Lower Bounds (cont'd):

Now we turn to $\text{MIN-SMIN}$ (or $\text{MAX-SMAX}$)

We have Brute-Force alg: $2n^3$

D. and C alg: $\left\lceil \frac{3}{2} n - 2 \right\rceil$ 25% improvement

Now an even better algorithm using the method of "tournaments" best illustrated with an example: $n = 8$. (We do $\text{MIN-SMIN}$)

A: $[1, 4, 8, 7, 12, 6, 5, 10]$

Q: Is it a lower bound further?

YES

Among blue elements is the second smallest

And then $\lceil \log n \rceil$ of this $\rightarrow$ To find best 4, then, we need $\lceil \log n \rceil - 1$ comparisons.
We show \( n + \lceil \log_2 n \rceil - 2 \) is a valid lower bound (2) for MIN-SMIN (MAX-SMAX) using adversary argument. Same Scenario as MAX-MIN with two players:

I: alg. designer (controls algorithm)
II: adversary (controls input)

Consider a valid algorithm. At the end of the algorithm, let us classify elements as follows: In each comparison one element wins (i.e. is smaller) and one "lose" (smaller). Each comparison is tagged with element that loses. Let

\[ N_0 = \# \text{ of elements that never lost} \]
\[ N_1 = \# \text{ lost exactly once} \]
\[ N_2 = \# \text{ lost twice} \]

These sets are mutually exclusive and collectively exhaustive. Total \# of comparisons:

\[ \text{Total } \# \text{ of comp } = N_1 + 2N_2 + 3N_3 + \ldots \]

There is another way to do this counting.
Let
\[ n_1 = \# \text{ of elements that lost at least once} \]
\[ n_2 = \# \text{ twice} \]
\[ n_3 = \ldots \]

\[ n_1 = N_1 + N_2 + N_3 + \ldots \]
\[ n_2 = N_2 + N_3 + \ldots \]
\[ n_3 = N_3 + \ldots \]

\[ n_1, n_2 \leq \sum_{i=1}^{\infty} n_i = N_1 + 2N_2 + 3N_3 + \ldots \]

All these are \( \geq 0 \); integers.

Moreover, for a valid algorithm, \( n_1 = n-1 \)

We will show \( n_2 \geq \lceil \lg n \rceil - 1 \)

Under a particular adv. scheme. This will imply \( n + \lceil \lg n \rceil - 2 \) is a valid lower bound.

(Can our tournament alg. is optimal?)

We describe the adversary strategy to show

\( n + \lceil \lg n \rceil - 2 \) is a valid lower bound.

Note Candidate for SMax must have lost the Max element

This part is from Baas & van Gelder's book
The adversary has weights $w_i$ for element $AC_iJ$. 

[These weights are only used by the adversary to answer comparison questions asked by the algorithm designer. Recall there is no fixed $A[1\ldots n]$; the adversary is attempting to find the worst input.]

Initially, $w_i = 1$, $i = 1, 2, \ldots, n$. At any stage of the process, $w_i \geq 0$, integer such that $\sum_{i=1}^{n} w_i = n$.

Assume at any stage of the process, the algorithm designer asks for the comparison $AC_iJ : A[C_j]$. The pseudocode describing the adversary strategy is:

1. If \( w_i > w_j \):
   
   - Subtract \( w_i \) from both
   - Add \( w_j \) to both

2. If \( w_i = w_j \):
   
   - Subtract \( w_i \) from both
   - Add \( w_j \) to both

3. If \( w_j > w_i \):
   
   - Subtract \( w_j \) from both
   - Add \( w_i \) to both

Else, \( w_i = w_j = 0 \) → be consistent, no change in \( w_i \)
So, \( \sum_{i=1}^{n} w_i \) remains constant = \( n \), and once a \( w_i = 0 \), it continues to remain 0.

Elements for which \( w_i > 0 \), have not "lost" so far.

