01OGD Algorithms and Programming

02/02/2015 – 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. Node names are integers in the range from 0 to 10.

2. (**1 point**)
Sort in ascending order with Shellsort the following array of integers:

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

Use Knuth's sequence and show relevant intermediate steps.

3. **(2 points**)
Suppose to have an initially empty priority queue implemented with a heap. Given the following sequence of integers and * character:

20 32 19 51 * * 28 * 74 9 81 * * 17 * 41 33 18

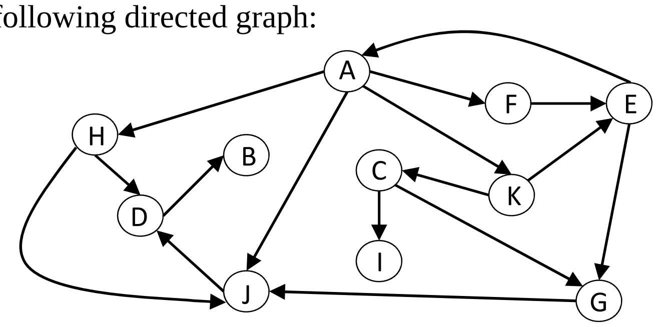
where each integer corresponds to an insertion into the priority queue and character * corresponds to an extraction of the maximum, show the priority queue after each operation and return the sequence of values extracted. At the end, change the priority of 41 into 11 and draw the corresponding priority queue.

4. (**1 point**)
Express in prefix and postfix notation the following arithmetic expression. To do so, visit the corresponding binary tree.

((A +B) * (C*(D-E))) /((E-F) + G*H)

5. **(2 points**)
Given the sequence of keys FEBBRAIO, where each character is identified by its index in the English alphabet (A=1, ..., Z=26), draw the final configuration of an initially empty hash table of size 19 where insertion of the the previous sequence character by character occurs. Assume open addressing with quadratic probing with $c_1 = 1$ and $c_2 = 1$. Show relevant intermediate steps.

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
02 February 2015

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 file contains a sequence of integer numbers on the same or on following lines. The first value indicates the number of integers following the first one. The following is a correct example of such a file:

```
5
100 140 200 80
50
```

All integer values stored in the file, but the first one, indicate the values (in million of euros) of a set of properties belonging to a wealthy man.
This man wants to dispose his estate among `n` heirs in a fair way.

To help the wealthy man, it is requested to write a C program able to:

- Receive two strings and an integer value on the command line. The first string is the name of the input file (with the format previously described). The second string is the name of an output file (which has to be generated by the program). The integer value is the number of heirs `n`.
- Call a recursive function able to split all properties in `n` sets, each one going to one of the `n` heirs, such that a specific optimum criteria is satisfy.
- Write the function to check the optimum, i.e., the sum of the absolute values of the differences among all the inheritances has to be minimum.
- Store all inheritances in the output file, writing each subset on a separate row of the file.

For example, for the previous set of properties the optimum splitting among `n` = 3 heirs, and the format of the output file are the following

```
200
100 80
140 50
```

For this solution, the optimization function, used to evaluate the solution, assume the following value:

```
|200-180| + |200-190| + |180-190| = 40
```

continue

2 (2.0 points)
Write function

```
void eraseDuplicate (char *str);
```

that erase duplicated characters in a string. In other words, given the string `str`, the function returns the same string in which, for all characters appearing in the string more than once, only the first occurrence is retained.
For example, if the string initially contains “`aa;;;bbbab;`” the function has to return string “`a;b`”.

3 (4.0 points)
A tree of degree equal to `n` is defined using the following C structure:

```
struct node {
  int key;
  struct node *child[N];
};
```

Write function

```
void visitLevelByLevel (struct node *root, int l1, int l2);
```

which visits the tree from depth `l1` to depth `l2` (with `l2` > `l1`) and prints out all visited keys on a level-by-level basis, i.e., it prints out all keys at depth `l1`, followed by all keys at depth `l1+1`, etc., followed by all keys at depth `l2`.
Notice that the tree can be visited more than once to obtain the desired result.

4 (6.0 points)
An alphanumeric acronym of length `n` is composed by selecting for each location of `acronym[i]` a character from the set `set[j]`. Each set `set[j]` is given as a string of maximum length equal 10. A file contains information items related to the length of the acronym `n` (integer on the first row) and to the strings that represent all sets `set[j]` (i.e., `n` strings stored on `n` rows of the file).
Write a recursive program which is able to read the file, to generate all possible acronyms, and to store them in a second file. The name of the two file are given on the program command line.
Example
Given the following content of the first file:

```
3
A
Xy
123
```

the function should write in the second file the acronyms:

```
AX1 AX2 AX3 Ay1 Ay2 Ay3
```