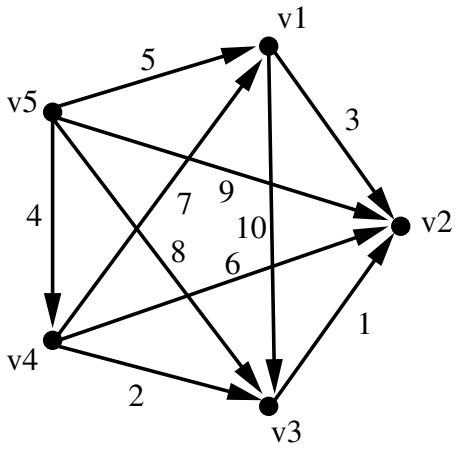
universität innsbruck

Discrete Structures/Mathematics | WS 2019/2020 | LVA 703070/703015

EXAM 1 | January 29th, 2020

This exam consists of three regular exercises (1–3) each worth 20 points. The time available is 1 hour and 45 minutes (105 minutes). The available points for each item are written in the margin. There are 60 points in total for the regular exercises. In addition, there are bonus exercises (4∗, 5∗) each worth 15 points. You need at least 30 points to pass.

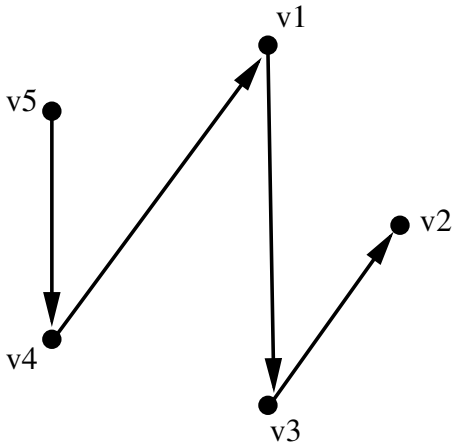
1 Let the weighted directed graph G with set of vertices $V = \{v1, v2, v3, v4, v5\}$ be:



Let R be the relation on vertices of graph G.

[6] (a) Give the Hasse diagram of R and show that R is a strict order.

The Hasse diagram H for a strict order is obtained by omitting transitive edges that can be reconstructed, i.e. we need a minimal subrelation/graph whose transitive closure is R. E.g. $v5 \to v1$ is omitted because of $v5 \to v4 \to v1$. Continuing like this yields:



A strict order is irreflexive and transitive. Irreflexivity was observed above. Alternatively, irreflexivity follows from the absence of infinite paths. Transitivity holds since ordering the nodes as $v5, v4, v1, v3, v2$ R contains *all* edges from a node to a node later in the order.

(b) Compute a shortest path from $v5$ to $v2$ in G using an algorithm of your preference. Indicate the algorithm used and give at least 2 intermediate stages of the algorithm. [7]

From R in the previous item being a strict order, it follows that G is in fact a dag. That is, the shortest path algorithm for dags can be applied. Indicating the (labelled) edges with their current predecessor and weight, and underlining definitive weights of nodes, the stages are:

- $\rightarrow v5, \underline{0}$
- $v5 \rightarrow v1, 5$, $v5 \rightarrow v2, 9$, $v5 \rightarrow v3, 8$, $v5 \rightarrow v4, \underline{4}$
- $v5 \rightarrow v1, \underline{5}$, $v5 \rightarrow v2, 9$, $v4 \rightarrow v3, 6$
- $v1 \rightarrow v2, 8$, $v4 \rightarrow v3, \underline{6}$
- $v3 \rightarrow v2, 7$

From which we read off that the shortest path is $v5 \rightarrow v4 \rightarrow v3 \rightarrow v2$ with weight 7.

