KØBENHAVNS UNIVERSITET

Introduktion til diskret matematik og algoritmer: Exam April 2, 2025 Problems and Solutions

Please note that the main purpose of this document is to explain what the correct solutions are and how to arrive at them. Thus, while the text below is certainly intended to provide good examples of how to solve problems and reason about solutions, these examples do not necessarily specify exactly how the handed-in exams were expected to look like. This is especially so since for many problems there are more than one correct way of solving them. As communicated during the course, the course notes published on Absalon, which contain many problems that we worked out in class, are probably the best indicator of what level of detail is expected from the exam solutions.

1 (50 p) This problem is about different representations of integers.

1a (10 p) Write the binary number $(110)_2$ in decimal notation.

Solution: We have $(110)_2 = 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 6$.

1b (20 p) Write the decimal number 110 in binary notation.

Solution: Dividing by 2 and outputting the remainders will give us the bits in the binary expansion in reverse order. We obtain

$$\begin{aligned} 110 &= 55 \cdot 2 + 0 \\ 55 &= 27 \cdot 2 + 1 \\ 27 &= 13 \cdot 2 + 1 \\ 13 &= 6 \cdot 2 + 1 \\ 6 &= 3 \cdot 2 + 0 \\ 3 &= 1 \cdot 2 + 1 \\ 1 &= 0 \cdot 2 + 1 \end{aligned}$$

from which we see that $(110)_{10} = (1101110)_2$. Just to verify that we have not made any mistake, we can check our answer by computing $(1101110)_2 = 1{\cdot}2^6{+}1{\cdot}2^5{+}0{\cdot}2^4{+}1{\cdot}2^3{+}1{\cdot}2^2{+}1{\cdot}2^1{+}0{\cdot}2^0 = 64 + 32 + 8 + 4 + 2 = 110$.

1c (20 p) Write the octal number $(2025)_8$ in decimal notation.

Solution: We have $(2025)_8 = 2 \cdot 8^3 + 0 \cdot 8^2 + 2 \cdot 8^1 + 5 \cdot 8^0 = 2 \cdot 512 + 2 \cdot 8 + 5 = 1045$.

2 (50 p) Use the algorithm we have learned for determining $d = \gcd(m, n)$ for the numbers below, showing details of all function calls made, and then express d as a linear combination of m and n.

2a (20 p) $m = 38$ and $n = 14$.

Solution: We use the Euclidean algorithm, which is based on the observation that $\gcd(m, n) = \gcd(n, m \bmod n)$.

For $m = 38$ and $n = 14$ we obtain

$$\begin{aligned} 38 &= 2 \cdot 14 + 10 \\ 14 &= 1 \cdot 10 + 4 \\ 10 &= 2 \cdot 4 + 2 \\ 4 &= 2 \cdot 2 + 0 \end{aligned}$$

from which we see that $\gcd(38, 14) = 2$. By considering the above equalities in reverse order we can write the greatest common divisor

$$\begin{aligned} 2 &= 10 - 2 \cdot 4 \\ &= 10 - 2 \cdot (14 - 1 \cdot 10) \\ &= 3 \cdot 10 - 2 \cdot 14 \\ &= 3 \cdot (38 - 2 \cdot 14) - 2 \cdot 14 \\ &= 3 \cdot 38 - 8 \cdot 14 \end{aligned}$$

as a linear combination of 38 and 14 as desired.

2b (30 p) $m = 117$ and $n = 69$.

Solution: For $m = 117$ and $n = 69$ we obtain

$$\begin{aligned} 117 &= 1 \cdot 69 + 48 \\ 69 &= 1 \cdot 48 + 21 \\ 48 &= 2 \cdot 21 + 6 \\ 21 &= 3 \cdot 6 + 3 \\ 6 &= 2 \cdot 3 + 0 \end{aligned}$$

from which we see that $\gcd(117, 69) = 3$. Processing these equalities in reverse order we obtain

$$\begin{aligned} 3 &= 21 - 3 \cdot 6 \\ &= 21 - 3 \cdot (48 - 2 \cdot 21) \\ &= 7 \cdot 21 - 3 \cdot 48 \\ &= 7 \cdot (69 - 1 \cdot 48) - 3 \cdot 48 \\ &= 7 \cdot 69 - 10 \cdot 48 \\ &= 7 \cdot 69 - 10 \cdot (117 - 1 \cdot 69) \\ &= 17 \cdot 69 - 10 \cdot 117 \end{aligned}$$

as a linear combination of 117 and 69 as desired.

