NTNU

Department of Computer and Information Science

# Examination paper for TDT4120 Algorithms and Data Structures

| | |
|---|---|
| **Academic contact during examination** | Magnus Lie Hetland |
| **Phone** | 918 51 949 |
| **Examination date** | December 17, 2016 |
| **Examination time (from–to)** | 09:00–13:00 |
| **Support material code** | D |
| **Other information** | The problem sheets are handed in, with answers in answer boxes under the problems |
| **Language** | English |
| **Number of pages (front page excluded)** | 8 |
| **Number of pages enclosed** | 0 |

**Informasjon om trykking av eksamensoppgave**

**Originalen er**

**1-sidig** ☑ **2-sidig** ☐

**sort/hvit** ☑ **i farger** ☐

**Skal ha flervalgskjema** ☐

**Quality assured by** Ole Edsberg

**Checked by**

Date Signature

Students will find the examination results in Studentweb. Please contact the department if you have questions about your results. The Examinations Office will not be able to answer these.

### Please read this thoroughly

(*i*) Read the entire exam thoroughly *before* you start!

(*ii*) The teacher will normally take one round through the exam hall. Have your questions ready!

(*iii*) Write your answers in the answer boxes and hand in the problem sheet. Feel free to use a pencil! Or draft your answers on a separate piece of paper, to avoid strikouts, and to get a copy for yourself

(*iv*) You are permitted to hand in extra sheets if need be, but the intention is that the answers should fit in the boxes on the problem sheets. Long answers do not count positively.

(*v*) The exam has 20 problem, worth 120 points in total. Of these, 20 are bonus points, so a score above 100 counts as 100. The point value is given next to each problem.

## Problems

(6 p) 1. You are to sort $n$ integers in the range 0 to $k$, where $k = O(n)$. Which sorting algorithm in the curriculum would it be most natural to use?

(6 p) 2. In the textbook, depth-first-search (DFS) is implemented using recursion. It is also possible to implement depth-first-search *without* recursion. How?

(6 p) 3. Which technique—incremental design, divide-and-onquer, dynamic programming, greedy algorithms or amortized analysis—is used in the common solution to the 0-1 knapsack problem?

(6 p) 4. Let the array $S[1..10]$ be $\langle 50, 70, 45, 15, 72, 41, 61, 22, 26, 64\rangle$ and let $S.top = 5$. Execute the following statements, in order:

```
1  x = POP(S)
2  y = POP(S)
3  PUSH(S, x)
4  PUSH(S, y)
```

What is the contents of the array $S$ now?

**Note:** The question is about the contents of the *entire* array, not just the stack.

(6 p) 5. After having executed an algorithm to find the shortest path from one vertex $s$ to all others in a graph (the single-source shortest path problem), we have $v.\pi = \text{NIL}$ for a given vertex $v$. What, then, is the value of $v.d$?

(6 p) 6. If you have a (possibly unbalanced) binary search tree with $n$ nodes and a height of $h$, how long does it take to find the smallest element (TREE-MINIMUM)? Express the answer in O-notation.

(6 p) 7. What is the amortized running time for insertion into a dynamic table (TABLE-INSERT)? Express the answer in O-notation.

(6 p) 8. If you use BUCKET-SORT on $n$ inependent, uniformly distributed numbers in the range $[0, 1)$, you get a running time of $O(n)$. During execution, as a part of BUCKET-SORT, several calls are made to INSERTION-SORT. What is the expected running time for *each individual call* to INSERTION-SORT? (We are asking for an expected value here.) Use O-notation and express the answer as a function of $n$.

**Note:** You are *not* to give your answer as a function of the input size for INSERTION-SORT, but as a function of the input size for BUCKET-SORT.

(6 p) 9. $A[1..n]$ is an array of integers. You are almost done writing an algorithm based on the design technique *divide-and-conquer* to find the smallest element (or one of the smallest elements) in $A$. What you have written so far is the following:

```
MINIMUM(A, p, r)
1  if p == r
2       return A[p]
3  else m = ⌊p + r/2⌋
4       x =
5       y =
6       if x < y
7            return x
8       else return y
```

To find the minimum of $A$, you use MINIMUM$(A, 1, n)$. Fill out what is missing in the two boxes in the pseudocode above.

(6 p) 10. Solve the following recurrence, where $n \geq 0$ is an integer:

$$T(n) = \begin{cases} 0 & \text{if } n = 1\,, \\ 10\,000\,T(n/10) + n^4 + n^2 & \text{if } n > 1\,. \end{cases}$$

Give your answer in $\Theta$-notation.

