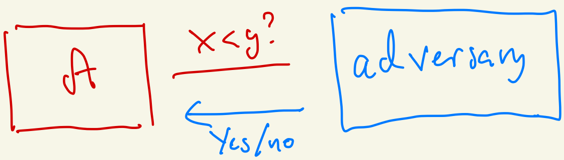
# Lower bounds for comparison based algorithms

Based on: Sara Baase, Computer Algorithms 2nd ed 1988 Addison Wesley
JBJ notes on lower bounds
Both available from homepage and Its learning

## What is a lower bound $L(n)$ for a problem P?

It is a proof that EVERY algorithm for solving P must use at least $L(n)$ operations (e.g. comparisons) on input of size n

Such proofs are often phrased in terms of an adversary who answers queries of the algorithm A consistently with previous answers while trying to force A to make many useless comparisons.




Example: Find maximum of n elements

We need at least n-1 comparisons since if z has never been compared, it could be the largest

n-1 comparisons suffice:

$m \leftarrow -\infty$
For $i \in 1$ to n
  if $x_i > m$ then $m := X$
return m

We will always measure the complexity of a comparison based algorithm in terms of the number of comparisons it makes when solving the instance at hand

**Problem max/min:** Given a set $S = \{x_1, x_2, \dots, x_n\}$ of $n$ numbers
Find the maximum and the minimum in $S$

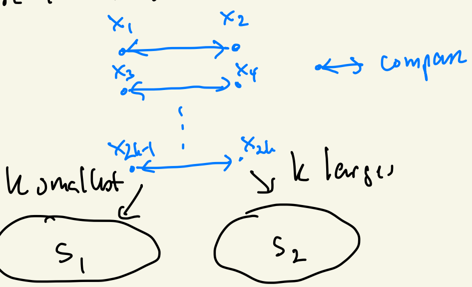
Naive algorithm:

- Find $M = \max\{x \mid x \in S\}$ in $n-1$ comparisons
- Find $m = \min\{x \mid x \in S - \{M\}\}$ in $(n-1)-1 = n-2$ comparisons

In total $2n-3$ comparisons

## Different algorithm ($\mathcal{A}_{max,min}$)

case 1 $n = 2k$




$m := \min\{x \mid x \in S_1\}$ $M := \max\{x \mid x \in S_2\}$

return $M, m$

\# comparisons:

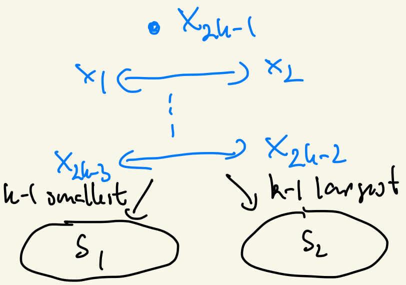
$k$ to construct $S_1, S_2$

$k-1$ to obtain $m$

$k-1$ to obtain $M$

$3k-2 = \frac{3}{2}n - 2$

case 2 $\quad n = 2k-1$




$m := \min\{x \mid x \in S_1\}$

$M := \max\{x \mid x \in S_2\}$

if $x_{2k-1} < m$ let $m = x_{2k-1}$

else if $x_{2k-1} > M$ let $M = x_{2k-1}$

return $M, m$

# comparisons

$k-1$ to obtain $S_1, S_2$

$k-2$ to obtain $m$

$k-2$ to obtain $M$

$+$ 1 or 2 to adjust via $x_{2k-1}$

$$\le 3k-3 = 3\left(\frac{n+1}{2}\right) - 3$$

$$= \frac{3n}{2} - \frac{3}{2}$$

Conclusion $A_{max,min}$ finds max and min using at most $\frac{3n}{2} - \frac{3}{2}$ comparisons

Can we do better?

No! as we can make an adversary who forces EVERY algorithm that finds min and max to use at least $\frac{3n}{2} - \frac{3}{2}$ comparisons

Adversary answers all comparison queries of $\mathcal{A}$ consistently with previous answers and forces $\mathcal{A}$ to make many redundant pieces of information

Observations. Let $\mathcal{A}$ be an arbitrary algorithm for finding min and max using only comparisons and let $m, M$ be the min and max elements in the input $S$

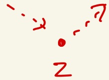
- Then $M$ is larger than all other elements so $\mathcal{A}$ has collected at least $n-1$ pieces of information: showing that $M$ is the maximum by ruling out each of the other elements via a lost comparison
- Similarly, to obtain $m$, $\mathcal{A}$ must collect at least $n-1$ pieces of information: showing that $m$ is the minimum, that is all other elements won at least one comparison
- In total $\mathcal{A}$ must collect at least $2n-2$ pieces of information to find $M$ and $m$

How many times can $\mathcal{A}$ collect 2 pieces of info in one comparison?

