H5.

- The assignment is due at Gradescope on Friday, February 24 at 11:59am. Late assignments will not be accepted. Submit early and often.
- You are permitted to study with friends and discuss the problems; however, *you must write up you own solutions, in your own words*. Do not submit anything you cannot explain. If you do collaborate with any of the other students on any problem, please do list all your collaborators in your submission for each problem.
- Finding solutions to homework problems on the web, or by asking students not enrolled in the class is strictly prohibited.
- We require that all homework submissions are prepared in Latex. If you need to draw any diagrams, however, you may draw them with your hand. Please use *a new page to begin each answer.*

PROBLEM 1 *Tug of War*

We want to play *roughly fair* tug of war in cs4800. You are given an integral array that holds the weights of n people in the class $W = (w_1, w_2, \ldots, w_n)$. Your goal is to divide the n people into two teams such that the total weight of the two teams is equal or as close as possible to equal. The total number of people on each team should differ by at most 1. Assume that M is the maximum weight of a person, i.e., $\forall i, w_i \leq M$. The running time should be $O(n^3 M)$. The output should be the list of people on each team and the difference in weight between the teams.

(a) Let $C(k, w, i)$ be *true* if it is possible to get the weight of team A to be w from the first k players, with i players on team A and $k - i$ players on team B. Describe a recurrence to compute $C(k, w, i)$ from $C(k-1, *, *)$. Hint: what are the possibilities for student k?

Solution:

(b) Describe a dynamic programming algorithm to compute all $C(k, w, i)$ $\forall i, k \leq n; 0 \leq w \leq \sum_i w_i$ and give its running time.

Solution:

(c) Describe how to compute the solution for our problem from the values $C(n, *, *)$ we computed above.

Solution:

PROBLEM 2 *Programming Tug of War*

In this part, you will implement your algorithm for Tug of War and test it on real data. Register and take on the challenge at `https://www.hackerrank.com/contests/cs4800-s17/`

Solution:

PROBLEM 3 *Lifeguards*

You run the pool on Saturdays. There must always be a lifeguard on duty. There are n lifeguards $1, 2, \ldots, n$ who can work, but they are busy college students with complex social obligations; lifeguard i can work starting at time a_i and ending at time b_i on Saturdays. Your goal is to cover the entire day from 8a–8p *using the fewest lifeguards as possible*.

(a) The first idea that comes to mind is to start with an empty schedule, and then add the guard who covers *the most amount of time that is not already covered*. Show that this algorithm fails by providing a counter-example.

Solution:

(b) Devise a greedy algorithm and prove that your strategy is optimal using an exchange argument. That is, consider an optimal solution. Prove that you can obtain a valid solution by exchanging one lifeguard in the optimal solution with the first lifeguard your algorithm picks.

Solution:

(c) Give pseudocode implementing your algorithm as fast as possible. Analyze the running time. You can make calls to algorithms we have learned in the course e.g. Karatsuba, mergesort, etc.

Solution: