+1/1/60+

EPFL

n/a

Teacher : Ola Svensson
Algorithms CS-250 - (n/a)
21 Jan 2022
Duration : 180 minutes

n/a

SCIPER : **999999**

Do not turn the page before the start of the exam. This document is double-sided, has 24 pages, the last ones possibly blank. Do not unstape.

- Place your student card on your table.
- You are only allowed to have a handwritten A4 page written on both sides.
- Communication, calculators, cell phones, computers, etc... are not allowed.
- The exam consists of two parts. The first part consists of four multiple choice questions and the second part consists of four open-ended questions.
- Use a **black or dark blue ballpen** and clearly erase with **correction fluid** if necessary.
- If a question is wrong, the teacher may decide to nullify it.

Good luck!

Respectez les consignes suivantes | Read these guidelines | Beachten Sie bitte die unten stehenden Richtlinien		
choisir une réponse | select an answer Antwort auswählen	ne PAS choisir une réponse | NOT select an answer NICHT Antwort auswählen	Corriger une réponse | Correct an answer Antwort korrigieren
		
ce qu'il ne faut **PAS** faire | what should **NOT** be done | was man **NICHT** tun sollte		
		

First part: multiple choice questions

For each question, mark the box corresponding to the correct answer. Each question has **exactly one** correct answer. You do not need to justify your answers in this part.

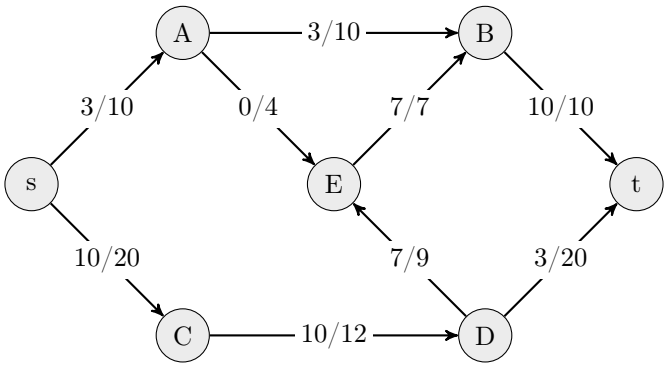
Question 1 : *(8 pts)* Consider an undirected graph $G = (V, E)$ where each vertex has degree three, i.e., each vertex is incident to exactly three edges. Now suppose we color the edges of G as follows:

> Each edge is independently colored red, green, or blue uniformly at random (i.e., each of the three colors is equally likely).

We say that a vertex is *monochromatic* if all its three incident edges are colored with the same color. What is the expected number of monochromatic vertices?

- ☐ The expected number of monochromatic vertices is $|V|/3$.
- ☐ The expected number of monochromatic vertices is $|E|/27$.
- ☐ The expected number of monochromatic vertices is $|V|/27$.
- ☐ The expected number of monochromatic vertices is $|V|/9$.
- ☐ The expected number of monochromatic vertices is $|E|/3$.
- ☐ The expected number of monochromatic vertices is $|E|/9$.

Question 2 : *(8 pts)* Assume the following flow network and corresponding flow of value 13 (the numbers on an edge determine its current flow and its capacity).



Starting with the depicted flow, what is the value of the flow obtained by running **one iteration** of the Ford-Fulkerson algorithm that chooses the *fattest augmenting path* (the one with the largest bottleneck).

- ☐ The value of the obtained flow is 16.
- ☐ The value of the obtained flow is 20.
- ☐ The value of the obtained flow is 18.
- ☐ The value of the obtained flow is 15.
- ☐ The value of the obtained flow is 14.
- ☐ The value of the obtained flow is 19.
- ☐ The value of the obtained flow is 13.
- ☐ The value of the obtained flow is 21.
- ☐ The value of the obtained flow is 17.

