EDAA40 Exam

31 May 2021

Solutions are set in blue, additional material and commentary in red.

1	2	3	PC	total
30	40	30	(5/10)	100

Total points: 100

points required for 3: 50

points required for 4: 67

points required for 5: 85

ID

Place your ID here when you scan this page.

Personnummer: ____________________

Name: ____________________

Signature: ____________________

Rules

Things you CAN use during the exam.
Any written or printed material is fine. Textbook, other books, the printed slides, handwritten notes, whatever you like. You may use electronic versions of the material, e.g. PDFs and the like.

In any case, it would be good to have a source for the relevant definitions, and also for notation, just in case you don't remember the precise definition of everything we discussed in the course.

You can use Clojure to test or validate your definitions, as long as all run is code belonging to the basic Leiningen distribution or that you wrote during the exam and do not access other software tools.

Things you CANNOT use during the exam.
Any communication facility, other than to interact with the examiner.

The exam is ongoing from the moment you receive this document to the moment you submit your answers. **During that time, you must not communicate with anybody other than the examiner and you must not solicit or accept help from any third party with any part of the exam.**

FAQ

Can I make auxiliary/helper definitions?
Yes.

Can I use definitions from previous tasks?
Yes. If you do, include a short note referencing the task you take the definition from.

Should you have made an error in the previous definition, it will **not** affect the score on a task in which you use it. In other words, if you use previous definitions, I will, for the purpose of grading the answer using them, assume that they are correct.

Do I need to provide only the answer or also the calculations I performed to get to it?

Unless I specifically ask for the path to an answer, the answer itself is sufficient.

Instructions

- This is a take-home exam. This document was sent to you electronically as a PDF.
- Print out this document and write your answers in the appropriate spaces.[1] You can use additional sheets if you need to.
  - Fill out the first page and sign.
- Once finished, scan or otherwise photographically capture the pages and produce a PDF from them (using software such as Office Lens, for example). **Include your photo ID on the first page.**
- The **name of the PDF file** must be your personnummer followed by "`.pdf`", i.e. it has the format

  `yymmdd-nnnn.pdf`
- **Return the PDF with your answers by replying to the email that you received the question sheet in**. The subject line must include "`[EDAA40 Exam]`" (without the quotes). Do not forget to attach the file. Make sure to **include the PDF as an email attachment –** do NOT send a link to your answers.
- If you have questions for the examiner during the exam, contact him by phone first (you will receive the phone number in the email with the exam).

Good luck!

Key points:

- create a legible PDF, name it with your personnummer in the above format
- attach the PDF to the email – no links to hosted files etc.
- return email to the address it came from, with [EDAA40 Exam] in the subject

1 If printing is not an option, you can answer the questions on empty sheets of paper. If you do, make sure to include, on your first page, your name, personnummer, and signature, **and scan it with your photo ID**.

- You need not reproduce complete multiple-choice tables in your answers. In that case, your answers will be matched to the questions in the table row by row, so make sure you match the order of your answers to that of the questions.

Programming contest

Were you member of a group that qualified for the EDAA40 programming contest this year? This is where you claim your bonus points.

	I was in a group that qualified but did not win.
	I was in the group that qualified and won.
	None of the above.

(please tick the appropriate box)

Group name (if applicable): ____________________________________

Some conventions you may use in the exam:

For any endorelation $R \subseteq A \times A$ and any set $X \subseteq A$, you can use $R^0(X) = X$ and for any $n \in \mathbb{N}$, $R^{n+1}(X) = R(R^n(X))$. Note that this means that $R^1(X) = R(X)$, i.e. the image of X under R.

In situations when ∞ is a meaningful value, you can use $\min\{\infty\} = \min \emptyset = \infty$, as well as $\min(S \cup \{\infty\}) = \min S$ and $x + \infty = \infty$ for any x.

About scoring multiple choice tables:

Every correct answer **adds** the indicated number of points per answer, while **every incorrect answer deducts the same number of points**. Marking “no answer” (or simply not marking any box in a given row) does not change the point score, so it counts as 0. Should the total score for the table be negative, it is counted as zero (0).

1 [30 p]

Suppose a directed graph (V, E). Assume that $V = \mathrm{dom}\ E \cup \mathrm{rng}\ E$. We say that an edge $(v, w) \in V$ should be ... $\in E$ is *going away from* v and *coming into* w.

