Lecture 08: Sorting Lower Bounds

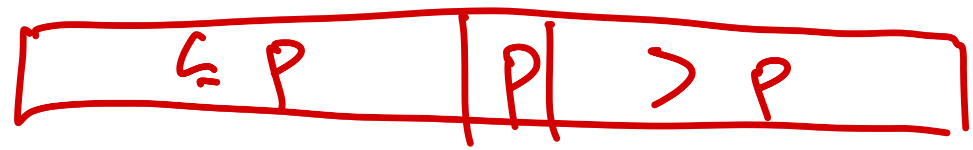
COSC 311 *Algorithms*, Fall 2022

Announcements

1. Homework 2 Draft Posted
2. Homework Late Days

Overview

1. QuickSort Again
2. Sorting Lower Bound



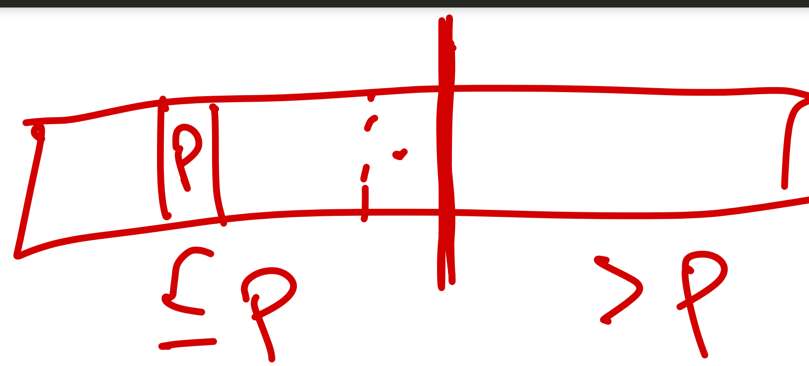
Last Time

```
QuickSort(a, i, j):
  if j - i <= 1 then
    return
  endif
  p <- GetPivot(a, i, j) # select a pivot
  k <- Split(a, i, j, p)
  QuickSort(a, i, k-1)
  QuickSort(a, k+1, j)
```

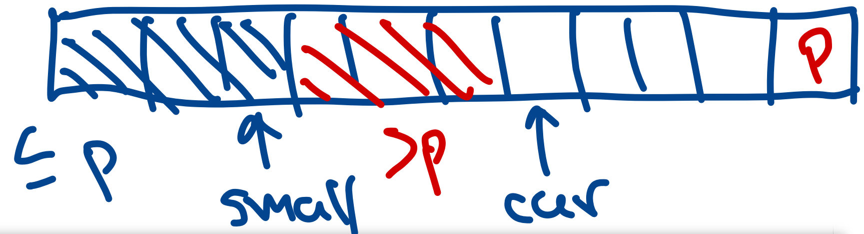
Split Before

```
Split(a, i, j, p):
  left <- i, right <- j
  while left < right do
    if a[left] > p and a[right] <= p then
       swap(a, left, right)
       left++, right--
    else
      if a[left] <= p then left++
      if a[right] > p then right--
    endif
  endwhile
  return right
```

What is the problem?



Split Updated




```
Split(a, i, j, pIndex):
  pivot <- a[pIndex]
  swap(a, pIndex, j);          # move pivot to last index
  small <- i - 1;
  for cur in range i..j do
    if a[cur] <= pivot then
      small <- small + 1
      swap(a, small, cur)
    endif
  endfor
  return small
```

Invariants:

1. elements ≤ pivot are at indices [i..small]
2. elements > pivot are at indices [small+1..cur]

So far

1. $O(n^2)$ sorting: SelectionSort, InsertionSort, BubbleSort
2. $O(n^2)$ worst-case, $O(n \log n)$ average case: QuickSort
3. $O(n \log n)$ worst-case: MergeSort

Question. Can we do better?

Lecture Ticket

$\geq c \cdot n$ for large n

some const.

Sorting arrays of size n requires $\Omega(n)$ operations.

Why?

$n, n-1, \ldots, 2, 1$

Today's Lower Bound

Any algorithm that accesses and modifies arrays using only compare and swap operations requires $\Omega(n \log n)$ comparisons.

