Linnaeus University
Department of Computer Science and Media Technology
Diego Perez

Exam in Algorithms and Advanced Data Structures, 1DV516, 4.5 cr.
Thursdayday 01 November 2018, 14.00–19.00

The exam consists of **4** questions. To pass you need 30 points.
Answer the questions in English.
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 $O(3N)$ means the same as $O(N)$ (2p)

(b) Assuming that addition and print methods take constant time. Write the asymptotic time complexity of the following methods (8p: 2p each)

```
public void alg1(int n) {
    if(n<=1){System.out.print(n);}
    else {
        for(int i=1; i<=n; i++) {
            System.out.print(i+" ");
        }
        System.out.println();
        alg1(n-1);
    }
}
```
(a)

```
public void alg2(int n) {
    if(n<=1){System.out.print(n);}
    else {
        for(int i=1; i<=n; i++) {
            System.out.print(i+" ");
        }
        System.out.println();
        alg2(n/2);
    }
}
```
(b)

```
public void alg3(int n) {
    if(n>1){
        if(n%2==0)  {alg3(n-2);}
        else        {alg3(n-1);}
    }
    System.out.print(n+" ");
}
```
(c)

```
public void alg4(int n) {
    if(n<=1){System.out.print(n);}
    else {
        for(int i=1; i<=n; i++) {
            System.out.print(i+" ");
        }
        System.out.println("    ");
        alg4(n/2);
        alg4(n/2);
    }
}
```
(d)

2. The University registration system processes the students applications following FIFO. That is, the first student who applies is the first to be processed by administrators. Students submit their applications to the system through a method *submit(StudentInfo information)*. Administrators at the University obtain the next application to process through method *retrieveNext()*, which gives them an object of class *StudentInfo*.

The *StudentInfo* class includes, among other information: 1) the ID of the student (we assume that the ID is a unique integer value, for instance, the personnummer), 2) a public method *getID()* which returns the ID value, 3) a value *averageGrade* in the set {A,B,C,D,E,F} that represents the average of the student grades in his/her previous courses, and 4) a method *getAverageGrade()* which returns the *averageGrade* value.

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, both the student and the administrators can update the information in the application through a method *updateInfo(int id, StudentInfo newInformation)*. In the following N is 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.

(a) Describe a solution for the University registration process where all the three *submit, retrieveNext,* and *updateInfo* operations execute in constant time in average. You must motivate that your solution works and argue that your solution actually executes in average O(1). You do not need to explain the internal execution of operations that you have studied during the lectures; you can just use them. (14p) [1]

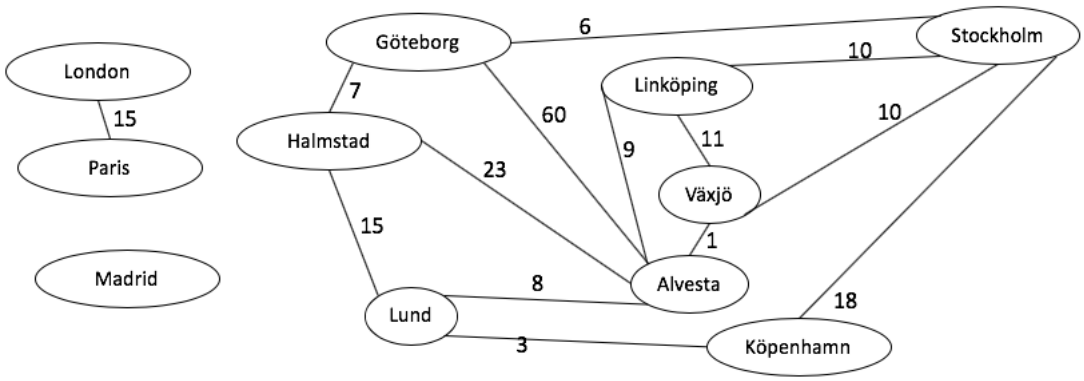
(b) The **A**dvanced **D**ata **S**tructures **e**xpert (ADSe) who is in charge of the registration system receives a system upgrade request. Now, at any point in time administrators would like to obtain an array with the applications in the submission system. That is, the applications that have been already submitted by students but not processed by administrators yet. Moreover, administrators would like to receive this array sorted by the *averageGrade* value of *StudentInfo* class. For this, they will use a method *getApplicationsSortedByGrade()*. This new method *getApplicationsSortedByGrade()* should run in O(N).
Describe an extension to your previous solution that includes the operation *getApplicationsSortedByGrade()* and satisfies the mentioned requirements: it executes in O(N) and the three other operations still run in average O(1). You must motivate that your solution works and argue that your new solution actually has the required time complexity. (6p) [2]

[1] *Alternative for obtaining half of the points (7p)*: In case you cannot find a solution that achieves the constant time in average, give a solution where each operation takes O(log N) in the worst-case.

[2] If you have chosen the *Alternative* option in part "a)", keep your *submit, retrieveNext,* and *updateInfo* in O(log N) and you will receive at most 3 points.

3. (a) We want to know the shortest travel time from Växjö to the rest of cities. Graph G(V,E) in next figure gives the amount of time required to travel between cities. Before calculating these times, we want to create a graph that only includes the cities that are connected to Växjö.

1. Give a method to discover the cities that can be reached from Växjo and whose time complexity is O(|V| + |E|). (3p)
2. Calculate the shortest travel times from Växjo to the rest of reachable cities applying Dijkstra's algorithm and show all the intermediate steps of the algorithm in a table. (5p)



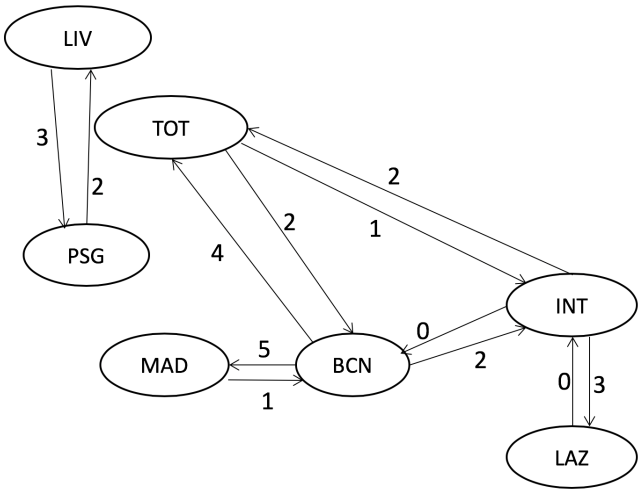
(b) We want to represent in a weighted directed graph G(V,E) the results of sport events during a championship. Each event consists of a match between two teams. Teams are represented by their 3-character acronym. There is a vertex in V for each team in the championship, and two edges for each match between two teams. An edge *(u,v)* with weight w means that team u scored w points to team v. If two teams have not played against each other yet, there is not any edge between them. For instance, the graph in next page shows that MAD team has already played against BCN team, and that the result was that BCN team scored 5 points and MAD scored 1 point. By the end of the season, each team will have played a match against all the rest of teams. The graph in the figure shows the status of the championship after only 6 matches. For a graph G(V,E) with $|V|$ teams and $|E|$ edges:

(a) What is the time complexity for finding the sum of the total points scored by a given team using adjacency lists? And using adjacency matrix? (3p)

(b) What is the time complexity for finding the sum of the total points scored against a given team using adjacency lists? And using an adjacency matrix? (3 p)

(c) What is the time complexity for finding the number of events won by a given team using adjacency lists? And using an adjacency matrix? (3p) (*Note:* An event between two teams u and v is won by u if the weight of (u, v) is greater than the weight of (v, u)).

(d) Which representation would you choose given the three operations above? (Adjacency lists or adjacency matrix). Motivate your answer. (2p)



4. Suppose you are starting a written exam which asks Q questions $q_1, ..., q_Q$. Each of these questions will give you some points in the exam, concretely, $p_1, ..., p_Q$ points (question q_1 is p_1 points, question q_2 is p_2 points, and so on). Assume that you know how much you have studied each chapter for the exam, so you can estimate the amount of time you will need to answer each question correctly. These times are $t_1, ..., t_Q$ (question q_1 will require t_1 minutes, question q_2 will require t_2 minutes, and so on). Unfortunately, the maximum time to finish the written exam is T minutes, which can be a value lower than the sum of the time needed to answer all the Q questions. Assume that all points $p_1, ..., p_Q$ and all times $t_1, ..., t_Q$ are integers strictly positive:

(a) Write an algorithm that executes in $O(Q \cdot T)$ time and that, given arrays $[p_1, ..., p_Q]$, $[t_1, ..., t_Q]$, and T, returns the maximum amount of points you can obtain if you answer correctly the most appropriate subset of questions. *Note 1:* You do not need to compute the concrete subset of questions that will give you the maximum score, just compute its value. *Note 2:* Dynamic Programming strategy can be applied to this problem. (8p)

(b) Given the same information ($[p_1, ..., p_Q]$, $[t_1, ..., t_Q]$, and T) and the decision problem, "*Is there a way in which I can obtain a score higher than S?*", to which class can you say that the decision problem belongs? (P, NP, NP-complete, NP-Hard). You must motivate your answer. (3p)