EPFL

1

Teacher : Michael Kapralov
Algorithms CS-250 - SC
19 Jan 2023
Duration : 180 minutes

Student Sample

SCIPER : **325664**

Do not turn the page before the start of the exam. This document is double-sided, has 20 pages, the last ones possibly blank. Do not unstaple.

- 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 multiple choice questions and the second part consists of 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

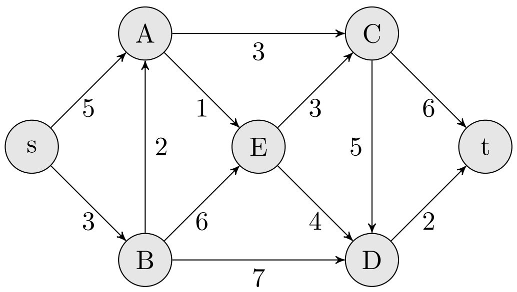
Corriger une réponse | Correct an answer
Antwort korrigieren

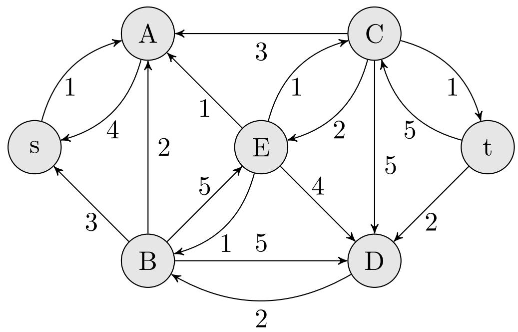
ce qu'il ne faut **PAS** faire | what should **NOT** be done | was man **NICHT** tun sollte

First part: multiple choice questions

For each question, mark the box corresponding to the correct answer. Each question has **exactly one** correct answer.

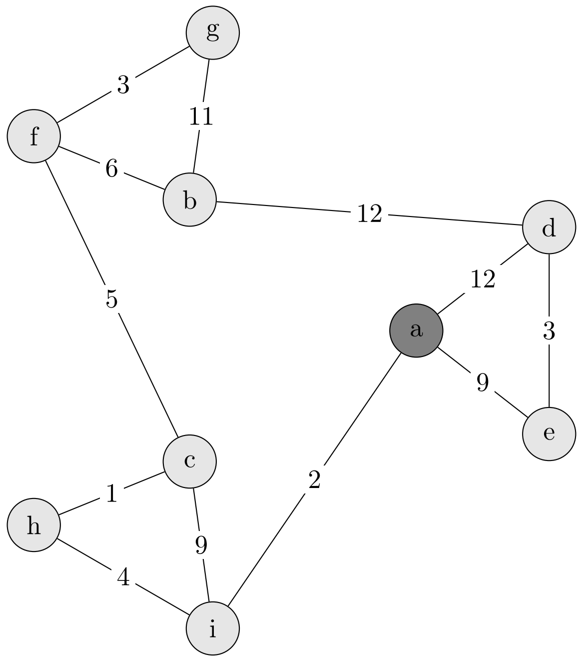
Question 1 : *(5 pts)* In this problem you are presented with a graph G with a source s and a sink t, as well as the residual graph corresponding to a maximum flow (see figure below). Select a minimum $s - t$ cut in the graph G from the options below:





- ☐ $\{s, A, B, C, D, E\}$
- ☐ $\{s, B, C, E\}$
- ☐ $\{s, A, B\}$
- ☐ $\{s, C, D, E\}$
- ☐ $\{s, A\}$
- ☐ $\{s, A, B, C, E\}$
- ☐ $\{s\}$
- ☐ $\{s, A, B, C, D, t\}$

Question 2 : *(5 pts)* Consider the graph below:



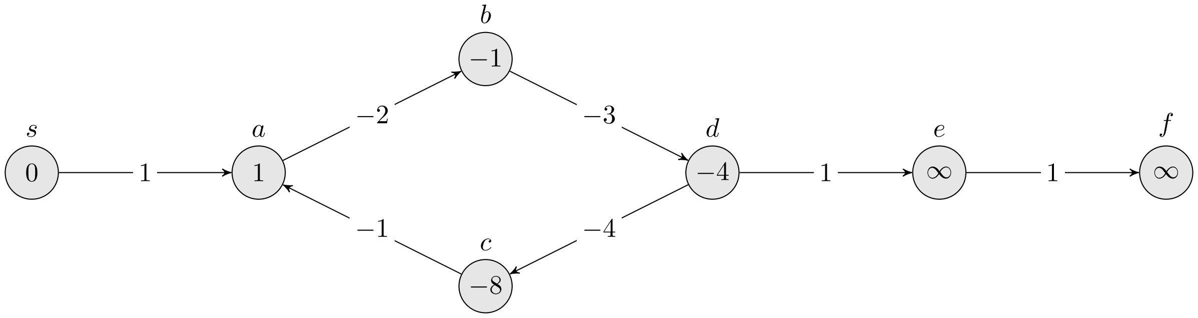
What is the sequence of weights of edges in the order they are added to the minimum weight spanning tree by **Prim's algorithm starting from vertex** a.

