01OGD Algorithms and Programming

03/09/2014 – Part I: Theory (12 points)

1. (1 point)

Apply on the following sequence of pairs, where notation i-j indicates that node i is adjacent to node j, an on-line connectivity algorithm with quickfind. At each step show the contents of the array. At the final step show the forest of trees. Nodes are integers in the range from 0 to 10.

2-10, 4-2, 0-4, 6-1, 2-7, 4-3, 0-8, 0-4, 7-8, 10-3

2. (**2 points**)

Given the following sequence of integers stored in an array:

21 2 11 3 7 33 13 5 16 9 17 1

- 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:

10 24 31 68 11 13 58 8 75

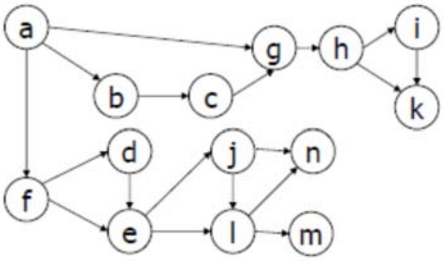
And, once insertion is completed, partition the BST around its median key.

4 (2 points)

Given the sequence of keys HUNGERGAMES, whwre each charcter is identified by its index in the English alphabet (A=1, ..., Z=26), show an initially empty hash table of size 23 where insertion of the the previous sequence character by character occurs. Assume open addressing with quadratic probing. Select proper values for c_1 and c2.

5. (2 points)

Apply a topological sorting algorithm to the following DAG. If necessary, consider vertices according to the alphabetic order and assume the adjacency list sorted in alphabetical order as well.

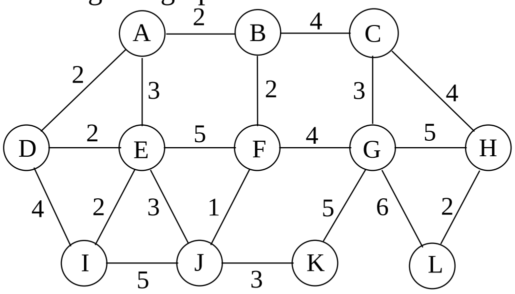


6. (1 point)

Transform the DAG of excercise 5 into the corresponding undirected graph and find its articulation points. Start from node **a** and, if necessary, consider vertices according to the alphabetic order.

7. (2 points)

Given the following undirected and weighted graph:



find a minimum spanning tree applying Kruskal's algorithm. Show intermediate steps, draw the tree and compute the minimum weight.

Algorithms and Programming

Examination Test

Programming Part

03 September 2014

No books or notes are allowed. Examination time: 100 minutes.

The programming part includes two *possibilities*:

- **Exercise number 1 (traditional) of a maximum value equal to 18 points.**
- **Exercises 2, 3 and 4 (easier) with a total maximum value of 12 points.**

Intermediate solutions are forbidden.

1 (18.0 points)

A designer must configure a network, which is given as a list of its vertices whose number in unknown. A file stores the lists of vertices. Each vertex is stored on a separate line, and has a unique identifier made up of a string of 20 characters at most. The file name is given on the command line.

The designer has to figure out which are the undirected (bi-directional) arcs to make the network connected. A network is defined as connected if for every vertex couple (v_i, v_j) there is a path connecting v_i with v_j. The solution has to satisfy the following conditions:

1. The number of arcs has to be minimum.
2. For each vertex couple (v_i, v_j) the length of the path connecting them has to be smaller than k, integer value given on the command line.
3. Each vertex has a degree smaller or equal to m, integer value given on the command line.

Write a C program that:

- Reads the input file defining the list of vertices and stores it in a proper data structure.
- Reads a second file storing a list of arcs, each one on a separate line, with the format

  ```
  v_i   v_j
  ```

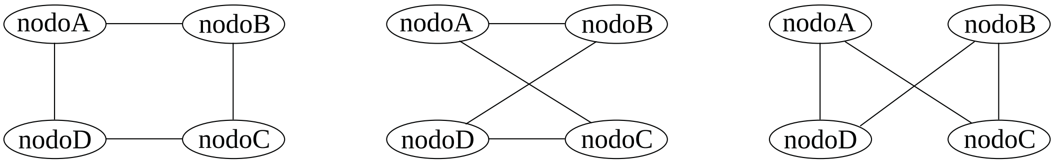
  and it verifies whether this set of arcs satisfy the conditions (2) and (3) defined above.