3 (60 p) Provide formal proofs of the following claims using proof techniques that we have learned during the course.

3a (30 p) Define $a_1 = 1$ and $a_n = 2a_{n-1} + 1$ for $n \geq 2$. Prove that for all positive integers $n \in \mathbb{N}^+$ it holds that $a_n = 2^n - 1$.

Solution: We prove this equality by induction over n.

Base case ***($n = 1$):*** We have $a_1 = 1 = 2^1 - 1$ by definition.

Induction step: Assume that the equality holds for $n - 1$ and consider n. We have

$$\begin{aligned} a_n &= 2a_{n-1} + 1 \\ &= 2 \cdot (2^{n-1} - 1) + 1 && \text{[by the induction hypothesis]} \\ &= 2^n - 2 + 1 \\ &= 2^n - 1 \end{aligned}$$

which is the desired equality.

The claim now follows by the induction principle.

3b (30 p) Prove that for all non-negative integers $n \in \mathbb{N}$ it holds that $3 \mid 4^n + 5$.

Solution: We prove this by induction over n.

Base case ***($n = 0$):*** For $n = 0$ we have that $4^0 + 5 = 6$ is divisible by 3.

Induction step: Suppose that $3 \mid 4^n + 5$, which is the same as saying that $4^n + 5 = 3 \cdot M$ for some integer M. Fixing this M, and working on the expression for $n + 1$, we get

$$\begin{aligned} 4^{n+1} + 5 &= 3 \cdot 4^n + 4^n + 5 && \text{[since } 4^{n+1} = (3+1) \cdot 4^n\text{]} \\ &= 3 \cdot 4^n + 3 \cdot M && \text{[by the induction hypothesis]} \\ &= 3 \cdot (4^n + M) \end{aligned}$$

which shows that this expression is divisible by 3.

The claim follows by the induction principle.

4 (90 p) Let $A = \{1, 2, 3, 4\}$ and consider the following binary relations on A:

$$\begin{aligned} R &= \{(2,1), (3,1), (3,2), (4,1), (4,2), (4,3)\} \\ S &= \{(1,1), (2,2), (3,3), (4,4)\} \\ T &= \{(1,1), (1,4), (2,2), (2,3), (3,2), (3,3), (4,1), (4,4)\} \end{aligned}$$

4a (60 p) For each of the relations above, determine whether it is

1. reflexive,
2. symmetric,
3. antisymmetric,
4. transitive.

Please make sure to explain, briefly but clearly, what these properties mean and why they are satisfied for a relation when they are. For any relation that fails to satisfy a property, make sure to provide a specific counterexample.

Solution: Recall that a relation R is:

- *reflexive* if for all x it holds that $(x, x) \in R$;
- *symmetric* if whenever $(x, y) \in R$ it also holds that $(y, x) \in R$;
- *anti-symmetric* if $(x, y) \in R$ and $(y, x) \in R$ implies $x = y$;
- *transitive* if whenever $(x, y) \in R$ and $(y, z) \in R$ it also holds that $(x, z) \in R$.

The relation R specified in the problem statement is anti-symmetric (simply since there is no pair (x, y) such that $(x, y) \in R$ and $(y, x) \in R$) and transitive (which can be verified by case analysis, or by observing that R is the greater-than relation). It is not reflexive since, e.g., $(1, 1) \notin R$, and it is not symmetric since, e.g., $(2, 1) \in R$ but $(1, 2) \notin R$.

The relation S is the identity relation, which vacuously satisfies all the properties listed.