Alternatively, Floyd's algorithm for directed graphs may be applied. Ordering nodes as $v5, v4, v1, v3, v2$, G is represented by the matrix:

$$\begin{pmatrix} 0 & 4 & 5 & 8 & 9 \\ \infty & 0 & 7 & 2 & 6 \\ \infty & \infty & 0 & 10 & 3 \\ \infty & \infty & \infty & 0 & 1 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

This ordering of nodes makes the matrix an 'upper triangular' matrix in the sense that all entries below the diagonal are ∞, which is convenient as this makes that executing Floyd's algorithm only ever changes the '1st quadrant', in particular leaving all ∞s unchanged. As a consequence the first and last iterations do not change the matrix. In the 2nd iteration, $v5 \rightarrow v3$ is updated from 8 to 6 via $v4$:

$$\begin{pmatrix} 0 & 4 & 5 & 6 & 9 \\ \infty & 0 & 7 & 2 & 6 \\ \infty & \infty & 0 & 10 & 3 \\ \infty & \infty & \infty & 0 & 1 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

In the 3rd iteration, $v5$ to $v2$ is updated from 9 to 8 via $v1$:

$$\begin{pmatrix} 0 & 4 & 5 & 6 & 8 \\ \infty & 0 & 7 & 2 & 6 \\ \infty & \infty & 0 & 10 & 3 \\ \infty & \infty & \infty & 0 & 1 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

In the 4th iteration, $v5$ to $v2$ is updated from 8 to 7, and $v4$ to $v2$ from 6 to 3, both via $v3$

$$\begin{pmatrix} 0 & 4 & 5 & 6 & 7 \\ \infty & 0 & 7 & 2 & 3 \\ \infty & \infty & 0 & 10 & 3 \\ \infty & \infty & \infty & 0 & 1 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

Thus we find the same shortest path of weight 7 as above.

(c) Write a recursive specification for the function f mapping a node v in G to the sum of the weights of all paths from v to $v2$. For instance, evaluating $f(v1)$ should result in 14. Your specification should not use any concrete weights in G, but be stated in terms of the function w assigning weights to edges. Using your recursive specification, stepwise evaluate $f(v4)$. [7]

Observe that if there is an edge from v to v' with weight w, then $f(v) = f(v')+w$ times the number of paths from v' to $v2$. Specifying the latter by means of a function n, we accordingly we define $f(v) = \sum_{vRv'} w(v,v') \cdot n(v') + f(v')$ where $n(v) = \sum_{vRw} n(w)$ is the number of paths from v to $v2$. We first compute n and then f, both starting from $v2$ (i.e. from right to left/bottom–up).

	$v5$	$v4$	$v1$	$v3$	$v2$
n	$8 = 4+2+1+1$	$4 = 2+1+1$	2	1	1
f	$95 = (4 \cdot 4 + 37) + (5 \cdot 2 + 14) + (8+1) + 9$	$37 = (7 \cdot 2 + 14) + (2+1) + 6$	14	2	0

That is, the total weight of the paths from $v4$ to $v2$ is 37.

Alternatively, instead of working backward from $v2$, one can work forward from a given note using an accumulator s initially set to 0, defining $g(v,s) = \sum_{vRv'} g(v, s+w(v,v'))$ if $v \neq v2$, and s otherwise, and defining $f(v) = g(v,0)$. We then compute $f(v4) = g(v4,0) = g(v1,7) + g(v3,2) + 6 = g(v3,17) + 10 + 3 + 6 = 18 + 19 = 37$.

2 (a) [5] Compute $7^{100} \bmod 11$, and compute an inverse of 12 modulo 17 and verify that it indeed is inverse. Justify the steps taken to compute the results.

Since 11 is a prime number, we have by Fermat's little theorem that $a^{10} \equiv 1 \pmod{11}$ so that $7^{100} \equiv 7^{100 \bmod 10} \equiv 7^0 \equiv 1 \bmod 11$.

Alternatively, one can evaluate by means of fast exponentiation: $7^{100} \equiv (7^{50})^2 \equiv ((7^{24})^2 \cdot 7)^2 \equiv \ldots \equiv (((((7^2 \cdot 7)^2)^2)^2 \cdot 7)^2)^2 \equiv ((((2^2)^2)^2 \cdot 7)^2)^2 \equiv ((5^2 \cdot 7)^2)^2 \equiv (10^2)^2 \equiv 1 \bmod 11$.