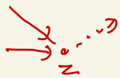
Only when comparing two elements $x$ and $y$ such that before this comparison they are both candidates for being both max and min. So at most $\lfloor \frac{n}{2} \rfloor$ times during the run of $\mathcal{A}$

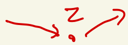
# Adversary strategy when A wants to compare x and y

notation $a \bullet\!\!\longrightarrow\!\!\bullet b \iff a<b$ (assume numbers in S are distinct)

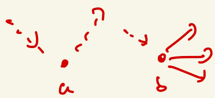
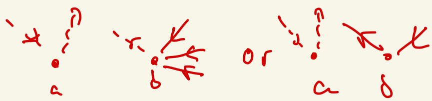
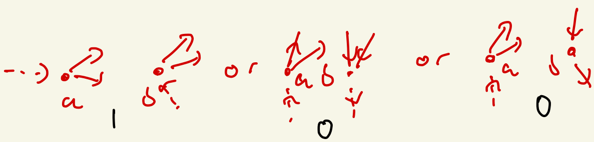
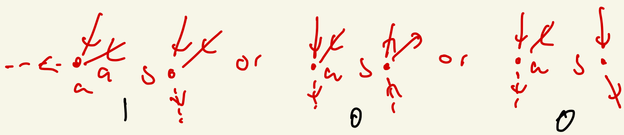
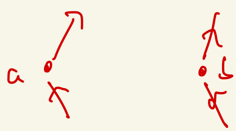
 : z was not in a comparison yet

 : z has lost all comparisons so far

 : z has won all comparisons so far

 : z has lost and won at least one comparison so far

## answer to a<b?

| # pieces of info obtained by A | status of a, b | answer |
| --- | --- | --- |
| 2 |  | $a<b$ |
| 1 |  | $b<a$ |
| 1 |  | $a<b$ |
| 0 or 1 |  | $a<b$ |
| 0 or 1 |  | $a>b$ |
| 0 |  | $a<b$ or $b<a$ so that consistency is maintained! |

Conclusion: $A$ can only obtain 2 pieces of info in the first case which can happen at most $\lfloor \frac{n}{2} \rfloor$ times

so $A$ can collect at most $n$ pieces of info in $\lfloor \frac{n}{2} \rfloor$ such comparisons.

Then remains $2n-2-2\lfloor \frac{n}{2} \rfloor \geq n-2$ pieces of information to collect

Each requires at least one extra comparison

Hence when $n$ is even $A$ must make at least

$$\frac{n}{2} + n - 2 = \frac{3n}{2} - 2 \text{ comparisons}$$

and when $n = 2k+1$ is odd $A$ can make at most $\lfloor \frac{n}{2} \rfloor = k$ comparisons which give 2 pieces of info

so there remains at least $2n - 2 - 2k = 2(2k+1) - 2 - 2k = 2k$ pieces of info to collect, each requirering a new comparison.

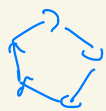
Hence when $n = 2k+1$ $A$ must make at least

$$k + 2k = 3k = 3\left(\frac{n-1}{2}\right) = \frac{3n}{2} - \frac{3}{2} \text{ comparisons}$$

How can the adversary answer consistently when both a and b have lost and won at least one comparison?

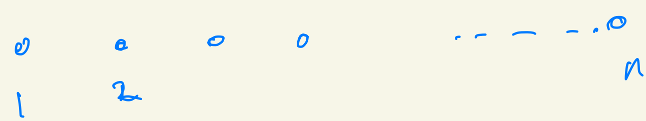
**Answer:** it uses acyclic orderings of partially oriented complete graphs

- A digraph $D=(V,A)$ is acyclic if D contains no directed cycle

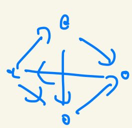


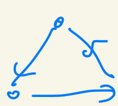
- A digraph is acyclic if and only if its vertices have an acyclic order $v_1, v_2, \dots v_n$ such that $v_i \to v_j \in A \Rightarrow i<j$

