Policies:

- It is strongly recommended that you try to solve the problems yourself before consulting other sources.
- You may use any source to solve the problems, but please cite your sources.
- For questions about problems 1 and 2, contact Harsha. For problem 3, contact Liu.

1 Single Source Shortest Path
The input for an instance of the SSSP is a connected weighted graph $G$ of $n$ vertices and $m$ edges, positive integer weights (representing lengths) on the edges, and a particular vertex $v_0$ marked as the source. The output assigns to each vertex its shortest distance from $v_0$.

(a) [10 points] Show that $\Omega(m + n \log n)$ is a lower bound on the running time of the Dijkstra’s algorithm for SSSP.

(b) [Extra Credit] Is $\Theta(m + n \log n)$ the optimal time for any algorithm solving the SSSP problem?

(c) [10 points] Given a supposed solution to the SSSP problem (the distances from the source to all vertices are specified), verify if the solution is correct in $O(m + n)$ time.

2 Median Data Structure
(a) [10 points] Suppose that we wanted a data structure to maintain a set of elements and quickly query for the median of the set (in case of even sized set of elements, call either of the two middle elements a median). We want to support the following operations with the specified amortized costs:
• **Build**(n): Build a data structure consisting of n elements in time $O(n)$.

• **makeDS**(x): Build a data structure consisting of 1 element in time $O(1)$

• **findMedian**(D): Find the median of D in time $O(1)$

• **insert**(x, D): Insert element x in to a structure D consisting n elements in time $O(\log n)$

• **deleteMedian**(D): Delete the median of D in time $O(\log n)$.

(b) [2 points] Build a data structure that supports all the above operations, but with the specified costs being worst case instead of amortized.

(c) [8 points] Is it possible to improve the amortized cost of the **insert** operation to $O(1)$, all other requirements being the same? If not, why?

3 **Union-Find**

Lecture 10 (Kozen’s book) on Union-Find uses two heuristics to improve the performance: (a) in an union, always merge the smaller tree into the larger, and (b) in a find, use path compression. Consider a sequence of m union and find operations on a set of n elements.

(a) [5 points] If we do not perform path compression and randomly pick one of the two elements to link to the other when we perform an union, what is the worst-case (over all sequences of operations) average cost (averaged over all choices of tree merges in union).

(b) [15 points] If we perform path compression but still randomly pick one of the two elements to link to the other when we perform an union, what is the worst-case (over all sequences of operations) average cost (averaged over all choices of tree merges in union).