# Episode

Sorting Lower Bound

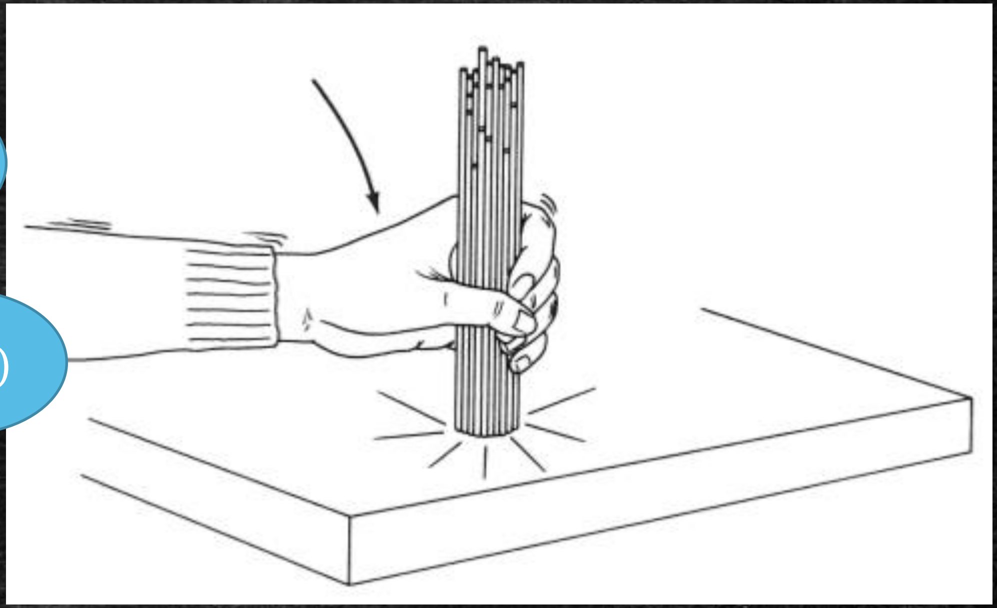
# Sorting Algorithms

- $\Theta(n^2)$
  - Selection Sort
  - Bubble Sort
  - Insertion Sort
- $\Theta(n \log n)$
  - Quick Sort
  - Merge Sort
  - Balance Tree
- Guide Question
  - **Can we do better?**

# Spaghetti Sort

$O(n)$

- For each number, **break** a piece of spaghetti whose length is that number. $O(1)$
- Take all spaghetti and **push** them against the table! $O(n)$
- Keep **touching** and **removing** spaghetti from the top by your other hand.



# Is it what we want?

- Need to think
  - What can we do?
  - What can not we do?

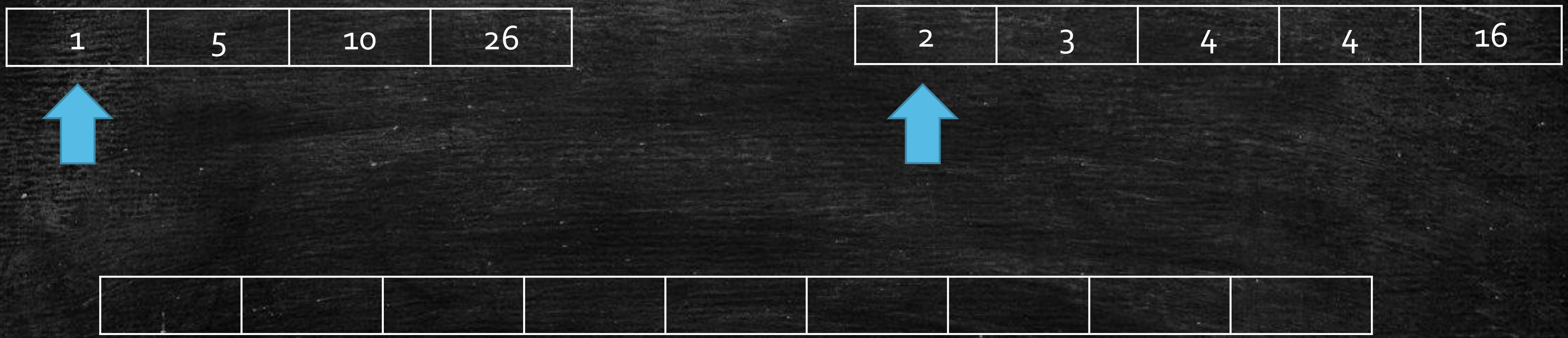
# Is it what we want?