- MergeSort is (asymptoticaly) optimal?

Exercise. Imp. MS w/ only compare & swap

can swap between arrays

Ingredients of Lower Bound

Consider. Arrays are *permutations* of $1, 2, \ldots, n$

= # $1 \ldots n$ in array, every val appears once

Main idea. a and b are distinct arrays and A is an algorithm

1. A **distinguishes** a and b if $A(a)$ and $A(b)$ both make a call to compare$(\cdot, i, j)$ with compare$(a, i, j) \neq$ compare(b, i, j)
   and $a \neq b$
2. if A does not distinguish a and b, then it does not sort *both* a and b
   $\leq c \cdot n \log n$
3. If A performs too few compare operations, then it cannot distinguish all arrays of size n
   - $\implies$ A does not sort all arrays of size n

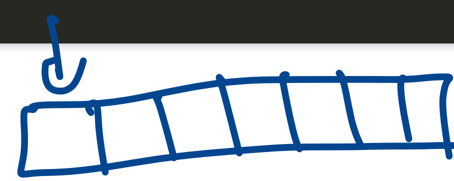
Decision Trees I

Consider:

1. Fixed sorting algorithm A

```
InsertionSort(a):
  for i = 2 to n do
    j <- i
    while j > 1 and compare(a, j-1, j) do
      swap(a, j, j-1);   j ← j-1
    endwhile
  endfor
```

2. Fixed set of inputs of size n



- S_n = permutations of $1, 2, \ldots, n$
- How many arrays in S_n? $n! = n \cdot (n-1) \cdot (n-2) \cdots 1$

Decision Trees II

Follow execution of A on all inputs from S_n

Define a binary tree:

1. each node corresponds to a single compare operation
2. each node has two children corresponding to two possible outcomes of compare

Form this tree for all comparisons made on all inputs in S_n

- label each node with inputs consistent with all compare outcomes

Decision Tree Example, n = 3

```
for i = 2 to n do
| j <- i
| while j > 1 and compare(a, j-1, j) do
| | swap(a, j, j-1)
```

; $j \leftarrow j-1$

What is first comparison?

$S_3 = \{123,$
$132,$
$213,$
$231,$
$312,$
$321\}$

F

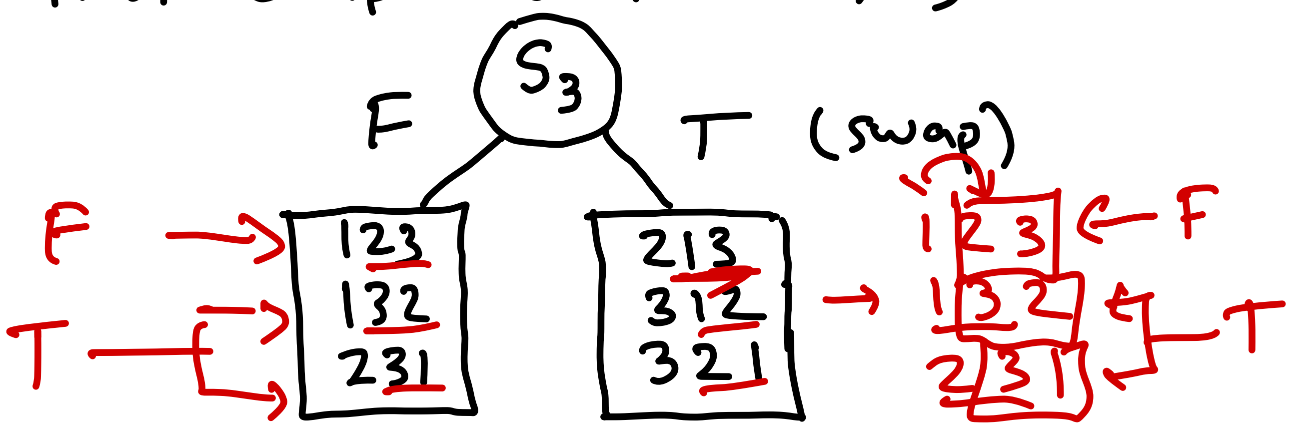
T

Decision Tree Example, n = 3

```
for i = 2 to n do
| j <- i
| while j > 1 and compare(a, j-1, j) do
| | swap(a, j, j-1); j ← j-1
```