- ☐ $2, 9, 3, 4, 1, 5, 6, 3$
- ☐ $2, 4, 1, 5, 3, 6, 9, 3$
- ☐ $12, 11, 9, 12, 9, 6, 5, 4$
- ☐ $12, 9, 12, 11, 9, 6, 5, 4$
- ☐ $12, 12, 11, 9, 6, 5, 9, 4$
- ☐ $2, 3, 4, 1, 5, 6, 9, 3$
- ☐ $2, 5, 3, 4, 1, 9, 6, 3$

Question 3 : *(5 pts)* **Randomized QuickSort.** Let $A[1 \ldots 9] =$ | 5 | 1 | 6 | 7 | 2 | 9 | 3 | 8 | 4 | be an array consisting of 9 numbers. Which of the options below lists the elements of A in **non-increasing** order of the probability of being compared to element $A[9] = 4$ during the execution of RANDOMIZEDQUICKSORT on A? (Note that element 4 is never compared with itself, and is therefore the last element in all options below)

- ☐ $3, 5, 2, 6, 1, 7, 8, 9, 4$
- ☐ $2, 6, 8, 1, 3, 5, 7, 9, 4$
- ☐ $9, 8, 7, 6, 5, 3, 2, 1, 4$
- ☐ $8, 9, 6, 7, 5, 2, 3, 1, 4$
- ☐ $5, 6, 7, 8, 9, 3, 2, 1, 4$
- ☐ $1, 2, 3, 5, 6, 7, 8, 9, 4$

Question 4 : *(5 pts)* **Shortest paths.** Consider running the Bellman-Ford algorithm on the following graph from vertex s.



In the figure, each vertex v is labelled with its current shortest path estimate $\ell(v)$. What are the shortest path estimates of the vertices after relaxing the following edges (in the written order) $\text{Relax}(a, b, w)$, $\text{Relax}(b, d, w)$, $\text{Relax}(d, e, w)$, $\text{Relax}(e, f, w)$, $\text{Relax}(c, a, w)$, $\text{Relax}(a, b, w)$, $\text{Relax}(b, d, w)$, $\text{Relax}(e, f, w)$, $\text{Relax}(d, e, w)$?
Recall that Relax takes as input the endpoints of a directed edge and the weights w on the edges.

- ☐ $\ell(s) = 0, \ell(a) = 1, \ell(b) = -7, \ell(c) = -4, \ell(d) = -4, \ell(e) = -3, \ell(f) = -2$
- ☐ $\ell(s) = 0, \ell(a) = 1, \ell(b) = -1, \ell(c) = -8, \ell(d) = -4, \ell(e) = \infty, \ell(f) = \infty$
- ☐ $\ell(s) = -1, \ell(a) = -9, \ell(b) = -8, \ell(c) = -8, \ell(d) = -4, \ell(e) = -3, \ell(f) = -2$
- ☐ $\ell(s) = 0, \ell(a) = -9, \ell(b) = -11, \ell(c) = -8, \ell(d) = -14, \ell(e) = -13, \ell(f) = -2$
- ☐ $\ell(s) = 0, \ell(a) = -9, \ell(b) = -8, \ell(c) = -8, \ell(d) = -4, \ell(e) = -3, \ell(f) = -2$
- ☐ $\ell(s) = 0, \ell(a) = 1, \ell(b) = -8, \ell(c) = -8, \ell(d) = -4, \ell(e) = -3, \ell(f) = -2$
- ☐ $\ell(s) = -8, \ell(a) = -9, \ell(b) = -8, \ell(c) = -8, \ell(d) = -4, \ell(e) = -3, \ell(f) = -2$
- ☐ $\ell(s) = 0, \ell(a) = -8, \ell(b) = -11, \ell(c) = -7, \ell(d) = -14, \ell(e) = -15, \ell(f) = -2$

Collisions. Suppose that we hash integers $\{1, 2, 3, \ldots, n\}$ into a table with m slots using a hash function $h : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, m\}$. Assume simple uniform hashing, i.e. that h is chosen uniformly at random. A collision is a pair of distinct integers $1 \leq i < j \leq n$ hashing into the same slot. In other words, i and j participate in a collision if and only if $h(i) = h(j)$.