(6 p) 11. Your friend Lurvik thinks the following recurrence has the solution $T(n) = n^3 - n^2$:

$$T(n) = \begin{cases} 0 & \text{if } n = 1\,, \\ 8T(n/2) + n^2 & \text{if } n > 1\,, \end{cases}$$

where $n = 2^k$, for a positive integer $k$. Show that she is right.

(6 p) 12. You have an undirected graph $G = (V, E)$, where each edge $(u, v) \in E$ has a weight $w(u, v) = -1$. You are given two vertices $s$ and $t$, and wish to find the longest (i.e., heaviest) path from $s$ to $t$, that is, the one with the greatest sum of weights. How would you solve this problem?

(6 p) 13. In Transitive-Closure, the binary variable $t_{ij}^{(k)}$ is used to indicate whether there is a path from $i$ to $j$, if all vertices on the path between them *must* lie in the set $\{1, 2, \ldots, k\}$. For example, $t_{ij}^{(0)} = 1$ if and only if $(i, j) \in E$. What is the expression for $t_{ij}^{(k)}$, when $k > 0$?

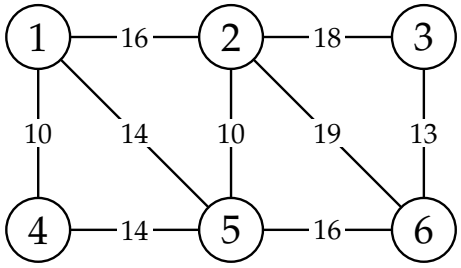


Figure 1: A weighted, undirected graph for use in problem 14

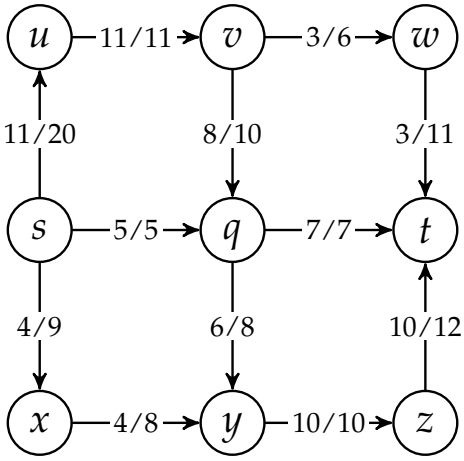


Figure 2: Flow network used in problem 15

(6 p) 14. If you execute MST-Kruskal on the graph in fig. 1, which edge would be the fifth to be chosen? That is, which edge would be the fifth to be added to the solution?

Give the edge in the form $(i, j)$, where $i < j$.

(6 p) 15. Figure 2 shows the flow network $G$, with source $s$, sink $t$ and flow $f$. Is the flow maximum? Please answer yes or no.

*If yes*, also list the vertices that are reachable (i.e., that there are paths to) from $s$ in $G_f$.

*If no*, also list the vertices of an augmenting path, in order.

$D^{(2)}$

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | 3 | $\infty$ | 7 |
| 2 | $\infty$ | 0 | $\infty$ | 4 |
| 3 | 1 | 2 | 0 | 6 |
| 4 | $\infty$ | $\infty$ | $-1$ | 0 |

$\Pi^{(2)}$

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | NIL | 1 | NIL | 2 |
| 2 | NIL | NIL | NIL | 2 |
| 3 | 3 | 3 | NIL | 2 |
| 4 | NIL | NIL | 4 | NIL |

Figure 3: Previous state during execution of Floyd-Warshall, used in problem 16

(6 p) 16. During a run of FLOYD-WARSHALL, $D^{(2)}$ and $\Pi^{(2)}$ are as shown in fig. 3. What will $D^{(3)}$ and $\Pi^{(3)}$ be?

Fill in your answer in the tables below.

$D^{(3)}$

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |

$\Pi^{(3)}$

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |

(6 p) 17. A flow network is a directed graph $G = (V, E)$ where each edge $(u, v) \in E$ has a *capacity* $c(u, v) \geq 0$. If $(u, v) \notin E$, we let $c(u, v) = 0$. A *flow* in $G$ is a real function $f : V \times V \to \mathbb{R}$ that satisfies two properties. Which properties are they?

The properties shouldbe expressed in mathematical notation. A brief textual description may give up to 3 points. It is *not* sufficient to simply give their names. (Nor is it *necessary* to give their names.)

(6 p) 18. Is the following statement correct?

> "For every language $L$, if $L$ is in NP, then $L$ is not in co-NP."

