ECE 406, Winter 2021, Final Exam

- The exam will be available at 00:01 ET Monday, April 19. It is due by 23:59 ET Friday, April 23. I will use Crowdmark.
- There are 6 problems in total. Every problem is worth the same, with an equal split across sub-parts if there are any. Problems (2) and (4) are the only ones with sub-parts. Problem (2) has sub-parts (a)–(c), and Problem (4) has sub-parts (a) and (b).
- Every problem will be marked.
- No questions on the content of the exam nor the format are allowed. I will configure Piazza to be read-only during the time of the exam. If you feel you need to make an assumption, or there is a bug in the problem, state it clearly up-front in what you turn in.
- No collaboration with another human being, whether that person is in the course or not, is allowed.
- You are allowed to freely consult any other sources (e.g., course materials, online discussion forums) in a "read only" way. As always, you should cite and briefly credit any source you use. Of course you need to vet any information you lift from such sources; there is a lot of crap information out there on this topic.
- For brevity of exposition, you are allowed to simply mention any algorithm/definition/theorem we have discussed as part of the course without copying and writing it down again in your solutions. (E.g., "by Definition 5 of Lecture Notes 9(c), ...," or "We first run Breadth First Search (BFS) to identify whether the graph is connected...")
- Your brevity of exposition is very important. The lengthier your response, the more likely you'll lose points for poor exposition.
- The problems are ordered in what I think is non-decreasing difficulty.

1. Recall that an *independent set* of an undirected graph $G = \langle V, E \rangle$ is a subset of the vertices $I \subseteq V$ with the property: $u, v \in I \implies \langle u, v \rangle \notin E$.

   Alice claims that the problem of computing an independent set of maximum size possesses the following optimal substructure. If I is an independent set of maximum size in G, then for any $u \in I$, $I \setminus \{u\}$ is an independent set of maximum size in G', where G' is G with u and the edges incident on u removed.

   Show via counterexample that Alice's claim is not true.

2. Consider the problem in which we are given as input a positive integer n, and want to compute $1^2+2^2+\ldots+n^2$. Following is a candidate algorithm for this. It is first invoked as $\textsc{SqSum}(1, n)$.

   $\textsc{SqSum}(p, q)$

   if $p = q$ **then return** $p \times p$
   else
   $\quad m \leftarrow \left\lfloor \frac{p+q}{2} \right\rfloor$
   $\quad l \leftarrow \textsc{SqSum}(p, m)$
   $\quad r \leftarrow \textsc{SqSum}(m+1, q)$
   $\quad$ **return** $l + r$

   (a) Is $\textsc{SqSum}$ correct? Whether you say 'yes' or 'no,' give a <u>short</u> proof.

   (b) Notwithstanding whether $\textsc{SqSum}$ is correct, is $\textsc{SqSum}$ a polynomial-time algorithm? Why or why not?

   (c) Give a characterization of the space-efficiency of $\textsc{SqSum}$.

3. Devise an algorithm for the following decision problem. Your algorithm should certainly run in time $O(n^2 \log n)$. For full points, it should run in time $O(n^2)$.

   Given as input: (i) an undirected graph $G = \langle V, E, l \rangle$ where $l \colon E \to \mathbb{Z}^+$, and, (ii) $T \subseteq E$ a minimum spanning tree of G, is T unique? That is, your algorithm should return `true` if G has no minimum spanning tree other than T, and `false` otherwise.

   Your mindset behind an algorithm and its correctness are more important than pseudo-code. Your exposition needs to be crisp and terse for full points.

4. Assume that checking whether an input integer ≥ 2 is prime can be done in polynomial-time. Prove that each of the following problems is in **NP**.

   (a) Given as input two integers a, b such that $2 < b < a$, does there exist $c \in \{2, 3, \ldots, b\}$ that divides (i.e., is a factor of) a?

   (b) Given as input two integers a, b such that $2 < b < a$, does no $c \in \{2, 3, \ldots, b\}$ divide a?

5. You have access to a method FairCoinToss which, with time-efficiency $\Theta(1)$ per invocation, returns *tails* or *heads* with equal probability. Devise an algorithm that given input $\langle a, b\rangle$ where a and b are both positive integers with $a < b$, returns an integer $i \in \{a, a+1, \ldots, b\}$ with i equally likely to be any of those values.

6. Recall the algorithm Split from Lecture 3(c). Assume a version of Split that returns the new index of the pivot it chose within. It is now possible to devise a sorting algorithm, Quicksort. It is first invoked as Quicksort$(A, 1, n)$, where A is an array of n items indexed $1, \ldots, n$.

```
Quicksort(A, p, r)
    if p < r then
        q ← Split(A, p, r)
        Quicksort(A, p, q − 1)
        Quicksort(A, q + 1, r)
```

   Suppose, in the input array $A[1, \ldots, n]$, the entries are all distinct. Note that A is not necessarily sorted at the time it is input to Quicksort. Suppose the items in A, once they are sorted, are $s_1, \ldots, s_n$. That is, $s_1 < s_2 < \ldots < s_n$, and every $s_i \in A[1, \ldots, n]$.

   Prove: for every $S_{i,j} = \{s_i, s_{i+1}, \ldots, s_j\}$, where $i < j$, some member of $S_{i,j}$ is guaranteed to be chosen as pivot at some point in a run of Quicksort$(A, 1, n)$.