- Need to think
  - ~~What can we do?~~
  - ~~What can not we do?~~
  - What can computer do?
  - What can not computer do?
- A proper Computation Model!
  - Allowed Operations: Comparison
  - Not allowed Operations: Break spaghetti, push spaghetti, touch spaghetti.

# Comparison-based Sorting

- Only allowed operation: **comparison.**
- Can do
  - $Compare(a, b)$, answer $a > b$ or $b \geq a$.
- Can not do
  - Ask what $a$ is, what $b$ is.
- Examples
  - Merge Sort
  - Insertion Sort
  - Quick Sort
  - ……

# Recall Merge Sort

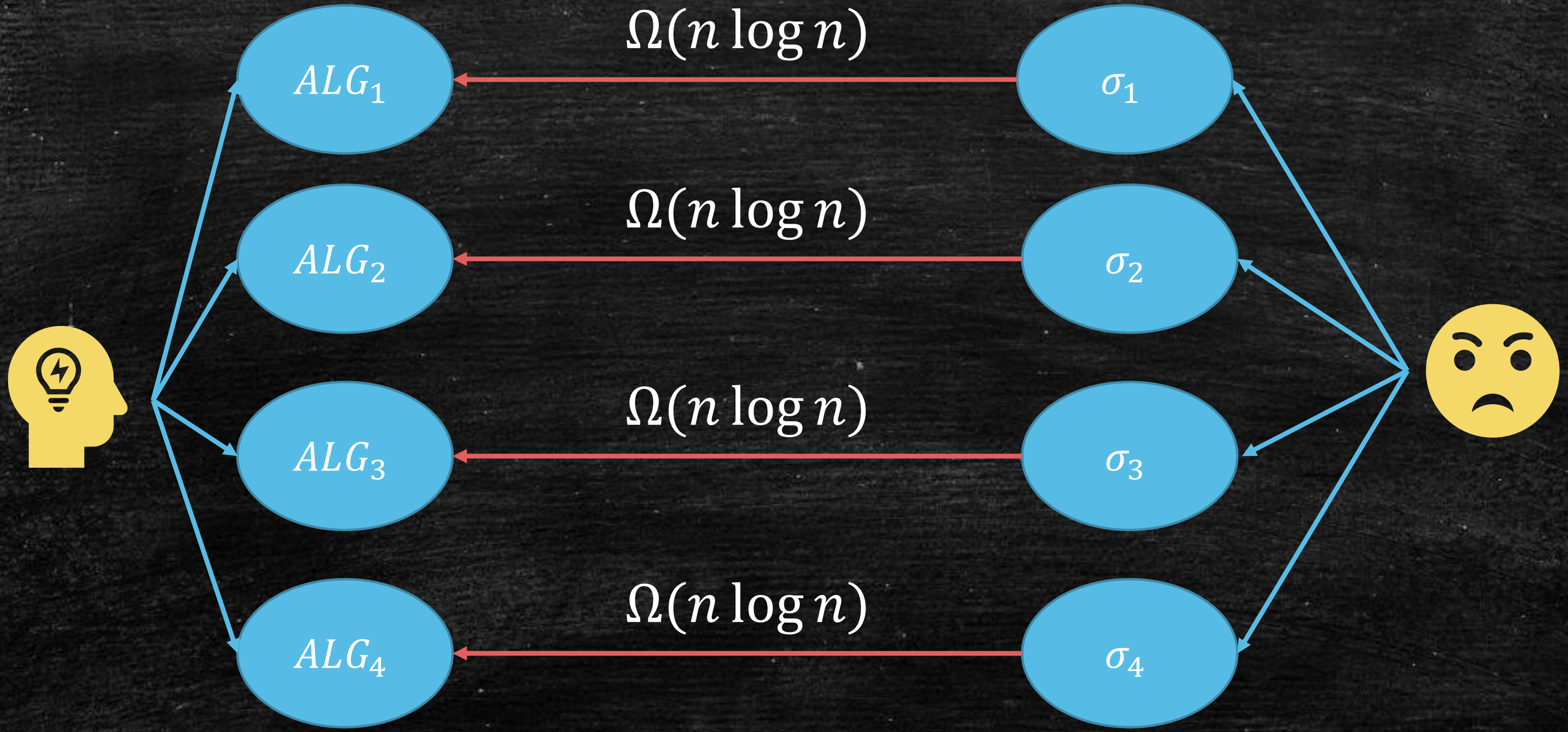



- Plan
  - Maintain 2 pointers $i = 1, j = 1$
  - Repeat
    - Append $\min\{a_i, b_j\}$ to $C$
    - If $a_i$ is smaller, then move $i$ to $i + 1$; If $b_j$ is smaller, then move $j$ to $j + 1$.
    - Break if $i > n$ or $j > m$
  - Append the reminder of the non-empty list to $C$

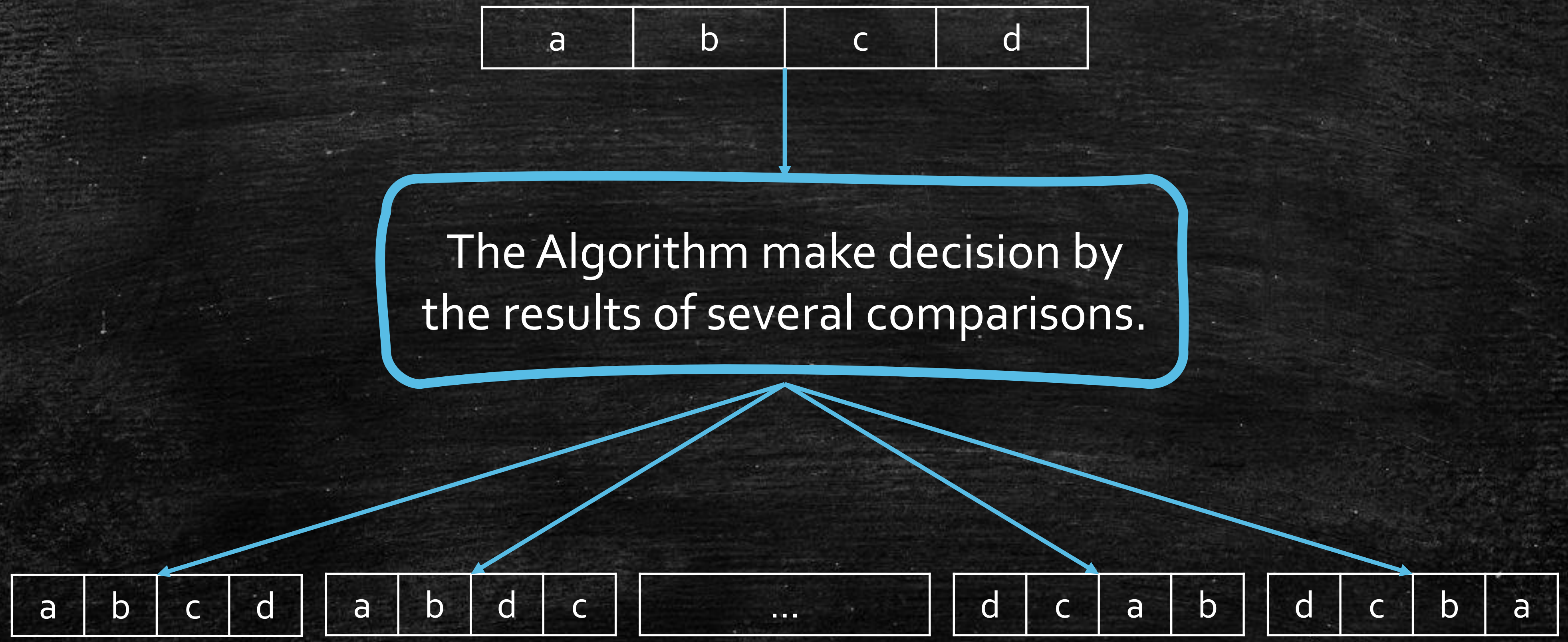
# How to prove lower bounds?

