Problem 1. Free marks.

Problem 2. Suppose that such S_1 and S_2 exist. Let u be any vertex in $S_1 \cap S_2$, and v be any vertex in $S_2 \setminus S_1$. For any vertex $w \in S_1$, because w can reach u in S_1 which in turn can reach v in S_2, we know w can reach v. On the other hand, because v can reach u in S_2 which in turn can reach w in S_1, we know v can reach w. Thus, $S_2 \cup \{w\}$ is a set of vertices that are mutually reachable. This violates the maximality of S_2 as an SCC.

Problem 3. We first prove LHS $\leq$ RHS. Suppose that the RHS is minimized at $u \in \mathrm{IN}(t)$. Thus, there is a path from s to t that first goes to u with distance $spdist(s, u)$ and then crossing the edge (u, t). This path has length $spdist(s, u) + w(u, t)$, implying LHS $\leq$ RHS.

Next, we prove LHS $\geq$ RHS. Consider an arbitrary shortest path π from s to t. Let u be the vertex preceding t on π. Clearly, $u \in \mathrm{IN}(t)$. The length of π, namely the LHS, must be $spdist(s, u) + w(u, t)$. Note that $spdist(s, u) + w(u, t)$ is merely one of the terms considered in the minimization of the RHS. It thus follows that LHS $\geq$ RHS.

Problem 4. First build a complete undirected graph $G(V, E)$ where

- $V = P$;
- for every two points $u, v \in P$, the edge $\{u, v\} \in E$ carries a weight equal to the two points' distance.

Then, a cycle defined in the problem statement is a Hamiltonian cycle in G. Thus, a cycle with length at most $2 \cdot \mathrm{OPT}$ can be found using the 2-approximate algorithm taught in the class.

Problem 5. We can cast the problem as a set cover problem. For the i-th column, define a set S_i of integers such that an integer $j \in [1, n]$ belongs to S_i if and only if $M[j, i] = 1$. Now, we can apply the $\ln n$-approximate set-cover algorithm taught in the class to solve this problem.

Problem 6. The algorithm is correct (Prof. Goofy finally got it right!). First, if an SCC has at least two distinct vertices u, v, then G has a path from u to v and also a path from v to u, which make a cycle. Second, if every SCC has only one vertex, then G itself is the SCC graph G^{scc}, which must be acyclic as proven in the class. It thus follows that G is acyclic.

Problem 7. First, find the median m of S in $O(n)$ expected time. Then, create another set of integers $T = \{|x - m| \mid x \in S\}$. Use k-selection to find the k-th smallest number $t \in T$. Then, scan S once to output every integer $x \in S$ satisfying $|x - m| \leq t$.

Problem 8. We can assume, w.l.o.g., that n is a power of 2. Let $S = P \cup Q$. Divide S using a vertical line ℓ such that exactly $n/2$ points fall on each side of ℓ. Let P_1 (resp., P_2) be the set of points in P on the left (resp., right) of ℓ. Define Q_1 and Q_2 similarly for Q. Recurse on (P_1, Q_1) and then on (P_2, Q_2).

When we return from recursion, we have obtained, for each point $q_1 \in Q_1$, the number c_1 of points in P_1 dominated by q_1. The count c_1 is precisely $dom_P(q_1)$ and be output directly. For each point $q_2 \in Q_2$, the recursion has found the number c_2 of points in P_2 dominated by q_2. To obtain $dom_P(q_2)$, we still need to find the number c_2' of points in P_1 dominated by q_2, after which $dom_P(q_2)$ can be set to $c_2 + c_2'$.

Next, we will explain how to find c_2' for each point $q_2 \in Q_2$. First, obtain the set Y of y-coordinates of the points in P_1. Sort Y in ascending order using $O(n \log n)$ time. Then, for each point q_2, the count c_2' is the number of values in Y that are less than or equal to $q_2[y]$. The count can be obtained with binary search in $O(\log n)$ time.

Let $f(n)$ be the worst-case running time of our algorithm when the input size is n. It is clear from the above discussion that $f(1) = O(1)$ and for $n \geq 2$

$$f(n) \leq 2 \cdot f(n/2) + O(n \log n).$$

Solving the recurrence gives $f(n) = O(n \log^2 n)$.

Problem 9.

1. $C^* = \{b, e\}$ and $r(C^*) = 1$.

2: Let $C = \{o_1, o_2\}$ be the set returned by the k-center algorithm. Assume that o_1 (resp., o_2) is the first (resp., the second) point added into C.

- When $o_1 \in \{a, b, c\}$, o_2 must be f. We have $r(C) = 2$.
- When $o_1 \in \{d, e, f\}$, o_2 must be a. We also have $r(C) = 2$.

Therefore, the radius of the centroid set returned by the k-center algorithm is always $2 \cdot r(C^*)$.

Problem 10. First, find the shortest path distance from s to each vertex $u \in V$. This can be done in $O((n + m) \log n)$ time by Dijkstra's algorithm.

Second, find the shortest path distance from every vertex $u \in V$ to t. This can also be done in $O((n + m) \log n)$ time. For this purpose, obtain a graph G^{rev} from G by reversing the direction of every edge in G. Then, run Dijkstra's algorithm to find the shortest path distance from t to every vertex $u \in V$ in G^{rev}. This distance is precisely the shortest path distance from u to t in the original graph G.

An edge (u, v) is feasible if and only if $spdist(s, u) + w(u, v) + spdist(v, t) \leq \sigma$. It is now trivial to report all the feasible edges in $O(m)$ time.