Dugga/Partial Exam
Data Structures (DAT037/DAT036)

2014-12-01

- Lecturer: Ramona Enache
- For an assignment to be accepted as *solved*, it needs to basically be a correct solution. Small errors could be accepted. However, please note that the Dugga will be graded more strictly than the final exam.
- Don't leave solutions for several assignments on the same sheet.
- Write clearly; unreadable = wrong!
- Solutions which are unnecessarily complicated or poorly motivated won't be accepted.
- Use the uniform cost model for analyzing complexity
- Unless the assignment specifies otherwise, you don't need to explain the data structures and algorithms from the course, but to motivate their use.
- Since certain data structures have different implementations, which yield to different complexities for certain operations, you should specify briefly which implementation of the data structure you use if you need to compute the complexity of an algorithm which uses that data structure.

1. 1. What is the complexity of the following code ?

```
for (int j = 1; j <= n; j = j*2)
   a.add(j);
int v[] = a.toArray() ;
insert_sort(v);
```

   where

   - `n` is a positive integer
   - `a` is a dynamic array
   - `v` is a normal array
   - `a.toArray` is linear in the number of elements from a
   - comparisons take O(1) time
   - `insert_sort` is insertion sort

   Compute the time complexity in the O notation and justify it.

2. Write a program that verifies if a singly-linked list of n integers is a palindrome in O(n) time complexity.

   For this purpose you can only use any of the following data structures from without having to implement it from scratch: *linked lists (singly or doubly linked), stacks, queues, heaps, graphs.* No *arrays/dynamic arrays*, except if they are used for implementing one of the previous data structures.

   Motivate the final time complexity. What is your space complexity ?

   Can you come up with a couple of tests for your solution ? Test how it works on them.

   Write Java/Haskell/pseudocode.

   - For Java/pseudocode, use the following representation for linked lists

     ```
     class Node
             {int info;
              Node next};
     ```

     The list is represented by its `head` and the last element of the list has `next = NULL`.
   - For Haskell, use Haskell lists. However, you can only use the basic list constructs (`:` and `[]`) and no other standard/predefined list function (for example `++`). You need to implement from scratch any function for lists that you wish to use in your solution.

3. Write a program that tests that if a binary tree given as input is a BST (binary search tree). The binary tree contains generic information. Your program should be a function taking a binary tree as input and returning

`true/false`. Additionally you can implement and use helper functions. Also you can use any data structure from the course without having to implement them from scratch.

What is the time and space complexity of your solution ? Justify!

Can you come up with a couple of tests for you solution ? Test how it works on them. Write Java/Haskell/pseudocode.

The binary tree representation is the one from the lectures, just adapted for arbitrary type of information.