Course "Data Structures and Algorithms" – PART2

Prof. Roberto Sebastiani

February 17^{th}, 2011

Test code:	32
Name:	**Riccardo**
Surname:	**Steffan**
Matriculation #:	9296

[COPY WITH SOLUTIONS]

Instructions: for each question, insert your answer below the text of the questions itself. You can use auxiliary empty paper, but these will not be corrected.

The text of the answers returned by the student must be clear, understandable and unambiguous. The handwriting must be readable. Ambiguities, lack of clearness and understandability affect negatively the evaluation of the answers. The examiners' interpretation of the handwritten text is indisputable.

Score: Each exercise EXi has a given weight $w(EXi)$, s.t. $\sum_i w(EXi) \geq 30$:

EXi:	Ex.1	Ex.2	Ex.3	Ex.4	Ex.5
w(EXi):	7	5	7	7	7

During the correction it will be assigned a given percentage $p(EXi)$ of correctness, from 0% (wrong) to 100% (perfect). No penalties are given for wrong answers. Then the score, expressed in 30's, is computed as:

$$score = round(\sum_i w(EXi) \cdot p(EXi)/100).$$

The part is passed if $score \geq 18$.

Examples:

- solving 100% of all exercises: 33 (30L) $\Longrightarrow$ passed
- solving 100% of first 3 exercises, 0% the others: 19 $\Longrightarrow$ passed
- solving 100% of first 2 exercises, 50% of the third, 0% of the others: 16 $\Longrightarrow$ failed

This page is willingly left empty.

Ex.1 Write a method in java:

```
public static int linearSearch(int a[], int v)
```

which, taking as input an array of integers and an integer value v, returns the position of the *last* occurrence of the element v in a, if present, -1 otherwise. The method must implement an iterative version of the linear search algorithm. It should be as efficient as possible.

NOTE: "by "position" it must return the user-perspective notion of position: e.g., if `v` is the 3rd element of the array, it must return 3, not 2.

[Solution: One possible implementation is the following:

```
public static int linearSearch(int a[], int v) {
     int i = a.length-1;
     while (i >=0 && a[i] != v) i--;
     if(i >= 0 )
       return i+1;
     else
       return -1;
 }
```

]

Ex.2 Consider the recurrence

$$T(n) = 3^5 \cdot T(n/9) + 4 \cdot n^2\sqrt{n}.$$

1. Is it possible to apply Master Method?
2. If yes, use one condition of Master theorem to compute the asymptotic behavior of $T(n)$. If not, explain why in detail.

[Solution:

1. Yes. In fact, $T(n)$ is in the form $a \cdot T(n/b) + f(n)$ s.t. $a = 3^5$, $b = 9$ and $f(n) = 4 \cdot n^2\sqrt{n}$; hence $log_b(a) = \frac{log_2(a)}{log_2(b)} = 5/2$, s.t. $n^{log_b(a)} = n^{5/2} = n^2\sqrt{n}$.
2. Since $f(n) = \Theta(n^{log_b(a)})$, it is possible to apply the second condition of Master theorem, from which we can conclude that $T(n) = \Theta(n^{log_b(a)} \cdot log(n)) = \Theta(n^2\sqrt{n} \cdot log(n))$

]

Ex.3 Write a method in java:

```
public static int[] merge(int a[], int b[])
```

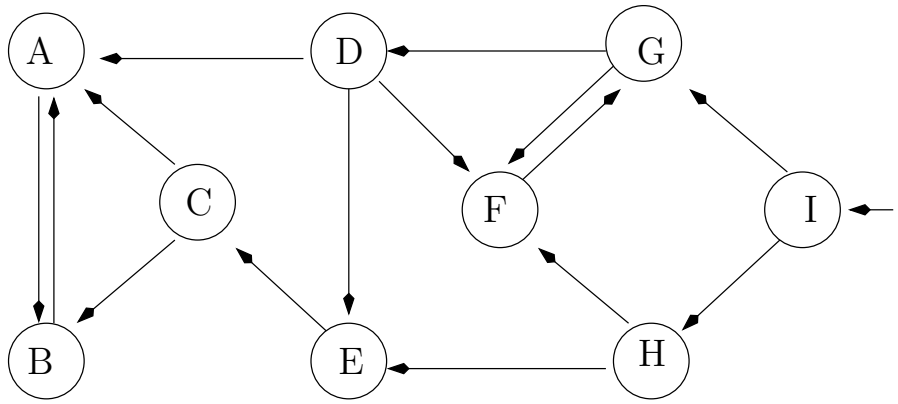
which, taking as input two arrays of integers which are sorted in increasing order, returns the array resulting from the merging of the two arrays.

[Solution: One possible implementation is the following:

```
public static int[] merge(int a[], int b[]) {
  int[] c = new int[a.length+b.length];
  for (int i=0,j=0,k=0;k<a.length+b.length;k++) {
    if (j==b.length || (i<a.length && a[i]<b[j])) {
       c[k]=a[i];
       i++;
    }
    else {
      c[k]=b[j];
      j++;
    }
  }
  return c;
}
```

]

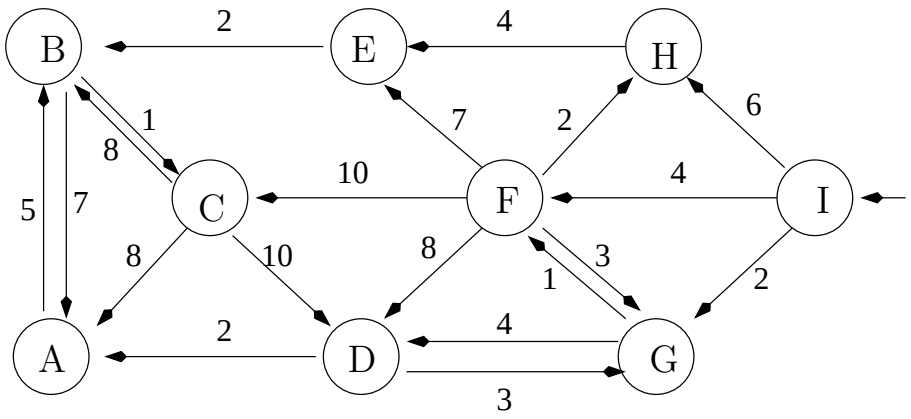
Ex.4



Given the graph in the figure, draw the sequence of arcs explored by a breadth-first search (BFS) starting from the node I. (Notation: use e.g., "AB" to denote the arc "$A \longrightarrow B$".)

[Solution:
IH, IG, HF, HE, GF, GD, FG, EC, DF, DE, DA, CB, CA, AB, BA]

Ex.5 Consider the following undirected weighted graph:



Compute the shortest-path tree from node I using Dijkstra's algorithm. In particular:

- list the arcs of the final shortest-path tree, in the order they have been added to the tree.
- for each node, indicate its distance from I.

[Solution:

- Tree: IG, GF, FH, GD, DA, HE, EB, BC
-

Node	*Dist. from I*
I	0
G	2
F	3
H	5
D	6
A	8
E	9
B	11
C	12

]