# Set 04: Pseudo-Random Generators, and Computational Lower Bounds

## CS240: Data Structures and Data Management

Jérémy Barbay

# Outline

# Pseudo-Random Number Generators

- Computers are built to be deterministic
- A Real Random source is not always necessary.
- Independant "Pseudo-Random" sources are often sufficient.

# Requirements of a Pseudo-Random source

We should not be able to observe any biases such as:

- Some numbers occur significantly more often than other numbers
- Odd numbers occur significantly more often than even numbers
- The sequence contains arithmetic progressions that are longer or shorted than would be expected by chance
- ...

# Example: Linear Congruential Generator

- Initialize a variable *seed* to an independant value (computer clock).
- Generate pseudo-random numbers as follows:

```
seed = ( a*seed + c ) mod max
return seed
```

# Disadvantages of this Generator:

- Values of $a$, $c$, and $max$ must be set with care.
- Value of $max$ must be large.

In practice, more complicated functions are used, but the principle stays the same.

# Disadvantages of Randomized Algorithms

- Computer Architecture tries to predict the future behavior of algorithms to optimize its execution.
- Algorithms randomized at the lowest level lose the benefit of this optimisation.
- Randomization usefull only for very hard problems.

# Outline

# Computational Complexity Lower Bounds: Why?

Given a problem to solve,

- Till when can you improve your algorithm?
- Where can you improve your technique?

For difficult problems where lower bounds are not known, other techniques $\Rightarrow$ NP-hardness.

Example

- Is $O(n \log n)$ optimal for sorting?
- Is $O(2^n)$ optimal for solving $f(x_1, \ldots, x_n) = 1$?

# The Comparison Model

When you don't manage to prove results on all possible algorithms, consider a restricted class of algorithms.

### Definition

An algorithm is in the Comparison Model if it access the data only using the following operations:

- comparing two elements.
- moving elements around.

This is a common example of model.
Other models are used for more specific problems.

### Example

- Selection Sort is in the Comparison Model: True
- Merge Sort is in the Comparison Model: True
- Counting Sort is in the Comparison Model: False

# Comparison-based Search

Consider a sorted array $A$ of $n$ elements, and an element $x$.
Is $x$ in $A$?

$$x = 76$$

| | | | | |
|---|---|---|---|---|
| 3 | 18 | $\cdots$ | 169 | 453 |

Which algorithm to use?

- Binary Search: $O(\log n)$ (in the worst case and on average).
- Interpolation Search
  - on average: $O(\log \log n)$
  - in the worst case: $O(n)$
  - Not in the comparison model!

How many comparisons are necessary to answer? $\Omega(\log n)$

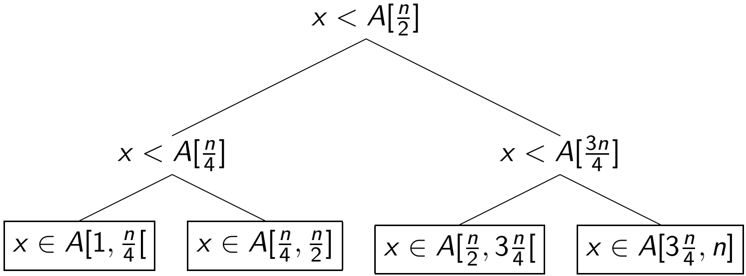
# Deterministic Lower Bound: Adversary Strategy

Define a game between the deterministic algorithm $A$ and an adversary $E$.

- $A$ doesn't see the array, compares $x$ to values of the array.
- $E$ can choose the content as long as it is coherent with its past answers.
- $E$ maintains two positions $l$ and $r$ s.t. $A[l] < x < A[r]$.
- When $A$ asks if $x > A[k]$:
  - $E$ forces $A$ to search in the largest interval:
  - $E$ answers yes if $r - k > k - l$, no otherwise
- $A$ will have to perform $1 + \log_2 n$ comparisons. $\square$

# Det/Rand Lower Bound: Decision Tree

Consider the comparisons performed by a binary search:




They form a Decision Tree: a binary tree where the $2n+1$ leaves corresponds to a possible result.
For any such tree:

- height is $\Omega(\lg n)$ $\Rightarrow$ deterministic lower bound.
- average branch length is $\Omega(\lg n)$ $\Rightarrow$ randomized lower bound.

# Randomized Lower Bounds

In most interesting problems, the tree is not as nice.

## Example

Consider an element $x$, and $k$ sorted array $A_1, \ldots, A_k$ of $n$ elements each. Is $x$ in $A_1 \cap \ldots A_k$?

$$x = 76$$

| | | | | | |
|---|---|---|---|---|---|
| $A$ | 3 | 18 | $\cdots$ | 169 | 453 |
| $B$ | 22 | 37 | $\cdots$ | 102 | 103 |
| $C$ | 1 | 12 | $\cdots$ | 982 | 1200 |
| $D$ | 5 | 12 | $\cdots$ | 300 | 389 |

This is called the (elementary) intersection problem.
It has application in databases, and search engines.