- Some Algorithm A want to say: A's time complexity is something better than $O(n\log n)$.
  - Prove: For all input $\sigma$ with size $n$, $T_A(\sigma) \le T(n) = O(g(n))$.
  - $g(n) = o(n\log n)$, like $g(n) = n$.
- Disprove it!
- I want to say: For all Algorithm A, there is some input show that $A$ can not do better than $O(n\log n)$!
  - Prove: Exists input $\sigma$ with size $n$, $T_A(\sigma) \ge T(n) = \Omega(n\log n)$.
- Play as an **adversary**!

# Play as An **Adversary**!

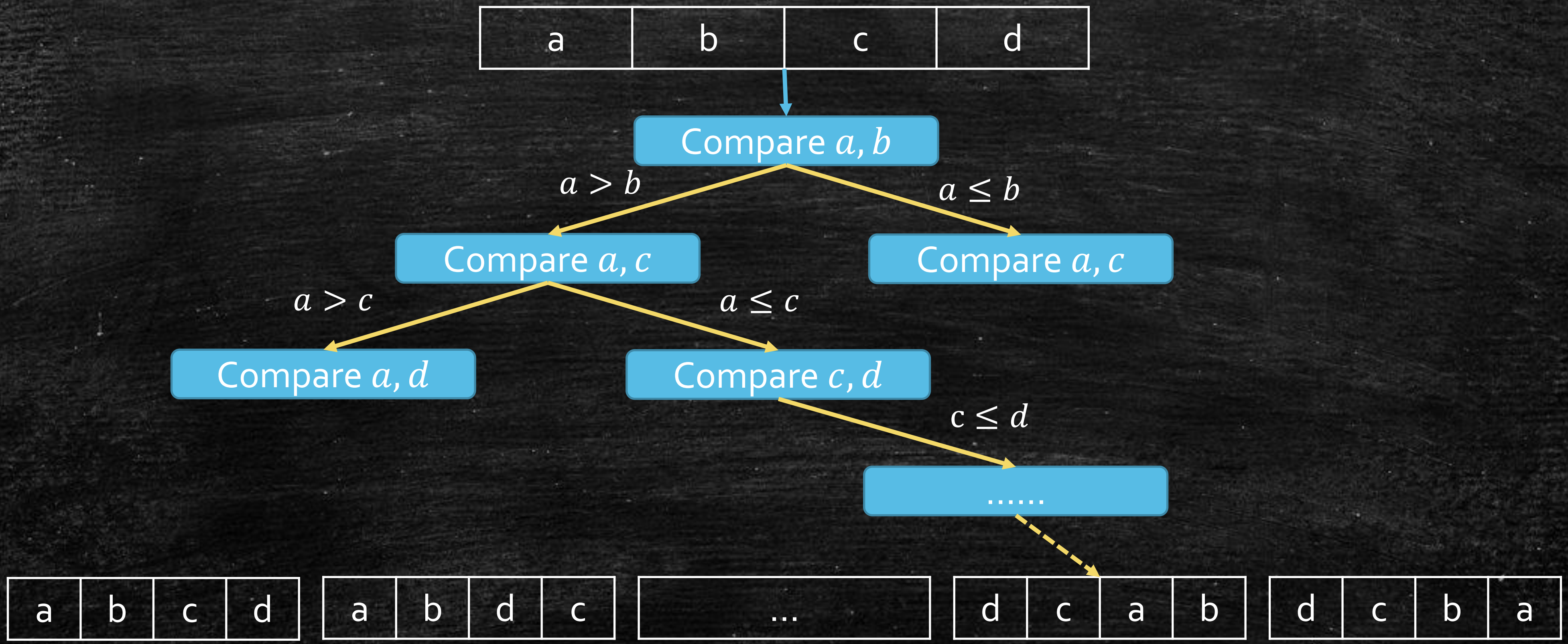


# Comparison-based Sorting




- How many possible ordering?