We keep track of which element is known at any stage to be larger than which other using a structure which is best illustrated via an example.

In this example \( n = 5 \). \( w_1 = w_2 = w_3, w_4 = w_5 = 1 \) at the start.


We represent this by

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array}
\]


\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array}
\]

\( w_1 \leftarrow 3 \)

\( w_3 \leftarrow 0 \)
Suppose next request is $A_{[4]}: A_{[5]}$.
Adversary answers $A_{[4]} > A_{[5]}$

$\begin{align*}
W_1 &= 3, \quad W_4 = W_5 = 0 \\
W_2 &= W_3 = 1
\end{align*}$

Suppose next request is $A_{[1]}: A_{[4]}$.
Adversary answers $A_{[1]} > A_{[4]}$

$\begin{align*}
W_1 &= 3, \quad W_4 = 2 \\
W_2 &= W_3 = W_5 = 0
\end{align*}$

At this point, alg. designer knows $A_{[1]} = \text{MAX}$

N node,

We have only listed comparisons among undefeated elements. Number of these comparisons = $n-1$. 

\[
\begin{align*}
W_1 &= 3, \quad W_2 = W_3 = W_4 = W_5 = 0 \\
W_4 &= W_5 = 1
\end{align*}
\]

Suppose next request is $A_{[4]}: A_{[5]}$.
Adversary answers $A_{[4]} > A_{[5]}$

$\begin{align*}
W_4 &= 2 \\
W_5 &= 0
\end{align*}$
If we concentrate on how the structure that includes (the eventual) overall \textbf{MAX}:

\begin{align*}
\text{Initial} & : \quad \text{number of nodes} = 1 \\
\text{Final} & : \quad n = n
\end{align*}

At each step, this number can at most double. Elements which "lost" to overall \textbf{WINNER} but were undefeated till then are \underline{potential candidates} for \textbf{SMAX}. Their number equals the number of steps it took the structure containing overall \textbf{MAX} to go from \(1\) to \(n\).

This number \(\leq \lceil \log_2 n \rceil\)

\(\therefore n_2 \leq \lceil \log_2 n \rceil - 1\)

\(\therefore \text{Total \# of Comparison} \leq n + \lceil \log_2 n \rceil - 2\)

\underline{Example 4:} \quad \textbf{Input:} \quad \text{Sorted Array} \quad A[1 \ldots n]

A[1] \leq A[2] \quad \ldots \quad \leq A[n]


One alg: \underline{Compare} consecutive elements; if \underline{Equal Value}, one such must be \underline{Consecutive}.

\# of Comparison (w.o): \(n-1\)

Q: \(2^{\lceil \log_2 (n-1) \rceil} \) a lower bound for this problem?
Example 5: Input: An unsorted array \( A[1,...,n] \)
Q: Are there duplicates?

One alg: Sort: \(\Theta(n\log n)\)
Check as in Example 4: \(n-1\)
\therefore Lower Bound: \(\Omega(n\log n)\)?
YES: Needs Algebraic Decision Tree

Practice Adv. arguments for (1) Sorting
(2) Binary search
(3) Example 4

Example 6: Convex-Hull in \(R^2\)
D-and-C alg: \(\Theta(n\log n)\) (presorting etc)
Lower Bound: \(\Omega(n\log n)\) ?  Yes

Proof Method: We know Lower Bound for Sorting is \(\Omega(n\log n)\)
We show how to use an alg. for Convex hull to Sort.

Input: \(x_1, x_2, ..., x_n\) numbers; Want to Sort.
Create \(P_i = (x_i, x_i^2)\) \(i=1,...,n\); There are all points on Parabola \(y = x^2\).
all are vertices, know \textit{Succ}(v) for each \( v \) gives you the sorted order.

Hence lower bound for \( CH \) is \( \Omega(n \log n) \).

This process is known as \textbf{REDUCTION}.

End of Lower Bounds.