UNIVERSITY of OSLO

Faculty of Mathematics and Natural Sciences

Exam in:	**INF 4130/9135:** ***Algorithms: Design and efficiency***
Date of exam:	**15th December 2017**
Exam hours:	**09:00 – 13:00 (4 hours)**
Exam paper consists of:	**7 pages (including attachment)**
Attachment:	**1 page**
Permitted materials:	**All written and printed**

Make sure that your copy of this examination paper is complete before answering.
You can give your answers in English or in Norwegian, as you like.

Read the text carefully, and good luck!

Assignment 1 Undecidability (5%)

Let L = {M | Turing machine M solves the Halting problem}.

Question 1.a

Is the language (problem) L decidable (solvable by an algorithm)? Prove your answer.

Assignment 2 NP-completeness (12%)

We have proven in class that the three-dimensional matching (3DM) problem is NP-complete. We shall ask you to extend or generalize this problem to 4DM and provide some answers.

Question 2.a

Write a definition of the 4DM problem.

Question 2.b

Is 4DM NP-complete? Carefully prove your answer. Your challenge is to show us that you understand what exactly needs to be argued and in what way.

Assignment 3 Complexity (15%)

Evaluate the provided statements as TRUE or FALSE, and provide a brief explanation.

Question 3.a
If there is no Turing machine that can solve a certain given problem L, then there is no algorithm that can solve L on any other realistic computer model.

Question 3.b
By defining problems as formal languages we become able to develop a theory of complexity.

Question 3.c
If we don't know whether a problem can be solved, or don't know an algorithm that solves it, then we say that the problem is "undecidable".

Question 3.d
The linear average time Hamiltonicity algorithm that was discussed in class, which works in three stages, can give us an idea how to create algorithms for NP-complete or computationally hard problems that have good average-case or "practical" performance.

Question 3.e
Parallel computing is not a practical solution to NP-completeness.

Assignment 4 Flow in Networks (12%)

We have the following network where *s* is the source and *t* is the sink. The numbers on the edges are capacities:

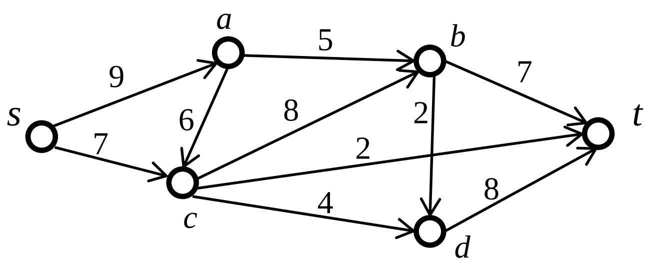


Figure 1 A network with capacities.

Question 4.a
We want to find the largest possible flow from *s* to *t* within the given capacities, by using the Ford-Fulkerson algorithm. We will this time use the variant that, in each main step, chooses an augmenting path that increases the flow as much as possible. We assume that the initial flow is zero, and you should for each main step indicate:

- which augmenting path you used.
- how much increase this gave in the total flow

Use one line for each main step, and describe a path as a sequence of nodes, including *s* and *t*. At the end, give the resulting total flow. You need not explain how you here found the best augmenting paths (but see question 4.b).

Question 4.b
As part of the algorithm in 4.a we, in each main step, have to find an augmenting path that gives as large increase in the flow as possible. You shall here describe a good algorithm for finding such a path when a flow (and the capacities) are given. You may refer to other "known" algorithms, and you then only need to indicate how the algorithm should be adjusted for the current purpose, and why. You need not prove that your algorithm is correct.

Assignment 5 Dynamic Programming (20%)

We shall look at finding the edit distance between two strings P and T by using dynamic programming, and a suitable two-dimensional table. However, we shall here assume that we have only two operations at our disposal, which is to *delete a character* and to *insert a character* (thus, substitution is not allowed, but it can obviously be done in two steps with deletion and insertion).

Question 5.a
We will, as for the standard case in the textbook, use a two-dimensional integer array D to remember already computed results. Decide whether the above limitation will change the way we should initialize D, compared to how it is done the standard case (with all three operations). If so: How should it be done now? Explain!