# Comparison-based Sorting




- How many possible ordering? --- $n!$

# Comparison-based Sorting: Time Complexity

1. Comparison-based Sorting forms a binary tree.
2. It should have at least $n!$ leaves.
3. Adversary: the **longest** path from root to leaves.
4. Best deterministic algorithm make the tree shallowest.
5. Shallowest tree is a balanced tree.
6. Height: $\log(n!)$

# Last step!

- The lemma we have
  - Any deterministic comparison-based algorithms must take $\log(n!)$ steps to sort an array in the worst case.
- The theorem we want
  - Any deterministic algorithm comparison-based algorithms must take $\Omega(n \log n)$ time.

**Proof**

$$\begin{aligned}
\log(n!) &= \log 1 + \log 2 + \log 3 + \cdots + \log n \\
&\geq \log\left(\frac{n}{2} + 1\right) + \log\left(\frac{n}{2} + 2\right) + \cdots + \log n \\
&\geq \frac{n}{2} \log \frac{n}{2} = \frac{n}{2} \log n - \frac{n}{2} \log 2 \\
&= \Omega(n \log n)
\end{aligned}$$

# Good News!

Merge Sort is the Optimal Deterministic Comparison-based Algorithm

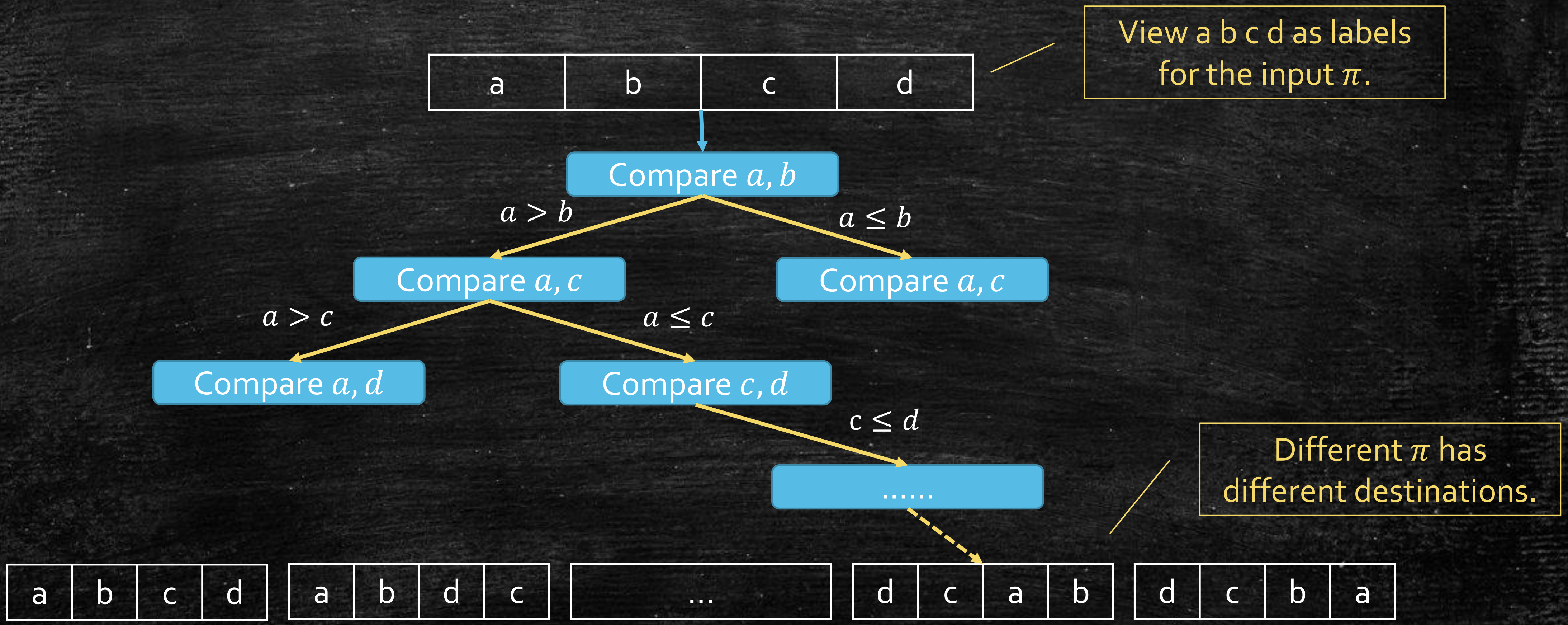
# What about random algorithms?