The relation T can be verified to be reflexive, symmetric, and transitive. It is not anti-symmetric since $(2, 3) \in T$ and $(3, 2) \in T$ but $2 \neq 3$.

4b (30 p) Which of the relations above, if any, are equivalence relations or partial orders? Please make sure to justify your answers.

Solution: An *equivalence relation* is a relation that is reflexive, symmetric, and transitive. The relations S and T satisfy these conditions as argued above.

A *partial order* is a reflexive, anti-symmetric, and transitive relation. The identity relation S is formally speaking also a partial order, since it satisfies all the required properties (but it is of course a very boring partial order).

It might be worth pointing out that the relation R is *not* a partial order, since it is not reflexive. This is just a special case of the general fact that non-strict order relations define partial orders but strict order relations do not.

5 (60 p) For each of the propositional logic formulas below, determine whether it is a tautology or not. If the formula is not a tautology, show how to add a single connective to make it into a tautology. Please make sure to justify your answers (e.g., by presenting truth tables, or by using rules for rewriting logic formulas that we have learned in class).

5a (30 p) $\neg\big((p \to q) \vee r\big) \to \big((\neg q \wedge \neg r) \wedge p\big)$

Solution: This formula is a tautology, i.e., it is always true. To see this, note first that we know that

$$p \to q \;\equiv\; \neg p \vee q \tag{1}$$

since the only way the implication $p \to q$ can be false is that p is true and q is false, and this is one of the basic equivalences we have learned in the course. Furthermore, from De Morgan's laws it is easy to derive that

$$\neg(a \vee b \vee c) \;\equiv\; \neg a \wedge \neg b \wedge \neg c \tag{2}$$

(i.e., the only way a disjunction can be false is that all its disjuncts are false).

By combining (1) and (2), and using that conjunction is commutative, we see that the premise $\neg\big((p \to q) \vee r\big)$ is in fact equivalent to the conclusion $(\neg q \wedge \neg r) \wedge p$, and so the implication in Problem 5a will always evaluate to true.

5b (30 p) $\big((p \wedge q) \to r\big) \leftrightarrow \big((q \vee r) \vee \neg p\big)$

Solution: This is not a tautology. If we set p and q to true but r to false, we get that $(p \wedge q) \to r$ evaluates to false but $(q \vee r) \vee \neg p$ evaluates to true, and so the whole formula is false.

If we change q on the right-hand side to $\neg q$, we obtain the formula

$$\big((p \wedge q) \to r\big) \leftrightarrow \big((\neg q \vee r) \vee \neg p\big)\ , \tag{3}$$

which is a tautology. To see this, we can rewrite the left-hand side of (3) as

$$\begin{aligned}(p \wedge q) \to r &\equiv \neg(p \wedge q) \vee r &\text{(4a)}\\ &\equiv (\neg p \vee \neg q) \vee r &\text{(4b)}\\ &\equiv (\neg q \vee r) \vee \neg p &\text{(4c)}\end{aligned}$$

by using the property (1) of implication, De Morgan's laws, and commutativity of disjunction. This show that the left-hand and right-hand sides of of the formula (3) are equivalent, and so the formula will always evaluate to true.

6 (80 p) Recall that the Fibonacci numbers are defined as

$$\begin{aligned}F_1 &= 1\\ F_2 &= 1\\ F_n &= F_{n-1} + F_{n-2} \qquad\qquad \text{for } n \geq 3.\end{aligned}$$

Prove that consecutive Fibonacci numbers F_{n+1} and F_n are relatively prime, and show that for $n \geq 2$ the Euclidean algorithm when run on F_{n+1} and F_n makes exactly $n-1$ function calls to determine that this is so (i.e., it reaches remainder 0 after exactly $n-1$ function calls).

Solution: We prove that two consecutive Fibonacci numbers are relatively prime, i.e., that they have greatest common divisor 1, by using the Euclidean algorithm. While doing so, we count the number of function calls, i.e., the number of times the relation $\gcd(m, n) = \gcd(n, m \bmod n)$ is applied before reaching the trivial base case where n divides m and the remainder is 0.

