01OGD Algorithms and Programming

24/02/2014 – 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:

3-10, 5-3, 1-5, 7-2, 3-8, 5-4, 0-9, 1-5, 8-9, 10-4

apply an on-line connectivity algorithm with quick union, showing at each step the contents of the array and the forest of trees at the final step. Nodes are named with integers in the range from 0 to 10.

2. (1 point)

Sort in ascending order with counting-sort the following array of integers:

3 4 5 2 6 5 4 2 8 1 4 1 8 0 4

Show the contents of the data structures used at intermediate steps.

3. (2 punti)

Suppose to have an initially empty priority queue implemented with a heap. Given the following sequence of integers and * character: 10 22 29 71 * * 58 * 34 9 31 * * 14 * 41 27 18 where each integer corresponds to an insertion into the priority queue and character * corresponds to an extraction of the maximum, show the priority queue after each operation and return the sequence of values extracted.

4. (**2 points**)

Suppose to have a 12-node binary tree. During the visits the following 3 sequences of nodes are generated:

preorder	33	41	23	1	7	11	17	19	13	3	27	5
inorder	23	41	7	1	11	33	13	19	17	27	3	5
postorder	23	7	11	1	41	13	19	27	5	3	17	33

Draw the original binary tree.

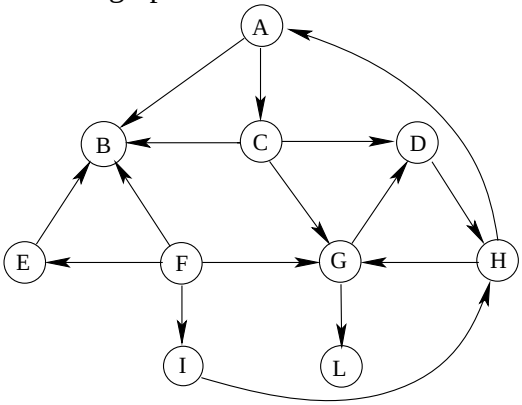
5. (1 point)

Suppose that all integers in the range 1-1000 are stored in a Binary Search Tree. Suppose you are looking for key 531. Which ones among these sequences **cannot** be a sequence found during search? Why?

500	550	510	540	533	539	532	531				
500	600	580	570	510	520	530	540	535	531		
570	100	200	300	400	500	660	550	520	525	530	531
610	30	600	60	588	117	519	299	301	477	531	

6.

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.

Algorithms and Programming

Examination Test

Programming Part
24 February 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.**

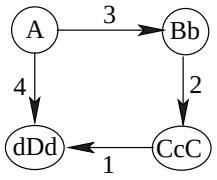
Intermediate solutions are forbidden.

1 (18.0 points)
A weighted directed graph is stored in a file throughout the list of its arcs.
Each line of the file has the following format:

`idV1 val idV2`

which indicates that vertex `idV1` is linked with an oriented arc of weight `val` (integer value) to vertex `idV2`. Note that the number of vertices of the graph is undefined, and that each vertex is identified with an alphanumeric identifier of maximum length equal to 20 characters.
The following figure illustrates an example of such a file, and the corresponding graph.

```
A 3 Bb
Bb 2 CcC
A 4 dDd
CcC 1 dDd
```



Write a C program such that:

- Receives the name of the file storing the graph description on the command line.
- Reads the file and stores the graph in a proper data structure.
- Verifies whether the graph is a regular graph. An oriented graph is said to be regular if each vertex has the same number of neighbors, i.e., if and only if all vertices have the same input and output degrees and those values are the same for all vertices.
- Performs a cycle in which a vertex identifier is read.i
  - If the identifier is `‘‘end’’` the program terminates.
  - Otherwise, the program has to find the simple path with source in the read vertex for which the sum of the weights is larger. For such a path, the program has to print-out on standard output the list of weights, the lists of arcs, and the total weight.

continue

2 (2.0 points)
Given an array of N integer values, write a function `ROTATE` which right or left-shift the array of P positions, where P is an integer value (positive for right-shifts. negative for left-shifts).
It is forbidden to implement the shift of P positions as the iteration of P shifts of 1 position each (which would have cost $O(N \cdot P)$).
Instead, it is requested to use an algorithm of cost $O(N)$, which works as follow. It uses an auxiliary array of P positions, dynamically allocated, on which P elements of the original array are temporarily stored. Then, it moves the remaining $N - P$ elements in their final destination. Finally, the P elements stored in the auxiliary array are moved to their final position. The function has to free the allocated memory at the end.

3 (4.0 points)
A matrix is said "sparss" when the majority of its elements are equal to zero. In such a case, it may be convenient to use a data structure in which only the element which differ from zero are allocated.
Write a C function for a sparse matrix of `float` whose prototypes is the following one:

```
void matInsert (matr_t *mat, int r, int c, float val);
```

which insert the value `val` into the element of row `r` and column `c` in the matrix `mat`.
The type for `matr_t` is a structure which includes the two dimensions (number of rows `nr` and number of columns `nc`), it points to the array of pointers (of size `nr`) to row, each of which is implemented through a list, containing only the elements which differ from zero.
Define the structure `matr_t`, the data structure used for the lists along each row, and write the function `matInsert`.

4 (6.0 points)
Write a recursive function able to generate all strings of length N formed by the 5 capital vocal letters 'A', 'E', 'I', 'O', 'U'. The function has to satisfy the following constraints:

- The value of `N` has to be received by the program on the command line, and it has to be $N \geq 5$.
- In the generated string all vocals have to appear at least once.

The recursive function has to fix the i-th letters of the final string at the recursion level i-th.