# Before that.....

# Average Case for Deterministic Algorithms

- **Input**
  - $n$ integers.
- **Shuffle**
  - Uniformly random a permutation $\pi \in \Pi$.
- **Sort**
  - Run the deterministic algorithm on $\pi$.
- Average Comparisons
  - $E[T(n)] = \frac{1}{n!}\sum_{\pi \in \Pi}(T(\pi))$

# Comparison-based Sorting

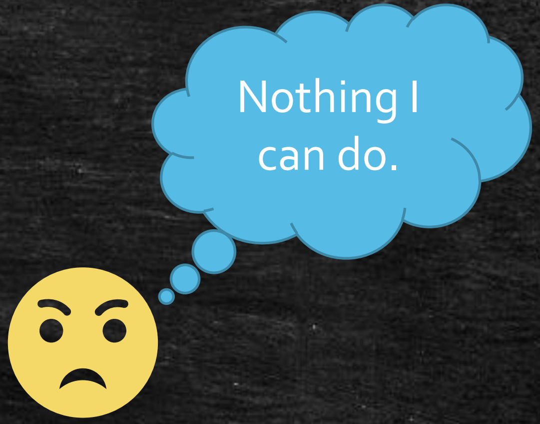



- How many possible ordering? --- $n!$

# How to relate the Comparisons to the tree?

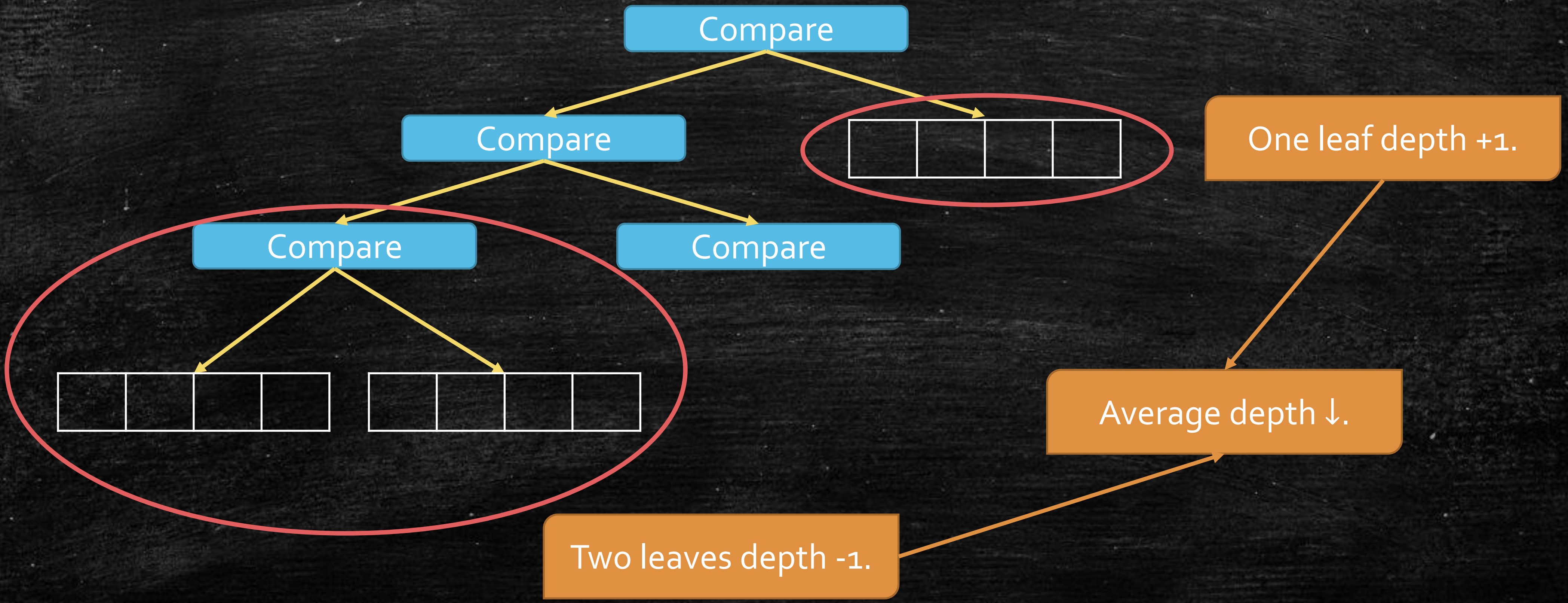
- Worst Case Comparisons
  - Longest path from root to leaves.
  - Balanced tree: The **best** we can do.