What is first comparison? (1, 2)

$S_3 = \{123, 132, 213, 231, 312, 321\}$

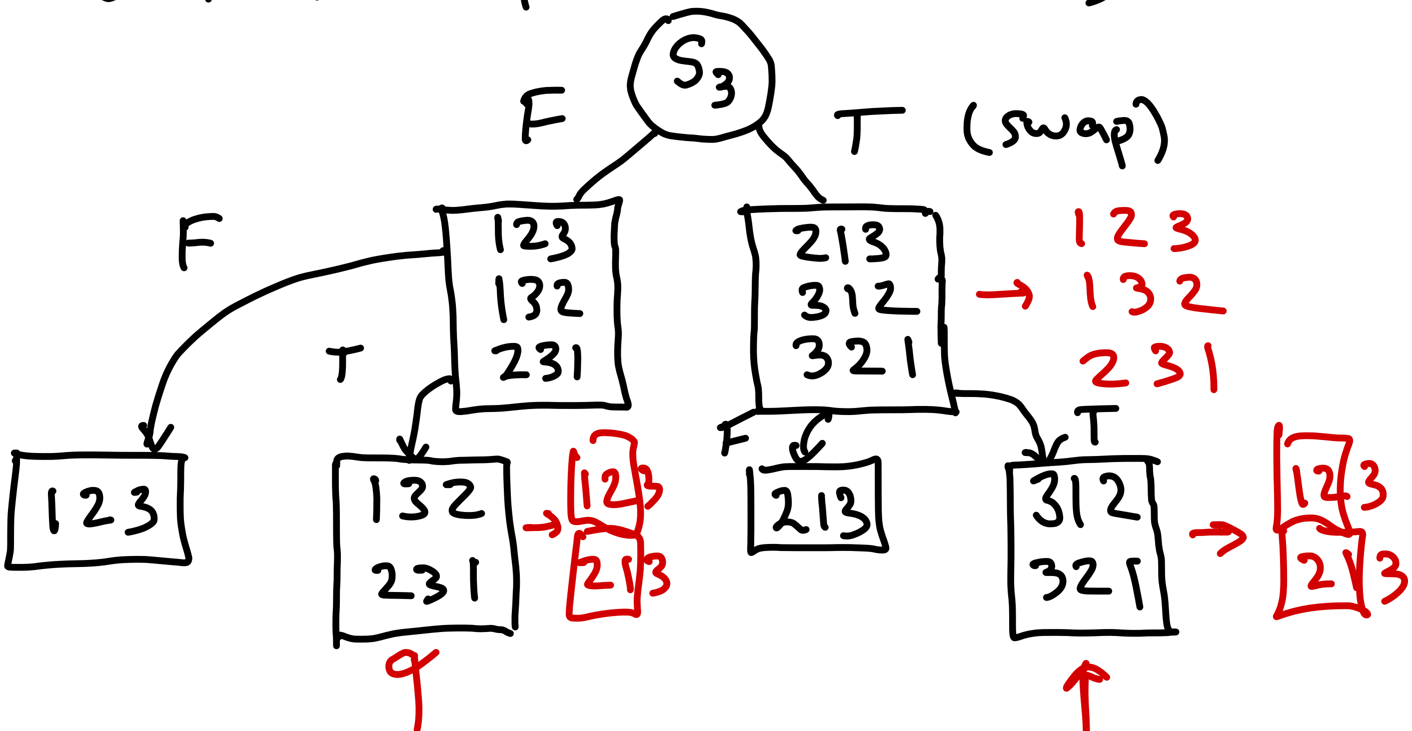


Decision Tree Example, n = 3

```
for i = 2 to n do
| j <- i
| while j > 1 and compare(a, j-1, j) do
| | swap(a, j, j-1)
```

What is first comparison? (1, 2)

