CS 577: Introduction to Algorithms — Homework 9

Out: 04/13/21 — **Due: 04/20/21**

Name: _____ Wisc ID: _____

Ground Rules

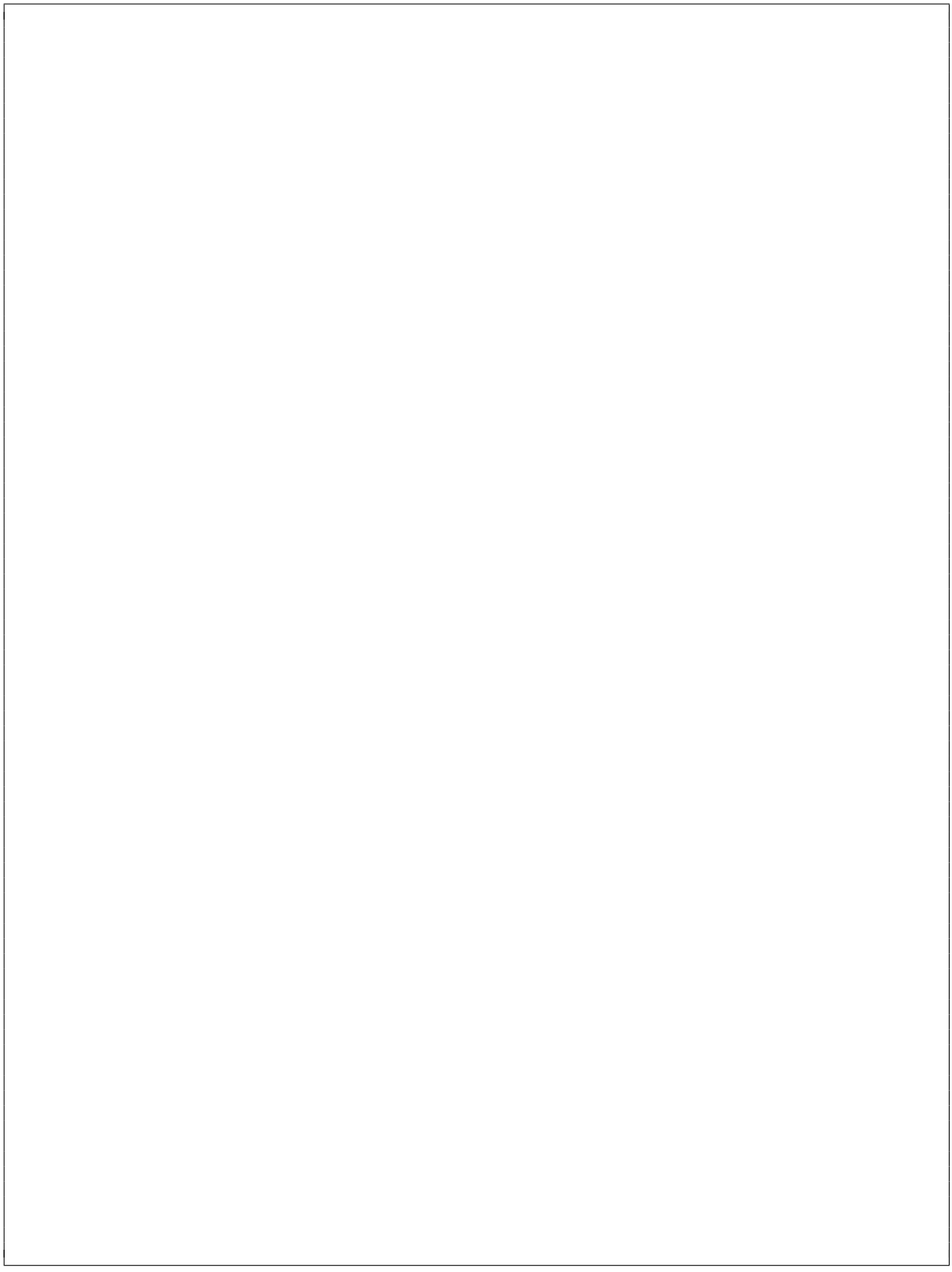
- Answer the questions in the boxes provided on the question sheets. If you run out of room for an answer, add a page to the end of the document. Do **not** feel obligated to fill the entire solution box. The size of the box does **not** correspond to the intended solution length.
- The homework is to be done and submitted individually. You may discuss the homework with others in either section but you must write up the solution *on your own*.
- You are not allowed to consult any material outside of assigned textbooks and material the instructors post on the course websites. In particular, consulting the internet will be considered plagiarism and penalized appropriately.
- The homework is due at 11:59 PM CT on the due date. No extensions to the due date will be given under any circumstances.
- Homework must be submitted electronically on Gradescope.