- Average Comparisons
  - Average length of paths from root to leaves.
  - Balanced tree: Still the **best**?



# Balanced tree: Still the best?

- Unbalanced tree: What if we try to make it balanced?



# Balanced tree is still the best!

Best Average Comparisons: $\boldsymbol{\log n!} = \boldsymbol{\Omega(n \log n)}$

# The Input-ALG Game

- Two Players
  - One give Input Permutation.
  - One give ALG.

| | $ALG_1$ | $ALG_2$ | $ALG_3$ | $ALG_4$ |
|---|---|---|---|---|
| Input = $\pi_1$ | $\log n!$ | $\log n!$ | $1$ | $\log n!$ |
| Input = $\pi_2$ | $1$ | $n^2$ | $\log n!$ | $\log n!$ |
| Input = $\pi_3$ | $n^2$ | $\log n!$ | $1$ | $\log n!$ |
| Input = $\pi_4$ | $\log n!$ | $1$ | $\log n!$ | $\log n!$ |

# Deterministic Lower Bound: Worst Case

- For Each ALG
  - A row has cost $\geq \log n!$

| | $ALG_1$ | $ALG_2$ | $ALG_3$ | $ALG_4$ |
|---|---|---|---|---|
| Input = $\pi_1$ | $\log n!$ | $\log n!$ | $1$ | $\log n!$ |
| Input = $\pi_2$ | $1$ | $n^2$ | $\log n!$ | $\log n!$ |
| Input = $\pi_3$ | $n^2$ | $\log n!$ | $1$ | $\log n!$ |
| Input = $\pi_4$ | $\log n!$ | $1$ | $\log n!$ | $\log n!$ |

# Deterministic Lower Bound: Average Case

- For Each ALG
  - The row $(\frac{1}{4}\pi_1, \frac{1}{4}\pi_2, \frac{1}{4}\pi_3, \frac{1}{4}\pi_4)$ has cost $\geq \log n!$

| | $ALG_1$ | $ALG_2$ | $ALG_3$ | $ALG_4$ |
|---|---|---|---|---|
| Input = $\pi_1$ | $\log n!$ | $\log n!$ | $1$ | $\log n!$ |
| Input = $\pi_2$ | $1$ | $n^2$ | $\log n!$ | $\log n!$ |
| Input = $\pi_3$ | $n^2$ | $\log n!$ | $1$ | $\log n!$ |
| Input = $\pi_4$ | $\log n!$ | $1$ | $\log n!$ | $\log n!$ |
| Input =$(\frac{1}{4}\pi_1, \frac{1}{4}\pi_2, \frac{1}{4}\pi_3, \frac{1}{4}\pi_4)$ | $\geq \log n!$ | $\geq \log n!$ | $\geq \log n!$ | $\geq \log n!$ |

# What is a randomized algorithm?

- Randomized is a **distribution** of all deterministic algorithms.
  - Randomized Algorithm is a deterministic algorithms with a random bit tape.
  - Fix a sample "$s$" of the random bit tape → An original deterministic algorithm $ALG_s$.

Random $ALG$ = Det $ALG$ +

| | | | | | |
|---|---|---|---|---|---|
| 5 | 3 | 2 | 100 | 5 | …… |

# What is a randomized algorithm?

Random $ALG$ = $ALG_1$ with 10%

$ALG_2$ with 30%

$ALG_3$ with 30%

$ALG_4$ with 30%

# Expected Running Time

- Randomized is a **distribution** of all deterministic algorithms.
  - Randomized Algorithm is a deterministic algorithms with a random bit tape.
  - Fix a sample "$s$" of the random bit tape → An original deterministic algorithm $ALG_s$.
    随机的一条纸带
- The expected running time of Randomized Algorithms
  - Input $\pi$
  - $E_s\left[T_{ALG_s}(\pi)\right]$
  - Consider the worst case!

