ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Final Exam, Algorithms 2016-2017

- You are only allowed to have a handwritten A4 page written on both sides.
- Communication, calculators, cell phones, computers, etc... are not allowed.
- 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.
- Attached at the end of the exam is a French translation.
- **Do not touch until the start of the exam.**

Good luck!

Name: _____ **N° Sciper:** _____

Problem 1 / 23 points	Problem 2 / 12 points	Problem 3 / 16 points	Problem 4 / 18 points	Problem 5 / 19 points	Problem 6 / 12 points

Total / 100

1 *(23 pts)* **Basic questions.** This problem consists of five subproblems (1a-1e) for which you **do not** need to motivate your answers.

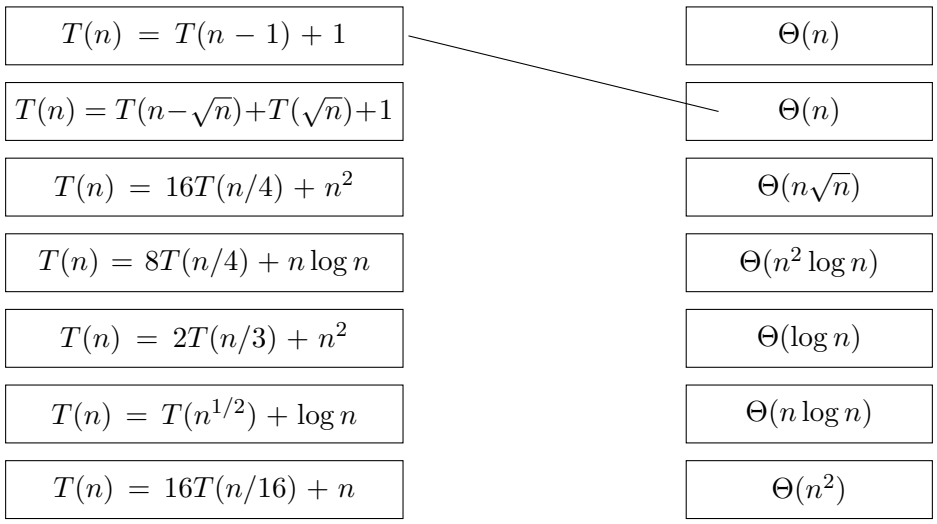
1a *(5 pts)* Consider the task of hashing n keys into m different hash table slots $\{1, 2, \ldots, m\}$. We assume simple uniform hashing. Define a 2-*collision* as a pair of distinct keys $\{k_1, k_2\}$ that are hashed to the same hash table slot. What is the largest value of n (asymptotically as a function of m) which ensures that the expected number of 2-collisions is less than $1/10$?

Solution:

The answer is _____

1b *(6 pts, 1 pt for each correct pair)* Pair up each of the recurrence relations on the left side with the correct asymptotic growth function on the right side (assuming that $T(1) = 1$). For example the recurrence $T(n) = T(n-1) + 1$ has asymptotic growth $\Theta(n)$.

Solution:



1c *(6 pts)* Consider the code for the function UNKNOWN below:

```
UNKNOWN(n):
1. if n < 100
2.       return
3. q = ⌊n/4⌋
4. UNKNOWN(q)
6. UNKNOWN(n − 2q)
7. PRINT "Almost done!"
8. UNKNOWN(n − 2q)
```

Let $T(n)$ denote the runtime of UNKNOWN(n). Write down a recurrence relation for $T(n)$:

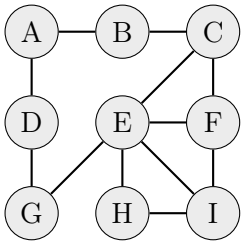
Solution: $T(n) =$ _____ *(3pts)*

Which of the following statements about the asymptotic growth of $T(n)$ as a function of n are true:

A: $T(n) = \Omega(n)$
B: $T(n) = O(n^2)$
C: $T(n) = \Omega(n^3)$
D: $T(n) = O(n)$

Solution: The correct statements are _____ *(3pts)*

1d *(3pts, 1.5pts for correct BFS and 1.5pts for correct DFS)* Consider the following undirected graph:



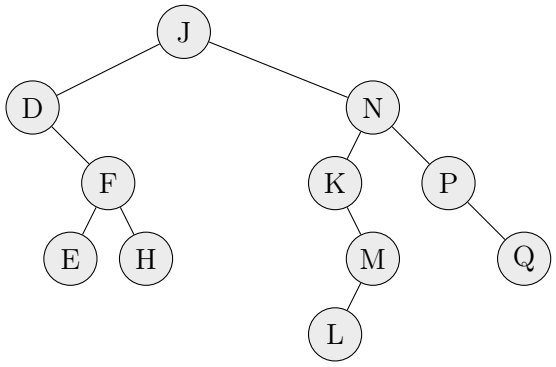
Write down the labels of the vertices in the order in which they are discovered by traversing the graph using breadth-first-search (BFS) and using depth-first-search (DFS), starting from the vertex labeled by A. Assume that the vertices are chosen in alphabetical order: that is, whenever the BFS/DFS procedure has several options, it chooses the next vertex according to the alphabetical order.

