Problem Set 8

MIT students: This problem set is due in lecture on **Monday, November 21, 2005.** The homework lab for this problem set will be held 2–5 P.M. on Sunday, November 20, 2005.

Reading: Section 22.1-22.2, Chapter 24.

Both exercises and problems should be solved, but *only the problems* should be turned in. Exercises are intended to help you master the course material. Even though you should not turn in the exercise solutions, you are responsible for material covered in the exercises.

Mark the top of each sheet with your name, the course number, the problem number, your recitation section, the date and the names of any students with whom you collaborated. **Please staple and turn in your solutions on 3-hole punched paper**.

You will often be called upon to "give an algorithm" to solve a certain problem. Your write-up should take the form of a short essay. A topic paragraph should summarize the problem you are solving and what your results are. The body of the essay should provide the following:

1. A description of the algorithm in English and, if helpful, pseudo-code.
2. At least one worked example or diagram to show more precisely how your algorithm works.
3. A proof (or indication) of the correctness of the algorithm.
4. An analysis of the running time of the algorithm.

Remember, your goal is to communicate. Full credit will be given only to correct solutions *which are described clearly*. Convoluted and obtuse descriptions will receive low marks.

Exercise 8-1. Do Exercise 22.2-6 on page 539 of CLRS.

Exercise 8-2. Do Exercise 24.1-3 on page 591 of CLRS.

Exercise 8-3. Do Exercise 24.2-4 on page 595 of CLRS.

Exercise 8-4. Do Exercise 24.3-2 on page 600 of CLRS.

Exercise 8-5. Do Exercise 24.5-2 on page 613 of CLRS.

Exercise 8-6. Do Exercise 24.5-7 on page 614 of CLRS.

Problem 8-1. No left turns

You've just stolen a brand-new Nexus Nano, but the owner has locked the car with *The Spade*, a mechanical lock on the steering wheel, which prevents you from making left turns. In addition, to avoid car accidents and otherwise attracting attention, you aren't going to make any U-turns either. You want to reach your chop-shop[1] as quickly as possible, and since you can drive down straight streets very quickly, your primary goal is to minimize the number of (right) turns you make. But if two paths have the same number of right turns, then you want to minimize the straight-line distance.

Fortunately, you have a map of the city, so you can plan out a route that uses no left turns and no U-turns. Even better, the map comes in electronic form as an $m \times n$ grid of cells, each marked as either ***empty*** (navigable) or ***blocked*** (unnavigable). At each cell you visit, you can continue in the direction you were going, or turn right. Two examples of no-left-turn maps are shown in Figure 1.

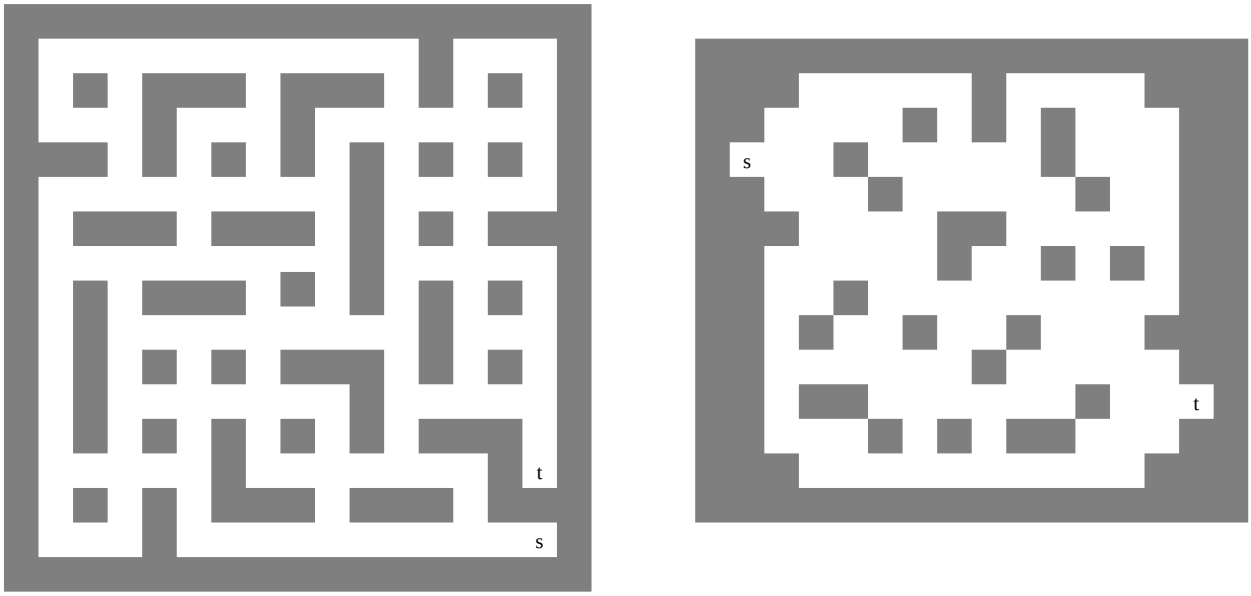

Figure 1: Two no-left-turn maps. Find a path from s to t with as few right turns as possible, and no left- or U-turns. You can play with these maps using a Java applet on `http://www.clickmazes.com/noleft/ixnoleft.htm`.