Base case ***($n = 2$):*** For $F_3 = 2$ and $F_2 = 1$, we clearly have $\gcd(2, 1) = 1$. If we run the Euclidean algorithm, we get that $F_3 = 2 \cdot F_2 + 0$, and so we reach remainder 0 after a single step.

Induction step: Suppose that for $n-1$ it holds that $\gcd(F_{n-1}, F_{n-2}) = 1$ and that the Euclidean algorithm reaches remainder 0 after $(n-1) - 1 = n - 2$ function calls.

As a first step when computing $\gcd(F_n, F_{n-1})$, the Euclidean algorithm divides F_n by F_{n-1} to compute the remainder. This remainder is F_{n-2}, since by the definition of Fibonacci numbers we have that $F_n = 1 \cdot F_{n-1} + F_{n-2}$, and so the equalities

$$\gcd(F_n, F_{n-1}) = \gcd(F_{n-1}, F_n \bmod F_{n-1}) = \gcd(F_{n-1}, F_{n-2}) \tag{5}$$

hold. By the induction hypothesis, the Euclidean algorithm computes $\gcd(F_{n-1}, F_{n-2}) = 1$ with $n-2$ additional recursive calls. Hence, we conclude that the Euclidean algorithm will determine that F_n and F_{n-1} are relatively prime after $n-1$ function calls.

The claim in the problem statement now follows by the induction principle.

7 (90 p) For a few years now the Copenhagen metropolitan area (including Lund) has had an unusually large number of researchers in computational complexity theory, and a team of such researchers have decided to submit a joint grant application to create the *Copenhagen Computational Complexity Centre* focusing on research in this scientific field. Since gender balance is a serious issue in computer science, a noteworthy aspect of the team of co-applicants is that the male professors Amir, Jakob, and Srikanth at the University of Copenhagen are balanced by the female professors Nutan and Paloma at the IT University of Copenhagen and Susanna at Lund University.

For the subproblems below, please make sure to answer not just with numbers but with more combinatorial-looking expressions, and to expand these expressions out to show that you understand the meaning of any notation used. Also make sure to explain how you reason to reach your answers.

7a (50 p) Together with the application documents, the co-applicants are planning to enclose a group photo, and much thought has gone into how to choose the seating arrangement. All the researchers will be placed in a single row, but they have agreed that a great way to highlight the gender balance would be to make sure that male and female researchers alternate, so that every second person in the row is male or female, respectively. In how many different ways can the 6 researchers be arranged on the photo to satisfy this constraint?

Solution: The seating arrangement is uniquely specified by determining whether the leftmost person on the photo is male or female, and then by specifying the internal order of the female and male researchers, respectively. This gives us:

- 2 choices for a male or female researcher at the leftmost position;
- $3! = 6$ ways of arranging the 3 female researchers from left to right;
- likewise $3! = 6$ ways of arranging the 3 male researchers from left to right;

for a total of $2 \cdot 3! \cdot 3! = 2 \cdot 6 \cdot 6 = 72$ different arrangements.

7b (40 p) Any serious research centre application these days should also identify a steering committee for the centre. After long deliberations, the co-applicants have decided that this committee should:

- consist of 4 persons all in all;
- include co-applicants representing all 3 partner institutions, i.e, the University of Copenhagen, the IT University of Copenhagen, and Lund University;
- have perfect gender balance, i.e., two male and two female members.

In how many different ways can the steering committee be composed?

Solution: Since Susanna is the only co-applicant from Lund University, she has to be on the steering committee.

This means that we need exactly one more female committee member, who will be from ITU, and this gives us 2 choices for either Nutan or Paloma.

Finally, we need two male members, who will both have to come from the University of Copenhagen. We can think of choosing either two persons among Amir, Jakob, and Srikanth in $\binom{3}{2} = 3$ ways, or choosing one person to leave out in $\binom{3}{1} = 3$ ways.

