Assignment 5

Available Since: 30 May 2014  Due Date: 5 June 2014, 12:00 a.m.
You are permitted and encouraged to work in groups of two.

Exercise 1: SP-Graphs  5 Points

An outerplanar graph has an embedding in the plane such that all vertices belong to the unbounded face of the embedding. In a biconnected graph if any vertex were to be removed, the graph will remain connected. Show that any outerplanar biconnected graph is series-parallel.

Exercise 2: st-Ordering  6 Points

Apply the algorithm for computing an st-ordering that was discussed in the lecture to the graph below. Show your steps in the style it was done in the lecture. In the depth first search, process the edges in the order indicated by their labels.

Exercise 3: st-Ordering  4 Points

Consider the algorithm for computing an st-ordering that was discussed in the lecture. In function process_ears the inner vertices of the path $P : u \rightarrow v \leftarrow w$ were inserted directly before or after $u$ into the list $L$ representing the st-ordering. Give an example to prove that the algorithm would not necessarily compute an st-ordering if the inner vertices of $P$ were directly inserted before or after $w$. 
Exercise 4: Biconnected Components

Implement the algorithm for determining biconnected components. Ensure linear time complexity.

- Highlight affiliations to individual components by different colors. Also, mark the cut vertices.
- To visualize your results, you can use the following command to layout the graph:
  ```java
  new OrganicLayouter().doLayout(graph);
  ```
- For changing the node/edge color, you can use, e.g., the following command:
  ```java
  ((Graph2D).graph).getRealizer(node).setFillColor(Color.red)
  ((Graph2D).graph).getRealizer(edge).setLineColor(Color.red)
  ```
- Make sure that the line width is thick enough:
  ```java
  EdgeRealizer().setLineType(LineType.createLineType(...))
  ```

Test the efficiency of your implementation by generating random networks with `material.RandomGraphGenerator` which is a GNP random graph generator. Note that by using probability $(\log n)/n$, the resulting graph is connected with high probability. Measure your running times and create a plot (i.e. using gnuplot, R, Excel) that documents your results. Submit the plot with the theoretical exercises and via svn.