

Finding the Second Largest Element

Andreas Klappenecker

Problem

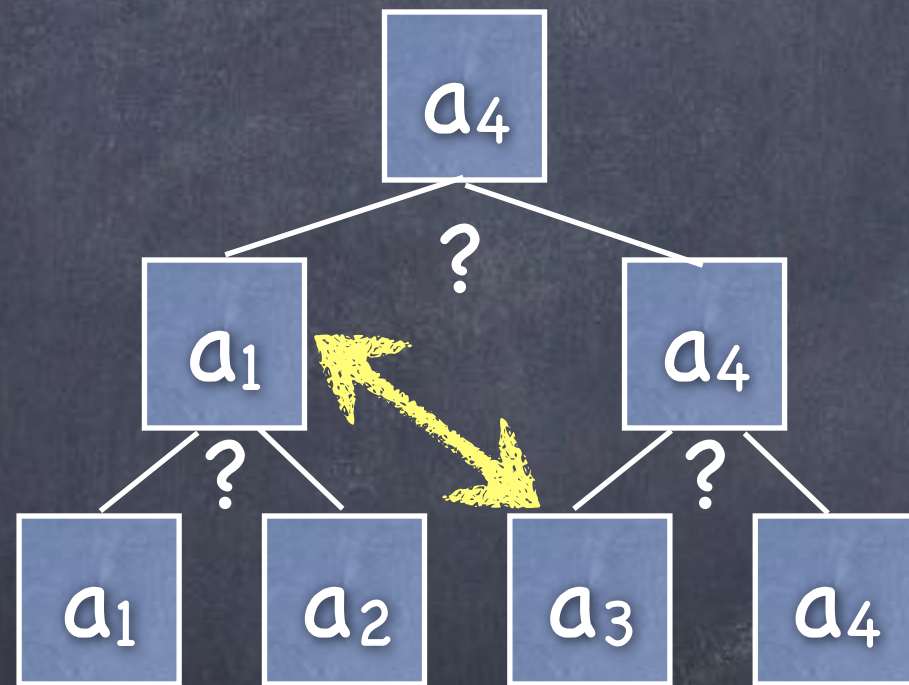
Given a set of n elements from a totally ordered domain, our goal is to find the second largest element m_2 .

How many queries are needed to determine m_2 ?

Upper Bound

We can compare the elements pairwise in a tournament style.

If $x < y$, then we say that y wins.



- If an element was never compared to the largest element, then it cannot be second largest (e.g. a_2).
- Find second largest among the one's who have lost to the largest.
- So $\leq (n-1) + \lceil \lg n \rceil - 1$ comparisons

Lower Bound

Any algorithm to determine the second largest element of a totally ordered set n elements needs at least $(n-2) + \lceil \lg n \rceil$ comparisons in the worst case.

Lower Bound

Let m_1 be the largest element and m_2 the second largest element.

An algorithm to determine m_2 **needs to find the largest element m_1** for otherwise an adversary would be able to exchange m_1 for m_2 .

Furthermore, the $n-2$ elements below m_2 must be identified by the algorithm, meaning that they must have lost in comparison to m_2 or some element below m_2 . This means that there are **$n-2$ comparisons that do not involve m_1** .

It remains to show that an adversary can force any algorithm to do **at least $\lceil \lg n \rceil$ comparisons with the largest element m_1** .

Adversary 1

Our goal is to show that an algorithm Z needs to make $\lg n$ or more comparisons with the largest element.

We construct an adversary that answers comparisons "Is $a \leq b$?" consistent with a total order of the n elements.

For each element x , we let $K(x)$ denote the set of elements y known to Z that satisfy $y \leq x$. Initially $K(x) = \{x\}$.

The adversary uses previous query history of Z and $K(a)$ and $K(b)$ to create answer for questions such as "Is $a \leq b$ ".

Adversary 2

The adversary behaves as follows:

- If "Is $a \leq b$?" was asked before, give same answer.
- If "Is $a \leq b$?" was not asked before, then answer
 - yes if $|K(a)| \leq |K(b)|$. Update $K(b) := K(a) \cup K(b)$
 - no, if $|K(a)| > |K(b)|$. Update $K(a) := K(a) \cup K(b)$

Adversary 3

Let S be the totally ordered domain of n elements.

- At the beginning $|K(a)| = 1$ holds for all a in S .
- For each query involving a , $|K(a)|$ can at most double.
- Since Z needs to determine largest element, $|K(m_1)| = n$ must hold at the end.
- The number k of queries involving m_1 satisfies $2^k \geq n$, so $k \geq \lg n$ and since k must be an integer, we have $k \geq \lceil \lg n \rceil$.

Conclusions

Any algorithm to determine the second largest element of a totally ordered set n elements needs at least $(n-2) + \lceil \lg n \rceil$ comparisons in the worst case.

We have given an optimal algorithm that attains this lower bound.