Solution:

The order in which vertices are discovered by BFS: _____

The order in which vertices are discovered by DFS: _____

1e *(3 pts)* Draw the binary search tree obtained by deleting the root from the following binary search tree:



The deletion is done by the procedure TREE-DELETE explained in class.

Solution:

2 *(12 pts)* **Angry birds.**

A number k of birds live in a city, and need to get from their homes to their common workplace every morning (the birds choose to walk to work in the morning, as they only get their first cup of coffee at work and are not sufficiently alert for flying before then). They also insist that the paths they take to work every morning do not overlap except possibly on road intersections – they are somewhat grumpy before their morning coffee, and get angry if there is any delay on the narrow streets.

In this problem you are given the map of streets of the city (i.e. a directed graph where edges are the streets and vertices are the street intersections), the k distinct locations (vertices) $s_i, i = 1, \ldots, k$, where the birds live, and the workplace t (a vertex). Your task is to find a set of k paths, one path from each s_i to t, so that no two paths share an edge, and output `NONE` if such a set of paths does not exist. Your solution should run in time $O((m + k + n)k)$ where m is the number of roads in the city and n is the number of intersections.

(In this problem you are required to (a) explain a correct algorithm with the desired running time and to (b) analyze its running time.)

Solution:

3 *(16 pts)* **Maintaining the median of a sequence.** Design a data structure that supports the following two operations: (a) insert a number and (b) return the median of numbers inserted so far. All operations should be performed in time $O(\log n)$, where n is the number of elements currently stored in the data structure. *Hint: use two heaps.*

Recall that the median of n numbers $a_1 \leq a_2 \leq \cdots \leq a_n$ is $a_{\lceil n/2 \rceil}$ if n is odd and $\frac{a_{(n/2)}+a_{(n/2+1)}}{2}$ if n is even.

(In this problem you are required to (a) explain how the numbers are stored in the data structure, to (b) explain the implementations of the "insert" and "median" operations with running time $O(\log n)$, and to (c) analyze the running time of your algorithms.)

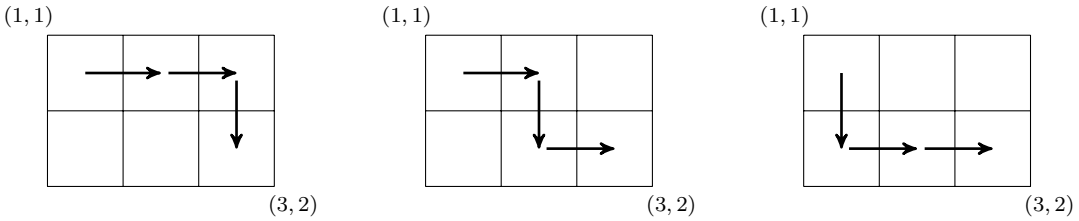
Solution:

Continuation of the solution to 3:

4 *(18 pts)* **Rectangular city.** Bob lives in a two-dimensional n by m city. He lives in cell $(1, 1)$ and works in cell (n, m). He wants to get to work by passing through the city. The time he has for getting to work is T, and entering any cell (i, j) takes t_{ij} time. Also, there are c_{ij} candies in each cell (i, j) and he can collect them if he enters the cell. Bob can only go right or down through the city. **Design and analyze** an algorithm that finds the maximum number of candies that Bob can collect while still getting to work on time.

- **Input:** The input consists of integers n, m, T and t_{ij}, c_{ij} for all $1 \leq i \leq n$, $1 \leq j \leq m$.
- **Output:** Output the maximum number of candies that Bob can collect while still getting to work on time. If there is no way that he can get to work on time, output 0.
- He can move from cell (i, j) to cells $(i + 1, j)$ and $(i, j + 1)$.
- All the t_{ij}, c_{ij} values are positive integers.

Here are three examples of paths that Bob can take from his home to work in a 2 by 3 city.



Your solution should run in time polynomial in n, m and T.

