The first page of your homework submission must be a cover sheet answering the following questions. Do not leave it until the last minute; it's fine to fill out the cover sheet before you have completely finished the assignment. Assignments submitted without a cover sheet, or with a cover sheet obviously dashed off without much thought at the last minute, will not be graded.

- How many hours would you estimate that you spent on this assignment?

- Explain (in one or two sentences) one thing you learned through doing this assignment.

- What is one thing you think you need to review or study more? What do you plan to do about it?

A particular instantiation of the Ford-Fulkerson framework, known as *Dinitz' Algorithm*, can be implemented to find maximum flows in a network in $O(V^2E)$ time. For this problem set, you may assume $O(V^2E)$ as the time needed to solve a max flow problem.

Question 1 (K&T 7.6)**.** Consider a set of mobile computing clients in a certain town who each need to be connected to one of several possible *base stations*. We'll suppose there are n clients, with the position of each client specified by its (x, y) coordinates in the plane. There are also k base stations; the position of each of these is specified by (x, y) coordinates as well.

We wish to connect each client to exactly one of the base stations, but our choice of connections is constrained in the following ways. First, there is a *range parameter*, denoted by r: a client can only be connected to a base station that is within distance r. There is also a *load parameter* L: no more than L clients can be connected to any single base station.

1

Your goal is to design a polynomial-time algorithm for the following problem. Given the positions of a set of clients and a set of base stations, as well as the range and load parameters r and L, decide whether every client can be connected simultaneously to a base station, subject to the constraints r and L.

Be sure to justify the correctness of your algorithm and analyze its asymptotic running time. Also, be sure to express the running time in terms of n, k, L, and r as appropriate; do not just say it takes $O(V^2E)$.

Question 2. Given a list of n classes, a list of r rooms, a list of t time slots, and a list of a therapy animals, our goal is to assign the final exam for each class to a particular room at a particular time, with a visit from one of a list of therapy animals.

- Each class i has a size S_i.
- Each room j has a capacity C_j; classes may only be assigned to rooms that are large enough to fit them, that is, if $S_i \leq C_j$.
- Each room is only available during some of the time slots, and may have at most one class assigned to it per time slot.
- Each therapy animal is only available during certain time slots, and an animal may only visit one exam per time slot.
- To avoid overwhelming the therapy animals, each animal may be assigned to visit a maximum of three exams.

Design an algorithm to find a valid way to assign each class to a room, time, and therapy animal that satisfies all the constraints (or report that no such assignment is possible). Be sure to justify the correctness of your algorithm and analyze its asymptotic running time.

Question 3 (CLRS 26.1-7)**.** Suppose that, in addition to edge capacities, a flow network has *vertex capacities*. That is, each vertex v has a limit $c(v)$ on how much flow can pass through v. Show how to transform a flow network $G = (V, E)$ with vertex capacities into an equivalent flow network $G' = (V', E')$ without vertex capacities, such that a maximum flow in G' has the same value as a maximum flow in G. How many vertices and edges does G' have?

Question 4 (CLRS 26-1b)**.** An $n \times n$ *grid* is an undirected graph consisting of n rows and n columns of vertices, as shown in Figure 1. We denote the vertex in the ith row and the jth column by (i, j). All vertices in a grid have exactly four neighbors, except for the boundary vertices.

Given $m \leq n^2$ starting points $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$ in the grid, the *escape problem* is to determine whether or not there are m vertex-disjoint paths (*i.e.* paths that do not share any vertices or edges) from the starting points to any m different points on the boundary. For example, the grid on the left of Figure 1 has an escape, as shown, but the grid on the right does not.

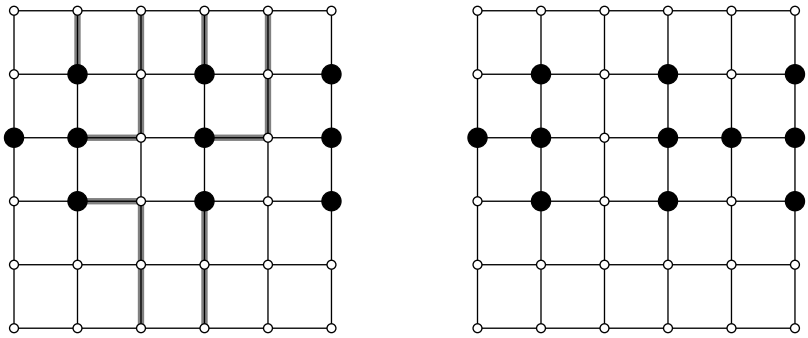

Figure 1: Grids with and without escapes

Describe an efficient algorithm to solve the escape problem, and analyze its running time.