St. Francis Xavier University
Department of Computer Science

CSCI 435: Algorithms and Complexity
Assignment 2
Due March 12, 2025 at 2:30pm

Assignment Regulations.

- This assignment must be completed individually.
- Please include your full name and email address on your submission.
- You may either handwrite or typeset your submission. If your submission is handwritten, please ensure that the handwriting is neat and legible.

[8 marks] 1. Suppose we are given a set of n items, where the size of each item i is $0 < s_i < 1$. The *bin packing problem* asks us to pack all items into the minimum number of capacity-1 bins.

One common heuristic for the bin packing problem is the *first fit heuristic*: take each item in order and place it in the first bin that can fit it. As it happens, the first fit heuristic gives a 2-approximation algorithm for the bin packing problem.

(a) Let $S = \sum_{i=1}^{n} s_i$. Prove that the optimal number of bins required is at least $\lceil S \rceil$.

(b) Prove that the first fit heuristic leaves at most one bin less than half full and, as a consequence, the number of bins used by the first fit heuristic is no more than $\lceil 2S \rceil$.

[7 marks] 2. In an undirected graph $G = (V, E)$, a *triangle* is a triplet of vertices u, v, and w with the property that the three edges (u, v), (v, w), and (w, u) all appear in the graph. Similarly, in a directed graph, an *oriented triangle* of vertices u, v, and w occurs when the graph contains either the three directed edges $u \to v \to w \to u$ or the three directed edges $u \leftarrow v \leftarrow w \leftarrow u$.

Consider the following problem: given an undirected graph $G = (V, E)$, convert G to a directed graph by assigning orientations to each edge $e \in E$ in such a way that the number of oriented triangles in the resultant directed graph is maximized.

Give a randomized algorithm that produces a directed graph whose *expected* number of oriented triangles is at least $1/4 \cdot \mathrm{OPT}(G)$, where $\mathrm{OPT}(G)$ is the maximum possible number of oriented triangles.

[8 marks] 3. Suppose you have a Las Vegas randomized algorithm solving a given problem in expected polynomial time. Show that you can always convert this to a Monte Carlo randomized algorithm solving the same problem with one-sided error in polynomial time.

Hint 1. Remember that your Monte Carlo algorithm must produce a correct answer with probability $\geq 1/2$. This probability can be modelled by the expression $\mathbb{P}[T_{\mathrm{LV}}(n) < T_{\mathrm{MC}}(n)]$, where $T_{\mathrm{LV}}(n)$ is the runtime of the Las Vegas algorithm and $T_{\mathrm{MC}}(n)$ is the runtime of the Monte Carlo algorithm.

Hint 2. Building on the previous hint, you may find *Markov's inequality* useful: given a nonnegative random variable X and a value $a > 0$, we have that $\mathbb{P}[X \geq a] \leq \mathbb{E}[X]/a$.

[7 marks] 4. Suppose we want to implement the LRU algorithm for the paging problem. To do so, we must implement an appropriate data structure to efficiently maintain both page IDs and timestamps for each page in the cache. Describe a data structure that maintains a cache index in this way and provides the following three operations:

- EXISTS(p): checks whether a page with ID p exists in the cache and returns true or false.
- OLDEST(): returns the page ID corresponding to the page with the oldest timestamp.
- UPDATE(p, t): changes the timestamp of the page with ID p to time t (where t is always larger than the existing timestamp of page p).

Your data structure should implement each of these operations with an average time complexity of $O(1)$, and it should have a space complexity of $O(k)$ for a cache of size k. You can make use of as many standard data structures as you need (e.g., arrays, hash tables, heaps, linked lists, etc.).