Reductions
& Randomization
Problem Set 10 – CS6515 (Spring 2025)

- This problem set is due on **Thursday April 17th**.
- Submission is via Gradescope.
- Your solution must be a typed pdf (e.g., via LaTeX, Markdown, etc. Anything that allows you to type math notation) – no handwritten solutions.
- Please try to make your solutions as concise and readable as possible. Most problems will have solutions that are no more than a page long. Consider using bullet points and adding space to break up large paragraphs into smaller chunks.
- **There are 2 problems**. Each problem is graded with 20p. **There is +1p bonus per problem** for stating (i) how long it took you to solve that problem, and (ii) how long it took you to type the answer.

28 QR Code Detection

QR codes can have different sizes, but they always have 4 indicators ("fiducial markers") in their corners. The four indicators are 3 large squares and one smaller square. See the two examples below.

Large squares in 3 corners →

Small square in this corner →

In order to detect a QR code, your computer (or phone or whatever) must detect these 4 indicators within the image taken by the camera. Let us simplify this computational problem and assume the QR code appears flat within the image (i.e., we don't need to rotate or skew the image to get proper alignment of the code). Let us also assume we already detected all shapes that look like those squares. With all these simplifying assumptions, how hard is it to detect the QR code within the image?

We can describe the task of detecting the QR code as follows:

QR-detection problem: Given a matrix $\mathbf{M}$ of size $n \times n$ where the entries are 0, 1, or 2. The algorithm must decide (i.e., return TRUE/FALSE) if there exist row indices $i < j$ and column indices $k < \ell$ with $\mathbf{M}_{i,k} = \mathbf{M}_{i,\ell} = \mathbf{M}_{j,\ell} = 1$ and $\mathbf{M}_{j,k} = 2$.

(This is our QR detection problem because $\mathbf{M}$ is the image taken by our camera and the matrix entries represents what kind of shape we detected within the image, i.e., 1 = large square, 2 = small square, 0 anything else. Finding indices $i < j$, $k < \ell$ means to find a rectangular sub-matrix within $\mathbf{M}$ where the corners $\mathbf{M}_{i,k} = \mathbf{M}_{i,\ell} = \mathbf{M}_{j,\ell} = 1$=large square, and the bottom left corner $\mathbf{M}_{j,k} = 2$=small square.)

Problem:

1. Show that if there is a $T(n)$-time algorithm for the QR-problem, then there is an $O(T(n))$-time algorithm for detecting triangles in undirected n-node graphs. (Give algorithm and complexity analysis.)

2. Explain why under the "boolean matrix multiplication conjecture," there is no efficient algorithm for the QR detection problem with $O(n^{3-\epsilon})$ time complexity.

Hint: Consider the adjacency matrix of graph G. This is already a 0/1 matrix. Where should we add 2-entries so that the QR detection algorithm can detect triangles?

You may assume $T(n) \geq n^2$ as any algorithm must read the $n \times n$ sized input matrix $\mathbf{M}$.

29 APSP and Min-Plus-Products

The typical matrix product between matrices $\mathbf{A}, \mathbf{B}$ is

$$(\mathbf{AB})_{i,j} = \sum_{k=1}^{n} \mathbf{A}_{i,k} \cdot \mathbf{B}_{k,j}.$$

This is also referred to as a $(+, \cdot)$-product because we have a sum $(+)$ of products $(\ \cdot\ )$.

The $(\min, +)$-product of matrices is correspondingly a minimum of sums, i.e.,

$$(\mathbf{A} \star \mathbf{B})_{i,j} = \min_{1 \leq k \leq n} (\mathbf{A}_{i,k} + \mathbf{B}_{k,j}).$$

Another problem is APSP (all pairs shortest paths), where we are given a directed and weighted graph $G = (V, E)$ and must compute for every pair $u, v \in V$ the distance (ie, length of the shortest path) from u to v. (It suffices to compute only the pair-wise distances. We do not need to return the paths.)

Problem: We want to show that $(\min, +)$-product and APSP are equivalent:

1. If there is an algorithm for computing APSP on n-node graphs in time $T(n)$, then there is an $O(T(n))$-time algorithm for $(\min, +)$-products of $n \times n$ matrices. (Give algorithm and complexity analysis.)

2. If there is a $T(n)$-time algorithm for computing $(\min, +)$-products of $n \times n$ matrices, then there is a $O(T(n) \log n)$-time algorithm for APSP. (Give algorithm and complexity analysis.)

Hint: Consider weighted adjacency matrix $\mathbf{A}$. How does $\mathbf{A} \star \mathbf{A}$ compare to paths that use only 2 edges?

You may assume $T(n) \geq n^2$ as any algorithm must at least read the input.

You may also assume that $T(O(n)) = O(T(n))$ (this is true for all polynomial time algorithms).