1. [5 p] Define the set I of vertices $v \in V$ that have at least one edge going away from and no edges coming into them.

   $I = \mathrm{dom}\ E \setminus \mathrm{rng}\ E$

2. [5 p] Define the set T of vertices that have at least one edge coming into and no edges going away from them.

   $T = \mathrm{rng}\ E \setminus \mathrm{dom}\ E$

3. [5 p] Define the function $d : \mathcal{P}(V) \times \mathcal{P}(V) \longrightarrow \mathbb{N} \cup \{\infty\}$ such that $d(X, Y)$ is the length of the shortest path from any vertex in X to any vertex in Y. If there is no such path, that distance is ∞. If $X \cap Y \neq \emptyset$, it is 0.

   $d : \mathcal{P}(V) \times \mathcal{P}(V) \longrightarrow \mathbb{N} \cup \{\infty\}$

   $X, Y \mapsto \min\{n : E^n(X) \cap Y \neq \emptyset\}$

   Many answers used recursion of some kind. Note that any recursive formulation would have to handle the case that a path does not exist, for example like this

$$X, Y \mapsto \begin{cases} 0 & \text{if } X \cap Y \neq \emptyset \\ \infty & \text{if } E(X) \subseteq X \\ 1 + d(X \cup E(X), Y) & \text{otherwise} \end{cases}$$

   Since d's arguments are already sets, there is no need for a helper function, one can simply accumulate the vertices by growing the first argument. Finiteness of V then guarantees termination.

   But accumulate the vertices one must. An alternative is to explicitly test whether the function will terminate, like so:

$$X, Y \mapsto \begin{cases} 0 & \text{if } X \cap Y \neq \emptyset \\ \infty & \text{if } E^+(X) \cap Y = \emptyset \\ 1 + d(E(X), Y) & \text{otherwise} \end{cases}$$

On the first call, this tests whether repeated applications of E will eventually yield a vertex in Y, and only proceeds with the recursive call if it does. Try to see what a well-founding for the recursion of this function might look like. However, while the above function always terminates, a very similar one does not:

$$X, Y \mapsto \begin{cases} 0 & \text{if } X \cap Y \neq \emptyset \\ \infty & \text{if } E^+(X) \cap Y = \emptyset \\ 1 + \min\ \{d(\{x\}, Y) : x \in E(X)\} & \text{otherwise} \end{cases}$$

Keeping previous vertices around to see whether new vertices can be reached or testing explicitly is necessary, however. Consider these two incorrect implementations:

a) $$X, Y \mapsto \begin{cases} 0 & \text{if } X \cap Y \neq \emptyset \\ 1 + d(E(X), Y) & \text{otherwise} \end{cases}$$

b) $$X, Y \mapsto \begin{cases} 0 & \text{if } X \cap Y \neq \emptyset \\ 1 + d(E(X) \setminus X, Y) & \text{otherwise} \end{cases}$$

The first looks at the vertices that can be reached from the current set of vertices, the second the front of "newly discovered" vertices $E(X) \setminus X$, until it intersects with Y. Either won't terminate if Y cannot be reached from X. Try running them on $(\{a, b\}, \{(a, a), (b, a)\})$ with $X = \{a\}$ and $Y = \{b\}$.

4. [5 p] Define the $M \subseteq V$ of vertices $v \in V$, such that the shortest path from any vertex in I to v is the same length as the shortest path from v to any vertex in T.

$$M = \{v \in V : d(I, \{v\}) = d(\{v\}, T)\}$$

5. [5 p] Define a function $e : V \longrightarrow \mathbb{N}$ that maps every vertex $v \in V$ to the number of edges it occurs in, i.e. that are coming into v or going away from v or both.

   $e : V \longrightarrow \mathbb{N}$

   $$v \mapsto \#\{(a, b) \in E : a = v \vee b = v\}$$

   Many answers used the cardinalities of $E(v)$ and $E^{-1}(v)$ to compute the number of edges. In that case care must be taken to accurately account for a possible self-loop (v, v) not be counted twice. Another approach was to take the cardinality of the union of those sets, sometimes reusing the answer to the next task. However, consider the graph $(\{a, b, c\}, \{(a, b), (b, a), (b, c), (c, b)\})$ and the value for $e(b)$, which should be 4.

