Today

- Lower bounds
  - Counting arguments
  - Ad hoc arguments
  - Adversary arguments

Run Time Bounds

Worst Case running times of classic sorting algorithms:
- Bubble-sort: $\Theta(n^2)$
- Insertion-sort: $\Theta(n^2)$
- Merge-sort $\Theta(n \log(n))$
- Heap-sort $\Theta(n \log(n))$
- Quick-sort $\Theta(n^2)$

Comparison-based sorting

A comparison-based sorting algorithm is one that doesn’t need to read the input, provided it is given the size of the input and a comparison oracle.

Lower Bound for Sorting

Theorem: Any comparison-based sorting algorithm has a worst-case running time that is $\Omega(n \log(n))$.

Proof of Theorem

A decision tree describes the queries of a comparison-based algorithm on input size $n$.

A root to leaf path represents the sequence of queries for a particular input.
Proof of Theorem cont.

Each leaf corresponds to the permutation that sorts the input.

Proof cont.

- There must be at least \( n! \) leaves.
- A binary tree with \( n! \) leaves has a path with length at least \( \log(n!) \).
- By Stirling's approximation, \( \log(n!) = \Omega(n \log(n)) \).

Today

- Lower bounds
  - Counting arguments
  - Ad hoc arguments
  - Adversary arguments

FIND-MIN

- How many comparisons does it take to find the minimum in a set of integers?
  - Answer: \( n-1 \)

FIND-MIN

- How many comparisons does it take to find the minimum in a set of integers?
  - Answer: \( n-1 \)

Upper Bound for FIND-MIN

Upper Bound Theorem: Finding the minimum in a set of \( n \) integers requires no more than \( n-1 \) comparisons.

Proof: Give algorithm
Lower Bound for FIND-MIN

Lower Bound Theorem: Finding the minimum in a set of integers requires at least \( n-1 \) comparisons.

Proof of Lower Bound:
- Consider an algorithm \( A \) on input of size \( n \).
- Let \( G \) be a graph with a vertex for each input integer. Initially \( G \) has no edges. When \( A \) compares two input values, we'll add an edge between the corresponding vertices of \( G \).
- \( A \) cannot conclude until \( G \) has \( \text{edges} \).
- Thus \( A \) cannot conclude until it has made \( \text{comparisons} \).

Today
- Lower bounds
  - Counting arguments
  - Ad hoc arguments
  - Adversary arguments

Upper Bound for FIND-MIN/MAX
- Upper Bound Theorem: Finding the minimum and maximum in a set of \( n \) integers requires no more than \( \lceil \frac{3n}{2} \rceil - 2 \) comparisons.
- Proof: Give an algorithm

Proof of Upper Bound:
- Algorithm for even \( n \):
  - Make \( n/2 \) pair-wise comparisons
  - Find the maximum of the winners with \( n/2-1 \) comparisons
  - Find the minimum of the losers with \( n/2-1 \) comparisons
- Algorithm for odd \( n \):
  - Run even algorithm on first \( n-1 \) integers
  - Compare the min and max to the last integer

Lower Bound for FIND-MIN/MAX
- Lower Bound Theorem: Finding the minimum and maximum in a set of \( n \) integers requires at least \( \lceil \frac{3n}{2} \rceil - 2 \) comparisons.
- Proof: Adversary argument
Example of an adversary

You pick a number \( y \) between 1 and 100
I have to guess \( y \) by posing queries of the form “Is it \( x \)?”
You answer "yes, \( x\leq y \)" or "no, \( y<x \)"

- How many queries can you force me to make?
- Prove it!

Adversary Argument to prove bound \( B \)
(for FIND MIN/MAX)

**Adversary Algorithm**

**Find-Min/Max Algorithm**

A set of integers \( S \):
A makes at least \( B \) comparisons on input \( S \)

**FIND-MIN/MAX Adversary - Accounting**

- Adversary = interactive comparison oracle
- Accounting scheme: For \( x \) in \( S \)

\[
\begin{align*}
b_{\text{MAX}}(x) &= 1 & \text{if the algorithm can rule out } x \text{ as the largest integer} \\
&= 0 & \text{otherwise} \\
b_{\text{MIN}}(x) &= 1 & \text{if the algorithm can rule out } x \text{ as the smallest integer} \\
&= 0 & \text{otherwise}
\end{align*}
\]

**FIND-MIN/MAX Adversary - Strategy**

- On query "Is \( x \leq y \)?"
- Answer consistently with previous answers
- If yes and no both consistent then answer so as to minimize the changes in \( b_{\text{MAX}} \) and \( b_{\text{MIN}} \) variables

**FIND-MIN/MAX Adversary - Analysis**

Consider a query "Is \( x \leq y \)?"

- If NO: \( b_{\text{MIN}}(x) \to 1 \) and \( b_{\text{MAX}}(y) \to 1 \)
- If YES: \( b_{\text{MAX}}(x) \to 1 \) and \( b_{\text{MIN}}(y) \to 1 \)

**Proof of Lower Bound:**

- Claim: At most \( \lceil n/2 \rceil \) queries can result in the change of two \( b_{\text{MIN/ MAX}} \) variables
- Claim: \( 2n-2 \) changes must occur before the algorithm concludes

\[ \Rightarrow \lceil n/2 \rceil + (2n-2) - 2\lceil n/2 \rceil \text{ queries are necessary} \]
**Find-gap**

- Input: $S: x_1, x_2, \ldots, x_n$ is a list of distinct integers sorted in ascending order.
- Question: Is there an index $i$ such that $x_{i+1} < x_i$?
- How many elements of $S$ have to be read (in worst case) in order to answer?

---

**Exercise**

- What is a good adversary strategy?
- What is a good algorithm strategy?

---

**Double 0's**

- Input: $B: b_1, b_2, \ldots, b_n$ $n$-bit vector of 0/1's
- Question: Are there two adjacent 0's?
- How many bits of $B$ have to be read (in worst case) in order to answer?

---

**Exercise**

- What is a good adversary strategy?
- What is a good algorithm strategy?

---

**Upper Bound**

Claim: Double 0's can be solved with $f(n)$ queries where:

- $f(n) = n - 1$ if $n \equiv 1 \mod 3$
- $f(n) = n$ otherwise

---

**Lower Bound**

Double 0's cannot be solved with fewer than $g(n)$ queries where:

- $g(n) = n - 1$ if $n \equiv 1 \mod 3$
- $g(n) = n$ otherwise