# Minimax Principle, Yao-von Neumann Theorem

## Theorem (Yao Principle)

*The average complexity of the best randomized algorithm*
*on the worst instance*
*is equal to*
*the complexity of the best deterministic algorithm*
*on the worst probability distribution.*

$\Rightarrow$ Just define the worst probability distribution you can.

### Example

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $A$ | 3 | 18 | $\cdots$ | 76 | $\cdots$ | 169 | 453 |
| $B$ | 22 | 37 | $\cdots$ | 76 | $\cdots$ | 102 | 103 |
| $C$ | 1 | 12 | $\cdots$ | 76 | $\cdots$ | 982 | 1200 |
| $D$ | 5 | 12 | $\cdots$ | 77 | $\cdots$ | 300 | 389 |

$$\Omega(k \log(n))$$

# Lower bound for Comparison-based Sorting

Many Sorting Algorithms:

| Sort | Running time | Analysis |
|---|---|---|
| Selection Sort | $\Theta(n^2)$ | worst-case |
| Merge Sort | $\Theta(n \log n)$ | worst-case |
| Heap Sort | $\Theta(n \log n)$ | worst-case |
| Deterministic Quick Sort | $\Theta(n^2)$ | worst-case |
| Randomized Quick Sort | $\Theta(n \log n)$ | expected |

**Theorem**

*Any correct comparison-based sorting algorithm requires at least $\Omega(n \log n)$ comparison operations.*

# Proof: lower bound

Proof.

- Any algorithm sorting an array of $n$ elements will have $n!$ possible outputs.
- The corresponding decision tree has $n!$ leaves, and height at least $\log(n!)$.
- $\log(n!) \approx n\lg(n) - n + \frac{\ln n}{2} + \frac{\ln 2\pi}{2}$ [Stirling].

Any comparison based algorithm performs $\Omega(n \log n)$ comparisons to sort an array of $n$ elements. $\square$

Similar technique used to prove a lower bound on *randomized* algorithms.

# Summary

- Many Techniques to prove Lower Bounds and optimality:
  - adversary strategy,
  - decision tree,
  - minimax principle (worse distribution).
- comparison-based search is $\Omega(\log n)$.
- comparison-based sort is $\Omega(n \log n)$.

# Outline

# Summary about Algorithmic Analysis

| | Topic | Concept(s) |
|---|---|---|
| 1 | Math Fundamentals | $\log, \sum$ |
| 2 | Asymptotic Notations | $O(), \Omega(), \Theta(), o(), \omega()$ |
| | Selection Sort | $O(n^2)$ |
| | Merge Sort | $O(n \lg n)$ |
| | Counting Sort | $O(n)$ |
| 3 | Average Case Analysis | $E(\sum_i X_i) = \sum_i E(X_i)$ |
| | Randomized Algorithms | prob. dist. on det. algos. |
| 4 | Simple Lower Bounds | Decision Tree |
| | Application to Sorting | Comparison model, $\Omega(n \lg n)$ |

# Summary about Algorithmic Analysis

| | Topic | Concept(s) |
|---|---|---|
| 1 | Math Fundamentals | $\log, \sum$ |
| 2 | Asymptotic Notations | $O(), \Omega(), \Theta(), o(), \omega()$ |
| | Selection Sort | $O(n^2)$ |
| | Merge Sort | $O(n \lg n)$ |
| | Counting Sort | $O(n)$ |
| 3 | Average Case Analysis | $E(\sum_i X_i) = \sum_i E(X_i)$ |
| | Randomized Algorithms | prob. dist. on det. algos. |
| 4 | Simple Lower Bounds | Decision Tree |
| | Application to Sorting | Comparison model, $\Omega(n \lg n)$ |

# Summary about Algorithmic Analysis (cont')

- Asymptotic Analysis simplifies the study,
- Worst case and Average case analysis simplify it further.
- Randomized Algorithms is linked to Average Analysis.
- Many Techniques to prove Lower Bounds and optimality.

# Reading Materials

| | Topic | GT | CLRS |
|---|---|---|---|
| 1 | Math Fundamentals | 4-9, 21-24 | 5-12, 21-22, 51-56 |
| 2 | Asymptotic Notations | 13-20 | 15-26, 41-50 |
| | Selection Sort | - | 15-20 |
| | Merge Sort | 219-224, 263-273 | 62-66 |
| | Counting Sort | 235-238 | 168-170 |
| 3 | Average Case Analysis | 235-237 | 91-114, 145-164 |
| | Randomized Algorithms | 238 | (same) |
| 4 | Simple Lower Bounds | 239-240 | - |
| | Application to Sorting | (same) | 165-168 |

- GT = Algorithm Design, by Goodrich & Tamassia
- CLRS = Introduction to Algorithms, by Cormen, Leisersen, Rivest & Stein