$\longrightarrow$ all arcs forward

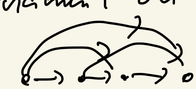



- A tournament is an orientation of a complete graph $K_n$



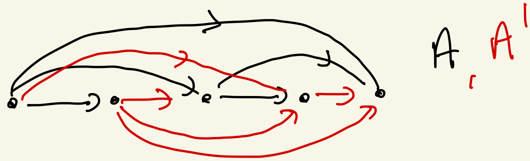


There is exactly one acyclic tournament on n vertices



$TT_4$

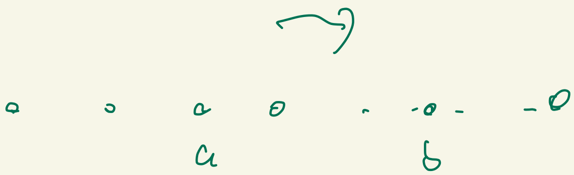
**Lemma** If $D=(V,A)$ is acyclic and $n=|V|$, then we can add a set of arcs $A'$ s.t. $D+A'$ is the transitive tournament $TT_n$




- The adversary orients edges of $K_n$ while answering the queries of $\mathcal{A}$. Let $D$ be the current oriented part
- The adversary can use the final acyclic digraph $D$ and an acyclic ordering of $V(D)$ to produce an input to $\mathcal{A}$ which is consistent with all answers given by the adversary
- We may chose this input as some permutation of $\{1,2..n\}$ determined by an acyclic ordering of the final $D$

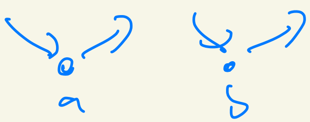
## Using acyclic orderings to answer queries:

Illustrated on max and min problem




current acyclic ordering

- initially no edges are oriented
- when a and b are compared the adversary checks the status ( • , , , ) of a and b and answers as we showed
- when




it also checks whether there is a directed path from a to b in D. If yes, it answers $a < b$ otherwise it answers $b < a$

## Problem: minimum and 2nd smallest element among n numbers

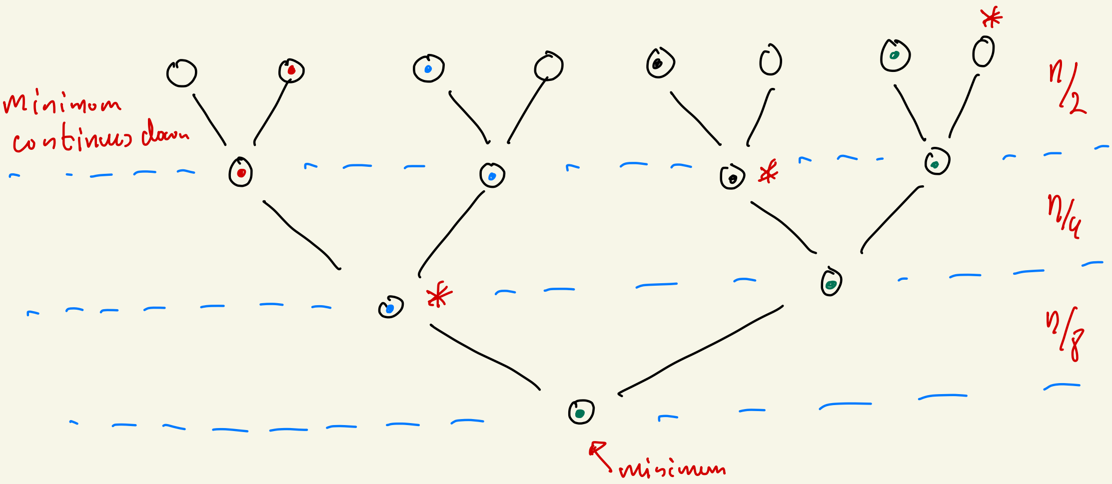
Naive algorithm:

1. Find $m = \min\{x \mid x \in S\}$ — $n-1$ comparisons
2. $S' \leftarrow S - \{m\}$
3. Find $m' = \min\{x \mid x \in S'\}$ — $n-2$ comparisons

Total $2n-3$ comparisons

## Tournament method:

assume $n = 2^k$ for some $k$




Where is second smallest?

Only candidates are those that were compared (and won) to min.

