Exam – Datastrukturer

DIT960 / DIT961, VT-18
Göteborgs Universitet, CSE

Day: 2018-10-12, *Time:* 8:30-12.30, *Place:* SB

Course responsible

Alex Gerdes, tel. 031-772 6154. Will visit at around 9:30 and 11:00.

Allowed aids

One hand-written sheet of A4 paper. You may use both sides. You may also bring a dictionary.

Grading

The exam consists of *six questions*. For each question you can get a U, a G or a VG. To get a G on the exam, you need to answer at least *four* questions to G or VG standard. To get a VG on the exam, all answers need to be correct and you need to answer at least five questions to VG standard.

A fully correct answer for a question, including the parts labelled "For a VG", will get a VG. A correct answer, without the "For a VG" parts, will get a G. An answer with minor mistakes might be accepted, but this is at the discretion of the marker. An answer with large mistakes will get a U.

Inspection

When the exams have been graded they are available for review in the student office on floor 4 in the EDIT building. If you want to discuss the grading, please contact the course responsible and book a meeting. In that case, you should leave the exam in the student office until after the meeting.

Note

- Begin each question on a new page.
- Write your anonymous code (not your name) on every page.
- When a question asks for pseudocode, you don't have to write precise code, such as Java. But your answer should be well structured. Indenting and/or using brackets is a good idea. Apart from being readable your pseudocode should give enough detail that a competent programmer could easily implement the solution, and that it's possible to analyse the time complexity.
- Excessively complicated answers might be rejected.
- *Write legibly!* Solutions that are difficult to read are not evaluated!

Exercise 1

The following algorithm takes as input an array, and returns the array with all the duplicate elements removed. For example, if the input array is $\{1, 3, 3, 2, 4, 2\}$, the algorithm returns $\{1, 3, 2, 4\}$.

```
S = new empty set
A = new empty dynamic array
for every element x in input array
  if not S.member(x) then
    S.insert(x)
    A.append(x)

return A
```

What is the big-O complexity of this algorithm, if the set is implemented as:

a) an AVL tree?

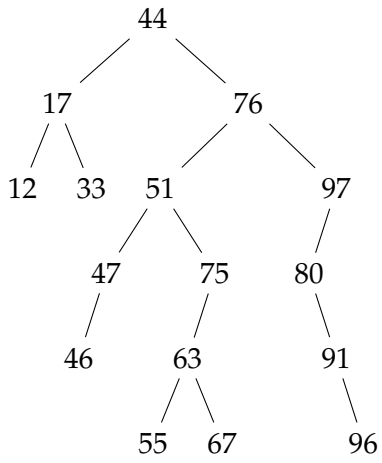
b) a hash table?

Write the complexity in terms of n, the size of the input array.

For a VG only: Write the complexity in terms of n and m, where n is the size of the input array and m is the number of distinct elements in the array (i.e., the number of elements ignoring duplicates).

Exercise 2

You are given the following binary search tree:



a) In which order could the elements 17, 33, 47, 63, 67 and 76 have been added to the tree? There are several correct answers, and you should choose them all.

A) 17, 33, 47, 63, 67, 76

B) 17, 76, 33, 63, 47, 67

C) 17, 76, 47, 33, 67, 63

D) 76, 63, 67, 47, 17, 33

E) 76, 67, 17, 33, 47, 63

b) Remove 76 from the tree, then insert it again. How does the tree look?

c) **For a VG only:**

Implement binary search in Java. Your method should have the following type:

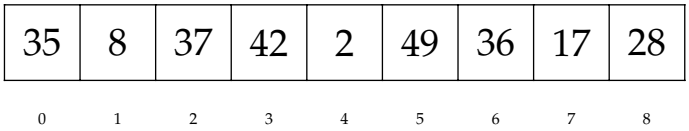
```
int search(Comparable[] array, Comparable key)
```

It should return the *index* of the object, or `-1` if the object could not be found. It should take $O(\log n)$ time, and must not use recursion, or any Java standard library methods.

Warning: there are many different ways to write a *wrong* binary search! To pass this question, your code must be correct. *Make extra sure that you cannot get into an infinite loop*. You do not, however, have to consider integer overflow.

Exercise 3

Perform a quicksort-partitioning of the following array:



Show the resulting array after partitioning it with the following pivot elements:

a) first element

b) middle element

c) last element

Also, highlight which subarrays that need to be sorted using a recursive call. (for example by drawing a line under each subarray)

For a VG only:

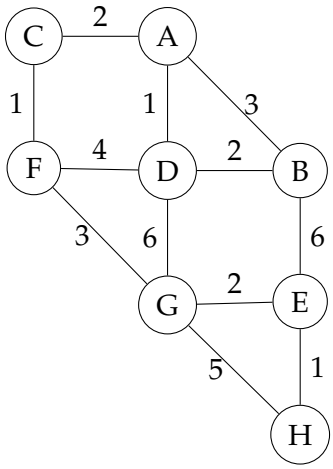
Define a Haskell function `sort` that sorts list of booleans. (When comparing booleans, False $<$ True.) It should have the following type signature:

```
sort :: [Bool] -> [Bool]
```

Your method should take $O(n)$ time. *Hint*: You can solve the problem in less than 10 lines of code.

Exercise 4

You are given the following undirected weighted graph:



a) Compute a minimal spanning tree for the following graph by manually performing Prim's algorithm using D as starting node.

Your answer should be the set of edges which are members of the spanning tree you have computed. The edges should be listed in the order they are added as Prim's algorithm is executed. Refer to each edge by the labels of the two nodes that it connects, e.g. DF for the edge between nodes D and F.

b) **For a VG only:** Suppose we perform Dijkstra's algorithm starting from node H. In which order does the algorithm visit the nodes, and what is the computed distance to each of them? There are several possible orders the algorithm might visit the nodes in – you may choose any of them.

Exercise 5

Suppose you have the following hash table, implemented using *linear probing*. For the hash function we are using the identity function, $h(x) = x \bmod 9$, where mod is the modulo operator that calculates the remainder of a division.

9	18		12	3	14	4	21	
0	1	2	3	4	5	6	7	8

a) In which order could the elements have been added to the hash table? There are several correct answers, and you should choose them all.

A) 9, 14, 4, 18, 12, 3, 21

B) 12, 3, 14, 18, 4, 9, 21

C) 12, 14, 3, 9, 4, 18, 21

D) 9, 12, 14, 3, 4, 21, 18

E) 12, 9, 18, 3, 14, 21, 4

b) **For a VG only:** Add 30 to the hash table (assume there is no rehashing). What does the hash table look like now?

c) **For a VG only:** Remove 3 from the hash table. What does it look like now?

Exercise 6

Design an algorithm that takes:

- An array containing n distinct natural numbers
- A number $k \leq n$

and calculates the sum of the k largest numbers in the array.

For example, if the array is $\{3, 7, 5, 12, 6\}$ and $k = 3$, then the algorithm should return 25 $(12 + 7 + 6)$.

Write down your algorithm as *pseudocode* – you don't need to write fully detailed Java code. *You may freely use standard data structures and algorithms from the course in your solution, without explaining how they are implemented.*

For a G: your algorithm should take $O(n \log n)$ time.

For a VG: your algorithm should take $O(n \log k)$ time.