Linnaeus University
Department of Computer Science and Media Technology
Diego Perez

Exam in Algorithms and Advanced Data Structures, 1DV516, 4.5 cr.

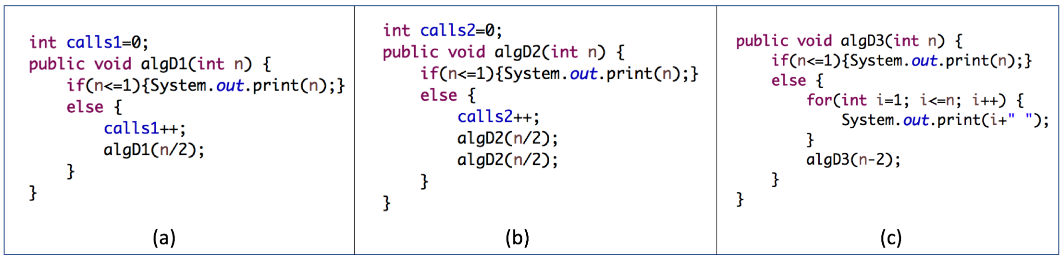
Friday 06 December 2019, 08.00–13.00

The exam consists of **4** questions. To pass you need 30 points.
Answer the questions in English.
Answer the questions using a pen. Do not use a pencil for your final answers.
When an algorithm is asked for, it should be understood that it should be as efficient as possible, and it should be expressed in such a way that it is understandable.
Allowed aids: Pen, pencil, eraser, and paper.
Not allowed: Consultation of any media outside of this document.

1. (a) Formally show that $O(N^2)$ means the same as $O(3N^2)$. (2p)
Help: A function f(N) is $O(g(N))$ if $\exists c, n_0 > 0 \mid 0 \leq f(N) \leq cg(N), \forall N \geq n_o$

(b) Formally show that the height of an AVL tree with N nodes is $O(\log N)$. (4p)

(c) Assuming that addition and print methods take constant time. Write the asymptotic time complexity of the following methods. Motivate your answer (for example, write the recurrence relations and solve them using one of the methods studied during the lectures). (6p: 2p each)



```
int calls1=0;
public void algD1(int n) {
    if(n<=1){System.out.print(n);}
    else {
        calls1++;
        algD1(n/2);
    }
}
```

(a)

```
int calls2=0;
public void algD2(int n) {
    if(n<=1){System.out.print(n);}
    else {
        calls2++;
        algD2(n/2);
        algD2(n/2);
    }
}
```

(b)

```
public void algD3(int n) {
    if(n<=1){System.out.print(n);}
    else {
        for(int i=1; i<=n; i++) {
            System.out.print(i+" ");
        }
        algD3(n-2);
    }
}
```

(c)

(d) Given a Hash Table that uses an array of length Z, write the maximum number of elements that we can guarantee that can be inserted in the following cases: (3p: 1p each)

- i) When the Hash Table solves collisions using separate chaining
- ii) When the Hash Table solves collisions using linear probing
- iii) When the Hash Table solves collisions using quadratic probing

2. The University registration system processes students' applications. Students submit their applications to the system through a method *submit(StudentInfo information)*. The *StudentInfo* class includes, among other information: 1) the ID of the student (we assume that the ID an alphanumeric value which concatenates the student's surname, name, birthdate; for instance Alison-Alice-19890908), 2) a public method *getID()* which returns the ID value, 3) a **real** value *averageGrade* between 0 and 10 that represents the average of the student grades in his/her previous courses, and 4) a method *getAverageGrade()* which returns the *averageGrade* value.

Administrators at the University obtain the next application to process through method *retrieveNext()*, which gives them an object of class *StudentInfo*. In the following, N represents the number of applications in the system; that is, the number of applications that have been submitted to the system but have not been processed by administrators yet. The University gives more priority to the registration of students with higher average grade. Therefore, method *retrieveNext()* returns the student's application that has the highest *averageGrade* value among the N applications in the system.

During the time interval between the moment when the student submits his/her information and the moment when the information is retrieved by an administrator, administrators can check the existence of an application. To do this, there is a method *approximateSearch(String partialID)* that returns the lowest and highest IDs in the system which match with the partialID. For example, if the system contains N=4 elements, with IDs `[Svenson-Sven-19800101, Fergusson-Fergus-19600606, Fredrikson-Fredrik-19730808, Fransson-Frans-18850505]`, then:

approximateSearch("F") returns [Fergusson-Fergus-19600606, Fredrikson-Fredrik-19730808
approximateSearch("Fr") returns [Fransson-Frans-18850505, Fredrikson-Fredrik-19730808]
approximateSearch("Fra") returns [Fransson-Frans-18850505]

(a) Describe a solution for the University registration process where all the three *submit, retrieveNext,* and *aproximateSearch* operations execute in logarithmic time. You must motivate that your solution works and argue that your solution actually executes in O(log N). You do not need to explain the internal execution of operations that you have studied during the lectures; you can just use them. (12p)

(b) The **A**dvanced **D**ata **S**tructures **e**xpert (ADSe) who is in charge of the registration system receives a system upgrade request. Now, at any point in time administrators would like to obtain an array with the N applications in the submission system. Moreover, administrators would like to receive this array sorted by the *ID* value of *StudentInfo* class. For this, they will use a method *getApplicationsSortedByID()*. This new method *getApplicationsSortedByID()* should run in O(N).
Describe an extension to your previous solution that includes the operation *getApplicationsSortedByID()* and satisfies the mentioned requirements: it executes in O(N) and the three other operations still run in O(log N). You must motivate that your solution works and argue that your new solution actually has the required time complexity. (6p)

3. A toothpaste company wants to carry out some statistical studies about the monthly amount of money that people spend in toothpaste. They know that nobody has spent during a month more than 1000 Swedish kronor (SEK). They have collected data and now they have an array with the monthly expenses in toothpaste of P people during the last M months. Therefore, the array contains $N = P \cdot M$ elements. They want to calculate the median value of the elements in the array. Give them a solution that calculates the median expenses in toothpaste in $O(N)$. You must motivate that your solution works and argue that your solution actually executes in $O(N)$. (8p)

4. Suppose there is a coffee machine, which works with cash, and that you have to implement the internal software that computes the coins returned to the customer as change. Given the amount of money to return M, you have to implement a method that always returns the minimum number of coins.

The typical *change-making problem* addresses the question of finding the minimum number of coins (of certain denominations) that add up to a given amount of money. The most appropriate algorithmic technique to use for solving this problem depends on the set of values of coins.

(a) Assuming that we can use only 3 types of coins, with values 1, 5, and 10; given array $Coins = [coin_1, ..., coin_C] = [1, 5, 10]$ and the amount of money to return M, write an algorithm that executes in $O(C)$ time (see that C is the number of elements in array $Coins$) that computes the minimum necessary number of coins. *Note 1:* You do not need to compute the concrete number of coins of each type, just compute the total number of coins. For instance, for M=16 you should return the value 3 (a coin of value 10, a coin of value 5 and a coin of value 1). *Note 2:* Assume that all arithmetic operations "+", "-", integer multiplication, integer division, and modulo operation execute in constant time (4p)

(b) Assuming that we can use only 4 types of coins, with values 1, 5, 8 and 10; given array $Coins = [coin_1, ..., coin_C] = [1, 5, 8, 10]$ and the amount of money to return M, write an algorithm that computes the minimum number of coins that are necessary and executes in $O(M \cdot C)$ time. *Note 1:* You do not need to compute the concrete number of coins of each type, just compute the total number of coins. For instance, for M=16 you should return the value 2 (two coins of value 8). *Note 2:* Dynamic Programming strategy can be applied to this problem. *Note 3:* Assume that all arithmetic operations execute in constant time. (12p)

(c) Given the same information (M, $[coin_1, ..., coin_C]$) and the decision problem, "*Is there a way in which I can give the change using less than S coins?*", to which class can you say that the decision problem belongs? (P, NP, NP-complete, NP-Hard). You must motivate your answer. (3p)