Please do all problems; we will grade a subset of the assigned problems (same subset for everyone). Please follow all of the instructions given in the handout for Homework 1.


2. (12 points) Suppose that you wish to do two types of operations on an infinite binary counter: Increment, which is as defined in class, and Reset, which resets all bits in the counter to zero. Explain, in words, how such a counter could be implemented so that any sequence of \( n \) Increment and Reset operations on an initially-zero counter takes \( O(n) \) time. Use the accounting (i.e., credits-based) method to do the analysis. State clearly the invariant you use and the amortized cost that you assign to each operation. (Assume that it takes constant time to examine or to modify a bit.)

   Hint: Consider keeping a pointer to the high-order 1-bit.

3. (12 points) This problem assumes familiarity with Ch. 6. (Note that Ch. 6 considers max-heaps, whereas the problem below is for min-heaps; however, the two notions are symmetric.)

   Consider an implementation of a binary min-heap as a binary tree. It is known that Insert and Extract-Min each take time \( O(\log n) \) in the worst case, where \( n \) is the size of the heap. It is possible to use amortized analysis to derive a more informative bound, as requested in part (a) below.

   (a) Use the potential method to prove the following: If an arbitrary sequence of \( n \) operations, consisting of Insert and Extract-Min, is done on an initially-empty heap, then the amortized cost of Insert is \( O(\log n) \) and that of Extract-Min is \( O(1) \). Do not change these operations in any way. Describe your potential function carefully and show that it works.

   Hint: Relate your potential function to the depths of the nodes in the heap.

   (b) Is it possible to achieve an amortized cost of \( O(\log n) \) for Extract-Min and \( O(1) \) for Insert? Justify your answer.

4. (12 points) Ex. 17.4-2, p. 471.

5. (15 points) Problem 19-1, p. 526–527. (Use the potential function given on page 509.)

6. (8 points) Consider the 2-approximation algorithm, Approx-Vertex-Cover, in Section 35.1 for finding a vertex cover in an undirected graph \( G \). Prove that the set of edges selected by the algorithm forms a maximal matching in \( G \). (A matching is a set of edges where no two share an endpoint. A matching is maximal if it is not contained properly in any other matching.)

7. (12 points) Let \( G = (V, E, w) \) be a complete, undirected, edge-weighted graph, where \( |V| = n \) and the edge weights \( w(\cdot) \) satisfy the triangle inequality. Consider the following heuristic to find an approximate traveling salesperson tour on \( G \).
Pick any vertex and consider this to be a trivial cycle, $C_1$, consisting of one vertex. In a general step, let $C_i$ be the cycle constructed so far on $i$ vertices, $1 \leq i < n$. Compute $C_{i+1}$ as follows: Find an edge $(x, y)$, where $x \notin C_i$ and $y \in C_i$, of minimum cost (ties broken arbitrarily). Let $z$ be a neighbor of $y$ on $C_i$. Replace the edge $(y, z)$ on $C_i$ by the edges $(x, y)$ and $(x, z)$ to obtain $C_{i+1}$. (When $i = 1$, $y$ and $z$ are the same vertex, so $C_2$ is just two copies of the edge $(x, y)$.) Return $C_n$ as the approximate tour.

Let $C^*$ be the minimum length tour. Prove that $w(C_n)/w(C^*) \leq 2$. (With a slight abuse of notation, we use $w(C_n)$ and $w(C^*)$ to denote the lengths of $C_n$ and $C^*$, respectively.)

Hints: (i) Use the triangle inequality to obtain an upper bound on $w(C_{i+1}) - w(C_i)$ in terms of $w((x, y))$. (ii) Observe that the edges $(x, y)$ found during the $n - 1$ iterations above form a minimum spanning tree. (Explain briefly why this is so.)