Final

Start Assignment

Due Wednesday by 11:59pm **Points** 100 **Submitting** a file upload
File Types pdf and tex

This is a take-home exam. You must LaTex your answer and submit both the compiled PDF and all the LaTex sources. No need to submit supporting files like images. **Do not submit archives.**

For problems 1 and 2 of the following problems, you must provide an algorithm in pseudo-code, proof of correctness, and analysis of the running time together with any explanation you deem necessary. Problem 3 just requires a proof of "hardness". **Do not provide an implementation in an actual programming language.**

1. Dynamic programming [35]: A thief is robbing a store that has n items. The ith item is worth v_i dollars, weighs w_i pounds and can be grabbed in t_i time, where v_i, t_i and w_i are integers. He wants to
   take as valuable a load as possible, but he can carry at most W pounds in his knapsack, and can only spend T time before the police arrive. Which items should he take to maximize the total value? All values involved are integers.
2. Network flow [35]: Given an undirected graph $G(V, E)$, determine the minimum number of edges that can be disconnected to partition the graph in such a way that two given vertices $s, t \in V$ are in separate partitions. Hint: reduce the problem to max flow or one of the variants discussed in class. Make sure you provide a complete algorithm and indicate a possibly specialized version of max flow that can be used. Indicate the complexity of this specialized algorithm.
3. Complexity [30]: Problem to consider -- given a set of integers $a_1, a_2, \ldots, a_n$, determine if there is a way to partition the set into two parts in such a way that the sum of the values for each subset is the same. Provide a complete proof that the problem is NP-complete. Hint: use a reduction from Subset-Sum; make sure you have all the elements of the proof in place.

Notes on the problems (based on questions I got)

1. Problem 1
   1. You can provide either an array-based or memoization-based solution, but for memoization-based solution, you have to provide a precise mechanism to implement it (that is correct). My suggestion is to provide an array-based solution since it follows what we did in class.
2. Problem 2
   1. There are no weights on the edges
   2. In class, we saw that the complexity of Ford-Fulkerson depends on "a constant". Depending on the problem, that constant might have a "good value". Your complexity analysis needs to take that

into consideration.

3. You need to provide the most efficient algorithm, thus "saving" whatever can be saved in terms of complexity is good

3. Problem 3
    1. This is simply a complexity problem. You just have to provide an NP-completeness proof. No need to provide any algorithm since I am not asking for it.