$S_3 = \{123, 132, 213, 231, 312, 321\}$

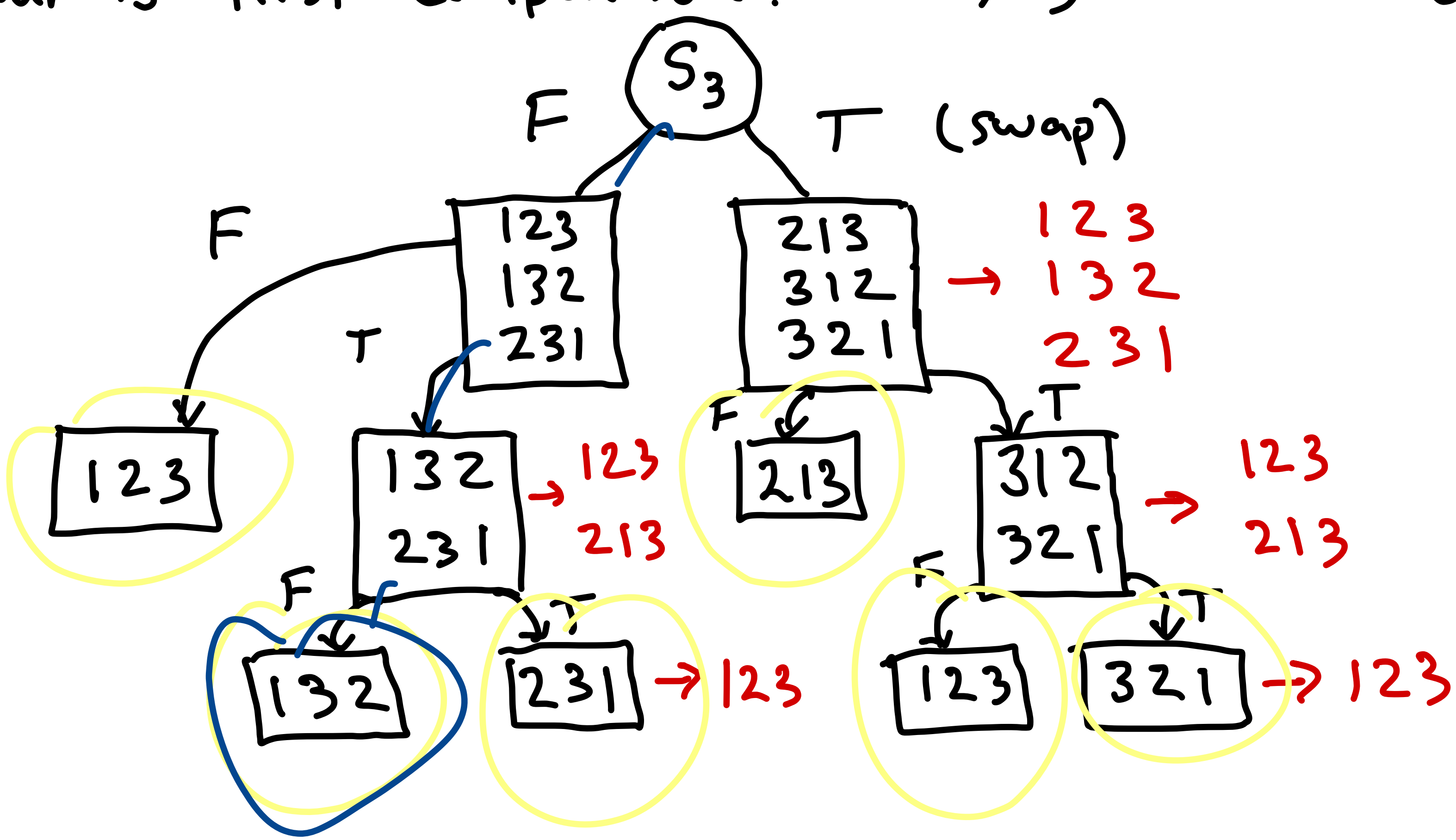


Decision Tree Example, n = 3

```
for i = 2 to n do
| j <- i
| while j > 1 and compare(a, j-1, j) do
| | swap(a, j, j-1)
```

What is first comparison? (1, 2)