Question 5 : *(4 pts)*
What is the expected number of collisions as a function of n and m?

- ☐ $\Theta(\sqrt{n/m})$
- ☐ $\Theta(m^2/n^2)$
- ☐ $\Theta(m^2/n)$
- ☐ $\Theta(m)$
- ☐ $\Theta(n)$
- ☐ $\Theta(\sqrt{nm})$
- ☐ $\Theta(\sqrt{m})$
- ☐ $\Theta(n^2/m)$
- ☐ $\Theta(n^2/m^2)$
- ☐ $\Theta(\sqrt{n^2/m})$
- ☐ $\Theta(\sqrt{n})$

Question 6 : *(4 pts)* What is the expected number of collisions among elements that hash to the **first hash table slot**?

- ☐ $\Theta(\sqrt{n^2/m})$
- ☐ $\Theta(n)$
- ☐ $\Theta(m)$
- ☐ $\Theta(\sqrt{n})$
- ☐ $\Theta(n^2/m^2)$
- ☐ $\Theta(\sqrt{m})$
- ☐ $\Theta(m^2/n)$
- ☐ $\Theta(\sqrt{nm})$
- ☐ $\Theta(m^2/n^2)$
- ☐ $\Theta(\sqrt{n/m})$
- ☐ $\Theta(n^2/m)$

Second part, open questions

This part consists of four questions, each worth 18 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 extra pages if you need additional space for your answers.
- Leave the check-boxes empty, they are used for the grading.

Question 7: Augmented data structure. *(18 pts)*

☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 ☐ 8 ☐ 9
☐ 10 ☐ 11 ☐ 12 ☐ 13 ☐ 14 ☐ 15 ☐ 16 ☐ 17 ☐ 18 *Do not write here.*

In this problem your task is to design an augmented stack data structure *MinStack* that supports all the stack operations, i.e. PUSH(), POP(), as well as an additional operation GETMIN() that returns the minimum element from the stack. All these operations of *MinStack* must have a time complexity of $O(1)$.

In this question you should **design** a data structure and **analyze** the runtime of PUSH(), POP() and GETMIN().

Hint: for example, this can be done using two stacks.

Question 8: Longest Modular Increasing Subsequence. *(18 pts)*

0 1 2 3 4 5 6 7 8 9
10 11 12 13 14 15 16 17 18

Do not write here.

In this problem you are given an array A of integers, and your task is to calculate the length of the longest *modular* increasing subsequence of the array. For an integer $k \geq 1$ we call a subsequence $1 \leq i_1 < \ldots < i_k \leq n$ of A a *modular* increasing subsequence if $A[i_1] < A[i_2] < \ldots < A[i_k]$ and the sum $\sum_{s=1}^{k} A[i_s]$ is divisible by 3.

Design an efficient algorithm for the following problem:

Input:	An array A of n integers.
Output:	The length of the longest modular increasing subsequence of A.

For example, on input $A = [5, 1, 2, 7, 5, 3, 6]$ your algorithm must output 4 (a maximizing subsequence is $i_1 = 2, i_2 = 3, i_3 = 6, i_4 = 7$).

Design a polynomial time algorithm for this problem and **analyze** its runtime.

Hint: for $b \in \{0, 1, 2\}$ let $r[j, b]$ denote the length of the longest increasing subsequence $1 \leq i_1 < \ldots < i_k = j$ of A that ends with j and whose sum is congruent to b modulo 3, i.e $\sum_{s=1}^{k} A[i_s] \pmod 3 = b$. Derive a recurrence for $r[j, b]$ and use it to design a dynamic programming solution.

Question 9: MST query. *(18 pts)*

0 1 2 3 4 5 6 7 8 9
10 11 12 13 14 15 16 17 18 *Do not write here.*

In this problem you are given a weighted graph $G = (V, E, w)$ with n vertices and m edges, as well as an edge $e \in E$. Your task is to design an efficient algorithm for checking whether there exists a minimum spanning tree (MST) of G that contains e.

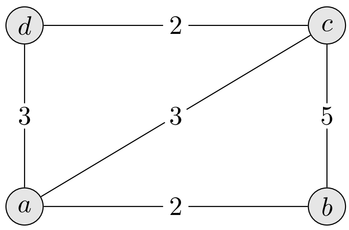
Design an efficient algorithm for the following problem:

Input:	A weighted graph $G = (V, E, w)$ with n vertices and m edges, and an edge $e \in E$.
Output:	YES if there exists a minimum spanning tree of G that contains e and NO otherwise.

