# CSE 101 Homework 5

Brian Masse, Kreshiv Chawla, Taira Sakamoto, Emily Xie, Annabelle Coles

October 31, 2025

1. **Cart horses**

   Consider the following problem: We have $n$ horses, $Horse_1, ..Horse_n$, each with a cart-pulling speed $s_i$. We need to pair the horses up into teams to pull carts.The faster horse will need to slow down for the slower horse,so if we pair $Horse_i$ and $Horse_j$, the cart will be pulled at speed $min(s_i, s_j)$. Each horse can only be in one pair. We want to maximize the sum of all the cart speeds.

   Give a greedy strategy for this problem, and prove that it is correct using any of the proof templates from class.

2. **Sensor maximization**

Suppose you are placing sensors on a one-dimensional road. You have identified $n$ possible locations for sensors, at distances $0 \leq d_1 \leq d_2 \leq ... \leq d_n$ from the start of the road. Each sensor must be at most $M$ distance from the previous one (so they can communicate reliably). The first one must be at 0 and the last at $d_n$. Given that, you want to minimize the number of sensors placed.

The following greedy algorithm, which places each sensor as far as possible from the previous one, will return a list $d_{i_1} \leq d_{i_2} \leq ...d_{i_k}$ of locations where sensors can be placed.

GreedySensorMin $[d_1...d_n, M]$

(a) Initialize a list to (0).

(b) Initialize $I = 1$, $PreviousSensor = 0$.

(c) While $I \leq n$ do:

(d) While $I \leq n$ and $d_I < PreviousSensor + M$ do:$I++$.

(e) If $I \leq n$ THEN append $d_{I-1}$ to list; $PreviousSensor = d_I$;$I++$.

(f) Append $d_n$ to list.

(g) Return list

(a) What constraints do solutions $d_{i_1}, ..d_{i_k}$ need to meet for this problem?

(b) What is the value of the objective function for a solution, of the form $d_0, d_{i_1}, ..d_{i_k} = d_n$, ?

(c) In using the "greedy stays ahead" proof technique to show that this is optimal, we would compare the greedy solution $d_{g_1}, ..d_{g_k}$ to another solution, $d_{j_1}, ..d_{j_{k'}}$ . In what sense is the greedy solution "staying ahead" of the other solution at each step $1 \leq t \leq k'$?

(d) Prove the claim you wrote above using induction on the step $t$.

(e) In O notation,how much time does the algorithm as written take?

3. **Quests**

In a role-playing game, your character will complete a series of $k$ quests. There is a list of $n > k$ possible quests $Q_i$, each with a first time it can be attempted $k \geq f_i \geq 1$ and the amount of gold earned on the quest, $g_i > 0$. You cannot complete the same quest twice, and you cannot put quest $i$ in a position before $f_i$ in your list. You want to maximize your total gold at the end of all your quests.

(a) Clearly state a greedy strategy that gives the most possible total gold

(b) Prove that this strategy is correct. Hint: use a modify-the-solution proof with two cases, based on whether the next quest the greedy algorithm performs is performed at some point in $OS$.

(c) Give an efficient algorithm carrying out this strategy and give a time analysis of your algorithm.

4. **Minimal spanning subgraph with negative edge weights**

   Say we allow negative edge weights into the spanning tree problem. We are given a connected undirected graph $G = (V, E)$ with possibly negative edge weights. We wish to find the set $E' \subset E$ so that $G' = (V, E')$ is connected that minimizes $\sum_{e \in E'} w(e)$.

   (a) Give an example of a graph with some negative edge weights where the minimum cost connected subgraph is not a tree.

   (b) Which edges are always in the minimum cost subgraph?

   (c) Once the edges above are added, what is an equivalent problem?

   (d) Describe how to use the answers to the above two questions to reduce this problem to minimum spanning tree. Give a time analysis for the resulting algorithm.

5. **Graph coloring heuristics**

The graph coloring problem is, given an undirected graph, give each vertex a color, so that neighboring vertices have different colors. You want to minimize the number of colors. This is an $NP$-complete problem, so we might want to use a greedy heuristic.

One greedy heuristic would be to order the vertices, then assign each the smallest color that is different from its neighbors.

Implement this heuristic on random graphs where we start with $|V|$ vertices and add $5|V|$ random edges. For a variety of sizes (e.g., $|V| = 2^k$ for $k = 3, ...12$), and several graphs of each size, try the heuristic ordering the vertices by increasing degree (number of neighbors) and decreasing degree. Plot the two numbers of colors used. Do these increase with $|V|$? How consistent are the results with different graphs of the same size?