   That example also puts the nail in the coffin for this solution:

   $$v \mapsto \#E(v) + \#E^{-1}(v) - \#(E(v) \cap \#E^{-1}(v))$$

   As in this case $E = E^{-1}$, and $E(b) = E^{-1}(b) = \{a, c\} = E(b) \cap E^{-1}(b)$, the result would be $e(b) = 2 + 2 - 2 = 2$, which is clearly incorrect.

   Consequently, since this question is about the number of edges that have a certain property, it is easiest to start with the edges in E, filter them by the property, and count the resulting set, rather than look at sets of vertices.

6. [5 p] Define the function $N : V \longrightarrow \mathcal{P}(V)$ that maps every vertex to the set of the vertices connected to it by an edge (in either direction).

   $N : V \longrightarrow \mathcal{P}(V)$

   $$v \mapsto E(v) \cup E^{-1}(v)$$

2 [40 p]

Suppose a set of vertices V, and two sets of edges $E_1 \subseteq V \times V$ and $E_2 \subseteq V \times V$ between them, such that (V, E_1) and (V, E_2) are two directed graphs with the same set of vertices. In addition, we have two functions $W_1 : E_1 \longrightarrow \mathbb{N}^+$ and $W_2 : E_2 \longrightarrow \mathbb{N}^+$ that assign the edges in each graph positive natural numbers as *edge weights*.

> *An example of such a situation could be cities connected by different kinds of transport, e.g. buses and trains, where the edge weights might measure e.g. the time it takes to use the corresponding transport.*

The goal is to measure a kind of *distance* $d(v, w)$ between any two vertices $v, w \in V$. It is based on the paths we can use to travel von v to w using the edges in both edge sets as follows.

The idea is that we travel from v to w on a path along the edges of both E_1 and E_2, adding the edge weights along that path as we go along. In addition, whenever we leave a vertex on that path using a edge from a different set than the edge that we used to reach it, we add a constant $s \in \mathbb{N}^+$ to the overall path weight as *switching cost*. So, for example, if we reached the current vertex using an edge from E_1, and we leave it using an edge from E_2, we add, in addition to the edge weights, s to the overall distance. If, on the other hand, we arrive at a vertex on an edge from, for instance, E_2, and we leave it again on an edge from E_2, no switching cost is added, and we just accumulate the corresponding edge weights.

We can switch back and forth between using edges from either edge set any number of times, adding s each time we do so.

> *The switching cost might be the additional delay incurred, for example, by walking from the train station to the bus terminal or vice versa. To simplify things, we assume that the switching cost is always the same, irrespective of the vertex the switching occurs on or whether we switch from E_1 to E_2 or vice versa.*

The distance $d(v, w)$ is the smallest path weight including switching costs for any path of the kind described above between v and w. If there is no way to reach w from v using the edges in E_1 and E_2, their distance is $d(v, w) = \infty$. For any vertex $v \in V$, the distance $d(v, v) = 0$.

Note that it is not necessarily the case that $E_1 \cap E_2 = \emptyset$, i.e. some edges (v_1, v_2) may be in both edge sets, and in that case their weights $W_1(v_1, v_2)$ and $W_2(v_1, v_2)$ need not be the same, so it is important to distinguish whether the edge was taken from E_1 or E_2 to properly keep track of weights and switching costs.

> *If you think of it in terms of buses and trains: even if two cities are connected by both a bus and a train, when you calculate the overall time of an itinerary, you need to distinguish the two in order to figure out (a) how long that part of the journey took (that's the potentially different weights of the same edge) and also (b) to figure out*

whether you need to account for trips between bus terminals and train stations (i.e. potential switching costs).

Also note that you cannot make the assumption that it is always better to avoid switching. Whether switching results in a shorter overall distance will always depend on the edge weights and s.

Define the distance function $d : V \times V \longrightarrow \mathbb{N} \cup \{\infty\}$ in the following two steps.