Computing by 100 times multiplying by 7, modulo 11, is in principle possible but leads to an unnecessarily long and time-consuming computation.

Since $\gcd(12,17) = 1$, there are u and v such that $1 = u \cdot 17 + v \cdot 12$ by Bézout's lemma, from which it then follows that v is the inverse of 12 modulo 17, as $1 \equiv u \cdot 17 + v \cdot 12 \equiv v \cdot 12 \pmod{17}$ To compute u and v we use Euclid's extended gcd algorithm:

$$\begin{aligned} 17 &= 1 \cdot 17 + 0 \cdot 12 \\ 12 &= 0 \cdot 17 + 1 \cdot 12 \\ 5 &= 1 \cdot 17 - 1 \cdot 12 \\ 2 &= -2 \cdot 17 + 3 \cdot 12 \\ 1 &= 5 \cdot 17 - 7 \cdot 12 \end{aligned}$$

That is $v = -7$, which is the same as 10 modulo 17. Verification: $10 \cdot 12 = 120 = 7 \cdot 17 + 1$, so $10 \cdot 12 \equiv 1 \pmod{17}$.

Alternatively, since the inverse must exists by $\gcd(12,17) = 1$, one can find it by trying out all products $v \cdot 12$ for $0 \leq v < 17$, yielding the same result.

(b) [5] Let R be the relation on pairs of natural numbers defined by: $(n,m)\ R\ (n',m')$ if $n + m' = n' + m$. For instance, $(5,3)\ R\ (2,0)$ since $5 + 0 = 5 = 2 + 3$, but not $(5,3)\ R\ (0,2)$ since $5 + 2 = 7 \neq 3 = 0 + 3$. Show that R is an equivalence relation, and give 4 elements of the R-equivalence class $[(0,2)]$ of $(0,2)$.

For R to be an equivalence relation, it must be reflexive, symmetric and transitive. We verify them in turn:

- $(n,m)\ R\ (n,m)$ since $n + m = n + m$.
- if $(n,m)\ R\ (n',m')$, then $n + m' = n' + m$. Therefore $n' + m = n + m'$, hence $(n',m')\ R\ (n,m)$.
- if $(n,m)\ R\ (n',m')\ R\ (n'',m'')$, then $n+m' = n'+m$ and $n'+m'' = n''+m'$. Adding these yields $n+m'+n'+m'' = n'+m+n''+m'$, from which we obtain $n+m'' = n''+m$ by cancelling n', m' and reordering summands, hence $(n,m)\ R\ (n'',m'')$.

Alternatively, one can observe that $(n,m)\ R\ (n',m')$ iff $m - n = m' - n'$, for $-$ subtraction on integers, and then use that any relation R specified in such a way,

namely by $x\ R\ y$ if $f(x) = f(y)$ for some function f (here $f((n,m)) = m - n$) is an equivalence relation (because equality $=$, is an equivalence relation).

We have $[(0,2)] = [(0,2),(1,3),(2,4),(3,5),(4,6),\ldots]$, or in general, all pairs of natural numbers (n,m) such that $m - n = 2$ are in the equivalence class.

(c) Suppose the complexity T of some algorithm A, as a function of the size n of its input, is an increasing function that satisfies $T(n) = 4 \cdot T\left(\frac{n}{2}\right) + 2 \cdot n^2$ if $n = 2^k$ for some
[5] positive natural number k, and 2 if $n = 1$. Determine a closed-form expression e such that $T(n) \in \Theta(e)$, and explain what the latter notation means.

We see that we have an instance of the Master theorem with $a = 4$, $b = 2$, $c = 2$, and $s = 2$. Since $a = 4 = b^s$, we are in the second case, hence $T(n) \in \Theta(n^2 \log n)$. That is, the complexity function T of algorithm A is asymptotically bounded both from below and above by a constant (not necessarily the same) times $n^2 \log n$.

[5] (d) Show that the set $\{M\#x \mid \text{TM } M \text{ loops on input } x\}$ is not recursive, where you may assume that (the code of) M and x are bit-strings.