Summing up (or, rather, multiplying together), we see that the steering committee can be composed in $1 \cdot 2 \cdot \binom{3}{2} = 1 \cdot 2 \cdot 3 = 6$ ways.

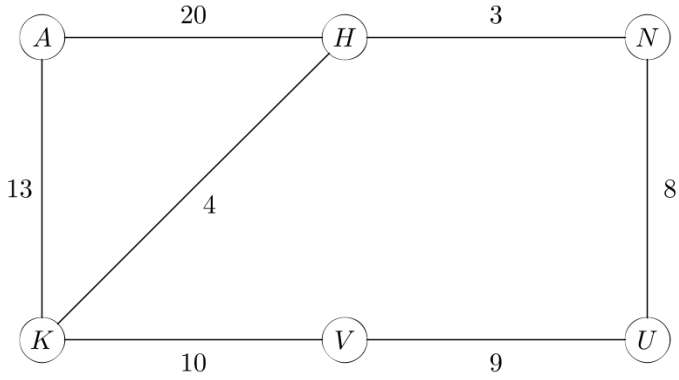


Figure 1: Graph modelling public transport options from Kastrup Airport to DIKU in Problem 8.

8 (90 p) When Jakob has international visitors, he needs to give them travel directions from Kastrup Airport to the Department of Computer Science (DIKU) at Universitetsparken. Jakob is aware of the following relevant public transport options in Copenhagen with travel times as stated (in either direction, and with time for switching transport mode included):

- Between Kastrup Airport and Kongens Nytorv by metro: 13 minutes.
- Between Kastrup Airport and København H by train: 20 minutes.
- Between København H and Nørreport by train: 3 minutes.
- Beween København H and Kongens Nytorv by metro: 4 minutes.
- Between Kongens Nytorv and Vibenshus Runddel by metro: 10 minutes.
- Between Vibenshus Runddel and Universitetsparken by foot: 9 minutes.
- Between Nørreport and Universitetsparken by bus: 8 minutes.

What is not so clear to Jakob is how he should use this information to find as fast a route as possible between the airport and DIKU to suggest to his visitors.

8a (20 p) Help Jakob by modelling this problem as a graph. Explain what the vertices and edges represent and what other information you need to add to the graph. Make sure to show concretely what graph you obtain for Jakob's problem above.

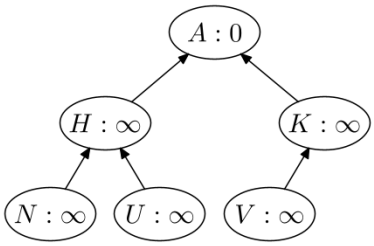
Solution: We create a graph with 6 vertices:

- vertex A for Kastrup Airport,
- vertex K for Kongens Nytorv,
- vertex H for København H,
- vertex N for Nørreport,
- vertex V for Vibenshus Runddel,
- vertex U for Universitetsparken and DIKU.

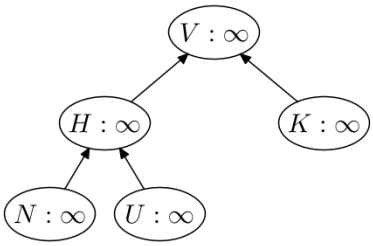
Between these vertices we add edges as specified by the travel options with weights equal to the travel times, and these edges are undirected since the time given is for travel in either direction:

- edge (A, K) with weight 13,
- edge (A, H) with weight 20,
- edge (H, N) with weight 3,
- edge (H, K) with weight 4,
- edge (K, V) with weight 10,
- edge (V, U) with weight 9,
- edge (N, U) with weight 8.

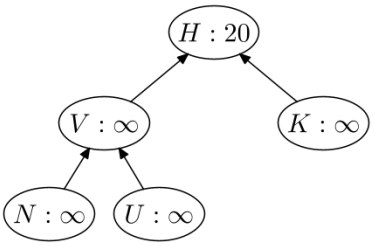
This yields the graph in Figure 1.