Answer yes or no, and explain very briefly.

(6 p) 19. An *independent set* in a graph $G = (V, E)$ is a subset $U \subseteq V$ of the vertices, such that each edge in $E$ is incident to at most one vertex in $U$ (i.e., no vertices in $U$ are neighbors).

A *free tree* is a connected, acyclic undirected graph (i.e., a tree without a specified root).

You are to solve the following problem.

> **Input:** A free tree $T = (V, E)$ where each vertex $v \in V$ has a weight $w(v) \neq 0$.
> **Output:** An independent set $U$ in $T$ with a maximum sum of weights $\sum_{v \in U} w(v)$.

See fig. 4 for an example. Note that the vertex weights *may be negative*.

To keep things simple, you do *not* need to describe how you would find the set $U$ itself; it suffices for you to find the correct sum of weights. Concisely describe an algorithm that solves the problem efficiently. Give the running time as precisely as possible, in asymptotic notation.

(6 p) 20. A *dominating set* in an undirected graph is a subset of the vertices, which collectively are neighbors to all the other vertices. That is, a dominating set for an undirected graph $G = (V, E)$ is a set $U \subseteq V$ such that for each vertex $v \in V - U$, there is at least one vertex $u \in U$ where $(u, v) \in E$. (See fig. 5 for an example.)

The so-called *dominating set problem* is to find a dominating set of minimal size (i.e., with as few vertices as possible) in a given graph. Expressed as a decision problem, it is to decide whether the graph has a dominating set of a given size $k$. As a language, we define

$$\text{DOMINATING-SET} = \{\langle G, k\rangle : \text{the graph } G \text{ has a dominating set with } k \text{ vertices}\}.$$

Assume that you already know that the language VERTEX-COVER is NP-complete. Use this knowledge to show that DOMINATING-SET is NP-complete.

**Note:** For full marks, all the components of NP-completeness proof must be included.

**Hint:** Assume that the vertex $w$ only has $u$ and $v$ as neighbors, and that $u$ and $v$ are adjacent, as in fig. 6. If $w$ is in our solution, we can always replace it with either $u$ or $v$, and still have a solution (a dominating set) with the same number of vertices.

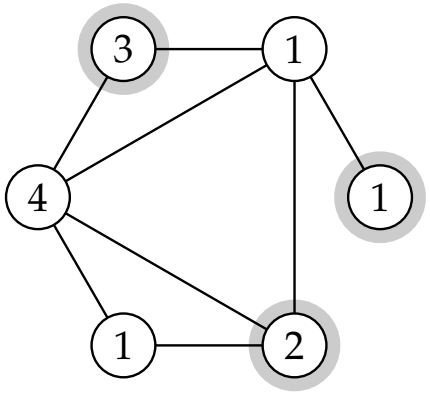


Figure 4: Example of maximum, weighted independent set (see problem 19). The three highlighted vertices are in the independent set $U$, which has a weight sum $\sum_{v \in U} w(u) = 1 + 2 + 3 = 6$

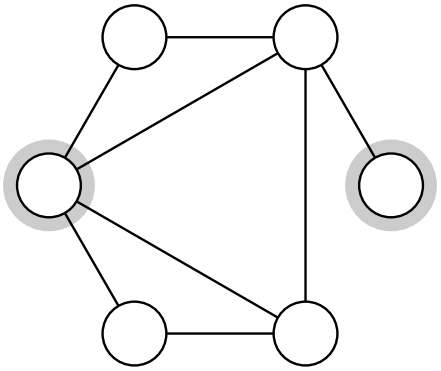

Figure 5: A dominating set (see problem 20) of size 2. The highlighted vertices are in the dominating set, and each of the other vertices has an edge to at least one of them

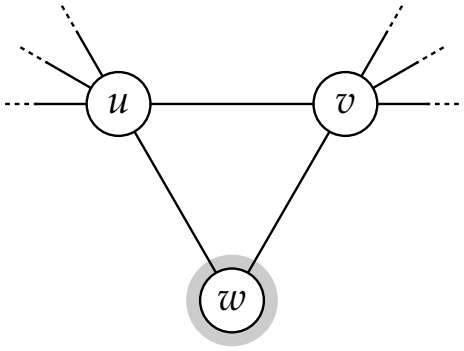


Figure 6: The vertices $u$ and $v$ may be adjacent to other vertices in the graph, but $w$ is only adjacent to $u$ and $v$. If $w$ is in a dominating set of size $k$, we may replace $w$ with either $u$ and $v$, and still have a dominating set of size $k$