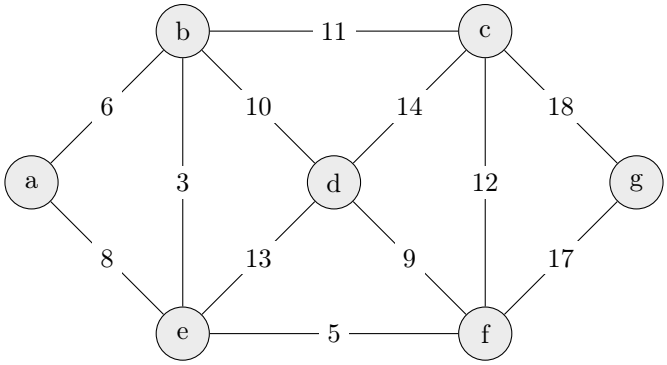
Question 3 : *(8 pts)* Arrange the following functions in increasing order according to asymptotic growth.

$$50\log n, \quad 10^{\sqrt{n}}, \quad 100\sqrt{n}\cdot\log\log n, \quad n/2, \quad n^3\cdot n^{\log n}, \quad (\log\log n)^{10}$$

The correct increasing order is

- ☐ $50\log n, \quad (\log\log n)^{10}, \quad n/2, \quad 100\sqrt{n}\cdot\log\log n, \quad 10^{\sqrt{n}}, \quad n^3\cdot n^{\log n}$
- ☐ $(\log\log n)^{10}, \quad 50\log n, \quad n/2, \quad 100\sqrt{n}\cdot\log\log n, \quad n^3\cdot n^{\log n}, \quad 10^{\sqrt{n}}$
- ☐ $50\log n, \quad (\log\log n)^{10}, \quad 100\sqrt{n}\cdot\log\log n, \quad n/2, \quad 10^{\sqrt{n}}, \quad n^3\cdot n^{\log n}$
- ☐ $50\log n, \quad (\log\log n)^{10}, \quad 100\sqrt{n}\cdot\log\log n, \quad n/2, \quad n^3\cdot n^{\log n}, \quad 10^{\sqrt{n}}$
- ☐ $50\log n, \quad (\log\log n)^{10}, \quad n/2, \quad 100\sqrt{n}\cdot\log\log n, \quad n^3\cdot n^{\log n}, \quad 10^{\sqrt{n}}$
- ☐ $(\log\log n)^{10}, \quad 50\log n, \quad n/2, \quad 100\sqrt{n}\cdot\log\log n, \quad 10^{\sqrt{n}}, \quad n^3\cdot n^{\log n}$
- ☐ $(\log\log n)^{10}, \quad 50\log n, \quad 100\sqrt{n}\cdot\log\log n, \quad n/2, \quad n^3\cdot n^{\log n}, \quad 10^{\sqrt{n}}$
- ☐ $(\log\log n)^{10}, \quad 50\log n, \quad 100\sqrt{n}\cdot\log\log n, \quad n/2, \quad 10^{\sqrt{n}}, \quad n^3\cdot n^{\log n}$

Question 4 : *(8 pts)*
Consider the following undirected graph with edge weights:



What are the weights of the edges (in the correct order) added to the solution by Dijkstra's and Prim's algorithms starting from a, i.e., with source a? Recall that Dijkstra's algorithm calculates a shortest path tree with source a and Prim's algorithm calculates a minimum spanning tree.

- ☐ The weights of the edges added by Dijkstra's algorithm (in the correct order) is $3, 5, 6, 9, 11, 17$.
  The weights of the edges added by Prim's algorithm (in the correct order) is $6, 8, 5, 10, 11, 17$.
- ☐ The weights of the edges added by Dijkstra's algorithm (in the correct order) is $6, 3, 5, 9, 11, 17$.
  The weights of the edges added by Prim's algorithm (in the correct order) is $6, 8, 5, 10, 11, 17$.
- ☐ The weights of the edges added by Dijkstra's algorithm (in the correct order) is $3, 5, 6, 9, 11, 17$.
  The weights of the edges added by Prim's algorithm (in the correct order) is $5, 6, 8, 10, 11, 17$.
- ☐ The weights of the edges added by Dijkstra's algorithm (in the correct order) is $5, 6, 8, 10, 11, 17$.
  The weights of the edges added by Prim's algorithm (in the correct order) is $6, 3, 5, 9, 11, 17$.
- ☐ The weights of the edges added by Dijkstra's algorithm (in the correct order) is $5, 6, 8, 10, 11, 17$.
  The weights of the edges added by Prim's algorithm (in the correct order) is $3, 5, 6, 9, 11, 17$.
- ☐ The weights of the edges added by Dijkstra's algorithm (in the correct order) is $6, 3, 5, 9, 11, 17$.
  The weights of the edges added by Prim's algorithm (in the correct order) is $6, 3, 5, 9, 11, 17$.
- ☐ The weights of the edges added by Dijkstra's algorithm (in the correct order) is $6, 3, 5, 9, 11, 17$.
  The weights of the edges added by Prim's algorithm (in the correct order) is $5, 6, 8, 10, 11, 17$.
- ☐ The weights of the edges added by Dijkstra's algorithm (in the correct order) is $6, 8, 5, 10, 11, 17$.
  The weights of the edges added by Prim's algorithm (in the correct order) is $3, 5, 6, 9, 11, 17$.
- ☐ The weights of the edges added by Dijkstra's algorithm (in the correct order) is $6, 8, 5, 10, 11, 17$.
  The weights of the edges added by Prim's algorithm (in the correct order) is $6, 3, 5, 9, 11, 17$.

