KØBENHAVNS UNIVERSITET

Diskret Matematik og Formelle Sprog: Exam April 12, 2023

Submission: Please write your solutions with ample margins on all sides, and make sure your handwriting is legible. *Start your solution of every new problem on a new page. Please mark every page with your name, exam number or something else that uniquely identifies your exam,* so that it is easy to see for every page which exam submission it is part of.
Exposition: Please try to be precise and to the point in your solutions and refrain from vague statements. Never, ever just state an answer, but always make sure to *explain why* the answer is what it is. *Write your solutions in such a way that a fellow student of yours could read, understand, and verify your solutions.*
Collaboration: All problems should be solved individually. No communication or collaboration is allowed, and solutions will be checked for plagiarism.
Reference material: All course material is allowed, including textbooks, lecture notes, exercise sheets, and individual notes. You are *not* supposed, and will not need, to copy substantial parts of text verbatim, and if your solutions make heavy use of reference material, then make sure to provide clear references to what you are using. Please note that you deviate from definitions and algorithms as described in the course material at your own risk!
Grading: A score of 200 points is guaranteed to be enough to pass the exam.
About the problems: Note that the problem are of quite different types, and are *not sorted in increasing order of difficulty. Please read through the whole exam first,* before you start working on any single problem, so that you can plan which order of working on the problems makes most sense to you. Partial answers to problems can also give points, and you can get a top grade even without solving all problems. Please do not hesitate to alert the exam administrators if any problem statement is unclear. Good luck!

1 (80 p) Consider the snippet of code

```
check (A, lo, hi)
    mid := floor ((lo + hi) / 2)
    success := TRUE
    i := lo
    while (i < mid and success)
        if (A[i] > A[mid])
            success := FALSE
        i := i + 1
    i := mid + 1
    while (i <= hi and success)
        if (A[i] < A[mid])
            success := FALSE
        i := i + 1
    if (lo < mid - 1 and success)
        success := check (A, lo, mid - 1)
    if (mid + 1 < hi and success)
        success := check (A, mid + 1, hi)
    return success
```

where `A` is an array indexed from 1 to `A.size` that contains elements that can be compared, and the function `floor` rounds down to the nearest integer.

1a Explain in plain language what the result is of an algorithm call `check (A, 1, A.size)` (i.e., do not just rewrite the pseudocode word by word, so that your text is just a more verbose version of the pseudocode, but interpret what the code is doing and why). In particular, what holds when the algorithm return `TRUE`?

1b Provide an asymptotic analysis of the running time as a function of the size of the array `A`.

1c Can you improve the algorithm (i.e., change the pseudocode) to run asymptotically faster while retaining the same functionality? In case you can design an asymptotically faster algorithm, analyse the time complexity of your new algorithm. For the best algorithm that you have—either your new one, or the old one—can you prove that the running time of this algorithm is asymptotically optimal?

2 (60 p) Post-pandemic life in academia has meant noticeable changes for both Jakob and his family, in ways both good and bad.

2a Last autumn Jakob attended his first non-virtual international conference in a very long time (since February 2020, to be precise), and the conference had more than 100 participants. A colleague at the conference pointed out to Jakob that this meant that either there were participants from more than 10 different countries, or else more than 10 people from some particular country attended the conference. Jakob, who had not studied the list of participants that closely, was amazed at this claim. Can you explain to Jakob, without even having looked at the list of participants, why the claim has to be true?

2b While Jakob was away, he suggested to his children that they should entertain themselves at home with the following game: Write down the numbers 1 to 20 on a sheet of paper. Erase any two distinct numbers a and b and replace them by the number $a + b - 1$. Now do the same again with all numbers currently on the sheet, i.e., pick any two members a' and b' of the multi-set $\big(\{1, 2, \ldots, 20\} \setminus \{a, b\}\big) \cup \{a+b-1\}$ of all numbers between 1 and 20 except a and b plus the new number $a+b-1$, and replace the two chosen numbers by their sum minus one $a' + b' - 1$. Repeat this procedure until there is only a single number left on the sheet of paper. The children found this game a little bit repetitive, however. Can you describe the range of possible outcomes for this game? For a full score, provide proofs of any claims you make.

3 (40 p) Let $a_1, a_2, a_3, \ldots$ be a sequence of numbers defined by

$$a_n = \begin{cases} 5 & \text{if } n = 1, \\ 13 & \text{if } n = 2, \\ 5a_{n-1} - 6a_{n-2} & \text{if } n > 2. \end{cases}$$

Prove that for all positive integers n it holds that $a_n = 2^n + 3^n$.

4 (100 p) Recall that a graph $G = (V, E)$ consists of a set of vertices V connected by edges E, where every edge is a pair of vertices. If there is an edge (u, v) between two vertices u and v, then the two vertices are said to be *neighbours* and are both *incident* to the edge. We say that a sequence of edges (v_1, v_2), (v_2, v_3), (v_3, v_4), $\ldots$, (v_{k-1}, v_k), in E is a *path* from v_1 to v_k.

In this problem, we wish to express properties of graphs in both natural language and predicate logic, and to translate between the two forms. We do this as follows:

- The universe is the set of vertices V of G.
- The binary predicate $E(u, v)$ holds if and only if there is an edge from u to v in G.
- The unary predicate $S(v)$ is used to identify a subset of vertices $S = \{v \mid v \in V, S(v) \text{ is true}\}$ for which some property might or might not hold.

Below you find five graph properties written as predicate logic formulas and six graph properties defined in natural language. Most of the predicate logic formulas have corresponding natural language definitions, but not all.

Your task is to determine which of the predicate logic formulas (a), $\ldots$, (e) match which—if any—of the natural language definitions (1), $\ldots$, (6). Please make sure to motivate your answers clearly.

Predicate Logic Formulas:

(a) $\forall u \forall v \big(E(u, v) \rightarrow E(v, u)\big)$

(b) $\forall u \forall v \big(E(u, v) \rightarrow S(u) \vee S(v)\big)$

(c) $\forall u \forall v \exists w \big(S(w) \wedge (E(u, w) \vee E(v, w))\big)$

(d) $\forall u \forall v \big(E(u, v) \rightarrow ((S(u) \wedge \neg S(v)) \vee (\neg S(u) \wedge S(v)))\big)$

(e) $\forall u \forall v \big((u \neq v \wedge S(u) \wedge S(v)) \rightarrow E(u, v)\big)$

Natural Language Definitions:

(1) S is a *dominating set* in G, i.e., every vertex v in the graph either is in S or is a neighbour of a vertex in S.

(2) S is a *clique* in G, i.e., a set of vertices that are all neighbours with each other.

(3) The graph G is *undirected*.

(4) The graph G is *connected*.

(5) S is a *vertex cover* in G, i.e., for every edge at least one of the vertices incident to it is in S.

(6) The graph G is *bipartite* with bipartition $(S, V \setminus S)$, i.e., all edges go between S and $V \setminus S$.

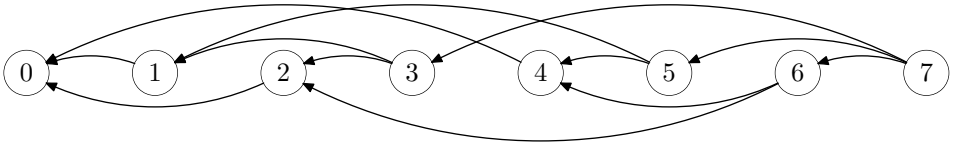

Figure 1: Directed graph D_S representing relation S in Problem 6.

5 (60 p) When Jakob is socializing in the evenings after conferences in the United States, he tries to convince his colleagues that poker should really be played with a *republican deck of cards*, i.e., with cards of all ranks 2–10 plus the aces, but without kings, queens, or jacks.

5a Suppose that you are dealt 5 cards from a perfectly shuffled republican deck of cards. What is the probability of getting a *full house*, i.e., three of a kind (three cards of the same rank) with a pair (two other cards of the same rank)? Explain clearly how you obtain the expression in your answer.

5b It is a sad fact that so far Jakob has had quite limited success in convincing colleagues that a republican deck of cards is the true American way of playing poker. Jakob believes this is because the colleagues have all been brainwashed by the big casinos in Las Vegas, and that this in turn is because with a republican deck of cards the probability for the players of getting a strong hand (like a full house) would be higher than with a standard deck of cards, making it more likely that casinos might lose money.

Ignoring the wider ramifications of the worldwide conspiracy that Jakob alleges to have discovered, is the factual claim true that the probability of getting, e.g., a full house is higher with a republican deck of cards than with a standard deck of cards when being dealt 5 cards from a shuffled deck? (Note that you should not have to do very precise calculations to be able to determine whether this claim is true or not, and a clear intuitive explanation of which answer should be the correct one can give generous partial credits.)

6 (60 p) Consider the relation S described by the directed graph D_S in Figure 1.

6a Write down the matrix representation M_S of the relation S and describe briefly but clearly how you constructed this matrix.

6b Let us write I to denote the inverse of the relation S. What is the matrix representation of I? Write it down and explain how you constructed it.

6c Now let T be the transitive closure of the relation I. What is the matrix representation of T? Write it down and explain how you constructed it.

6d Finally, let R be the reflexive closure of the relation T. Can you explain in words what the relation R is by describing how it can be interpreted? (In particular, is it similar to anything we have discussed during the course? Be as specific in your reply as you can.)

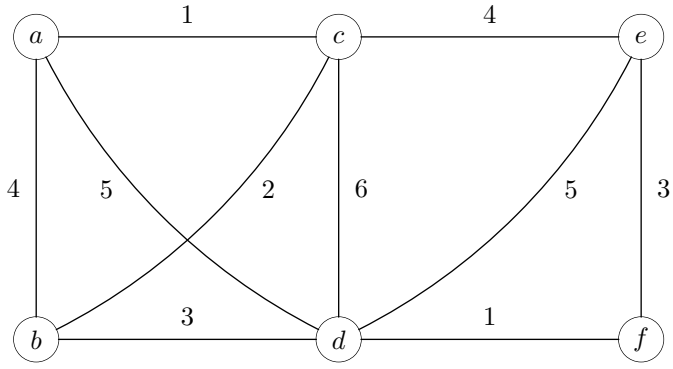

Figure 2: Graph G on which to run Prim's algorithm in Problem 7a.

7 (80 p) In this problem we wish to understand algorithms for minimum spanning trees.

7a Consider the graph G in Figure 2, which is given to us in adjacency list representation, with the neighbour lists sorted so that vertices are encountered in lexicographic order. (For instance, the neighbour list of c is (a, b, d, e) sorted in that order.)

Run Prim's algorithm on the graph G, starting with the vertex a. Show the minimum spanning tree T produced at the end of the algorithm. Use a heap for the priority queue implementation. Assume that in the array representing the heap, the vertices are initially listed in lexicographic order. During the execution of the algorithm, describe for every vertex which neighbours are being considered and how they are dealt with. Show for the first two dequeued vertices how the heap changes after the dequeueing operations and all ensuing key value updates in the priority queue. For the rest of the vertices, you should describe how the algorithm considers all neighbours of the dequeued vertices and how key values are updated, but you do not need to draw any heaps.

7b Jakob has designed a fairly nifty MST algorithm that works as follows for $G = (V, E)$:

1. Initialize $T = \emptyset$ and $S = \{v\}$, where v is a randomly chosen vertex of the graph.
2. Iterate $|V| - 1$ times:
   (a) Let w be the vertex most recently added to S.
   (b) Among all edges from w to vertices x not yet added to S, pick the edge (w, x) with lowest cost and add to T, and add x to S with predecessor w.
   (c) If w does not have any neighbours not already in S, go back to the predecessor of w, and then to the predecessor of the predecessor, et cetera, until you find a vertex z that has at least one neighbor not already in S. Use that vertex z instead of w in this iteration.

Jakob claims that T as computed by this algorithm is a minimum spanning tree, and that the algorithm is also faster than Prim's algorithm since it runs in time $O(|V| + |E|)$. Determine whether Jakob's algorithm is valid by giving a proof of correctness or a counter-example. Regardless of whether the algorithm is correct or not, did Jakob get the time complexity analysis right? Motivate your answer clearly.

8 (90 p) Consider the following context-free grammars, where a, b, c are terminals, S, T, U are non-terminals, and S is the starting symbol.

Grammar 1:

$$S \to TU \tag{1a}$$
$$T \to aTb \tag{1b}$$
$$T \to \tag{1c}$$
$$U \to bUc \tag{1d}$$
$$U \to \tag{1e}$$

Grammar 2:

$$S \to TU \tag{2a}$$
$$T \to aT \tag{2b}$$
$$T \to bT \tag{2c}$$
$$T \to \tag{2d}$$
$$U \to bU \tag{2e}$$
$$U \to cU \tag{2f}$$
$$U \to \tag{2g}$$

Grammar 3:

$$S \to TU \tag{3a}$$
$$T \to abT \tag{3b}$$
$$T \to \tag{3c}$$
$$U \to bcU \tag{3d}$$
$$U \to \tag{3e}$$

Which of these grammars generate regular languages? For each grammar, write a regular expression that generates the same language (and explain why), or argue why the language generated by the grammar is not regular. In your regular expressions, please use *only* the concatenation, alternative (|) and star (*) operators, and not the syntactic sugar extra operators that we just mentioned in class but never utilized.