1. [25 p] First, define a helper function recursively

$$d' : V \times V \times \mathbb{N}^+ \times \{1,2\} \times \mathcal{P}(V) \longrightarrow \mathbb{N}^+ \cup \{\infty\}$$

with the following meaning of the arguments of $d'(v, w, D, n, S)$:

v	is the current vertex
w	is the vertex we want to reach
D	is the distance accumulated so far, i.e. from the initial starting point up to v
n	is 1 or 2 and denotes the edge set we took the edge from with which we reached the current vertex, 1 for E_1 and 2 for E_2
S	is the set of vertices we have visited so far (including the current vertex v, so it is always the case that $v \in S$)

$v, w, D, n, S \mapsto$

$$\begin{cases} D & \text{for } v = w \\ \min\{d'(v', w, D + W_1(v,v'), 1, S \cup \{v'\}) : v' \in E_1(v) \setminus S\} \\ \cup\{d'(v', w, D + W_2(v,v') + s, 2, S \cup \{v'\}) : v' \in E_2(v) \setminus S\} & \text{for } v \neq w, n = 1 \\ \min\{d'(v', w, D + W_1(v,v') + s, 1, S \cup \{v'\}) : v' \in E_1(v) \setminus S\} \\ \cup\{d'(v', w, D + W_2(v,v'), 2, S \cup \{v'\}) : v' \in E_2(v) \setminus S\} & \text{for } v \neq w, n = 2 \end{cases}$$

There are various ways to make this more compact. For example, you can use the fact that 3-n yields 2 for n=1 and 1 for n=2 to shorten this somewhat:

$$v, w, D, n, S \mapsto \begin{cases} D & \text{for } v = w \\ \min\{d'(v', w, D + W_n(v, v'), n, S \cup \{v'\}) : v' \in E_n(v) \setminus S\} \\ \cup\{d'(v', w, D + W_{3-n}(v, v') + s, 3 - n, S \cup \{v'\}) : v' \in E_{3-n}(v) \setminus S\} & \text{for } v \neq w \end{cases}$$

One rather clever solution managed to shorten the expression in the final case to

$$\min\{d'(v', w, D + W_k(v, v') + s \cdot \#(\{k\} \setminus \{n\}), k, S \cup \{v'\}) : k \in \{1, 2\}, v' \in E_k(v) \setminus S\}$$

Key points of a correct solution are the following:

- termination when $v = w$
- recursive calls on d' with (a) the proper weight added to D depending on which edge set the edge was taken from and (b) additionally s added to the weight when that edge set was different from the one v was reached on,
- exploring all possible paths to w (filtered only by the growing S to avoid cycles), returning the minimum weight from all of them.

This last point bears emphasizing. One common mistake was to choose a particular edge from E_1 and E_2 based on their weight and switching cost, leading to many more cases with complicated conditions. However, this choice cannot be made locally, because it affects the subsequent weights one encounters after picking the "cheapest" next step.

Many solutions also explicitly tested for non-reachability of the target vertex and returned ∞ in that case, for example by using a case with a condition like $(E_1 \cup E_2)(v) \setminus S = \emptyset$. While not incorrect (and so no points were deducted from the score), it is redundant given the convention that $\min \emptyset = \infty$. If the rest was done correctly, the sets of recursive calls to d' would eventually become empty, and so $\min$ would automatically produce the desired result.

2. [10 p] Using the helper function d' from the previous step, define the distance function

   $$d : V \times V \longrightarrow \mathbb{N} \cup \{\infty\}$$

   as described above.

$$v, w \mapsto \begin{cases} 0 & \text{for } v = w \\ \min\{d'(v', w, W_1(v, v'), 1, \{v, v'\}) : (v, v') \in E_1)\} & \\ \cup\{d'(v', w, W_2(v_1, v'), 2, \{v, v'\}) : (v, v') \in E_2)\} & \text{otherwise} \end{cases}$$

Some solutions looked like this:

$$v, w \mapsto \min\{d'(v, w, 0, 1, \{v\}), d'(v, w, 0, 2, \{v\})\}$$

That computes a correct result, but technically violates the definition of d', which requires a positive value as a third argument. (A few solutions noticed this and worked around it by passing a 1 into d' and subtracting it again at the end.) I concluded that this hint was too subtle and let it slide, accepting solution that passed 0 to d'.

3. [5 p] In order to ensure that d' terminates, we require a **well-founded strict order** $\prec$ on its domain $V \times V \times \mathbb{N}^+ \times \{1, 2\} \times \mathcal{P}(V)$, such that for any (v, w, D, n, S) that d' is called on, it will only ever call itself on $(v', w', D', n', S') \prec (v, w, D, n, S)$. Define such an order:

$$(v', w', D', n', S') \prec (v, w, D, n, S) \iff S' \supset S$$

Note: You are NOT required to *prove* that the order you define is well-founded, it suffices that it is. More specifically, a correct answer to this question must have three properties.