Second part, open questions

This part consists of four questions, each worth 17 points. Please follow the following instructions:

- Your explanations should be clear enough and in sufficient detail that a fellow student can understand them. In particular, do not only give pseudocode without explanations. A good guideline is that a description of an algorithm should be such that a fellow student can easily implement the algorithm following the description.
- You are allowed to refer to material covered in the lectures including algorithms and theorems (without reproving them). You are however *not* allowed to simply refer to material covered in exercises.
- Please answer all questions within the designated boxes (otherwise your answer may not be accurately scanned). At the end of the exam there are five extra pages if you need additional space for your answers.
- Leave the check-boxes empty, they are used for the grading.

Question 5: Recurrences. *(17 pts)*

☐0 ☐1 ☐2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9
☐10 ☐11 ☐12 ☐13 ☐14 ☐15 ☐16 ☐17

Do not write here.

Consider the following procedure UNKNOWN that takes as input an integer n:

```
UNKNOWN(n):
1. if n < 100
2.       return
3. q = ⌊n/5⌋
4. UNKNOWN(3 · q)
5. UNKNOWN(4 · q)
6. for i = 1, 2, ..., n
7.       PRINT "Almost done!"
```

Let $T(n)$ be the time it takes to execute UNKNOWN(n). Give the recurrence relation of $T(n)$ and prove that $T(n) = O(n^2)$. To simplify notation and calculations, you may assume that $n/5$ always evaluates to an integer.

(In this question you are asked to give the recurrence relation of $T(n)$ and to give a mathematically rigorous proof that $T(n) = O(n^2)$. Recall that you are allowed to refer to material covered in the lectures.)

Begin writing your answer on the next page.

Question 6: Check shortest path distances. *(17 pts)*

☐0 ☐1 ☐2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9
☐10 ☐11 ☐12 ☐13 ☐14 ☐15 ☐16 ☐17

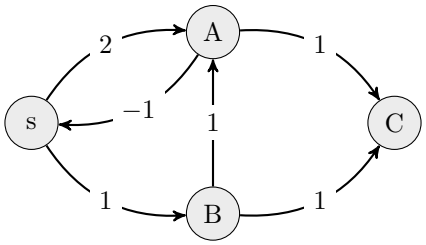
Do not write here.

Design and analyze an algorithm that **runs in linear time**, i.e., in time $O(|V| + |E|)$, for the following problem:

> **INPUT:** A directed graph $G = (V, E)$, a source $s \in V$, edge-weights $w : E \to \mathbb{Z}$, and a value $v.d \in \mathbb{Z}$ for each vertex $v \in V$.
>
> **OUTPUT:** Print "CORRECT" if $v.d$ equals the shortest path distance from s to v for every vertex $v \in V$; otherwise print "INCORRECT."

You may assume that every vertex is reachable from s and that the graph only contains non-negative cycles but it may contain negative edge weights.
We remark that there is no assumption on the values $v.d$. A solution that is fully correct under the additional assumption that, for every vertex $v \in V$, $v.d$ is at least the shortest path distance from s to v is rewarded 10 points.

Example: Consider the following input (the values $v.d$ are not depicted):



The output of your algorithm should be "CORRECT" if and only if $s.d = 0, A.d = 2, B.d = 1$, and $C.d = 2$.

(In this question you are asked to design and analyze an algorithm that runs in time $O(|V| + |E|)$ for the problem of verifying that the values $v.d$ equals the shortest path distance from s to v for every vertex $v \in V$. Recall that you are allowed to refer to material covered in the lectures.)

Question 7: Short walk of maximum value. *(17 pts)*

☐0 ☐1 ☐2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9
☐10 ☐11 ☐12 ☐13 ☐14 ☐15 ☐16 ☐17

Do not write here.

Consider a $(n+1) \times (n+1)$ grid of cells with coordinates (i, j) for $0 \leq i \leq n$ and $0 \leq j \leq n$. We are going to analyze walks from the bottom-left cell $(0, 0)$ to the top-right cell (n, n) where the available moves are:

UP: Standing in a cell (i, j) with $j < n$, we can go to $(i, j+1)$.

RIGHT: Standing in a cell (i, j) with $i < n$, we can go to $(i+1, j)$.

UP-RIGHT: Standing in a cell (i, j) with $i < n$ and $j < n$, we can go to $(i+1, j+1)$.

Notice that any such walk makes at least n moves and at most $2n$ moves. We say that the walk is **short** if it makes at most $3n/2$ moves. In addition, each cell (i, j) has a value $v(i, j)$, and the value of a walk is the total value of the cells it visited. Your task is to design and analyze an efficient algorithm that calculates the maximum value any short walk from $(0, 0)$ to (n, n) can achieve.
The running time of your algorithm should be polynomial in n. While your algorithm should run in polynomial time, it is not important that you achieve an optimal running time in this question. We also remark that partial credits will be given to solutions that output the maximum value that any (not necessarily short) walk can achieve.

Example: Consider the following $(n+1) \times (n+1)$ grid with $n = 2$:

$(0,2)$	$(1,2)$	$(2,2)$
$(0,1)$	$(1,1)$	$(2,1)$
$(0,0)$	$(1,0)$	$(2,0)$

The value of the gray cells is 1 and the value of the white cells is 0 (in this example cells take values 0 and 1 but in general cells can take any value). Hence, the maximum value of any walk from $(0, 0)$ to $(2, 2)$ is 3. This value is obtained by the walk that visits the cells $(0, 0), (1, 0), (2, 0), (2, 1), (2, 2)$. However, this walk makes $2n = 4$ moves and is thus not short. The maximum value of a short walk (that makes at most $3n/2 = 3$ moves) is 2. This is achieved by the short walk that makes 3 moves and visits the cells $(0, 0), (1, 0), (2, 1), (2, 2)$. On this input, the output of your algorithm should thus be 2.

(In this question you are asked to design and analyze an algorithm that runs in time polynomial in n for the problem of calculating the maximum value of any short walk from $(0, 0)$ *to* (n, n)*. Recall that you are allowed to refer to material covered in the lectures.)*

Begin writing your answer on the next page.

Question 8: Many spanning trees. *(17 pts)*

☐0 ☐1 ☐2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9
☐10 ☐11 ☐12 ☐13 ☐14 ☐15 ☐16 ☐17

Do not write here.

Design and analyze an efficient algorithm for the following problem:

> **INPUT:** A connected undirected graph $G = (V, E)$ with integer edge-weights $w : E \to \mathbb{Z}$.
>
> **OUTPUT:** Print "ONE" if G with edge-weights w has a unique minimum spanning tree; otherwise print "MANY."

The running time of your algorithm should be polynomial in $|V|$. While your algorithm should run in polynomial time, it is not important that you achieve an optimal running time in this question. For full credit, you should also explain in words why your algorithm is correct.

Example: Consider the following two inputs:

The output of your algorithm should be "MANY" on the left input since that graph contains four minimum spanning trees; on the right instance, the output should be "ONE" since that graph contains a unique minimum spanning tree.

(In this question you are asked to design and analyze an algorithm that runs in time polynomial in $|V|$ for the problem of verifying whether the graph $G = (V, E)$ with integer edge-weights w has a unique minimum spanning tree. While you do not need to give a rigorous proof of correctness, you should argue why your algorithm is correct in addition to analyzing its running time. Recall that you are allowed to refer to material covered in the lectures.)