Problem 1

A group of people are carpooling to work together, and want to come up with a fair schedule for who will drive on each day. This is complicated by the fact that the subset of people in the car varies from day to day due to different work schedules. We define fairness as follows. There are n people and d days. Let $S = \{1, \cdots, n\}$ denote the set of people. Suppose that on the i-th day a (nonempty) subset $S_i \subseteq S$ go to work. A schedule for the d days is an assignment of drivers to days. The driver for day i must belong to the subset S_i. Note that a schedule may be partial in that not all days get assigned a driver. A schedule is called "fair" if for each driver j, the total number of times they drive is at most

$$\Delta_j = \left\lceil \sum_{i:j \in S_i} \frac{1}{|S_i|} \right\rceil .$$

(a) Design and analyze an efficient algorithm for computing a fair schedule that maximizes the number of days to which a driver is assigned. *Hint: Reduce the problem to network flow. What can you say about the value of the maximum flow in your constructed network?*



(b) Prove that for any choice of sets $S_1, ..., S_d$, there exists a fair schedule that assigns a driver to *each* day.

Problem 2

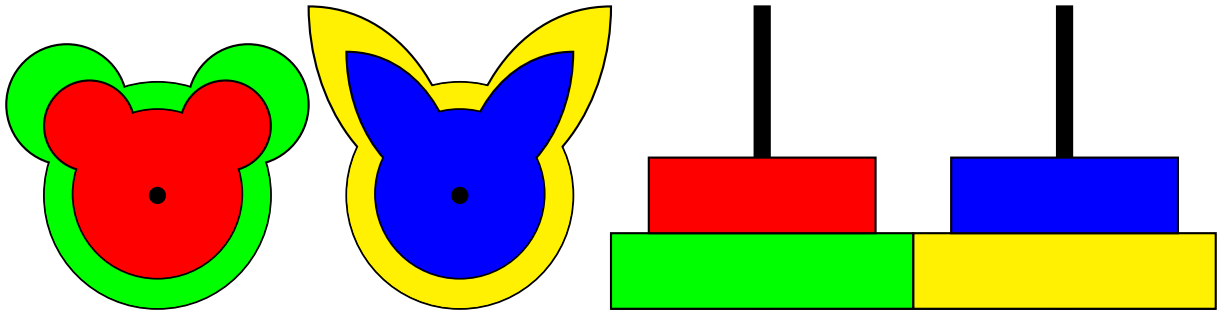
Children's toymaker AlgosRUs is creating a stacking puzzle where n oddly shaped discs need to be placed on k pegs, and has employed you to help with the task. Each disc can either be placed directly on a peg or must be placed on top of another disc that is strictly larger than it in all directions. Your task is to decide whether n given shapes can be assembled into k "stacks" to be placed on the pegs.

As input to the problem, you are given an $n \times n$ matrix C, where for all $i, j \in [n]$, $C_{ij} = 1$ if disc i can be placed on disc j and $C_{ij} = 0$ otherwise. You can assume that the given matrix represents a collection of shapes that are physically possible. Given n, C, and k as input, your goal is to design an algorithm that will output "Yes" if a feasible arrangement of the n discs on k pegs exists, and "No" otherwise.

For example, suppose that $n = 4$ and the matrix C is given by:

$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Then the discs can be arranged on two pegs as shown in the picture below. However, the discs cannot be arranged on a single peg.



