01OGD Algorithms and Programming

27/06/2014 – Part I: Theory (12 points)

1. (1 point)

Using MergeSort, sort in ascending order the following array of integers:

11 9 12 1 13 3 39 23 29 17 21 10 18 16 8 100

Report relevant steps.

2. (2.5 points)

Using a greedy approach find an optimal Huffman code for the following characters. Each character has an associated integer frequency:

a: 6 b: 16 c: 4 d: 3 e: 22 f: 11 g: 14 h: 9 i: 8 l: 10

3. (2 points)

Given the sequence of keys THELASTSTAND, where each character is identified by its progressive order in the alphabet (A=1, ..., Z=26), draw an initially empty hash table of size 19, after the insertion of the above characters in sequence. Use linear chaining and the hash function *h(k) = k mod 19*.

4. (1 point)

Execute in sequence the following operations on an initially empty BST (+ means insertion in leaf, − means deletion):

+10 +4 +3 +16 +21 +7 +11 +1 +8 +19 +12 +15 -4 -10 -7

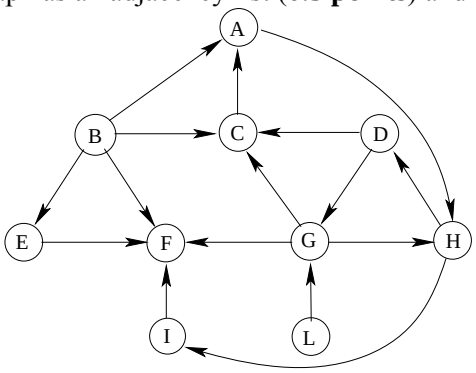
5. (2.5 punti)

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

10 4 3 16 21 7 11 1

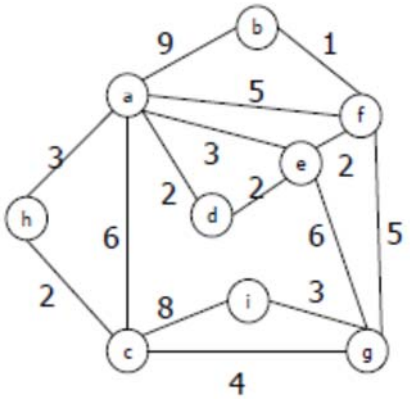
6. (2 x 0.5 points)

Represent the following directed graph as an adjacency list (**0.5 points**) and as an adjacency matrix (**0.5 points**).



8. (2 points)

Given the following undirected and weighted graph:



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

Algorithms and Programming

Examination Test

Programming Part
27 June 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 data base stores information about car manufactures, specifying the list of producers, car models (for each producer), and car accessories (for each car model). The data base is organized in the following way:

- A first file stores the list of producers. For each producer its name is stored on a single file line, followed by the file name containing the name of the car models it manufactures. The number of producers in unknown.
- For each producer there is a file storing the car models fabricated. The name of each car model is specified on a single line of the file, followed by the file name storing the related accessory list.
- For each car model there is a file storing the list of accessories. Each accessory is specified on a file line, followed by its price in Euro.

Notice that all names are alphanumeric strings where spaces are substituted with underscores ‘‘_’’, and all strings have a maximum length of 100 characters. Moreover, all producers, car models, and accessory names are unique in the entire data base.

Example

Producer file
Audi Audi_models.txt
BMW BMW_models.txt
FIAT FIAT_models.txt
Ford Ford_models.txt
Renault Renault_models.txt
Toyota Toyota_models.txt
Volkswagen Volkswagen_models.txt

File `FIAT_models.txt`
Cinquecento Cinquecento_accessories.txt
Bravo Bravo_accessories.txt
Doblo' Doblo'_accessories.txt
Freemont Freemont_accessories.txt
Panda Panda_accessories.txt
Punto Punto_accessories.txt

File `Bravo_accessories.txt`
Surfing_Bars 150.00
Spoiler 220.50
Alloy_wheels 75.25
Viva-voce 450.50
Sensor_parking 700.50

Write a C program able to:

- Receive the producer file name on the command line.
- Read the producer, all model and all accessory files and store all data in a proper data structure.
- Run through a user menu, in which the user is able to:
  - Read (from standard input) a manufacturer name, and print-out (on standard output) the model list produced by that manufacturer. The complexity has to be at most logarithmic in the number of manufacturers.
  - Read (from standard input) a model name, and print-out (on standard output) the accessory list for the model. The complexity has to be at most logarithmic in the total number of models.
  - Erase a producer, with all its car models, and all accessories of those car models.
  - Erase a car model and all its accessories.
  - Erase a single accessory.

continue

2 (2.0 points)
Two arrays of integers `vet1` e `vet2` of size `d1` and `d2`, respectively, are given. Write a function

```
int search (int *vet1, int *vet2, int d1, int d2);
```

that searches the array `vet2` as a sub-array of the array `vet1`. In case `vet2` does appear in `vet1`, the function has to return the array index from which `vet2` is starting within `vet1`. In case `vet2` does not appear in `vet1` the function has to return `-1`.

Example
Let us suppose that the arrays `vet1` e `vet2` have sizes `d1=10` and `d2=4`, and they have the following content:

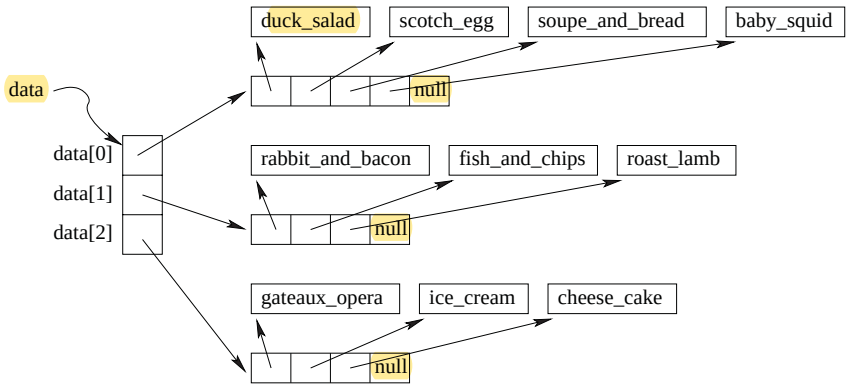
```
vet1: 1 2 3 4 5 6 7 8 9 10
vet2: 4 5 6 7
```

The search has to end positively, and to return the value `3` (index in `vet1` of the value `4`).

3 (4.0 points)
A bi-linked list (linked on both right and left sides) of integer is given. Let `node_t` be the struct type of each element list. The definition of `node_t` has to be given by the student. Implement the following two functions:

- `void listInsert (node_t **left, node_t **right, int key, int leftRight)`
  that insert the value `key` at the left-hand side (`left`) of the list if the parameter `leftRight` is equal to `0`, or to the right-hand side (`right`) if the parameter `leftRight` is equal to `1`.
- `void listDisplay (node_t *left, node_t *right, int leftRight)`
  that print-out (on standard output) the content of the entire list, visiting it from left-to-right if the parameter `leftRight` is equal to `0`, or from right-to-left if `leftRight` is equal to `1`.

4 (6.0 points)
A restaurant has a fixed-price menu made-up of `n` courses. For each course, the customer has to select among a list of possibilities. The menu is stored in an array of `n` elements. Each element is a pointer to an array of pointers to strings. This last array indicates the dishes available within that course. The number of dishes is not given, but a `NULL` pointer in the array indicates the end of the list. Each string referenced by the array indicates one dish. The following picture represent the status for a restaurant with three main courses ,and with 4, 3 and 3 dishes per courses 0, 1, and 2, respectively. The data structure is called `data`, and an example it is represented in the following picture.



Write the function

```
void buildMenu (char **data[], int n);
```

that, adopting recursion, is able to print-out (on standard output) all possible sequences of dishes, i.e., all menu composition made-up of one dish extracted from the corresponding course. In the above example, the output can be something like:

```
menu_1: duck_salad rabbit_and_bacon gateaux_opera
menu_2: duck_salad rabbit_and_bacon ice_cream
menu_3: duck_salad rabbit_and_bacon cheese_cake
menu_4: duck_salad fish_and_chip gateaux_opera
menu_5: ...
```