Question 5.b
Write down the recurrence relation you want to use for this version of edit distance. Explain the differences between your recurrence relation and the one for the standard case. We assume that *i* is the index for the P-string (from 1 to *n*) and that *j* is the index for the T-string (from 1 to *m*).

Question 5.c
Fill in a matrix D, including initialization, for finding the edit distance between P = 'abcd' and T = 'xbdc' according to the above restriction. What is the edit distance?

Question 5.d
We shall now assume that we only allow *substitution* and *insertion* of a character (not deletion). As the situation now has become "asymmetric" (we can insert, but not delete), we must specify what is the starting string and what is the target string, and we will talk about the edit distance *from* a string P *to* a string T.

For this case we need (at least logically) a new kind of value for the table entries. What situation should this value indicate? Choose a suitable numerical value to represent this situation. How would you now initialize the table?

Question 5.e
Write down the recurrence formula for the situation as described in Question 5.d. and find the edit distance from 'abcd' to 'babda' by filling in a new version of matrix D.

Assignment 6 Computational Geometry (12%)

In Figure 2 below we are given ten points in the plane, and the corresponding Voronoi diagram.

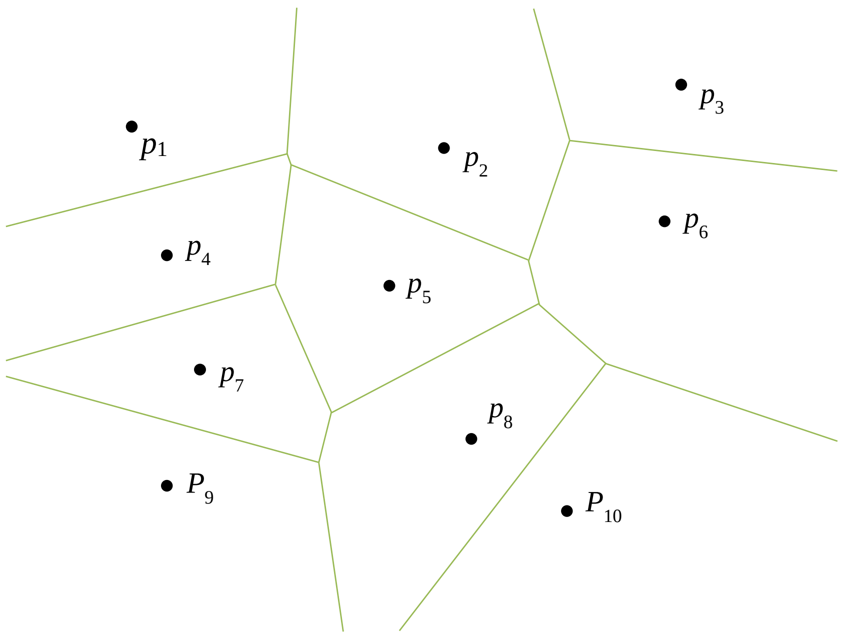


Figure 2 Ten points in the plane, and the corresponding Voronoi diagram.

Question 6.a

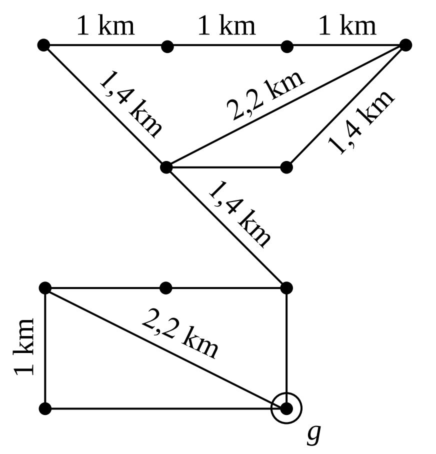
Find the Delaunay triangulation of the ten points. Use the copy of Figure 2 in the appendix to give your answer. Tear out the page and hand it in with the rest of your exam. Justify your answer.