top view side view

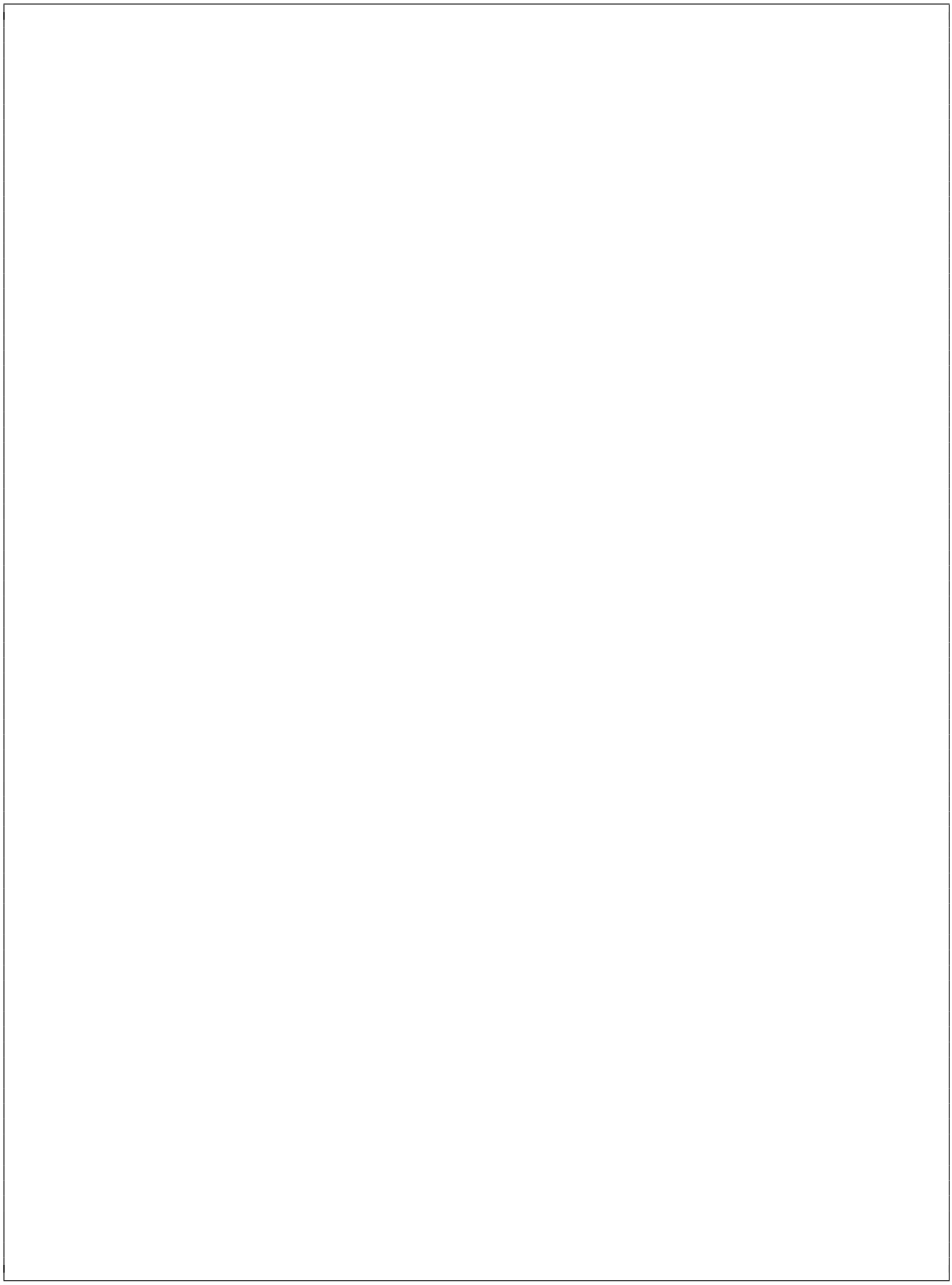
(a) Show that the following greedy algorithm does NOT solve this problem correctly by providing a counterexample. Make sure you include the matrix C and either a description of the shapes consistent with C or a diagram showing the shapes.

Algorithm 1: Greedy

Input: The list of discs D, the compatibility matrix C, and the number of pegs k

1 **for** $i \leftarrow 1$ **to** k **do**
2 Find the largest number of discs from the list D that can be put on one peg, call it n_i, and remove these discs from the list D;
3 **if** $n_1 + n_2 + ... + n_k = n$ **then**
4 **return** YES
5 **else**
6 **return** NO

Line 2 can be done with a dynamic programming algorithm, but you do not need to specify the details. If your counterexample relies on a particular tie-breaking rule (i.e., a rule that specifies which discs to remove in line 2 when there are multiple longest sequence of discs that can be put on one peg), please specify it. However, to obtain full marks for this part, your counterexample should not rely on the tie-breaking rule.



(b) Design an algorithm that solves this problem and has runtime polynomial in n. Provide a brief proof of correctness, as well as an analysis of the running time of your algorithm.

Hint: Try to reduce this problem to maximum flow, minimum cut, or some other application of network flow discussed in the lectures. You do not need to present an algorithm for solving the problem you reduce to, but make sure you explain how the reduction is made and argue its correctness.