$S_3 = \{123, 132, 213, 231, 312, 321\}$



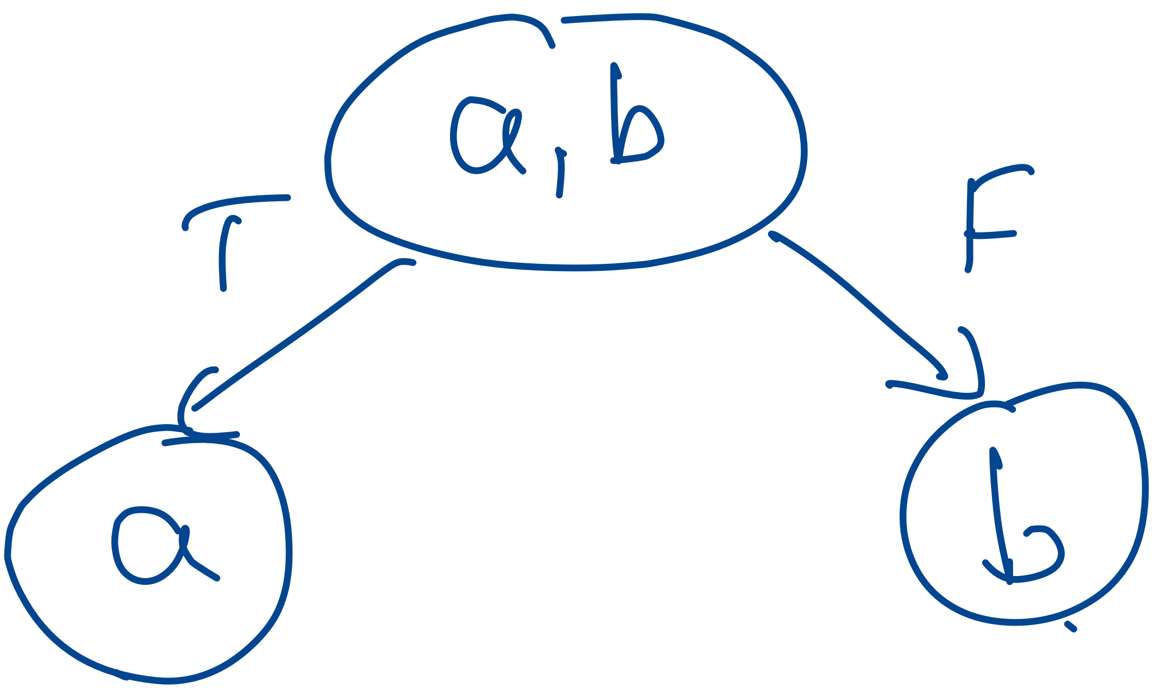
Decision Tree Depth

Question. What does the *depth* of decision tree correspond to in terms of execution of A?

= Max # of comparisons on any input.

Indistinguishability

Question. If A distinguishes a and b, what can we say about nodes labeled with a and b?