Question 6.b

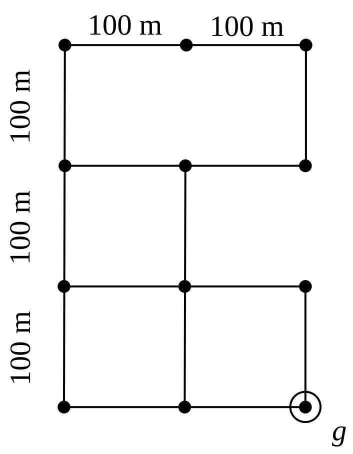
Find the convex hull of the ten points in Figure 2.

Assignment 7 A* Search (12%)

We are given the following two maps, one of a rural area, and one of a city. Distances are indicated for both maps (graphs). The goal node is indicated with a circle.



Map 1



Map 2

Figure 3 Two maps given as input to the A algorithm.*

Question 7.a

The following heuristic is proposed for the A* algorithm when we want to find the shortest path to the goal node g:

$h_1(n)$ = Euclidian distance from node n to the goal node g.

The Euclidian distance between two points $n = (n_x, n_y)$ and $g = (g_x, g_y)$ is the length of the straight line between the two points. It is given by the formula $\sqrt{(n_x - g_x)^2 + \left(n_y - g_y\right)^2}$.

1. Evaluate how suited the proposed heuristic is for use in the A* algorithm on Map 1.

2. Evaluate how suited the proposed heuristic is for use in the A* algorithm on Map 2.

Question 7.b

The following heuristic is proposed for the A* algorithm when we want to find the shortest path to the goal node g:

$h_2(n)$ = Manhattan distance from node n to the goal node g.

The Manhattan distance between two points $n = (n_x, n_y)$ and $g = (g_x, g_y)$ is the sum of the horizontal distance, along the x-axis, and the vertical distance, along the y-axis. It is given by the formula $|n_x - g_x| + \left|n_y - g_y\right|$.

1. Evaluate how suited the proposed heuristic is for use in the A* algorithm on Map 1.

2. Evaluate how suited the proposed heuristic is for use in the A* algorithm on Map 2.

Assignment 8 Priority Queues (12%)

Trees of various kinds are common data structures used in priority queues. We have looked at binary trees and binomial trees.

Question 8.a

Fibonacci heaps were implemented with a fast `decreaseKey()` operation in mind. Assume that we are in a situation where our priority queue looks like Figure 4 below.

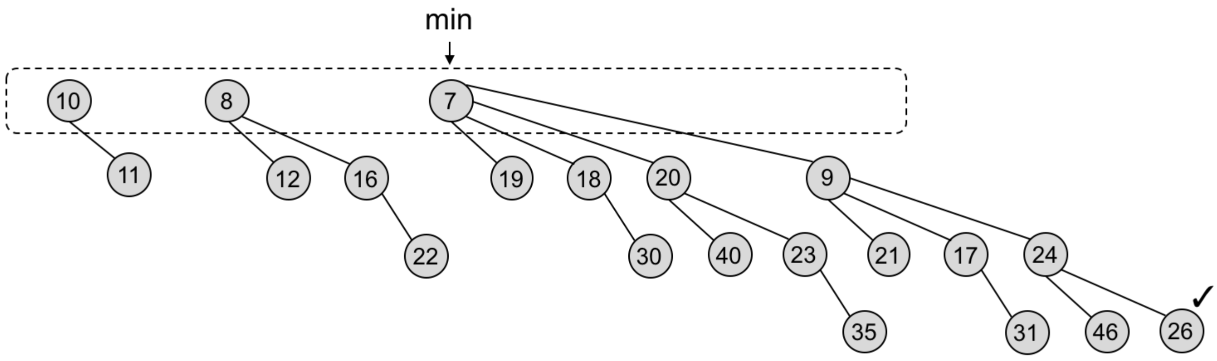


Figure 4 A Fibonacci heap at a certain point in the execution of an algorithm. One element is marked because of previous actions.