# Randomized Lower Bound: Average Case

- For the column $(p_1 ALG_1, p_2 ALG_2, p_3 ALG_3, p_4 ALG_4)$
  - The row $(\frac{1}{4}\pi_1, \frac{1}{4}\pi_2, \frac{1}{4}\pi_3, \frac{1}{4}\pi_4)$ has cost $\geq \log n!$

| | $ALG_1$ | $ALG_2$ | $ALG_3$ | $ALG_4$ | $(p_1 A_1, p_2 A_2, p_3 A_3, p_4 A_4)$ |
|---|---|---|---|---|---|
| Input = $\pi_1$ | $\log n!$ | $\log n!$ | $1$ | $\log n!$ | |
| Input = $\pi_2$ | $1$ | $n^2$ | $\log n!$ | $\log n!$ | |
| Input = $\pi_3$ | $n^2$ | $\log n!$ | $1$ | $\log n!$ | |
| Input = $\pi_4$ | $\log n!$ | $1$ | $\log n!$ | $\log n!$ | |
| Input =$(\frac{1}{4}\pi_1, \frac{1}{4}\pi_2, \frac{1}{4}\pi_3, \frac{1}{4}\pi_4)$ | $\geq \log n!$ | $\geq \log n!$ | $\geq \log n!$ | $\geq \log n!$ | |

# Randomized Lower Bound: Average Case

- For the column $(p_1 ALG_1, p_2 ALG_2, p_3 ALG_3, p_4 ALG_4)$
  - The row $(\frac{1}{4}\pi_1, \frac{1}{4}\pi_2, \frac{1}{4}\pi_3, \frac{1}{4}\pi_4)$ has cost $\geq \log n!$

| | $ALG_1$ | $ALG_2$ | $ALG_3$ | $ALG_4$ | $(p_1A_1, p_2A_2, p_3A_3, p_4A_4)$ |
|---|---|---|---|---|---|
| Input = $\pi_1$ | $\log n!$ | $\log n!$ | $1$ | $\log n!$ | |
| Input = $\pi_2$ | $1$ | $n^2$ | $\log n!$ | $\log n!$ | |
| Input = $\pi_3$ | $n^2$ | $\log n!$ | $1$ | $\log n!$ | |
| Input = $\pi_4$ | $\log n!$ | $1$ | $\log n!$ | $\log n!$ | |
| Input =$(\frac{1}{4}\pi_1, \frac{1}{4}\pi_2, \frac{1}{4}\pi_3, \frac{1}{4}\pi_4)$ | $\geq \log n!$ | $\geq \log n!$ | $\geq \log n!$ | $\geq \log n!$ | $\geq \log n!$ |

# Randomized Lower Bound: Average Case

- For the column $(p_1ALG_1, p_2ALG_2, p_3ALG_3, p_4ALG_4)$
  - The row $(\frac{1}{4}\pi_1, \frac{1}{4}\pi_2, \frac{1}{4}\pi_3, \frac{1}{4}\pi_4)$ has cost $\geq \log n!$

| | $ALG_1$ | $ALG_2$ | $ALG_3$ | $ALG_4$ | $(p_1A_1, p_2A_2, p_3A_3, p_4A_4)$ |
|---|---|---|---|---|---|
| Input = $\pi_1$ | $\log n!$ | $\log n!$ | $1$ | $\log n!$ | |
| Input = $\pi_2$ | $1$ | $n^2$ | $\log n!$ | $\log n!$ | |
| Input = $\pi_3$ | $n^2$ | $\log n!$ | $1$ | $\log n!$ | $\geq \log n!$(the worst one) |
| Input = $\pi_4$ | $\log n!$ | $1$ | $\log n!$ | $\log n!$ | |
| Input =$(\frac{1}{4}\pi_1, \frac{1}{4}\pi_2, \frac{1}{4}\pi_3, \frac{1}{4}\pi_4)$ | $\geq \log n!$ | $\geq \log n!$ | $\geq \log n!$ | $\geq \log n!$ | $\geq \log n!$ |

# Conclusion

- For any randomized algorithm, there is an input $\pi$, such that $E\big(T(\pi)\big) \geq \log n! = \Omega(n \log n)$

# More Questions

- Do we have **linear time** sorting algorithms that is not comparison-based?
  - Yes! But with some restriction.