01OGD Algorithms and Programming

18/09/2017 – Part I: Theory (12 points)

1. (1 point)

Given the following sequence of pairs, where the relation i-j means that node i is adjacent to node j:

1-7, 4-3, 4-7, 6-2, 5-10, 5-6, 0-9, 3-5, 6-9, 10-1

apply an on-line connectivity algorithm with **weighted** quickunion, showing at each step the contents of the array and the forest of trees at the final step. Node names are integers in the range from 0 to 10.

2. (2 points)

Given the following sequence of integers stored in an array:

31 2 21 3 7 43 13 50 16 9 71 12

- turn it into a heap, assuming to use an array as underlying data structure. Draw each step of the heap-building process, as well as the final result. Assume that, at the end, the largest value is stored at the heap's root
- execute the first two steps of the heapsort algorithm on the heap bulit at the previous step.

NB: assume that the sequence is already stored in the array and that it represents an intermediate configuration on which the heap property doesn't necessarily hold.

3. (2 points)

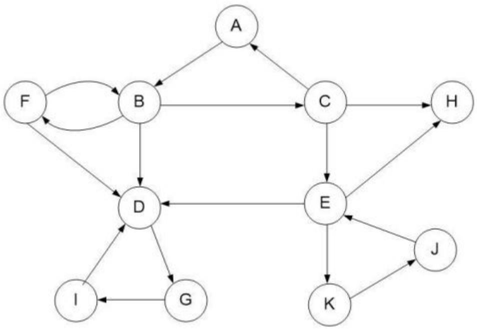
Insert in the leaves of an initially empty BST the following keys in sequence:

11 7 4 22 12 9 20 16 13

Once insertion is completed, partition the BST around its median key.

4. (2 + 1.5 + 1.5 points)

Suppose to have the following directed graph:

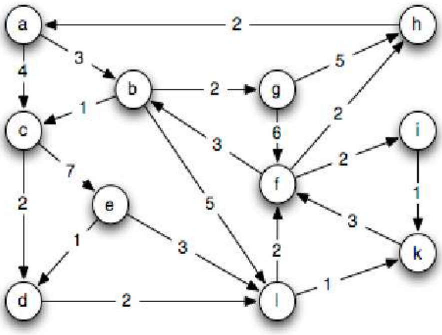


- visit it in depth-first starting at node **A.** Label nodes with discovery and end-processing times in the format time1/time2 (**2 points**)
- visit it in breadth-first starting from node **A** (**1.5 points**)
- redraw it labelling each edge as T (tree), B (back), F (forward), C (cross), starting at node **A**. (**1.5 points**).

Whenever necessary consider nodes in alphabetical order.

5. (2 points)

Given the following undirected and weighted graph:



find a minimum spanning tree using Prim's algorithms starting from vertex **a**, draw the tree and return the minimum weight as a result. Show intermediate steps.