22 Adversary Arguments (April 20)

22.1 Minimum and Maximum

Let's consider the complexity of finding the largest *and* smallest elements in a set of n numbers. More formally:

> Given a sequence $X = \langle x_1, x_2, \ldots, x_n \rangle$ of n distinct numbers, find indices i and j such that $x_i = \min X$ and $x_j = \max X$.

How many comparisons do we need to solve this problem? An upper bound of $2n - 3$ is easy: find the minimum in $n - 1$ comparisons, and then find the maximum of everything else in $n - 2$ comparisons. Similarly, a lower bound of $n - 1$ is easy, since any algorithm that finds the min and the max certainly finds the max.

We can improve the upper and the lower bound to $\lceil 3n/2 \rceil - 2$. The upper bound is given by the following algorithm. Compare all $\lfloor n/2 \rfloor$ consecutive pairs of elements x_{2i-1} and x_{2i}, and put the smaller element into a set S and the larger element into a set L. if n is odd, put x_n into both L and S. Then find the smallest element of S and the largest element of L. The total number of comparisons is at most

$$\underbrace{\left\lfloor \frac{n}{2} \right\rfloor}_{\text{build } S \text{ and } L} + \underbrace{\left\lceil \frac{n}{2} \right\rceil - 1}_{\text{compute } \min S} + \underbrace{\left\lceil \frac{n}{2} \right\rceil - 1}_{\text{compute } \max L} = \left\lceil \frac{3n}{2} \right\rceil - 2.$$

For the lower bound, we use an adversary argument. The adversary marks each element $+$ if it might be the maximum element, and $-$ if it might be the minimum element. Initially, the adversary puts both marks $+$ and $-$ on every element. If the algorithm compares two double-marked elements, then the adversary declares one smaller, removes the $+$ mark from the smaller element, and removes the $-$ mark from the larger one. In every other case, the adversary can answer so that at most one mark needs to be removed. For example, if the algorithm compares a double marked element against one labeled $-$, the adversary says the one labeled $-$ is smaller and removes the $-$ mark from the other. If the algorithm compares to $+$'s, the adversary must unmark one of the two.

Initially, there are $2n$ marks. At the end, in order to be correct, exactly one item must be marked $+$ and exactly one other must be marked $-$, since the adversary can make any $+$ the maximum and any $-$ the minimum. Thus, the algorithm must force the adversary to remove $2n - 2$ marks. At most $\lfloor n/2 \rfloor$ comparisons remove two marks; every other comparison removes at most one mark. Thus, the adversary strategy forces any algorithm to perform at least $2n - 2 - \lfloor n/2 \rfloor = \lceil 3n/2 \rceil - 2$ comparisons.

22.2 Median

Finally let's consider the *median* problem: Given an array X of n numbers, find its $n/2$th largest entry. (I'll assume that n is even to eliminate pesky floors and ceilings.) More formally:

> Given a sequence $\langle x_1, x_2, \ldots, x_n \rangle$ of n distinct numbers, find the index m such that x_m is the $n/2$th largest element in the sequence.

To prove a lower bound for this problem, we use a combination of information theory and two adversary arguments. We use one adversary argument to prove the following simple lemma:.

Lemma: Any comparison tree that correctly finds the median element also identifies the elements smaller than the median and larger than the median.

Proof: Suppose we reach a leaf of a decision tree that chooses the median element x_m, and there is still some element x_i that isn't known to be larger or smaller than x_m. In other words, we cannot decide based on the comparisons that we've already performed whether $x_i < x_m$ or $x_i > x_m$. Then in particular no element is known to lie between x_i and x_m. This means that there must be an input that is consistent with the comparisons we've performed, in which x_i and x_m are adjacent in sorted order. But then we can swap x_i and x_m, without changing the result of any comparison, and obtain a different consistent input in which x_i is the median, not x_m. Our decision tree gives the wrong answer for this "swapped" input. $\square$

This lemma lets us rephrase the median-finding problem yet again.

> Given a sequence $X = \langle x_1, x_2, \ldots, x_n \rangle$ of n distinct numbers, find the indices of its $n/2 - 1$ largest elements L and its $n/2$th largest element x_m.

Now suppose a "little birdie" tells us the set L of elements larger than the median. This information fixes the outcomes of certain comparisons — any item in L is bigger than any element not in L — and so we can "prune" those comparisons from the comparison tree. The pruned tree finds the largest element of $X \setminus L$ (the median of X), and thus must have depth at least $n/2 - 1$. In fact, an adversary argument similar to last lecture's implies that *every* leaf in the pruned tree must have depth at least $n/2 - 1$, so the pruned tree has at least $2^{n/2-1}$ leaves.

There are $\binom{n}{n/2-1} \approx 2^n/\sqrt{n/2}$ possible choices for the set L. Every leaf in the original comparison tree is also a leaf in exactly one of the $\binom{n}{n/2-1}$ pruned trees, so the original comparison tree must have at least $\binom{n}{n/2-1}2^{n/2-1} \approx 2^{3n/2}/\sqrt{n/2}$ leaves. Thus, any comparison tree that finds the median must have depth at least

$$\left\lceil \frac{n}{2} - 1 + \lg \binom{n}{n/2-1} \right\rceil = \frac{3n}{2} - O(\log n).$$

A more complicated adversary argument (also involving pruning the tree) improves this lower bound to $2n - o(n)$.

A similar argument implies that at least $n - k + \left\lceil \lg \binom{n}{k-1} \right\rceil$ comparisons are required to find the kth largest element in an n-element set.

22.3 Evasive Graph Properties

Another class of problems for which adversary arguments give good lower bounds is graph problems where the graph is represented by an adjacency matrix, rather than an adjacency list. Recall that the adjacency matrix of an n-vertex graph $G = (V, E)$ is an $n \times n$ matrix A, where $a_{ij} = \big[(i,j) \in E\big]$.

We are particularly interested in deciding whether a graph has or does not have a certain *property*. For example, is the input graph connected? Acyclic? Planar? Complete?

The model of computation we consider is a *boolean decision tree*, which is a binary decision tree where every query asks for the value of some entry a_{ij} in the adjacency matrix, or in other words, each query asks if two vertices are connected by an edge. We say that a graph property is *evasive* if any boolean decision tree that correctly decides if graph has that property has depth $\binom{n}{2}$. In other words, in order to decide whether a graph has an evasive graph property, *any* algorithm must examine *every* possible edge in the worst case.

A fairly obvious example of an evasive graph property is *emptiness*: Does the graph have any edges at all? We can show that emptiness is evasive using the following simple adversary strategy. The adversary maintains *two* graphs E and G. E is just the empty graph with n vertices. Initially G is the complete graph on n vertices. Whenever the algorithm asks about an edge, the adversary removes that edge from G (unless it's already gone) and answers "no". Now, if the algorithm terminates without examining every edge, then G is not empty. Since both G and E are consistent with all the adversary's answers, the algorithm must give the wrong answer for one of the two graphs.

22.4 Connectedness Is Evasive

Now let me give a more complicated example, *connectedness*. Once again, the adversary maintains two graphs, Y and M ("yes" and "maybe"). Y contains all the edges that the algorithm knows are definitely in the input graph. M contains all the edges that the algorithm thinks *might* be in the input graph, or in other words, all the edges of Y plus all the unexamined edges. Initially, Y is empty and M is complete.

Here's the adversary's strategy for answering edge queries. I'll assume that whenever an algorithm examines an edge, it's in M but not Y; in other words, algorithms never ask about the same edge more than once.

```
QUERY(e):
    if M \ {e} is connected
        remove e from M
        return 0
    else
        add e to Y
        return 1
```

Notice that Y and M are both consistent with the adversary's answers. The adversary strategy maintains a few other simple invariants.

- **Y is a subgraph of M.**
- **M is connected.**
- **If M has a cycle, none of its edges are in Y.** If M has a cycle, then deleting any edge in that cycle leaves M connected.
- **Y is acyclic.** Obvious from the previous invariant.
- **If $Y \neq M$, then Y is disconnected.** The only connected acyclic graph is a tree. Suppose Y is a tree and some edge e is in M but not Y. Then there is a cycle in M that contains e, all of whose other edges are in Y. This is impossible.

Now, if an algorithm terminates before examining all $\binom{n}{2}$ edges, then there is an edge in M that is not in Y. Since the algorithm cannot distinguish between M and Y, even though M is connected and Y is not, the algorithm cannot possibly give the correct output for both graphs. Thus, in order to be correct, any algorithm must examine every edge—*Connectedness is evasive!*

22.5 An Evasive Conjecture

A graph property is *nontrivial* is there is at least one graph with the property and at least one graph without the property. (The only trivial properties are "Yes" and "No".) A graph property is *monotone* if it is closed under taking subgraphs — if G has the property, then any subgraph of G has the property. For example, emptiness, planarity, acyclicity, and *non*-connectedness are monotone. The properties of being a tree and of having a vertex of degree 3 are not monotone.

Conjecture (Aanderraa, Karp, and Rosenberg): Every nontrivial monotone property of n-vertex graphs is evasive.

The Aanderraa-Karp-Rosenberg conjecture has been proven when $n = p^e$ for some prime p and positive integer exponent e—the proof uses some interesting results from algebraic topology[1]—but it is still open for other values of n.

[1] Let Δ be a contractible simplicial complex whose automorphism group $\mathrm{Aut}(\Delta)$ is vertex-transitive, and let Γ be a vertex-transitive subgroup of $\mathrm{Aut}(\Delta)$. Suppose there are normal subgroups $\Gamma_1 \lhd \Gamma_2 \lhd \Gamma$ such that $|\Gamma_1| = p^\alpha$ for some prime p and integer α, $|\Gamma/\Gamma_2| = q^\beta$ for some prime q and integer β, and Γ_2/Γ_1 is cyclic. Then Δ is a simplex.

No, this will not be on the final exam.