CS302 - Problem Set 4

This homework is I think a little more challenging than previous assignments (because of the spiral nature of the class, it takes a little while for things to ramp up, because we were doing a lot of set-up/review initially), so start early and give your brain plenty of time to work on these.

1. You have a shipping container that can hold W worth of weight (please ignore volume). You have n types of items that you can ship. Item i is w_i pounds and is worth v_i dollars. However, you can take a fraction of any item, so if you wanted, you could pack $f \times w_i$ pounds of item i, which would be worth $f \times v_i$ dollars, where $0 < f \leq 1$. You would like to pack the shipping container so that the total value is as large as possible.

   (a) **[3 points]** Describe a greedy strategy for selecting which items, and how much of each item, to pack.

   (b) **[11 points]** Prove your strategy is optimal, using an exchange argument similar to the one we did in class. (You may assume that the function that you use to choose an order for your greedy algorithm gives a unique values to each item, so there is a unique optimal strategy.) There are probably other ways to prove that your strategy is correct, but for practice, you should use an exchange argument. In other words, assume for contradiction that there is some optimal strategy σ^* that is not your greedy strategy, and show that you can modify σ^* in an exchangey kind of way to get a new strategy $\sigma^{*\prime}$ that is better than σ^*, a contradiction.)

   (c) **[6 points]** Briefly describe the steps of the algorithm that implements this strategy, and give the runtime.

2. In 3 dimensions, the distance between two points $p_i = (x_i, y_i, z_i)$ and $p_j = (x_j, y_j, z_j)$ is $D(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$.

   (a) **[9 points]** Sketch (either using pseudocode, or using English to describe the steps) a divide and conquer algorithm for 3D closest points that returns the smallest distance between any two points. You may assume the input to your algorithm is three arrays X, Y, and Z, of the n points sorted by x-, y-, and z-coordinate respectively. (Hint 1: in 2D - we ended up with a problem that looked almost like finding closest points on a line. In 3D, you will first reduce to a problem that looks like 2D closest points, and from there, to something that is similar to points on a line. Hint 2: you will actually need two recursive algorithms).

   (b) **[11 points]** Sketch a proof that your algorithm is correct. If things are the same in your proof as in the 2D proof, you don't have to fill in all the details, but can say something like "The idea here is the same as in the 2D proof."

(c) **[6 points]** What is the runtime of your algorithm? (You should not have to do a full master-method type analysis here. If you end up with a recurrence relation you previously analyzed, just look up the solution from a previous pset, or use the master method final result.)

3. **[6 points for each invariant/base case/termination]** Create loop invariants that could be used to argue the following algorithm for insertion sort is correct. You should explain the invariant, the base case, and the termination condition. You do not need to prove the maintenance step. (Hint 1: your invariant can be of the form "If a certain array is sorted, then...", Hint 2: Don't forget an invariant for each loop, and start with the inner loop, Hint 3: Your "invariant" can include several parts.)

Algorithm 1: `InsertionSort`(A)

Input : Array A of integers of length n
Output: Array containing sorted elements of A

```
1  for k = 2 to n do
2  |  for j = k to 2 do
3  |  |  if A[j] < A[j − 1] then
4  |  |  |  Swap A[j] and A[j − 1];
5  |  |  else
6  |  |  |  Break;
7  |  |  end
8  |  end
9  end
10 return A;
```

4. Approximately how long did you spend on this assignment (round to the nearest hour)?