In this problem you must **design** the algorithm, **analyze** its runtime and **prove** its correctness.

For full credit your solution should run in time $O(n+m)$. Slower solutions may receive some partial credit. You may assume that the graph G is connected and the weights of its edges are positive integers.

For example, consider the following graph G.



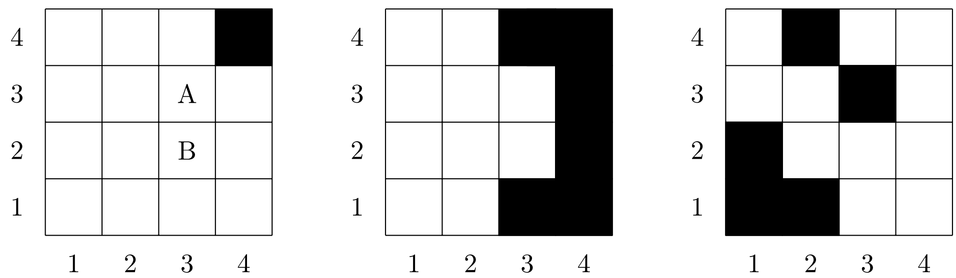
Here, for both edges (a, d) and (a, c) there exists an MST containing them, even though there is no MST containing both of them at the same time. Hence, the answer on inputs $G, (a, d)$ and $G, (a, c)$ is YES. Edge (b, c) is not part of any MST, so the answer on the input $G, (b, c)$ is NO.

Question 10: Escape from a haunted house. *(18 pts)*

0 1 2 3 4 5 6 7 8 9
10 11 12 13 14 15 16 17 18 *Do not write here.*

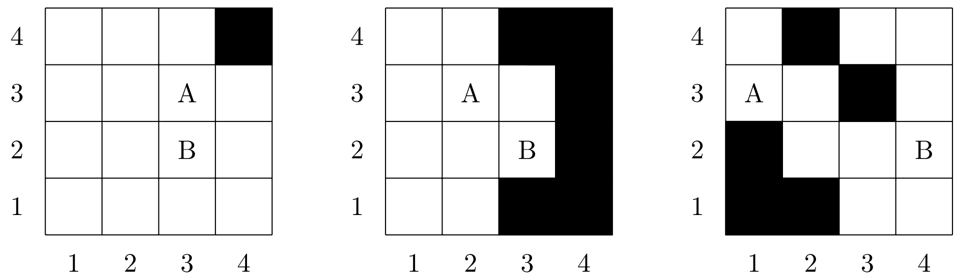
Suppose that k friends start at distinct locations in a haunted house (represented by an $n \times n$ grid) and need to get out (to the boundary of the grid) in T steps. The house is haunted: ghosts appear at various places in the house (cells in the grid) at times $t = 1, 2, \ldots, T$. For every $t = 1, 2, \ldots, T$ every one of the k friends can either move to an adjacent cell (up/down/left/right) or stay in the same cell. No two friends should occupy the same cell at any point in time, and none of the friends should ever move to a cell that contains a ghost at the corresponding point in time. Your task is: given the original positions of the k friends and positions in which ghosts appear at time $t = 1, 2, \ldots, T$, determine if the k friends can escape in T steps.

Consider the instance below, where $n = 4$, $T = 3$ and $k = 2$. The two friends, Alice and Bob, start at time $t = 1$ in cells $(3, 3)$ and $(3, 2)$ respectively (these cells are marked A and B). The cells occupied by ghosts at time $t = 1, 2, 3$ are shown as black.



time $t = 1$ time $t = 2$ time $t = 3$

In the example below Alice and Bob can reach the boundary without ever occupying the same cell as a ghost and without colliding in any cell. A possible solution is shown below. In this solution Alice moves to the left at time 2, while Bob stays in the same cell. Then Alice moves left again and Bob moves right at time $t = 3$, reaching the boundary, as required.



time $t = 1$ time $t = 2$ time $t = 3$

> **Input:** The size n of the grid, the number of time steps T, the number k of friends and their distinct locations $(x_i, y_i), i = 1, \ldots, k$, where $1 \le x_i, y_i \le n$ are integers. For every $t = 1, \ldots, T$ an $n \times n$ matrix G^t, where $G^t_{ij} = 1$ if there is a ghost at time t in cell (i, j) and $G^t_{i,j} = 0$ otherwise.
>
> **Output:** YES if all friends can escape and NO otherwise.

Design an algorithm for this problem with runtime polynomial in n, T and k.

Hint: convert the input instance into a graph and use max-flow.