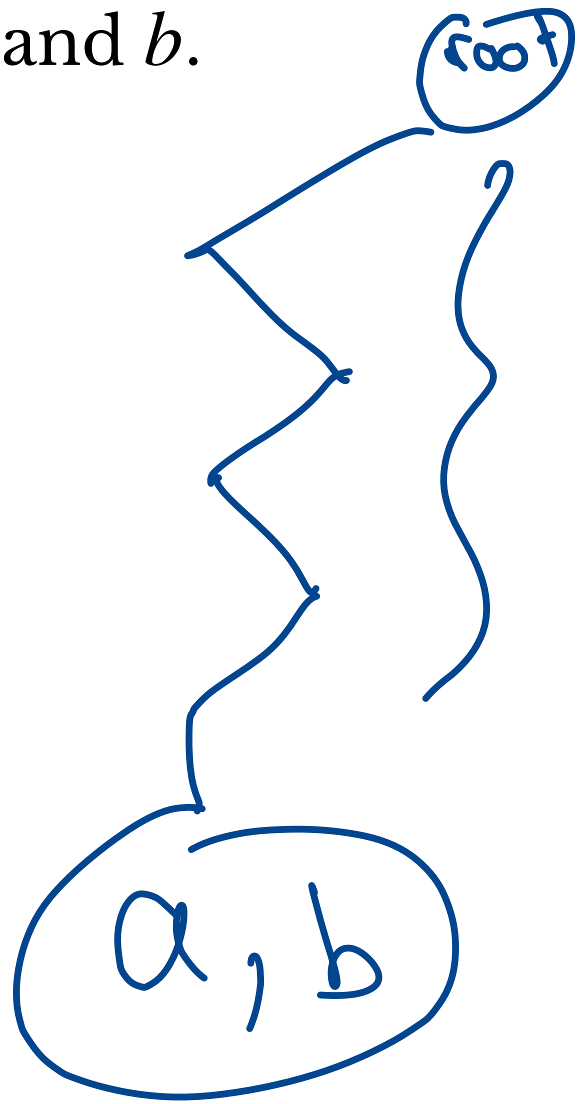


a and b in diff. leaves

Indistinguishability Claim

Claim. If A does not distinguish a and b with $a \neq b$, then A does not sort both a and b.

Why?



same operations performed $\Downarrow$ different outputs

Indistinguishability Consequence

Consequence. If A sorts all arrays of size n, then every leaf of A's decision tree is labeled with a single permutation array.

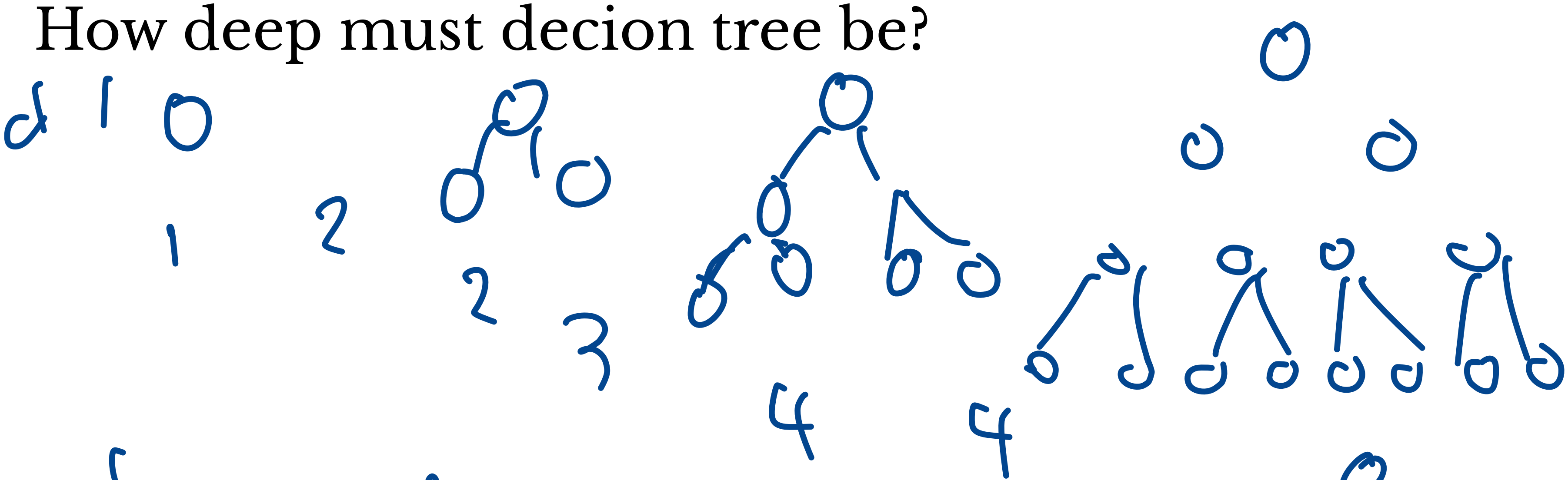
Why? True b/c prev. arg.

How Big is Decision Tree?

How many leaves must a correct decision tree have?

$\Rightarrow$ $n!$ leaves

How deep must decion tree be?



$$2^d \geq n! \iff d \geq \log(n!) \approx \Omega(n \log n)$$