(In this problem you are required to (a) explain a correct algorithm with the desired running time and to (b) analyze its running time.)

Solution:

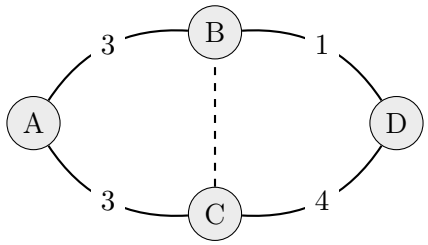
Continuation of the solution to 4:

5 *(19 pts)* **Minimum spanning trees.** In this problem you are given an instance to the minimum spanning tree problem (i.e., a connected undirected graph with edge weights). You are further considering adding a new edge $e = \{u, v\}$ to the graph, and you want to assign it the largest possible weight while maintaining that it is part of every minimum spanning tree. More formally, your task is to **design and analyze** an efficient algorithm for the following problem:

> Given an undirected graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}$ and an edge $e = \{u, v\} \notin E$, return the maximum weight α such that if $w(e) < \alpha$ then *every* minimum spanning tree of $(V, E \cup \{e\})$ contains e.

For full credit, your algorithm should run in time $O((|V| + |E|) \log |V|)$.

Example: Consider the following input where solid edges are those in E and the dashed edge is e.



The correct output of the algorithm is $\alpha = 3$ because of the following:

- If $w(e) < 3$, the minimum spanning trees of $(V, E \cup \{e\})$ are $\{\{B, D\}, \{B, C\}, \{A, B\}\}$ and $\{\{B, D\}, \{B, C\}, \{A, C\}\}$.
- If $w(e) = 3$, a minimum spanning tree of $(V, E \cup \{e\})$ is $\{\{B, D\}, \{A, B\}, \{A, C\}\}$ (so 3 is the maximum possible weight).

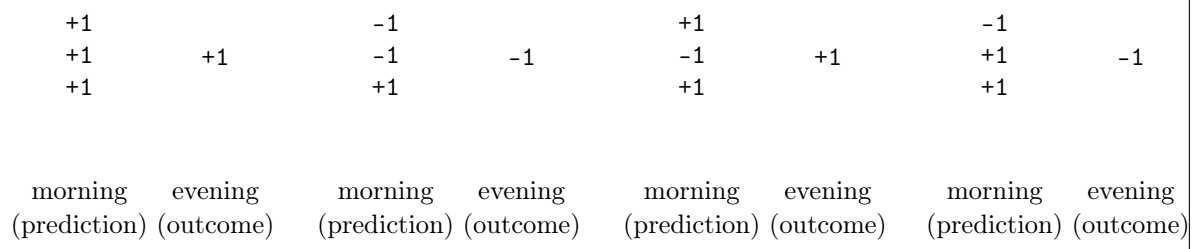
(In this problem you are required to (a) explain a correct algorithm with the desired running time, to (b) **prove that your algorithm outputs the correct value***, and to (c) analyze the running time of your algorithm.)*

Solution:

Continuation of the solution to 5:

6 *(12 pts)* **Investment strategy with advice.** In this problem you will design a strategy for playing a stock price prediction game with the help of n experts. Every morning each of the n experts makes a prediction: `+1` for **up** or `-1` for **down**, and based on their advice you make your own prediction (i.e., you output `+1` or `-1`). In the evening the true answer is revealed: `+1` if the stock actually went up and `-1` if the stock went down. It is guaranteed that one of the experts is perfect, i.e., gives the right answer every day, but other experts may behave arbitrarily and you do not know which of the experts is the perfect one. Show that it is possible to play the prediction game for any number T of rounds so as to output the wrong prediction at most $\lfloor \log_2 n \rfloor$ times.

An example timeline over $T = 4$ days with $n = 3$ experts is as follows:



Notice that after the first day, all experts look like they could be the perfect one. On the second day, the prediction of the third expert is wrong, so we then know that the perfect expert can only be the first or the second. By the same argument, after day 3 we know that the perfect expert is the first one. So after day 3 it is easy to always make the correct prediction. However, we are only allowed to make at most $\lfloor \log_2(n) \rfloor = 1$ erroneous prediction in total, meaning that we also have to perform fairly well even before we have identified the perfect expert (who may not be unique in general).

(In this problem you are required to (a) explain a strategy with the desired performance and to (b) analyze the number of wrong answers that your strategy can give in the worst case, i.e., prove that your algorithm only makes at most $\lfloor \log_2(n) \rfloor$ mistakes.)

Solution:

Continuation of the solution to 6: