1. (50 pts) The Engineers of Gondor are planning a big modification to their network of canals. Let $G = (V, E)$ be an unweighted, undirected graph representing their current network. An induced subgraph $G' = (V', E')$ is the set of vertices V' and edges E' that remain after deleting some vertex set $U \subseteq V$ from V and deleting every (x, v) or $(v, x) \in E$ such that $x \in U$.

   Describe and analyze (describe in detail, derive its running time, prove its correctness; use pseudocode, if necessary) an efficient algorithm that, given some graph G, represented as an adjacency list, and an integer k, will allow the Engineers to find the largest induced subgraph G' such that the degree of every vertex in G' is greater than or equal to k.

   Gandalf's hint: Determine which vertices to delete so that the remaining graph has the desired property, and think about how to keep track of how deleting a vertex may change the status of other vertices.

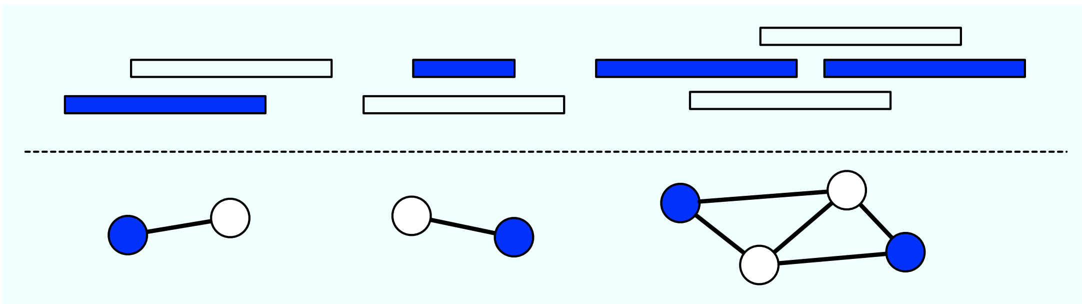
2. (50 pts total) Returning to the Shire after your long trek back from Mordor, you decide that you want to sell your stretch of river-front property (and move into a nice hobbit house). From various interested hobbits (and wizards?), you receive a set of bids for various intervals of the property. Wanting to maximize your profit across the set of sales, you must now decide which subset of bids to accept.

   Let $[A, B]$ denote the left- and right-endpoints of the river-front property on some real number line. Let the n bids you receive be denoted by the set X. Each bid is composed of (i) an interval $x_i = [\,L_i, R_i\,]$, where $A \leq L_i < R_i \leq B$, and (ii) a value $w(x_i) > 0$. Your task is to find the largest subset of bids $Y \subseteq X$ such that its value $w(Y) = \sum_{x_i \in Y} w(x_i)$ is maximized. Note that if two intervals overlap, then they both cannot be in Y, i.e., you cannot sell the same piece of land to two different bidders. See the upper half of the figure below.

   (a) (20 pts) Describe a naïve greedy approach to solving this problem. Explain what properties of X lead this approach to produce a suboptimal solution (a non-maximum $w(Y)$). Provide an example of X for which your algorithm returns a suboptimal solution, and identify the optimal solution Y.

   (b) (30 pts) Describe and analyze a dynamic programming algorithm that solves this problem correctly.

   Gandalf's hint: We can transform the input intervals X into an "interval graph" $G = (V, E)$ by letting each node in V correspond to an interval $x_i \in X$ and let two



nodes be connected $(x_i, x_j) \in E$ if and only if the corresponding intervals overlap. If we define a *valid* solution Y as a subset of interval such that no piece of land is sold twice, all valid solutions on the interval graph are then *independent sets* (do you see why?); see the lower half of the figure above. The goal, then, is reduced to finding the maximum-weight independent set on G.

3. (10 pts) Preparing for a big end-of-semester party in The Shire, you open your cellar and count n bottles of fine wine. Gandalf has previously warned you that exactly k of these bottles have been poisoned, and consuming poisoned wine will result in an unpleasant death. The party starts in one hour, and you do not want to poison any of your guests.

   Luckily, a family of ℓ docile rats has occupied one corner of the cellar, and they have graciously volunteered to be test subjects for identifying the poisoned bottles. Let $\ell = o(n)$ and $k = 1$, and assume it takes one hour for poisoned wine to kill a rat.

   Describe a scheme by which you can feed wine to rats and identify with complete certainty the poisoned bottle, prove that the scheme is correct and give a tight bound on the number of rats ℓ necessary to solve the problem.

   Gandalf's hint: Carefully chose subsets of the bottles, and feed wine from all bottles in a subset to a rat.

4. (**25 pts extra credit**) Prove that finding the second largest element in an n-element array requires exactly $n - 2 + \lceil \lg n \rceil$ comparisons in the worst case. Prove the upper bound by describing and analyzing an algorithm; prove the lower bound using an adversary argument.

   Gandalf's hint: An adversary argument is one that articulates exactly what the worst possible input could be, chosen by the adversary, regardless of the algorithm used. The adversary's strategy cannot depend on any predetermined order of operations or what the is done when elements are examined.

5. (**30 pts extra credit total**) Let S be a set of n points in the plane. A point p in S is called Pareto-optimal if no other point in S is both above and to the right of p.

   (a) (**15 pts extra credit**) Describe and analyze a deterministic algorithm that computes the Pareto-optimal points in S in $O(n \log n)$ time.

   (b) (**15 pts extra credit**) Suppose each point in S is chosen independently and uniformly at random from the unit square $[0, 1] \times [0, 1]$. What is the exact expected number of Pareto-optimal points in S?