COMPSCI 311: Introduction to Algorithms **Fall 2022**

Challenge Problems 2

due 10/5/2022 at 11:59pm in Gradescope

Instructions. Limited collaboration is allowed while solving problems, but you must write solutions yourself. List collaborators on your submission.

You can choose which problems to complete, but must submit at least one problem per assignment. See the course page for information about how challenge problems are graded and contribute to your homework grade. Since you don't need to complete every problem, you are encouraged to focus your efforts on producing high-quality solutions to the problems you feel confident about. There is no benefit to guessing or writing vague answers.

If you are asked to design an algorithm, please (a) give a precise description of your algorithm using either pseudocode or language, (b) explain the intuition of the algorithm, (c) justify the correctness of the algorithm; give a proof if needed, (d) state the running time of your algorithm, (e) justify the running-time analysis.

Submissions. Please submit a PDF file. You may submit a scanned handwritten document, but a typed submission is preferred. Please assign pages to questions in Gradescope.

1 Challenge Problems

Problem 1. Butterfly ID. K&T Ch 3 Ex 4. Some of your friends have become amateur lepidopterists (they study butterflies). Often when they return from a trip with specimens of butterflies, it is very difficult for them to tell how many distinct species they've caught—thanks to the fact that many species look very similar to one another.

One day they return with n butterflies, and they believe that each belongs to one of two different species, which we'll call A and B for purposes of this discussion. They'd like to divide the n specimens into two groups—those that belong to A and those that belong to B—but it's very hard for them to directly label any one specimen. So they decide to adopt the following approach.

For each pair of specimens i and j, they study them carefully side by side. If they're confident enough in their judgment, then they label the pair (i, j) either "same" (meaning they believe them both to come from the same species) or "different" (meaning they believe them to come from different species). They also have the option of rendering no judgment on a given pair, in which case we'll call the pair ambiguous.

So now they have the collection of n specimens, as well as a collection of m judgments (either "same" or "different") for the pairs that were not declared to be ambiguous. They'd like to know if this data is consistent with the idea that each butterfly is from one of species A or B. So more concretely, we'll declare the m judgments to be consistent if it is possible to label each specimen either A or B in such a way that for each pair (i, j) labeled "same," it is the case that i and j have the same label; and for each pair (i, j) labeled "different," it is the case that i and j have different labels. They're in the middle of tediously working out whether their judgments are consistent, when one of them realizes that you probably have an algorithm that would answer this question right away. Give an algorithm with running time $O(m + n)$ that determines whether the m judgments are consistent.

Problem 2. K&T Ch 3 Ex 9. Let $G = (V, E)$ be an n node undirected graph containing two nodes s and t, such that the distance between s and t is strictly greater than $n/2$. Show that there must be some node v, not equal to either s or t such that deleting v from G destroys all $s - t$ paths. In other words, the graph G' obtained by deleting v contains no paths from s to t. Give an algorithm with running time $O(m + n)$ to find such a node v. (Hint: consider a BFS starting at some node. Think about the layers.)

Problem 3. Directed Graphs. Given a directed acyclic graph G, give an algorithm with running time $O(m + n)$ to determine if the graph has a directed path that visits every vertex. Give a clear proof of correctness that argues that the algorithm outputs "yes" if and only if such a path exists.

Problem 4. Greedy stays ahead. You fail to land a good internship for the summer so you end up working in the UMass mail room. The job is really boring. You stand at a conveyor belt and put mail items from the conveyor belt into boxes. It turns out that all of the mail is headed to the CS department! Each box has a fixed limit W on how much weight it can hold, and the items arrive on the conveyor belt one by one: the ith item that arrives has weight w_i. The rules of the job are really draconian: you must fill one box at a time and send it to the CS department before starting on the next box, and you must pack items into boxes in exactly the order they arrive on the conveyor belt. So, your only real decision is how many items to pack in each box before you send it off to the CS department.

You decide to try a simple greedy algorithm: pack items into the current box in the order they arrive, and, whenever the next item does not fit, send the current box and start a new one.

Is it possible that this will cause you to use more boxes than necessary? That is, could you decrease the overall number of boxes by packing one box less full, so that items somehow fit more efficiently into later boxes?

Prove that, for a given set of items with specified weights, your greedy algorithm minimizes the number of boxes that are needed. Your proof should follow the type of analysis used in the book for the Interval Scheduling Problem: it should establish the optimality of this greedy packing algorithm by identifying a measure under which it "stays ahead" of all other solutions.

Here is some notation and a few definitions to help formulate the problem precisely.

- Assume the items are numbered $1, 2, \ldots, n$ and arrive in order, and that item i has weight w_i.
- Let i_k be the number of items packed in the first k boxes by the greedy algorithm (equivalently, i_k is the number of the last item packed in box k).
- Similarly, consider any optimal solution O and let j_k be the number of items packed in the first k boxes by O.

Here are some recommended steps to follow to develop a solution. (You don't need to submit answers to all of these. Your final proof will be graded based on clarity and completeness in solving the problem as stated above.)

- Create an example where you select a specific value for W, and make up weights for a sequence of n items that requires at least three boxes. Design your example so there are at least two different optima solutions. Indicate the values $i_1, i_2, \ldots, i_p$ for the greedy solution, as well as the values $j_1, \ldots, j_q$ for a *different* optimal solution.
- Write down an inequality that is always true for the quantities i_1 and j_1 and explain your reasoning.
- Formulate a "claim": an inequality comparing i_k and j_k that is true for $k \geq 1$, which you will prove by induction.
- Prove your claim by induction.
- Now argue that your claim implies that the greedy algorithm is optimal, i.e., that $p = q$ where p is the number of boxes used by the greedy algorithm and q is the number of boxes used by the optimal solution.