Let $LP = \{M\#x \mid \text{TM } M \text{ loops on input } x\}$. The idea is that LP is $\sim HP$ except that LP contains only those strings $y \in \sim HP$ that have shape $M\#x$, i.e. the bit-string of the code of a TM followed by a hash-symbol followed by another bit-string.

Accordingly we show $\sim HP \leq LP$ by a function f that first disposes of such *garbage* strings, strings not of the right shape. More precisely, we let f map a string y to itself if y has shape $M\#x$ for some TM M and bit-string x, and otherwise to $M'\#x'$ for a fixed TM M' and bit-string x' such that M' loops on x'. The function f is easily seen to be computable (by first testing y to be two bits-strings separated by a hash, and then trying to decompile the left bit-string into a TM).

- If $y \in \sim HP$, then either y has shape $M\#x$ for some TM M and bit-string x but M does not halt on x so M loops on x and $f(y) = y = M\#x \in LP$, or y is garbage so $f(y) = M'\#x' \in LP$ per construction of M' and x'.
- If $y \notin \sim HP$, then $y \in HP$ so has shape $M\#x$ for some TM M and bit-string x such that M halts on x, hence $f(y) = y \in HP$, so $f(y) \notin LP$.

Alternatively, we can diagonalise directly: Suppose $LP = L(K)$ for some total TM K.

- Define the behaviour of a TM M on input x to be ∞ (looping) if M loops on x, and $\downarrow$ (halting) otherwise. and let *cd* be the complement of the behaviours of TM M_x on x, for the usual enumeration $\epsilon, 0, 1, 00, \ldots$ of the bit-strings. Then *cd* is distinct from the behaviour of any TM (it is distinct from TM M_x at input x; from which we conclude since each TM occurs in the enumeration $M_\epsilon, M_0, M_1, M_{00}, \ldots$
- Let CD be the TM that first transforms an input x into $M_x\#x$ and feeds that to K. If K accepts, i.e. if M_x loops on x, then CD halts, and if K rejects, then CD loops.

Then CD exhibits behaviour *cd*: if CD loops/halts on x, then M_x halts/loops on x, so the complement of that behaviour is loop/halting. Therefore, CD cannot be a TM, so the assumption that $LP = L(K)$ for some total TM K must have been false, so LP is not recursive.

[20] 3 Determine whether the following statements are true or false. Every correct answer is worth 2 points. For every wrong answer 1 point is subtracted, provided the total number of points is non-negative.

statement

For every partial order $\leq$ on A, its complement $\sim(\leq) = (A \times A) - \leq$ is a partial order.

False. For instance, for the partial order $\leq$ on $\{a, b, c\}$ defined by $a \leq a \leq b \leq b$, the relation $\sim(\leq)$ is neither reflexive (not $a \sim(\leq) a$), nor transitive (although $a \sim(\leq) c \sim(\leq) b$, not $a \sim(\leq) b$), nor anti-symmetric (although $b \sim(\leq) c \sim(\leq) b$, not $b = c$).

For every regular language L over Σ, its complement $\sim L = \Sigma^* - L$ is regular.

True.

For every recursively enumerable language L over Σ, its complement $\sim L = \Sigma^* - L$ is recursively enumerable.

False. The complement $\sim HP$ of the halting problem HP is not r.e. (as otherwise both would be recursive).

For every set B and countable subset $A \subset B$, its complement $\sim A = B - A$ is countable.

False. For instance, take $A = \emptyset$ and B the set of real numbers.

The specification $f(x) = f(x) \cdot 2$ defines a function $f : \mathbb{N} \to \mathbb{N}$.

True. The unique solution to $y = y \cdot 2$ on the natural numbers is $y = 0$, so the specification defines the constant-0 function; $\forall n \in \mathbb{N}, f(n) = 0$.

Let $\mathbb{N}_2 = \mathbb{N} - \{0, 1\}$. For every $n \in \mathbb{N}_2$, n is prime iff n is minimal w.r.t. divisibility in $\mathbb{N}_2$.

