CS 473 ✧ Spring 2017

Homework 7

Due Wednesday, March 29, 2017 at 8pm

---

1. Suppose we are given an array $A[1..m][1..n]$ of non-negative real numbers. We want to ***round*** A to an integer matrix, by replacing each entry x in A with either $\lfloor x \rfloor$ or $\lceil x \rceil$, without changing the sum of entries in any row or column of A. For example:

$$\begin{bmatrix} 1.2 & 3.4 & 2.4 \\ 3.9 & 4.0 & 2.1 \\ 7.9 & 1.6 & 0.5 \end{bmatrix} \longmapsto \begin{bmatrix} 1 & 4 & 2 \\ 4 & 4 & 2 \\ 8 & 1 & 1 \end{bmatrix}$$

   (a) Describe and analyze an efficient algorithm that either rounds A in this fashion, or correctly reports that no such rounding is possible.

   (b) Prove that a legal rounding is possible *if and only if* the sum of entries in each row is an integer, and the sum of entries in each column is an integer. In other words, prove that either your algorithm from part (a) returns a legal rounding, or a legal rounding is *obviously* impossible.

2. Quentin, Alice, and the other Brakebills Physical Kids are planning an excursion through the Neitherlands to Fillory. The Neitherlands is a vast, deserted city composed of several plazas, each containing a single fountain that can magically transport people to a different world. Adjacent plazas are connected by gates, which have been cursed by the Beast. The gates between plazas are open only for five minutes every hour, all simultaneously—from 12:00 to 12:05, then from 1:00 to 1:05, and so on—and are otherwise locked. During those five minutes, if more than one person passes through any single gate, the Beast will detect their presence.[1] Moreover, anyone attempting to open a locked gate, or attempting to pass through more than one gate within the same five-minute period will turn into a niffin.[2] However, any number of people can safely pass through *different* gates at the same time and/or pass through the same gate at *different* times.

   You are given a map of the Neitherlands, which is a graph G with a vertex for each fountain and an edge for each gate, with the fountains to Earth and Fillory clearly marked. Suppose you are also given a positive integer h. Describe and analyze an algorithm to compute the maximum number of people that can walk from the Earth fountain to the Fillory fountain in at most h hours—that is, after the gates have opened at most h times—without anyone alerting the Beast or turning into a niffin. *[Hint: Build a different graph.]*

---

[1] This is very bad.

[2] This is very very bad.

Rubric (graph reductions): For a problem worth 10 points, solved by reduction to maximum flow:

- 2 points for a complete description of the relevant flow network, specifying the set of vertices, the set of edges (being careful about direction), the source and target vertices s and t, and the capacity of every edge. (If the flow network is part of the original input, just say that.)
- 1 point for a description of the algorithm to construct this flow network from the stated input. This could be as simple as "We can construct the flow network in $O(n^3)$ time by brute force."
- 1 point for precisely specifying the problem to be solved on the flow network (for example: "maximum flow from x to y") and the algorithm (For example: "Ford-Fulkerson" or "Orlin") to solve that problem. Do *not* regurgitate the details of the maximum-flow algorithm itself.
- 1 point for a description of the algorithm to extract the answer to the stated problem from the maximum flow. This could be as simple as "Return TRUE if the maximum flow value is at least 42 and FALSE otherwise."
- **4 points for a proof that your reduction is correct.** This proof will almost always have two components (worth 2 points each). For example, if your algorithm returns a boolean, you should prove that its True answers are correct and that its False answers are correct. If your algorithm returns a number, you should prove that number is neither too large nor too small.
- 1 point for the running time of the overall algorithm, expressed as a function of the original input parameters, *not* just the number of vertices and edges in your flow network. You may assume that maximum flows can be computed in $O(VE)$ time.

Reductions to other flow-based problems described in class or in the notes (for example: edge-disjoint paths, maximum bipartite matching, minimum-cost circulation) or to other standard graph problems (for example: reachability, topological sort, minimum spanning tree, all-pairs shortest paths) have similar requirements.