

## Practice Problems on Lower Bound Proofs

November 14, 2003

1. Prove that  $n - 1 + \min(k - 1, n - k)$  is a lower bound on the number of comparisons needed to find the  $k$ th largest element.

*Hint: Think about how to modify the lower bound given in class for finding the median of  $n$  elements*

2. Consider an undirected  $2n$  vertex graph  $G$  where it is not known which edges are in  $G$ . (In other words, you can think of the adjacency matrix as being hidden from the algorithm.) An algorithm works under a model of computation in which it can give any two vertices, and is told if there is an edge between them. The goal of the algorithm is to determine if the graph is connected while asking about the fewest vertex pairs possible. Use the adversary lower bound technique to prove that at least  $n^2$  vertex pairs must be tested (in the worst case) in order for any algorithm to determine if  $G$  is connected.

*Hint: The adversary may find it convenient to partition the vertices into two groups of  $n$  completely connected vertices. While it is possible to prove a better lower bound, you need not do so.*

3. Consider the problem of merging two sorted arrays  $a_1, a_2, \dots, a_n$  and  $b_1, b_2, \dots, b_n$ . Prove the best lower bound you can on the number of comparisons required.
4. Consider the following simplified version of the game Battleship. For positive integers  $n$  and  $k$ , there is  $kn \times kn$  grid in which a  $k \times k$  battleship has been hidden. The algorithm can only learn about the location of the ship by using *probes*. In a probe, the algorithm selects any grid entry  $(i, j)$  for  $1 \leq i \leq kn$  and  $1 \leq j \leq kn$  and is either told “hit” if the ship covers this grid entry or “miss” otherwise. Give the best lower bound you can on the number of probes required by the algorithm to determine the coordinates of the upper-left hand corner of the ship.

## Solutions (solve the problems before reading these)

1. Observe that using the adversary strategy given in class for the median lower bound, once there are  $\min(k - 1, n - k)$  elements in heaven and hell, one of the two will be “full.” We modify this strategy, by at this point having the adversary moves all but one of the elements in purgatory to whichever of heaven and hell is not full. After this step there are  $k - 1$  elements in heaven,  $n - k$  in hell, and one element in purgatory (which will be the  $k$ th largest). The  $\min(k - 1, n - k)$  comparisons between two elements in purgatory cannot be crucial (using the obvious modification of crucial where the median is replaced by the  $k$ th largest). There must be a crucial comparison made for all but the  $k$ th largest; otherwise, that element is still a possibility for the  $k$  largest meaning the algorithm is not done since there would be two possibilities for the  $k$ th largest. Hence at least  $n - 1 + \min(k - 1, n - k)$  comparisons must be made by any comparison-based algorithm to find the  $k$ th largest element.
2. Divide the vertices (arbitrarily) into  $V_1$  and  $V_2$  each of  $n$  vertices. Respond to each question of the algorithm as follows:

- (a) If the algorithm asks about  $u, v \in V_1$  say “yes,  $(u, v) \in E$ .”
- (b) If the algorithm asks about  $u, v \in V_2$  say “yes,  $(u, v) \in E$ .”
- (c) If the algorithm asks about  $u \in V_1$  and  $v \in V_2$  say “no,  $(u, v) \in E$ .”

Notice that that algorithm cannot know  $G$  is not connected until it has asked about every edge fitting under rule (c) above (since the existence of any one of those edges in  $G$  would connect the graph). The number of edges fitting rule (c) above is  $n^2$  and hence at least  $n^2$  vertex pairs must be tested.

3. The adversary strategy is simply to answer all queries based on the final order  $a_1 < b_1 < a_2 < b_2 \cdots < a_n < b_n$ . We now define a crucial comparison and argue that  $2n - 1$  crucial comparisons are needed before the algorithm has forced a unique merged order. We first argue that each comparison of the form  $a_i < b_i$  for  $1 \leq i \leq n$  must be made by the algorithm. Suppose one was omitted. Then both the order given above and one with  $a_i$  and  $b_i$  interchanged would be consistent with all queries and hence the algorithm could not be done. By the same argument, each comparison of the form  $b_i < a_{i+1}$  for  $1 \leq i \leq n - 1$  must be made. Hence at least  $n + (n - 1) = 2n - 1$  comparison must be made by any correct comparison-based algorithm to merge two lists of length  $n$  since without any one of the  $2n - 1$  crucial comparisons there would still be two valid solutions left.
4. Here is a simple adversary argument to give a  $n^2 - 1$  lower bound. Partition the  $kn \times kn$  grid into a grid of  $n \times n$  squares which are  $k \times k$  each. Then ship will be in one of these  $n^2$  positions. For the first  $n^2 - 1$  probes asked by the algorithm, the adversary will say “no”. Note that after any  $n^2 - 2$  probes, there must be at least two  $k \times k$  region not yet probed and the ship could be in either of those locations and hence the algorithm cannot have located the ship until making at least  $n^2 - 1$  probes. It is possible to know the ship location without making any probe on it and so this bound cannot be extended to  $n^2$ .

Here is a more sophisticated strategy. There are  $(kn - k + 1)^2 = k^2(n - 1)^2 + 2k(n - 1) + 1$  positions for the ship. The adversary strategy is as follows. The adversary will say “no” for the first  $(n - 1)^2 - 1$  probes made. Note that each “no” response can eliminate at most  $k^2$  possibilities for the ship location. Hence after these first  $(n - 1)^2 - 1$  queries there will be at least  $k^2 + 2kn - 2k + 1 \geq k(2n + k)$  possible ship locations left. At this point the adversary will respond to any probe by computing the number of ship locations  $h$  that will be left if “hit” is given as the answer, and the number of ship locations  $m$  that will be left if “miss” is given as the answer. The adversary will answer “hit” if and only if  $h \geq m$ . If there were  $x$  possible ship locations before the query, then since  $h + m = x$ , and the adversary responds according to the larger of  $h$  and  $m$ , there will be at least  $\lceil x/2 \rceil$  ship positions left after the query. Hence, since there are at least  $k^2$  ship positions left when this adversary begins this strategy, the number of additional probes required by the algorithm to reduce the number of ship positions to 1 is at least  $\lceil \log_2 k(2n + k) \rceil \geq \log_2 k + \log_2(2n_k)$ . Hence the total number of probes required by any algorithm is at least  $(n - 1)^2 - 1 + \log_2 k + \log_2(2n_k)$ . This is better than the above bound when  $k$  is very large with respect to  $n$ . By refining slightly the number of probes for which the adversary answers “no” before switching to the other strategy, you can improve this bound a little (depending on the relationship between  $k$  and  $n$ ).