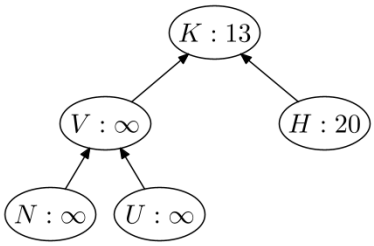
(a) Before dequeuing A.



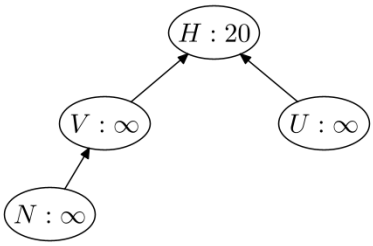
(b) After dequeuing A.



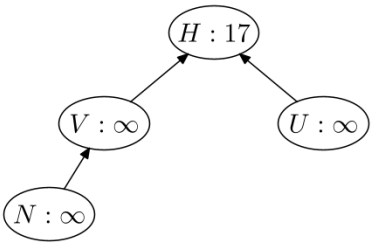
(c) After relaxing (A, H).



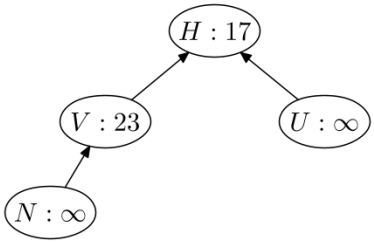
(d) After relaxing (A, K).



(e) After dequeuing K.



(f) After relaxing (K, H).



(g) After relaxing (K, V).

Figure 2: Heap configurations for priority queue in Problem 8.

8b (70 p) Propose a suitable graph algorithm to solve Jakob's problem. Explain what this algorithm is and why it is the right choice for this problem. Make a dry-run of the algorithm and explain the relevant steps in the execution (similarly to what has been done in class and in the lecture notes).

If your algorithm uses any auxiliary data structures, then explain in detail for the first two vertices processed how these data structures change. For the rest of the algorithm execution, just report what the relevant outputs of the data structures are without going into any details.

What travel directions for Jakob's visitors does your algorithm produce?

Solution: What Jakob is looking for is the fastest route from Kastrup to DIKU, i.e., the shortest path in a graph where the edge weights are the travel times between different locations. We have learned in the course to compute such shortest paths by using Dijkstra's algorithm.

Following the instructions in the problem statement, when running Dijkstra's algorithm we illustrate in Figure 2 how the heap used for the priority queue changes during algorithm execution, using the notation $v : k$ for a vertex v with key value k. At the outset, the vertex A corresponding to Kastrup Airport has key 0 and all other vertices have key ∞ as in Figure 2a.

1. After vertex A has been dequeued, vertex V is moved to the top of the heap and we have

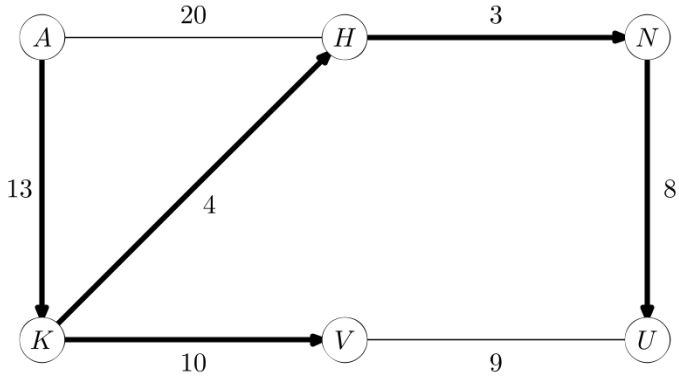


Figure 3: Shortest path tree for public transport from Kastrup Airport to DIKU in Problem 8.

the configuration in Figure 2b. Relaxing the edge (A, H) gives value 20 to H (i.e., the weight of the edge), and and the min-heap property is restored by letting H bubble up above V, yielding Figure 2c. Relaxing (A, K) decreases the key of K to 13 and makes H bubble up above K, yielding Figure 2d. These are all edges incident to A that we need to consider for relax operations.

2. Since vertex K is at the top of the heap, it is dequeued next. This adds the edge (A, K) to the spanning tree, which we indicate in Figure 3. When K is removed, U is moved to the top. This violates the min-heap property, since the key of U is not smaller than or equal to those of its children. Since H has smaller key than V, we swap U and H. This restores the heap property (since the left subtree of the root was not changed, and U is now the root of a singleton subheap), and so the heap after removal of K looks as in Figure 2e.

   Relaxing the edge (K, H) decreases the key value of H to 17. Since H is already the root, the heap does not change except for this key update and now looks like in Figure 2f. Relaxing (K, V) decreases the key value of V to 23, but again does not lead to any structural changes in the heap since V still has a larger key than its parent H (see Figure 2g). There are no further edges incident to K to relax. According to the instructions in the problem statement, we do not need to provide any further heap illustrations from this point on.

3. Since H is now the vertex with the smallest key value it is dequeued next, and the edge (K, H) is added to the spanning tree (since the latest update of the key value of H was when relaxing the edge (K, H)). When the edge (H, N) is relaxed the key of N is updated to 20.

4. Vertex N currently has the smallest key 20 and is dequeued next. This adds the edge (H, N) to the spanning paths tree, since the key of N was last updated when (H, N) was relaxed. Relaxing (N, U) updates the key of U to 28.

5. Now vertex V has the smallest key 23 and so is dequeued, adding (K, V) to the spanning tree. When we relax (V, U) nothing happens since $23 + 9 \geq 28$.

6. Finally, vertex U is dequeued, adding the edge (N, U) to the spanning tree. The queue is now empty, and there is nothing to relax.

The shortest path spanning tree computed as described above is indicated by the bold edges in Figure 3. From this tree we can read off that the fastest route for Jakob's guests is Kastrup Airport – Kongens Nytorv – København H – Nørreport – Universitetsparken/DIKU.

9 (120 p) Consider the graph in Figure 4 and the following ordered sequences listing the vertices in this graph:

1. a, c, e, b, h, g, f, d.
2. a, b, c, e, d, h, g, f.
3. a, c, b, e, h, g, f, d.
4. a, b, d, c, e, h, g, f.

For each of the sequences above, determine whether it can be the result of:

(a) a breadth-first search with vertices listed in order of visits;
(b) a depth-first search with vertices listed in order of discovery;
(c) a shortest-path computation with vertices listed in the order they are removed from the priority queue.

Each sequence above is the output of at most one of the algorithms, but since there are four sequences there could be a sequence that cannot be produced by any of the algorithms.

Partial credit is given for matching algorithms and vertex sequences correctly. For full credit, you need to give an overview of how and why the algorithms process the vertices in the given order (such as explaining the order of recursive calls, edges processed, or similar), or why none of the proposed algorithms could yield the sequence in question, but you do not have to provide detailed information about any auxiliary data structures used in the algorithms.

Solution: Let us consider the different algorithms in the order listed and try to find sequences that would match their outputs. Since we are promised that each sequence is the output of at most one algorithm, we are done with each sequence as soon as we find a match.

Sequence 2 is the result of breadth-first search starting with vertex a. To see this, consider how the different vertices are enqueued in and dequeued from the queue if we start the breadth-first search in vertex a:

Dequeued vertex	**Enqueued vertices**	**Queue**
—	—	(a)
a	$\{b, c, e\}$	(b, c, e)
b	$\{d\}$	(c, e, d)
c	—	(e, d)
e	$\{h\}$	(d, h)
d	—	(h)
h	$\{g\}$	(g)
g	$\{f\}$	(f)
f	—	$()$

Sequence 4 is the result of depth-first search starting with vertex a. Below follows an illustration of in which orders vertices are discovered and finished:

Discover a — undiscovered neighbours b, c and e
    Discover b — undiscovered neighbour d
        Discover d — undiscovered neighbour c
            Discover c — no undiscovered neighbours
            Finish c, since no undiscovered neighbours
        Finish d, since no undiscovered neighbours left
    Finish b, since no undiscovered neighbours left
    Discover e — undiscovered neighbour h
        Discover h — undiscovered neighbour g
            Discover g — undiscovered neighbour f

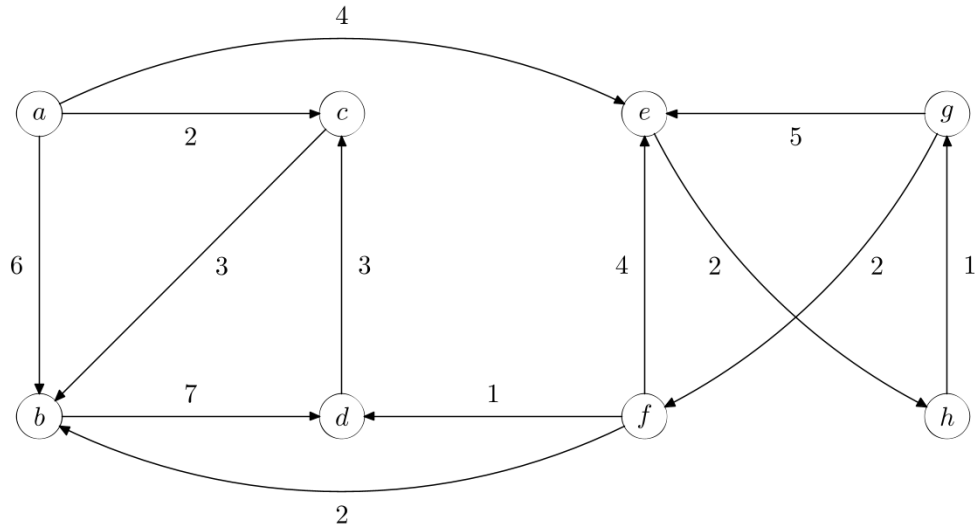


Figure 4: Graph for Problem 9

Discover f — no undiscovered neighbours
Finish f, since no undiscovered neighbours
Finish g, since no undiscovered neighbours left
Finish h, since no undiscovered neighbours left
Finish e, since no undiscovered neighbours left
Finish a, since no undiscovered neighbours left

Sequence 1 is the result of a call to Dijkstra's algorithm computing shortest paths from vertex a. To see this, consider in which order the vertices are dequeued from the priority queue and how vertex key values are updated as edges are relaxed:

Dequeued	**Updates**	**Priority queue (after relaxations)**
—	—	$\{(a:0),(b:\infty),(c:\infty),(d:\infty),(e:\infty),(f:\infty),(g:\infty),(h:\infty)\}$
a	$\{b,c,e\}$	$\{(c:2),(e:4),(b:6),(d:\infty),(f:\infty),(g:\infty),(h:\infty)\}$
c	$\{b\}$	$\{(e:4),(b:5),(d:\infty),(f:\infty),(g:\infty),(h:\infty)\}$
e	$\{h\}$	$\{(b:5),(h:6),(d:\infty),(f:\infty),(g:\infty)\}$
b	$\{d\}$	$\{(h:6),(d:12),(f:\infty),(g:\infty)\}$
h	$\{g\}$	$\{(g:7),(d:12),(f:\infty)\}$
g	$\{f\}$	$\{(f:9),(d:12)\}$
f	$\{d\}$	$\{(d:10)\}$
d	—	$\{\}$

Sequence 3, finally, is a spoiler sequence that cannot be the output of any of the algorithms listed. To see this, consider the following case analysis:

- If the sequence were the result of breadth-first search, then it would list vertices in order of the number of directed edges required to get to the vertex from the start vertex a, but the first deviation from this is that g is listed before d.
- A depth-first search starting with a, c, b would discover d next and not e.
- Finally, the sequence cannot be the output of Dijkstra's algorithms, since the vertices are not listed in increasing order of distance from a. The first deviation here is that e should come before b in a listing in increasing order of distance.