(a) Give an efficient algorithm to find a no-left-turn path through an $m \times n$ map using the minimum number of right turns, or report that no such path exists. Among all paths with the minimum number of right turns, your algorithm should return the path minimizing the straight-line distance (i.e., the number of straight steps). (*Hint:* For intuition, solve the left-hand map in Figure 1.)

(b) After mapping the route produced by your algorithm from part (a), you realize that you may lead the police right to the chop-shop if they are following you. You decide

[1] A garage that illegally disassembles cars in order to sell the parts.

that the best strategy to avoid pursuit is to make two right turns in quick succession, i.e., in adjacent squares, the 90-degree rotations intervened by only one unit of straight travel.

So now you want to find a route that makes at least two consecutive right turns, while still making no left turns, making as few right turns as possible, and among all such paths, minimizing the straight-line distance. Give an efficient algorithm to find such a path through the grid, or report that no such path exists.

Problem 8-2. ***Video Game Design***

Professor Cloud has been consulting in the design of the most anticipated game of the year: ***Takehome Fantasy XII***. One of the levels in the game is a maze that players must navigate through multiple rooms from an entrance to an exit. Each room can be empty, contain a monster, or contain a life potion. As the player wanders through the maze, points are added or subtracted from her ***life points*** L. Drinking a life potion increases L, but battling a monster decreases L. If L drops to 0 or below, the player dies.

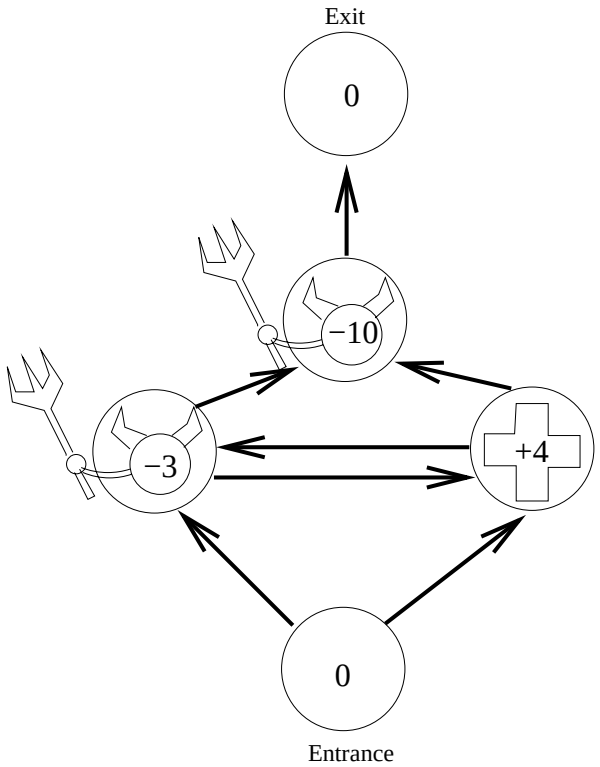

Figure 2: An example of a 1-admissible maze.

As shown in Figure 2, the maze can be represented as a digraph $G = (V, E)$, where vertices correspond to rooms and edges correspond to (one-way) corridors running from room to room. A vertex-weight function $f : V \to \mathbb{Z}$ represents the room contents:

- If $f(v) = 0$, the room is empty.
- If $f(v) > 0$, the room contains a life potion. Every time the player enters the room, her life points L increase by $f(v)$.
- If $f(v) < 0$, the room contains a monster. Every time the player enters the room, her life points L drop by $|f(v)|$, killing her if L becomes nonpositive.

The ***entrance*** to the maze is a designated room $s \in V$, and the ***exit*** is another room $t \in V$. Assume that a path exists from s to every vertex $v \in V$, and that a path exists from every vertex $v \in V$ to t. The player starts at the entrance s with $L = L_0 > 0$ life points. For simplicity, assume that the entrance is empty: $f(s) = 0$.

Professor Cloud has designed a program to put monsters and life potions randomly into the maze, but some mazes may be impossible to safely navigate from entrance to exit unless the player enters with a sufficient number $L_0 > 0$ of life points. A path from s to t is ***safe*** if the player stays alive along the way, i.e., her life points never become nonpositive. Define a maze to be ***r-admissible*** if a safe path through the maze exists when the player begins with $L_0 = r$ life points.

Help the professor by designing an efficient algorithm to determine the minimum value r for which a given maze is r-admissible, or determine that no such r exists.

(a) Find a safe path in the maze in Figure 2.

(b) Formulate the problem as an equivalent problem where the weights are on the edges, and prove equivalence.

(c) Assume for this problem part that there are no cycles whose traversal gives a net increase in life points. Given r, how would you check whether the maze is r-admissible?

(d) Now assume that there can be cycles in the graph whose traversal gives a net increase in life points. Given r, how would you check whether the maze is r-admissible?

(e) How can you find the minimum r for which the maze is r-admissible?