True. One of the given characterisations of primality is not being divisible by smaller numbers.

The function mapping n to the pair $(n \bmod 10, n \bmod 7)$ is a bijection between $\{0, \ldots, 69\}$ and the set of pairs of natural numbers $\{(x, y) \mid 0 \leq x < 10, 0 \leq y < 7\}$.

True. A direct consequence of the Chinese Remainder Theorem, since $\gcd(10, 7) = 1$.

For all natural numbers a, b, if $\gcd(a, b) = 1$, then $a^1, a^2, \ldots, a^b$ are all distinct modulo b.

False. For instance, although $\gcd(2, 3) = 1$ for $a = 2$, $b = 3$, we have $2^1 \equiv 2 \equiv 2^3 \pmod 3$, (It is *almost* true though: $a^1, a^2, \ldots, a^{b-1}$ *are* all distinct modulo b.)

If $f : A \to B$, $g : B \to C$, $h : C \to A$ are injective functions, then A and B are equinumerous.

True. Since injective functions are closed under composition, $h \circ g : B \to A$ is an injection, from which we conclude by the theorem of Schröder–Bernstein that there exists a bijection between A and B.

For every DFA A and each of its states q, there exist strings x, y such that $\hat{\delta}(q, x) = \hat{\delta}(q, y)$ and $x \neq y$.

True. By the idea of the pumping lemma: If A has n states then starting from any state q, an input string x of length $n+2$, makes the automaton A go through more than n states, so some state is repeated, and the corresponding non-empty substring y of x, can be pumped arbitrarily often.

4∗ Let $\sqsubseteq$ be the lexicographic order on pairs of natural numbers, with $\sqsubset$ its strict part. For instance, $(3,5) \sqsubset (5,3) \sqsubset (5,5) \sqsubseteq (5,5)$. Let R be the relation on pairs of natural numbers defined in Exercise 2(b).

[5] (a) Show that $\sqsubseteq$ is a total relation on pairs of natural numbers.

For $\sqsubseteq$ to be total we must have for all pairs (n, m) and (n', m') that either $(n, m) \sqsubseteq (n', m')$ or $(n', m') \sqsubseteq (n, m)$. Since $\leq$ is a total relation on natural numbers, either $n < n'$ or $n = n'$ or $n' < n$. In the first case $(n, m) \sqsubseteq (n', m')$, and in the last case $(n', m') \sqsubseteq (n, m')$, both by definition of the lexicographic order. If $n = n'$, then $m \leq m'$ or $m' \leq m$ (or both) again by $\leq$ being total. Thence, $(n, m) \sqsubseteq (n, m')$ or $(n, m') \sqsubseteq (n, m)$. More generally, if the two components orders are total, as is the case here for $\leq$ on the natural numbers, then their lexicographic product is total, by the reasoning given.

Alternatively, we can use that the lexicographic order is total on all sequences, so certainly on pairs.

(b) Show that each R-equivalence class has a unique $\sqsubseteq$-minimal element, give such $\sqsubseteq$-minimal elements for $[(3,5)]$, $[(5,3)]$, and $[(5,5)]$, and argue that taking $\sqsubseteq$-minimal
[5] elements yields a system of representatives of R.

The lexicographic order on pairs of natural numbers is a well-founded partial order. Elements of shape $(0, m)$ or $(n, 0)$ are even $\sqsubseteq$-least in their R-equivalence classes. If $(0, m)\ R\ (n', m')$ that holds since either $0 < n'$, or $0 = n'$ and $n = m'$. If $(n, 0)\ R\ (n', m')$ that holds since either $n = n'$ and $m' = 0$ or $n < n'$ (uniqueness of representatives). Moreover, every element (n', m') is R-equivalent to $(0, m' - n')$ if $n' \leq m'$ and to $(n' - m', 0)$ otherwise (existence of representatives). Thus, the elements of shapes $(0, m)$ or $(n, 0)$ give rise to a system of representatives.

The $\sqsubseteq$-minimal elements are, respectively $(0, 2)$, $(2, 0)$, and $(0, 0)$.

[5] (c) Show that there is a bijection between the set of integers $\mathbb{Z}$ and the set of equivalence classes of R.

The function f that maps $-n$ to $[(n, 0)]$ and n to $[(0, n)]$ for $n \in \mathbb{N}$, is a bijection. Injectivity is clear (noting that 0 is mapped to $[(0, 0)]$ either way), and that its inverse is injective follows from the previous item.

5∗ (a) Show that for any n, there is a dag having n nodes and $\frac{n \cdot (n-1)}{2}$ edges. (Cf. the graph
[5] G of Exercise 1, which has 5 nodes and $\frac{5 \cdot 4}{2} = 10$ edges),

For a given n, let the directed graph G_n have as nodes $\{1, \ldots, n\}$, and as edges $\{(i, j) \mid i < j\}$. G_n trivially is acyclic, so a dag. Taking the sum over all nodes of the number of edges to that node, we obtain (double counting) that the total number of edges of G_n is $\sum_{i=1}^{n} i - 1 = \sum_{i=0}^{n-1} i = \frac{n \cdot (n-1)}{2}$ as desired.

[5] (b) Show that the number given in the previous item is maximal, i.e. show that there are no dags having n nodes and more than $\frac{n \cdot (n-1)}{2}$ edges.

We prove the claim by induction on the number of nodes. The base case (no nodes) is trivial. Suppose G is a dag having $n > 0$ nodes. For a G-minimal node v (it exists by acyclicity), let G' be obtained from G by removing v and all k edges connecting

it. Then G' is a dag having $n-1$ nodes, so has at most $\frac{(n-1)\cdot(n-2)}{2}$ edges. By choice of v, all edges connecting v to G' are edges *from* v, so $k \leq n-1$. We conclude, since $\frac{(n-1)\cdot(n-2)}{2} + n - 1 = \frac{n\cdot(n-1)}{2}$.

Alternatively, for a proof without induction, first observe that in a dag G_n on n nodes having a maximal number of edges, there is *at least* one edge between any pair of distinct nodes v, v'. This holds, since either there is a path from one to the other, say w.l.o.g. from v to v', or there is no such path, and in both cases an edge from one to the other (from v to v') can be adjoined (if it was not there yet) without creating a cycle. Moreover, there is *at most* one edge between any pair of distinct nodes, since if there were two, they would be in opposite direction forming a cycle. Computing as above, the number of edges of G_n is found to be $\frac{n\cdot(n-1)}{2}$, the number of unordered pairs of distinct nodes, i.e. n^2 (the number of ordered pairs) minus n (for distinctness) divided by 2 (for unorderedness).

(c) [5] Give a *maximal* spanning tree T of the undirected graph U corresponding to the directed graph G of Exercise 1, i.e. a spanning tree having maximal weight among all spanning trees. Argue why your tree T is indeed a maximal spanning tree.

A maximal spanning tree of G is $v2 \leftarrow_9 v5 \rightarrow_8 v3 \leftarrow_{10} v1 \leftarrow_7 v4$ with weight 34, It is maximal since it must be spanning and connect 5 nodes, so must have 4 edges, and the weights of the edges selected are greater than all other edge weights.

More generally, the maximal spanning tree of an undirected weighted graph U can be obtained by first negating all the weights to yield an undirected weighted graph $-U$, then applying Kruskal's algorithm to yield a minimal spanning tree $-T$ of $-U$, and then negating the weights in $-T$ to yield a tree T. That T then is a spanning tree of U is immediate, since only the weights differ between T and $-T$, and U and $-U$. That T is maximal holds, since if there were a spanning tree T' having greater weight, then the tree $-T'$ with negated weights would be a spanning tree of $-U$ have smaller weight than $-T$. Formally, this is based on that $w < w'$ iff $-w' < -w$ so that $\max(w, w') = -\min(-w, -w')$, and $\sum_i -w_i = -\sum_i w_i$ so that $w(T) = -w(-T)$. It easily checked that applying this procedure to G yields the tree given above.