1. It must define a strict order on $V \times V \times \mathbb{N}^+ \times \{1, 2\} \times \mathcal{P}(V)$.
2. It must be well-founded, i.e. there cannot be an infinite descending chain in that order.
3. Your definition of d' must conform to it, i.e. any recursive call in your definition must be called on a smaller (according to the order) quintuple of arguments.

3 [30 p]

Given is a directed graph (V, E), a function $\lambda : E \longrightarrow A$, assigning each edge a label from a set A, and another function $W : E \longrightarrow \mathbb{R}^+$, assigning each edge a non-negative real number (its *weight*).

For any set of labels $S \subseteq A$, let $E_S = \lambda^{-1}(S) = \{e \in E : \lambda(e) \in S\}$, i.e. the set of edges whose label is in S.

With this, we define a family of distance functions $d_X : V \times V \longrightarrow \mathbb{R}^+ \cup \{\infty\}$ for every $X \subseteq A$, as follows:

$$d_X : V \times V \longrightarrow \mathbb{R}^+ \cup \{\infty\}$$

$$v, w \mapsto d'_X(v, w, \emptyset)$$

using a family of helper functions d'_X defined as follows:

$$d'_X : V \times V \times \mathcal{P}(V) \longrightarrow \mathbb{R}^+ \cup \{\infty\}$$

$$v, w, S \mapsto \begin{cases} 0 & \text{for } v = w \\ \min(\{d'_X(v', w, S \cup \{v\}) : v' \in E_X(v) \setminus S\} & \\ \qquad \cup \{W(v, v') + d'_X(v'w, S \cup \{v\}) : v' \in E_{A \setminus X}(v) \setminus S\}) & \text{otherwise} \end{cases}$$

We now use those distance functions to define a relation $R \subseteq \mathcal{P}(A) \times \mathcal{P}(A)$ on the subsets of A:

$$R = \{(X, Y) \in \mathcal{P}(A) \times \mathcal{P}(A) : \forall v \in V, v' \in V \ (d_X(v, v') \leq d_Y(v, v'))\}$$

In order to fill in the tables below, it helps to get a deeper understanding of these definitions. $d_X(v, w)$ is the smallest path distance between v and w, where the weights of the edges labeled with a label in X, i.e. the edges in E_X, are NOT counted, let's say those edges are *null-weighted*. This means that larger label sets can never make the path weight larger, only smaller, because adding labels to the label set means that all null-weighted edges remain null-weighted, plus whatever edges are labeled with the labels that were added. So $X \subseteq X' \rightarrow d_{X'}(v, w) \leq d_X(v, w)$ for any two $v, w \in V$.

Therefore, $(X, Y) \in R$ if and only if null-weighting the edges in E_X always leads to path weights that are not larger than null-weighting the edges in E_Y. Since supersets never make path weights larger, it follows that $X \subseteq X' \rightarrow (X', X) \in R$. Note that this means that for any $X \in \mathcal{P}(A)$, it is always true that $(A, X) \in R$ and $(X, \emptyset) \in R$. Also, for any $X \in \mathcal{P}(A)$, $(X, X) \in R$.

A few corner cases that might be interesting to think about:

- C_0 : the empty graph $(\emptyset, \emptyset)$ with an empty label set $A = \emptyset$. (Note that only a graph with no edges permits an empty label set – since $\lambda : E \longrightarrow A$ is a function, $A \neq \emptyset$ as soon as $E \neq \emptyset$.) Then, $\mathcal{P}(A) = \{\emptyset\}$ and $R = \{(\emptyset, \emptyset)\}$.

- C_1: a non-empty graph (V, E) with $V \neq \emptyset \neq E$, with a singleton label set $A = \{L\}$ (which therefore implies that all edges are labeled the same). Then, $\mathcal{P}(A) = \{\emptyset, A\}$ and $R = \{(\emptyset, \emptyset), (A, \emptyset), (A, A)\}$ (convince yourself that that is the case).

- C_2: a non-empty graph (V, E) with $V \neq \emptyset \neq E$, with label set $A = \{L_1, L_2\}$ and a constant labeling function $\lambda : e \mapsto L_1$. In other words, this is like the case C_1, except that we have an "unused" extra label L_2 in the label set. In this case, $\mathcal{P}(A) = \{\emptyset, \{L_1\}, \{L_2\}, A\}$ and

$$\begin{aligned} R = \{&(A, A), (A, \{L_1\}), (A, \{L_2\}), (A, \emptyset), \\ &(\{L_1\}, A), (\{L_1\}, \{L_1\}), (\{L_1\}, \{L_2\}), (\{L_1\}, \emptyset), \\ &(\{L_2\}, \{L_2\}), (\{L_2\}, \emptyset), \\ &(\emptyset, \{L_2\}), (\emptyset, \emptyset)\} \end{aligned}$$

- C_3: a non-empty, unconnected graph $(V, \emptyset)$ with $V \neq \emptyset$ and some non-empty label set $A \neq \emptyset$. Note that this means that distances are unaffected by the set of labels : $d_X(v, v')$ is 0 if $v = v'$ and ∞ otherwise, for any $X \in \mathcal{P}(A)$. Consequently, $R = \mathcal{P}(A) \times \mathcal{P}(A)$.

1. [10 p] The relation $R \subseteq \mathcal{P}(A) \times \mathcal{P}(A)$ depends on the combination of the graph (V, E), the label set A and the labeling and weight functions λ and w.

   In the following table, several properties are given. Tick the corresponding box under "always", if R has the property for every valid combination of graph, label set, and labeling and weight function, under "never" if it does NOT have that property for all such combinations, and under "sometimes" if it has that property for at least one combination and does not have the property for at least another one.
   Mark the corresponding box under "no answer" if you prefer to not give an answer.

   See p. 4 on how these tables are scored.

   [1 point per answer]

		always	**sometimes**	**never**	**no answer**
1	reflexive over $\mathcal{P}(A)$	X			
2	transitive	X			
3	symmetric		X		
4	antisymmetric		X		
5	asymmetric			X	
6	is a strict order			X	
7	is a non-strict order		X		
8	is a total order		X		
9	has a minimum element		X		
10	has a maximum element		X		

Note: Be sure of the distinction between minimum and minimal, and also between maximum and maximal.

Notes:

1. The condition for $(X, X) \in R$ is that for all $v, v' \in V$ it must be the case that $d_X(v, v') \leq d_X(v, v')$ (vacuously so if $V = \emptyset$), which is clearly the case no matter what X we choose, so R is always reflexive.
2. If $(X, Y) \in R$ and $(Y, Z) \in R$, it means that for any $v, v' \in V$, it is the case that $d_X(v, v') \leq d_Y(v, v')$, and also that $d_Y(v, v') \leq d_Z(v, v')$, so therefore $d_X(v, v') \leq d_Z(v, v')$, and consequently $(X, Z) \in R$. Hence, R is always transitive.
3. In general, R is not symmetric (larger sets tend to lead to smaller distances), but if the label set is empty as in case C_0, it is. Similarly, it is for unconnected graphs, such as case C_3.
4. It would seem that R might be antisymmetric – after all, it is generally not symmetric, but reflexive, and it certainly is antisymmetric in case C_1. But note how in case C_2, $(\{L_1\}, A) \in R$ and $(A, \{L_1\}) \in R$, so it is not always antisymmetric. Neither is it in case C_3.
5. Since R is always reflexive, and never empty, it cannot be asymmetric.
6. For the same reasons, it cannot be a strict order.
7. It is a non-strict order when it is antisymmetric, i.e. sometimes. For example, not in case C_2.
8. It is in cases C_0 and C_1, but in C_2 it is not an order at all.
9. It has a minimum (namely A) when R is an order.
10. It has a maximum (namely $\emptyset$) when R is an order.

2. [20 p] Similar to the previous task, the following table contains statements about the definitions above, such as the distance functions d_X and the relation $R \subseteq \mathcal{P}(A) \times \mathcal{P}(A)$. Whether they are true or false might depend on the specific combination of the graph (V, E), the label set A and the labeling and weight functions λ and w.

   Tick the corresponding box under "always", if the statement is true for every valid combination of graph, label set, and labeling and weight function, under "never" if it is false for all such combinations, and under "sometimes" if it is true for at least one combination and false for at least another one.
   Mark the corresponding box under "no answer" if you prefer to not give an answer.

   See p. 4 on how these tables are scored.