One for each level so $k = \log_2 n$ candidates $(n = 2^k)$

$\log_2 n$ candidates for 2min

- $n-1$ comparisons to get min and collect $\log_2 n$ candidates for 2min
- $\log_2 n - 1$ comparisons to get 2min

Total $n + \log_2 n - 2$ comparisons

Can we do better?

No!

Adversary wants to maintain as many candidates for 2min as possible while answering the queries of $A$ consistently

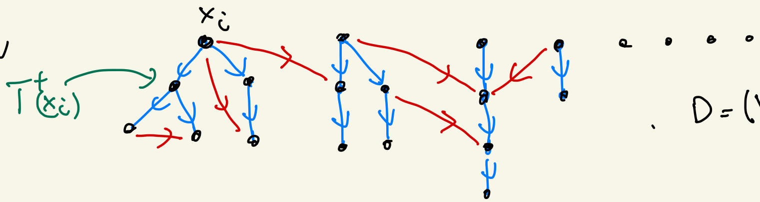
# Notation:

$x \longrightarrow y$ means adversary has answered that $x<y$

$\longrightarrow$ the arc is usefull for determining min

$\longrightarrow$ the arc provides no new info on min but possibly usefull for finding 2 min

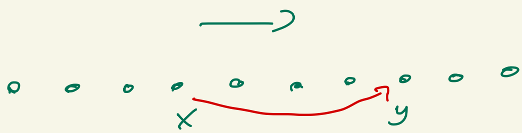
Initially $x_1 \; x_2 \; \ldots \; x_n$ (no arcs determined)



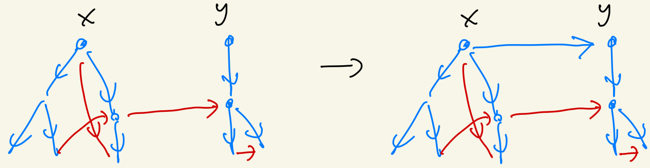

$D=(V,$ oriented arcs$)$

How adversary should answer query $(x,y)$: (adding $x \to y$ means also answer $x<y$)

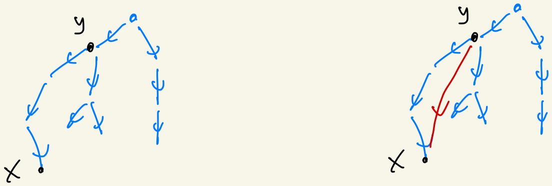
1. If $d^-_D(x)=d^-_D(y)=0$ ($x$ and $y$ are roots of blue out-trees)
   then if $|T^+(x)| \geq |T^+(y)|$ add a blue arc $x \to y$
   otherwise add $y \to x$
2. If $d^-_D(x)=0<d^-_D(y)$ add $x \to y$
3. If $d^-_D(y)=0<d^-_D(x)$ add $y \to x$
4. if $d^-_D(x), d^-_D(y)>0$ and no $(y,x)$-path in $D$ add $x \to y$
   else add $y \to x$



Example of rule 1:



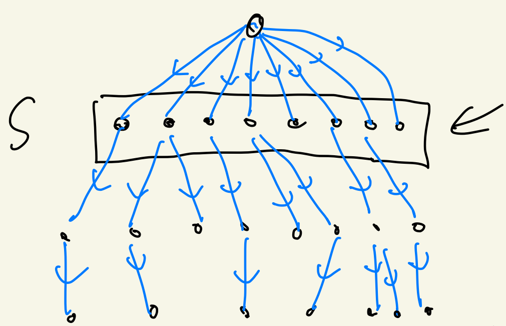

Example of rule 4:




Invariant: • the digraph D consisting of the oriented edges is always acyclic.

• The blue arcs form a forest of out-trees rooted at remaining candidates for being min

• When the algorithm terminates there is just one blue tree containing all vertices and its root is the minimum




• $|S| \geq \log_2 n$ by rule 1 (size of new blue tree is at most twice as large as largest of the two)

• A needs at least $|S| - 1 \geq \log_2 n - 1$ comparisons corresponding to red arcs to determine 2min

<u>Conclusion</u> The adversary forces $A$ to make at least $n-1$ comparisons (blue arcs) to find min and at least $\log_2 n - 1$ other comparisons (red arcs) to determine 2min

so $A$ uses at least $n-1+\log_2 n - 1 = n + \log_2 n - 2$ comparisons $\square$.