Algorithms and Data Management
Written Exam

G. Falquet, V. Daponte

January 29, 2018

Remark. The algorithms must be written in Scala. Minor syntax errors are allowed, provided the code remains non ambiguous

Question 1 - Algorithm on arrays

Find an algorithm that, given an array L of positive and negative numbers, constructs a new array N that

- contains the same elements as L
- is ordered so that a negative number is never followed by another negative number.

For example if

$$L = (-1, -2, 6, -6, 66, 4, 1, -5, -7, -9, 14)$$

a possible content for N is

$$(-1, 6, -2, 66, -6, 4, -9, 1, -5, 14, -7)$$

.

1. propose a necessary and sufficient condition (on L's content) under which this problem is solvable
2. write the algorithm in Scala

Question 2 - Complexity

suppose that
- t and u are arrays of integer numbers of size N and M respectively
- u is sorted in increasing order
- t is not sorted

1. what is the time complexity, as a function of N and M, of the following algorithm

```
def finder(t: Array[Int], u: Array[Int]):Int = {
val M = u.length
var k = 0
for (x <- t) {
   var p = 0
   var r = M-1
   var s = (p+r)/2
   while (u(s) != x && p <= r) {
      if (u(s) < x) r = s+1
      else r = s-1
      s = (p+r)/2
   }
   if (u(s) == x) k += 1 }
return k
}
```

2. What does k represent at the end of the algorithm's execution?

3. Now suppose that t is also sorted

   - explain how the algorithm could be optimized for this case
   - what would be its time complexity?

Question 3 - Recursion

In the following table, the cells in the first column and the cells on the diagonal contain 1's. The value of a cell not in the first column and not in the diagonal is is the sum of the cell just above it and the cell on the left of the one above it.

1					
1	1				
1	2	1			
1	3	3	1		
1	4	6	4	1	
1	5	10	10	5	1

1. Write a recursive function *f(i:Int, j:Int):Int* that computes the value of the j-th cell of the i-th row of this table. For instance:

   f(1,1) => 1
   f(2,1) => 1

f(2,2) => 1
f(3,2) => 2
f(5,3) => 6
f(6,3) => 10

2. Estimate the time complexity of this function.

3. Can you think of a more efficient way to compute $f(i, j)$?

Question 4 - Graphs

A *Graph* class is defined as follows:

```
case class Graph() {
  var nodes = Set[Node]()
  var adj = Map[Node, Set[Node]]()
    // the set of nodes adjacent to a given node
  var w = Map[(Node, Node), Double]()
   // w(n1,n2) is the weight of the arc (n1, n2)

}
```

with *Node* defined as (for instance)

```
case class Node(name : String)
```

1. Suppose that the map w has been filled with the weights of the (directed) edges of the graph. Write a method *fillAdj* that uses w to fill the adjacency map *adj* (a node n belongs to the adjacency set of a node k if there is a key → value pair $(k, m) \rightarrow x$ in w)

   Remark. the method _1 (resp. _2) extracts the first (resp. second) element of a pair.

   Remark 2. the method *keySet* yields the keys of a map

2. Write a method *minCostAt3* that performs the following operation : for a given node n, find the node at distance 3 with minimal cost, i.e. a node that can be reached by starting at n and traversing 3 edges and such that the added weights of these edges is minimal.

Question 5. Databases

Given the following relational schema (the primary keys are underlined)

```
country(name, population, area)
river(name, length)
cityOn(river, seqNo, city)
riverFlowsThrough(river, country)
city(name, population, country)
```

where a tuple (r, n, c) of *cityOn* represents the fact that c is the nth city along river r.

1. Write SQL queries to answer the following questions:

   (a) what are the countries that have a city with more than 1000000 inhabitants?

   (b) what are the cities that are on two (different) rivers?

   (c) what is the last city along river ’Congo’? (may require subqueries)

   (d) for each river what is the average density of population in the countries it flows through

2. For each table provide its foreign keys (if it has some)