Test Pearl 001 — Algorithmics

Pearls of Computer Science (201300070) / Introduction to BIT (201300073)
15 september 2017, 13:45–14:45
Module coordinator: Doina Bucur
Instructor: Arend Rensink

- You may use 1 A4 sheet with your own notes for this test, as well as a *simple* calculator
- Scientific or graphical calculators, laptops, mobile phones, books etc. are not allowed. ***Put those in your bag now (with the sound switched off)!***
- Total number of points: 100

15 points **Question 1** Suppose that you execute the following assignments in Python:

```
range = [17, 22]
```

`range` consists of the lowest student age and highest student age of the first-year students enrolled at the UT.

(a) Write a Python condition (*not* an entire `if`-statement) that tests whether the number `a` falls into the range from lowest up to (and including) highest student age.
(b) Assuming that `a` is higher than `22`, write a Python assignment that assigns to `range` an extended range that includes `a`. (Do not use the concrete values of `range`)
(c) Write a Python `while`-loop that assigns to a new variable `ages`, initialised to the empty list `[]`, the list of all actual ages from the lower bound to the upper bound of `range`. *(Hint: the function to add a single element to the end of a list is `append`.)*

15 points **Question 2** Analyse the following Python-function.

```
1  def compute(data, size, bound):
2      result = []
3      i = 0
4      while i < len(data) and len(result) < size:
5          print("i=", i, ", result=", result)
6          if data[i] < bound:
7              result.append(data[i])
8          i = i+1
9      return result
```

(a) Show what happens in a call of `compute([63,3,37,13,324],2,20)`, by writing down
  (i) the output of the `print`-statement in line 5 every time it is executed;
  (ii) the return value of the call.
(b) What does the function `compute` actually do? *(Do not give a step-by-step explanation of the execution, but describe the purpose of the function.)*

10 points **Question 3** How many steps does `compute` (in Question 2) need in terms of the length (say n) of the input list `data`: approximately ("in the order of") $\log_2 n$, approximately $\sqrt{n}$, approximately n or approximately n^2? Explain your answer.

10 points **Question 4** What would happen if the condition in line 4 of `compute` (Question 2) were replaced by only `len(result) < size`? Explain your answer.

20 points **Question 5** Consider the list `[20, -2, 7, -10, 21, 0]`.

(a) Show how bubble sort sorts this list, by writing down the list after every single modification.
(b) Show how merge sort sorts this list, by schematically showing how the list is split and zipped back together.

10 points **Question 6** Suppose you want to order a pack of cards so that all clubs (♣) come first, followed by all diamonds (♢), all hearts (♡) and all spaces (♠). Within each suit, the cards are to be ordered from lowest (the 2) to highest (the ace). Here are two ways to do this:

1. Apply merge sort to the entire pack, where one card is considered to be smaller than another if it has a smaller suit (according to ♣ < ♢ < ♡ < ♠) or, if the suit is the same, a smaller value.
2. First distribute the cards into piles corresponding to the four different suits, then apply merge sort to each suit, and then put the piles back together again.

Which of these two solutions is faster? Explain your answer by analyzing the complexity of each solution.

20 points **Question 7** You have a tightly packed bookshelf with a large collection of Science Fiction books, arranged in alphabetical order by author. Yesterday, from a friend you got 10 books (you do not yet know which ones) in a messy pile, ehich you now want to add to your collection while keeping the order intact.

(a) Write an algorithm in natural language, with unambiguous, numbered instructions, that allows you to reach your goal in as few steps as possible. Your instructions may only involve one or two books at a time, never an arbitrary set of them.

You may assume that you have ample space to arrange the books in any way you like. The books do not have to end up on the shelf they started on, but they should end up on a *single* shelf.

Do not try to give an answer in Python!

(b) How many steps does your algorithm need in the worst case, as a function of the number of books already on the shelf?