- Computes a solution satisfying conditions (1), (2) and (3) defined above, and it stores it as a list of arcs, one for each line of an output file, whose name is passed on the command line.

For example if the input file is the following one

```
nodoA
nodoB
nodoC
nodoD
```

and $k = 2$ and $m = 2$ there would be three possible solutions:



nodoA nodoB	nodoA nodoB	nodoA nodoC
nodoA nodoD	nodoA nodoC	nodoA nodoD
nodoB nodoC	nodoB nodoD	nodoB nodoC
nodoC nodoD	nodoC nodoD	nodoB nodoD

The program has to store in the output file one of the arc sets.

continue

2 (2.0 points)
Write the function

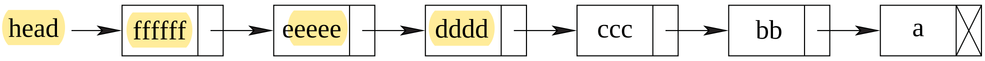
```
void searchStr (char *str, int *start, int *length);
```

which receives in input the `str` string and finds-out the longest sub-sequence of characters made-up by the same character. The function has to return the starting index of this sub-sequence and its length in the parameters `start` and `length`, respectively.
For example, if the function receives the string `abbcccddddeeeee` it should find-out the sequence `eeeee` and return the value `start = 10` and `length = 5`.

3 (4.0 points)
A string includes sub-strings, made-up of alphabetic and numeric characters, and delimited by the full-stop `‘‘.’’` character. Write the function

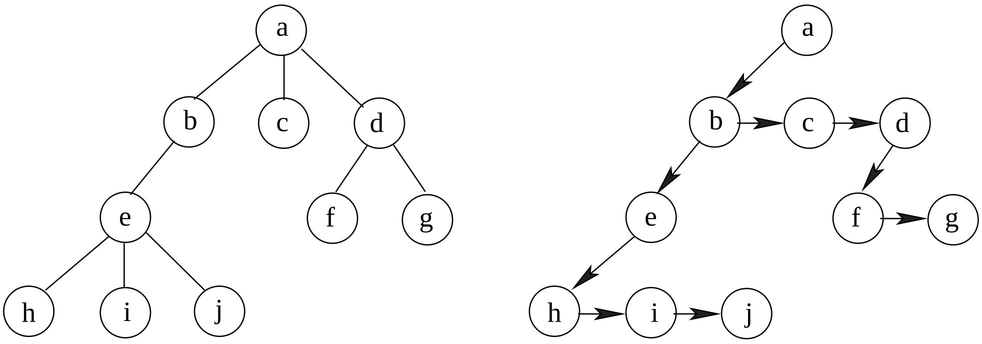
```
node_t *splitStr (char *str);
```

which receives such a string as parameter `str` and it returns the pointer to a list in which each element includes a sub-string defined in a dynamic way. The candidate has to define the node structure of the list as well.
For example, if the function receives the string `a.bb.ccc.dddd.eeeee.fffff` is should create the following list:



in which all strings are dynamically defined, and it should return the pointer `head` to the list head.

4 (6.0 points)
A tree of degree equal to n can be represented using nodes including n pointers (see left-hand picture), or, to avoid a too large number of pointers, using the left-child right-sibling technique. With this method each node includes a pointer to the leftmost child and to its nearest sibling to the right. The right-hand side picture shows the same tree of the left-hand side representation using this technique.



Let us suppose that each node of the tree is represented by a structure named `node_t`. `node_t` includes two strings (first and last name) and an integer value (an examination mark). Define the C structure to represent the tree, and write the recursive function which prints-out the entire content of a tree once received its root pointer as a parameter.