Theoretical Computer Science Preliminary Exam

First name: _____ Student ID: _____

Last name: _____ Additional papers: _____

Instructions.

- The exam lasts 3 hours.
- Each problem is worth 10 points.
- Sign each additional page you hand in. Please do not use the same page for different problems.
- You may use standard algorithms as black boxes only if you clearly state the relevant guarantee and running time.

Good luck!

Problem 1. Short questions

Please answer the following questions. **No justifications are needed.**

(a) State whether the following statement is True or False.

There exists a connected graph on 2026 vertices and 2026 edges that contains at least two cycles.

True	False

(b) State whether the following statement is True or False.

Consider a connected weighted undirected graph, where edge weights can also be negative. If we run Prim's algorithm on this graph, then Prim's algorithm correctly outputs a minimum-cost spanning tree of the input graph.

True	False

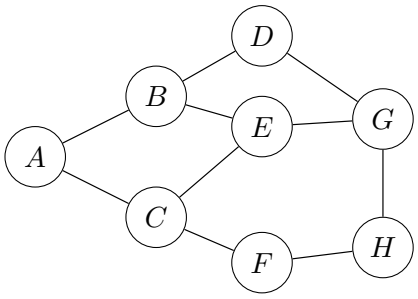
(c) You are given a sorted array A of length n, and you run binary search to check whether A contains a given element x. What is the worst-case running time of binary search?

$\Theta(1)$ $\quad \Theta(\log n)$ $\quad \Theta(\sqrt{n})$

$\Theta(n)$ $\quad \Theta(n \log n)$ $\quad \Theta(n^2)$

Problem 2. DFS and Kruskal's algorithm

(a) Consider the following undirected graph. The adjacency list of each vertex is sorted alphabetically.



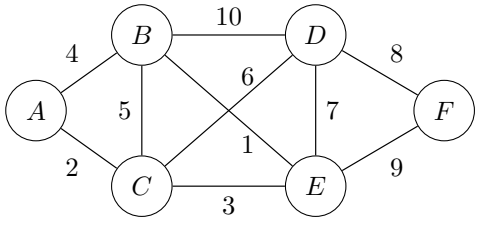
The adjacency lists are:

A	B, C
B	A, D, E
C	A, E, F
D	B, G
E	B, C, G
F	C, H
G	D, E, H
H	F, G

Run Depth-First Search (DFS) starting from vertex C; the neighbors are visited in the order given by the adjacency list. Write the order in which vertices are first discovered.

Discovery step	1	2	3	4	5	6	7	8
Vertex								

(b) Kruskal's algorithm is an algorithm for computing a minimum spanning tree of a connected weighted undirected graph. Consider the following connected weighted undirected graph.



Run Kruskal's algorithm on this graph. Write the edges in the order in which Kruskal's algorithm adds them to the minimum spanning tree.

Edges added, in order: _____

Problem 3. Maximum-value common subsequence

You are given two input strings S and T, each of length at most n. Each string consists of letters from the English alphabet. For every character c, you are also given a positive integer value $v(c)$.

A common subsequence of S and T is a string that can be obtained from S by deleting zero or more characters, and can also be obtained from T by deleting zero or more characters. The remaining characters must stay in their original order, but they do not need to be consecutive.

The value of a common subsequence is the sum of the values of its characters. Design an algorithm that outputs the maximum possible value of a common subsequence of S and T. If there is no common non-empty subsequence, the algorithm should output 0.

Example: If $S =$ `CCA`, $T =$ `ACC`, $v(A) = 6$, and $v(C) = 2$, then the correct output is 6. Indeed, one can match the single character A, which has value 6, while matching the two characters `CC` has value only 4.

For full score, your algorithm should run in $O(n^2)$ time. Analyze the running time and briefly justify correctness. Informal and brief, but informative, arguments for correctness will also receive full credit.

Problem 4. Undecidability

Prove that the following problem is undecidable: given two Turing machines A and B, is it the case that

$$L(A) = \overline{L(B)}?$$

Here $L(A)$ is the set of strings accepted by A.

Prove this by reduction from the "no-input halting problem": given a Turing machine M, does $M(\epsilon)$ halt, where ϵ is the empty string?

Problem 5. SAT-1-in-3

SAT-1-in-3 is a variant of 3SAT with no negations, and each clause is satisfied if exactly one of its three variables is true. Such an assignment is called a *1-in-3 satisfying assignment*. For example,

$$\varphi = (x \vee y \vee z) \wedge (x \vee a \vee b) \wedge (z \vee b \vee x)$$

is 1-in-3-satisfied by

$$x = 0, \qquad y = 1, \qquad z = 0, \qquad a = 0, \qquad b = 1.$$

Prove that SAT-1-in-3 is NP-complete.

Hint. To a given 3CNF formula with n variables, add n new variables intended to be the negations of the original variables. To do this, add two new variables t and f, and a clause where any 1-in-3 satisfying assignment assigns t to 1 and f to 0. Then use t and f to help add clauses that force the other new variables to be negations of the original variables.

Also observe that for existing variables x and y, with a new dummy variable i appearing only in one clause, the clause

$$(x \vee \text{notY} \vee i),$$

where notY is the new variable that, in any 1-in-3 satisfying assignment, is forced by other clauses to be the negation of the original variable y, enforces the implication $x \Rightarrow y$ in any 1-in-3 satisfying assignment.

Finally, show how to add clauses so that for a clause $(x \vee y \vee z)$ in φ, where x, y, z are literals, the clause $(c_1 \vee c_2 \vee c_3)$, where c_1, c_2, c_3 are new unnegated variables, is 1-in-3 satisfied if and only if the following implications hold:

$$c_1 \Rightarrow x, \qquad c_2 \Rightarrow y, \qquad c_3 \Rightarrow z.$$

Problem 6. Relativized complexity classes

Prove that if A is any PSPACE-complete language, then

$$\mathrm{P}^A = \mathrm{NP}^A.$$

Recall that for a complexity class $\mathcal{C}$ and language A, the class $\mathcal{C}^A$ is the set of problems decidable by machines defining $\mathcal{C}$ that have oracle access to the language A, i.e., the ability to query in $O(1)$ time whether a given string x is in A.

Hint. Show that both P^A and NP^A are equal to PSPACE.