- Homework solutions should be neatly written or typed and turned in through **Gradescope** by 11:59 pm on the due date. No late homework will be accepted for any reason. You will be able to look at your scanned work before submitting it. Please ensure that your submission is legible (neatly written and not too faint), or your homework may not be graded.
- Students should consult their textbook, class notes, lecture slides, instructor, and TAs when they need help with homework. Students should not look for answers to homework problems in other texts or sources, including the internet. Only post about graded homework questions on Piazza if you suspect a typo in the assignment or if you don't understand what the question is asking you to do.
- Your assignments in this class will be evaluated not only on the correctness of your answers but on your ability to present your ideas clearly and logically. You should always explain how you arrived at your conclusions using mathematically sound reasoning. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.
- For questions requiring pseudocode, you can follow the same format as we do in class or write pseudocode in your style, as long as you specify your notation. For example, are you using "=" to mean assignment or to check equality? You are welcome to use any algorithm from class as a subroutine in your pseudocode. For example, if you want to sort list A using `InsertionSort`, you can call `InsertionSort(`A`)` instead of writing out the pseudocode for `InsertionSort`.
- You may use any of the following known NP-complete problems to show that a given problem is NP-complete: 3-SAT, INDEPENDENT-SET, VERTEX-COVER, SET-COVER, HAMILTONIAN- CYCLE, HAMILTONIAN-PATH, SUBSET-SUM, 3-COLORING.

There are 5 questions for a total of 100 points.

---

1. Show that the following problems that were discussed in class are NP-hard:
   (a) (10 points) Minimum makespan
   (b) (10 points) Dominating set

2. (20 points) Consider the following problem:

   SPAN-TREE: Given an undirected graph $G = (V, E)$, determine if there is a spanning tree of G that has at most 10 leaves.

   Either show that SPAN-TREE is NP-complete or give a polynomial time algorithm (along with correctness proof and running time discussion).

3. (20 points) Consider the following problem:

   F-SAT: Given a boolean formula in CNF form such that (i) each clause has exactly 3 terms and (ii) each variable appears in at most 3 clauses (including in negated form), determine if the formula is satisfiable.

   Answer the following questions with respect to the above problem under the assumptions (i) $\mathsf{P} = \mathsf{NP}$, and (ii) $\mathsf{P} \neq \mathsf{NP}$. Give reasons.

   (a) Is F-SAT $\in$ NP?
   (b) Is F-SAT NP-complete?

(c) Is F-SAT NP-hard?

(d) Is F-SAT $\in$ P?

4. (20 points) Consider the following problem:

   LONG-PATH: Given a weighted, directed graph $G = (V, E)$, two vertices $s, t \in V$ and a number W, determine if there is a simple path between s and t such that the sum of weights of edges in this path is $\geq W$.

   Recall that a simple path is a path that does not have any vertices repeated. Either show that LONG-PATH is NP-complete or give a polynomial time algorithm (along with correctness proof and running time discussion).

5. Recall the Minimum Makespan problem discussed in class. We saw a greedy approximation algorithm and showed an approximation guarantee of $2 - \frac{1}{m}$. Consider the following slightly modified algorithm:

   ```
   GreedySortMakespan
     - Consider jobs in decreasing order of duration
     - While all jobs are not assigned
       - Assign the next job (as per the order defined) to a machine with the least load
   ```

   Let OPT denote the value of an optimal solution, and G be the value of the above greedy solution. Then we will argue that $G \leq (4/3) \cdot OPT$. WLOG, let us assume that $d(1) \geq d(2) \geq ... \geq d(n)$.

   Let us call a problem instance $(d(1), ..., d(n))$ *nice* if as per the above greedy algorithm, the job with the maximum finishing time is n. We will first argue that it is sufficient to analyze the approximation guarantee for nice instances.

   Claim 1.1: If our greedy algorithm gives a factor c approximation for nice instances, then it gives factor c approximation for all instances.

   We can now focus on only nice instances. Again, for any nice instance, we use G to denote the value of the greedy solution and OPT to denote the value of an optimal solution. Let us break the analysis into two cases: (i) $d(n) \leq OPT/3$, and (ii) $d(n) > OPT/3$. For case (i), we can make the following claim:

   Claim 1.2: If $d(n) \leq OPT/3$, then $G \leq (4/3) \cdot OPT$.

   Now, consider the case when $d(n) > OPT/3$. We first show that $n \leq 2m$.

   Claim 1.3: If $d(n) > OPT/3$, then $n \leq 2m$.

   Once we realize the above, the next thing we show is that, in this case, the greedy algorithm gives an optimal solution.

   Claim 1.4 If $d(n) > OPT/3$, then `GreedySortMakespan` returns an optimal solution.

   Combining all the above claims, we get that the above greedy algorithm gives an approximation guarantee of (4/3).

   For this question, you have to answer the following:

   (a) (5 points) Prove Claim 1.1

   (b) (4 points) Prove Claim 1.2

   (c) (1 point) Prove Claim 1.3

   (d) (10 points) Prove Claim 1.4