- This exam has 5 questions for a total of 150 marks, of which you can score at most 100 marks.
- The deadline for uploading your answers is 13:00 hours on September 11, 2020.
- You may answer any subset of questions or parts of questions. All answers will be evaluated.
- You are free to refer to the lectures/your notes/any other material that you wish, but you are **not** permitted to confer with one another in any manner.
- Warning: CMI's academic policy regarding cheating applies to this exam.

Unstated assumptions and lack of clarity in solutions can and will be used against you during evaluation. You may freely refer to statements from the lectures in your arguments. You don't need to reprove these unless the question explicitly asks you to, but you must be precise. That is, "As we saw in class ... " will not get you any credit, but "As we saw in Exercise x of Lecture y" will. Please feel free to ask me via the designated channels if you have questions about the questions.

1. This problem is about generating powersets without repetition.
   (a) Write pseudocode for a procedure which accepts an array $A$ of $n \geq 0$ distinct integers as input, runs in $\mathcal{O}(c^n)$ time for some constant c, and prints out each subset of these integers exactly once. Argue why your procedure is correct. **You will get the credit for this part only if both your pseudocode and your reasoning are correct.** [20]

   (b) Argue that your procedure runs in $\mathcal{O}(c^n)$ time for some constant $c$. [10]

2. Refer to the function DFSAll on page 3 of Lecture 16. Let $D$ be a directed graph, and let $Pre, Post$ be the two arrays computed by the call DFSAll$(D)$.
   (a) Prove that for any two vertices $u, v$ in $D$, the intervals $[Pre[u], Post[u]]$ and $[Pre[v], Post[v]]$ are either disjoint, or nested. [10]

   (b) Prove that the following statements are equivalent for any two vertices $u, v$ in $D$: [10]
      1. $[Pre[u], Post[u]]$ contains $[Pre[v], Post[v]]$.
      2. DFS$(v)$ is invoked during the execution of DFS$(u)$. That is, after DFS$(u)$ has been called, and before it has returned.
      3. Vertex $u$ is an ancestor of vertex $v$ in the DFS forest of $D$ which is (implicitly) computed by the call DFSAll$(D)$.

(c) Recall that a vertex $v$ is said to be *active* at some specific point in time during the invocation of DFSALL(D) if the call DFS($v$) has been made, but has not returned. Prove that at any point in time during the invocation of DFSALL(D), the set of active vertices at that point are part of a directed path in digraph D. [10]

3. Refer to the procedure PRIMSALGORITHM on page 8 of Lecture 21. Recall that the *amortized* running times of the various operations on the Fibonacci heap, when used as a priority queue in Prim's algorithm on graph G, are as follows: [30]

   1. ELEMENTOF: $\mathcal{O}(1)$
   2. INSERT: $\mathcal{O}(1)$
   3. MINIMUM: $\mathcal{O}(1)$
   4. EXTRACT-MIN: $\mathcal{O}(\log_2 |V(G)|)$
   5. DECREASE-KEY: $\mathcal{O}(1)$

   Prove that if the priority queue is implemented using a Fibonacci heap then PRIMSALGORITHM($G, w$) runs in $\mathcal{O}(|E(G)|+|V(G)|\log_2 |V(G)|)$ time in the **worst** case.

4. Write pseudocode for an algorithm which takes the adjacency list of an undirected graph G as input, runs in time $\mathcal{O}(|V(G)| + |E(G)|)$, and computes the following: [30]

   1. The number of cc of connected components of G.
   2. An array $CC[1, 2, \ldots, |V(G)|]$ of integers where
      - Each element of CC is a number between 1 and cc, both inclusive;
      - For any two vertices $v_i \neq v_j$ of G, $CC[i] = CC[j]$ if and only if $v_i$ and $v_j$ belong to the same connected component of G.

   Argue why your algorithm correctly computes all these things. **You will get the credit for this problem only if both your pseudocode and your reasoning are correct.**

5. Let D be a directed graph on $n$ vertices and $m$ edges, given as its adjacency list. Let Pre, Post be the two arrays computed by a call to DFSALL(D) (Lecture 16, page 3).

   (a) Write pseudocode for an algorithm that takes D, Pre, and Post as input, runs in $\mathcal{O}(n + m)$ time, and outputs whether D contains a directed cycle using the information contained in Pre and/or Post. You will *not* get any credit for presenting pseudocode similar to the procedure ISACYCLIC from the lectures, which does not refer to Pre or Post. [10]

   (b) Prove that your algorithm from part (a) is correct, and that it runs in $\mathcal{O}(n + m)$ time. [20]