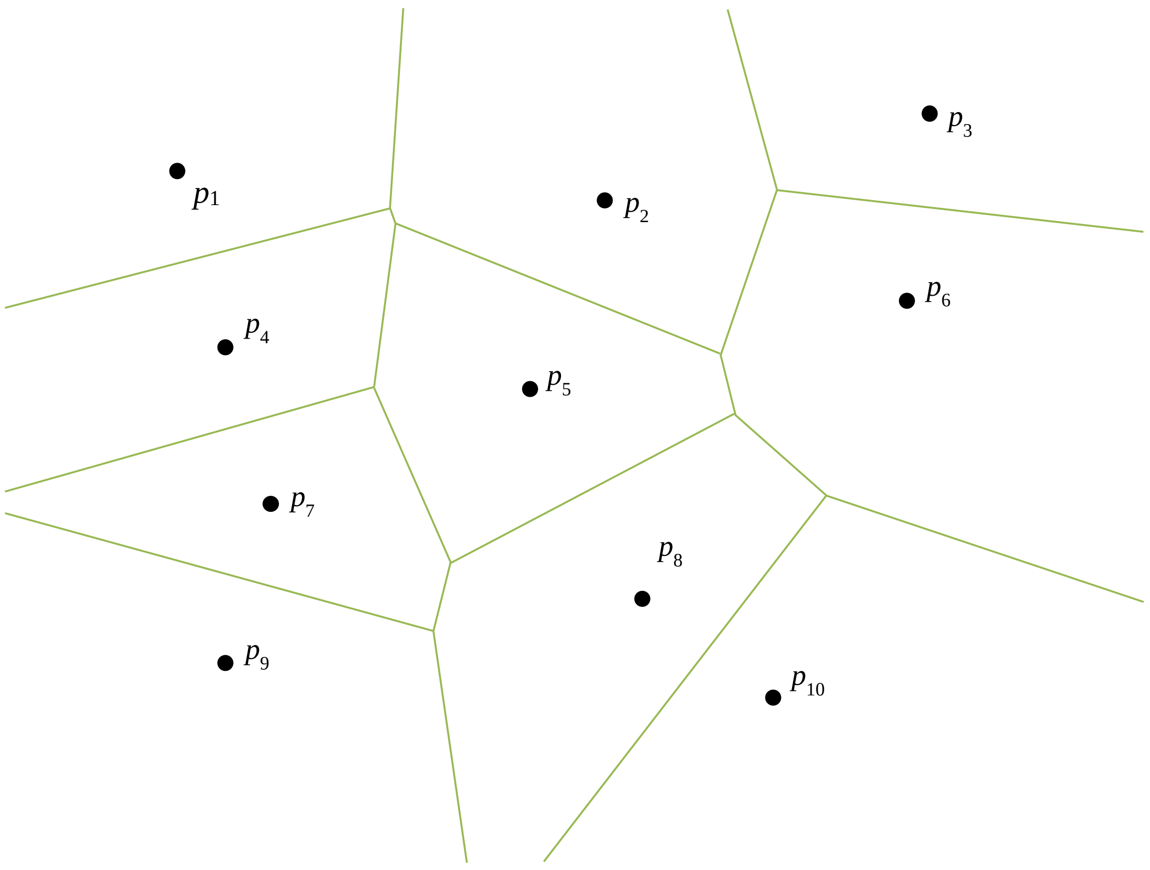
Draw the priority queue after the execution of the following three operations (you only need to draw the final result after executing all three operations):

```
/*
Element with current key value 17 gets key decreased to 6
Element with current key value 23 gets key decreased to 14
Element with current key value 21 gets key decreased to 5
*/
decreseKey(<element with key 17>, 6)
decreseKey(<element with key 23>, 14)
decreseKey(<element with key 21>, 5)
```

Question 8.b

Fibonacci heaps are implemented with binomial trees. Could we have implemented a similar data structure with binary trees? What, if any, would the major problems be? Your answer should contain a short discussion for each of the major operations: `insert()`, `decreaseKey()`, and `deleteMin()`.

[End of exam]


p_1
p_2
p_3
p_4
p_5
p_6
p_7
p_8
p_9
p_{10}