   [2 points per answer]

		always	**sometimes**	**never**	**no answer**
1	$\forall X, Y \in \mathcal{P}(A)\ \forall v, w \in V$ $(d_X(v,w) = \infty \leftrightarrow d_Y(v,w) = \infty)$	X			
2	$\forall X, Y \in \mathcal{P}(A)\ \forall v, w \in V$ $(d_X(v,w) = 0 \leftrightarrow d_Y(v,w) = 0)$		X		
3	$\forall X, Y \in \mathcal{P}(A)\ ((X,Y) \in R \to X \subseteq Y)$		X		
4	$\forall X, Y \in \mathcal{P}(A)\ (X \subset Y \to (X,Y) \in R)$		X		
5	$\forall X, Y \in \mathcal{P}(A)\ ((X \cup Y), X) \in R)$	X			
6	$\forall X, Y \in \mathcal{P}(A)\ (X, (X \cup Y)) \in R)$		X		
7	$\forall X, Y \in \mathcal{P}(A)\ ((X \cap Y), X) \in R)$		X		
8	$\forall X, Y \in \mathcal{P}(A)\ (X, (X \cap Y)) \in R)$	X			
9	$\exists X \in \mathcal{P}(A)\ \forall v, w \in V\ (d_X(v,w) = \infty)$		X		
10	$\exists X \in \mathcal{P}(A)\ \forall v, w \in V\ (d_X(v,w) = 0)$		X		

Hint: In deciding on these properties and statements, it can help to consider "corner cases": very small graphs, very sparse or very connected graphs (i.e. very few or very many edges), graphs where all edge weights are 0, where all edges are labeled the same or all are labeled differently etc.

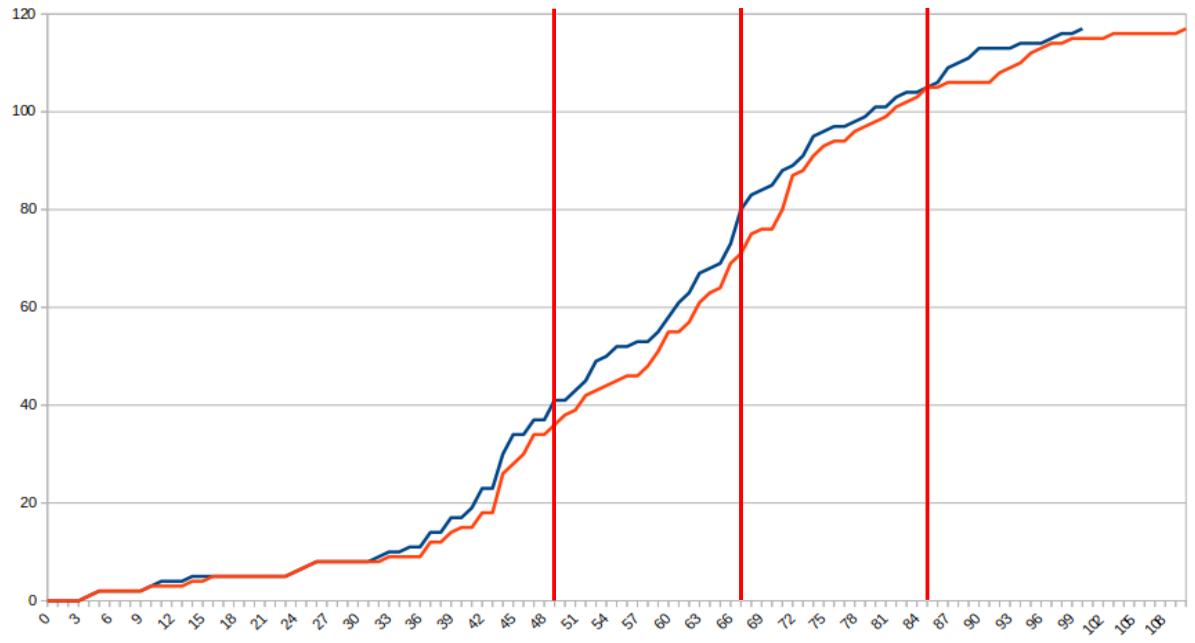
Notes:

1. The distance functions will only return the distance ∞ if there is no path from the first to the second vertex, so this is always the case irrespective of label set or vertices chosen.
2. This, however, is generally not true: the distance $d_X(v, v')$ can only be zero if $v = v'$ or all edges between the two vertices are labeled by labels in X. However, since this universally quantifies over the vertices, it is true for an empty graph, or for an unconnected graph, in which all distances are 0 or ∞ irrespective of the label set.
3. Generally, this would be false: shorter distances would signify larger label sets. Yet, in case C_0, the conclusion $X \subseteq Y$ is true for all $X, Y \in \{\emptyset\}$, therefore the implication holds, and thus the formula is true.
4. Once again, this formula would tend to be false in general. However, in C_0, the premise $X \subset Y$ is false for all $X, Y \in \{\emptyset\}$, so the implication holds, and therefore the formula is true.
5. As we observed, larger sets will always lead to distances that are not larger, so this is always true.
6. This would conversely tend to be false, but it is true, e.g. in case C_0 of an empty graph, and also case C_3, when the graph is unconnected.
7. This is the dual of 6: it would usually be false, but true for cases like C_0 and C_3.
8. This, in turn, is the dual of 5: smaller sets lead to not-smaller distances, so this is always true.
9. As observed before, the choice of the set X has no bearing on whether the distance $d_X(v, v')$ becomes infinite: if it is infinite for some $X \in \mathcal{P}(A)$, then it is for all of them. This can only be the case if there are no vertices and the inner universal quantifier becomes vacuously true (case C_0). Even in an unconnected non-empty graph (case C_3) the distances of the vertices to themlselves are 0.
10. This is true for graphs in which every vertex can be reached from every other vertex. Then, (at least) the set of all labels would lead to all-zero distances. However, in graphs where some vertices cannot be reached by some other vertices, their distance remains ∞, regardless of the choice of X.

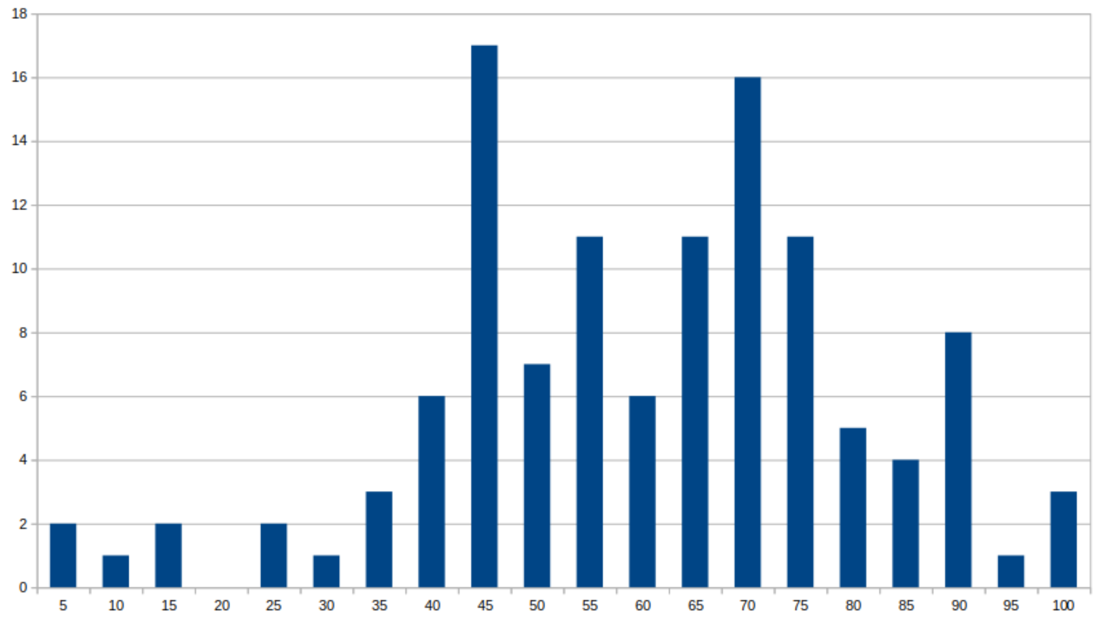
Statistics

117 students wrote the exam. The Gini coefficient of the scores is 0.21.

Below is the cumulative score graph, blue for the raw scores, red for the final adjusted scores including the bonus points for the programming contest. The x-axis is the score (in points), the y-axis shows the number of students whose score is less than or equal to the x-value. The vertical lines mark the minimal scores for 3, 4, and 5.



Below is a histogram of the score distribution (raw scores before adjustment), arranged by buckets of 5 points. The bucket label is the top score in the bucket. So, for example, the bucket labeled “65” shows the number of students scoring from 61 to 65 points.


18
16
14
12
10
8
6
4
2
0
5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100