at random on a circle with radius r and centered at the root node.
- Then, the descendant nodes of n0 are placed uniformly at random on annulus wedges with radius r and centered at the parent of each node respectively, according to the proposed angle assignment rules for child nodes.



Task 8

Angle Assignment Rules:

◎ We place the n child nodes of the root node uniformly at random on a circle (First Layer) :

$$\theta_1 = \frac{2(i-1)\pi}{n_1},\ n_1 \neq 0,$$

where the angle assigned to each child node is $\frac{2\pi}{n_1}$.

◎ For The Second Layer we follow:

$$\theta_2 = \begin{cases} \theta_1 & \text{if } n_2 = 1, \\ \theta_1 - \frac{\pi}{f_0} + \frac{2(i-1)\pi}{(n_2-1)f_0} & \text{otherwise,} \end{cases}$$

where the angle assigned to each child node is $\frac{2\pi}{n_2 f_0}$.

◎ For Further Layers:

$$\theta_d = \begin{cases} \theta_{d-1} & \text{if } n_d = 1, \\ \theta_{d-1} + \frac{2(i-1)\pi}{(n_d-1)\prod_{k=0}^{m} f_k} & \text{if } \theta_{d-1} < \theta_{d-2} \text{ and } n_d \neq 1, \\ \theta_{d-1} - \frac{2(i-1)\pi}{(n_d-1)\prod_{k=0}^{m} f_k} & \text{if } \theta_{d-1} > \theta_{d-2} \text{ and } n_d \neq 1, \\ \theta_{d-1} - \frac{\pi}{\prod_{k=0}^{m} f_k} + \frac{2(i-1)\pi}{(n_d-1)\prod_{k=0}^{m} f_k} & \text{otherwise,} \end{cases}$$

where $m = d-2$, and the angle assigned to each child node is $\frac{2\pi}{n_d \prod_{k=0}^{m} f_k}$.

Team

Aravind N.R.
Associate Professor
IIT Hyderabad

Sai Vardhan Malthi
MA19BTECH11010
Undergraduate Student
Maths and Computing

Varun Rao Chintu
MA19BTECH11009
Undergraduate Student
Maths and Computing

References

Orthogonal Layout:

- A better heuristic for orthogonal graph drawings (Link)
- Two Streamlined Depth-First Search Algorithms (Link)
- Biconnectivity and st-numbering (Link)

Radial Layout:

- Root Centric Layout (Link)
- A parent-centered radial layout algorithm for interactive graph visualization and animation (Link)
- PLANET: A radial layout algorithm for network visualization (Link)

Thank You