Warm up

Show $\log(n!) = \Theta(n \log n)$

Hint: show $n! \leq n^n$

Hint 2: show $n! \geq \left(\frac{n}{2}\right)^{\frac{n}{2}}$

$\log n! = O(n \log n)$

$$\begin{array}{cccccccccccc}
n! = & n & \cdot & (n-1) & \cdot & (n-2) & \cdot & \ldots & \cdot & 2 & \cdot & 1 \\
 & \shortparallel & & \wedge & & \wedge & & & & \wedge & & \wedge \\
n^n = & n & \cdot & n & \cdot & n & \cdot & \ldots & \cdot & n & \cdot & n
\end{array}$$

$$n! \leq n^n$$
$$\Rightarrow \log(n!) \leq \log(n^n)$$
$$\Rightarrow \log(n!) \leq n \log n$$
$$\Rightarrow \log(n!) = O(n \log n)$$

$$\log n! = \Omega(n \log n)$$

$$n! = \overbrace{n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot \frac{n}{2} \cdot \left(\frac{n}{2}-1\right) \cdot \ldots \cdot 2 \cdot 1}^{n}$$

$$\vee \qquad \vee \qquad \vee \qquad \| \qquad \vee \qquad \vee \quad \|$$

$$\left(\frac{n}{2}\right)^{\frac{n}{2}} = \underbrace{\frac{n}{2} \cdot \frac{n}{2} \cdot \frac{n}{2} \cdot \ldots \cdot \frac{n}{2}}_{n/2} \cdot \underbrace{1 \cdot \ldots \cdot 1 \cdot 1}_{n/2}$$

$$n! \geq \left(\frac{n}{2}\right)^{\frac{n}{2}}$$

$$\Rightarrow \log(n!) \geq \log\left(\left(\frac{n}{2}\right)^{\frac{n}{2}}\right)$$

$$\Rightarrow \log(n!) \geq \frac{n}{2} \log \frac{n}{2}$$

$$\Rightarrow \log(n!) = \Omega(n \log n)$$

Today's Keywords

- Divide and Conquer
- Quicksort
- Decision Tree
- Worst case lower bound
- Sorting

CLRS Readings

- Chapter 7
- Chapter 8

Homeworks

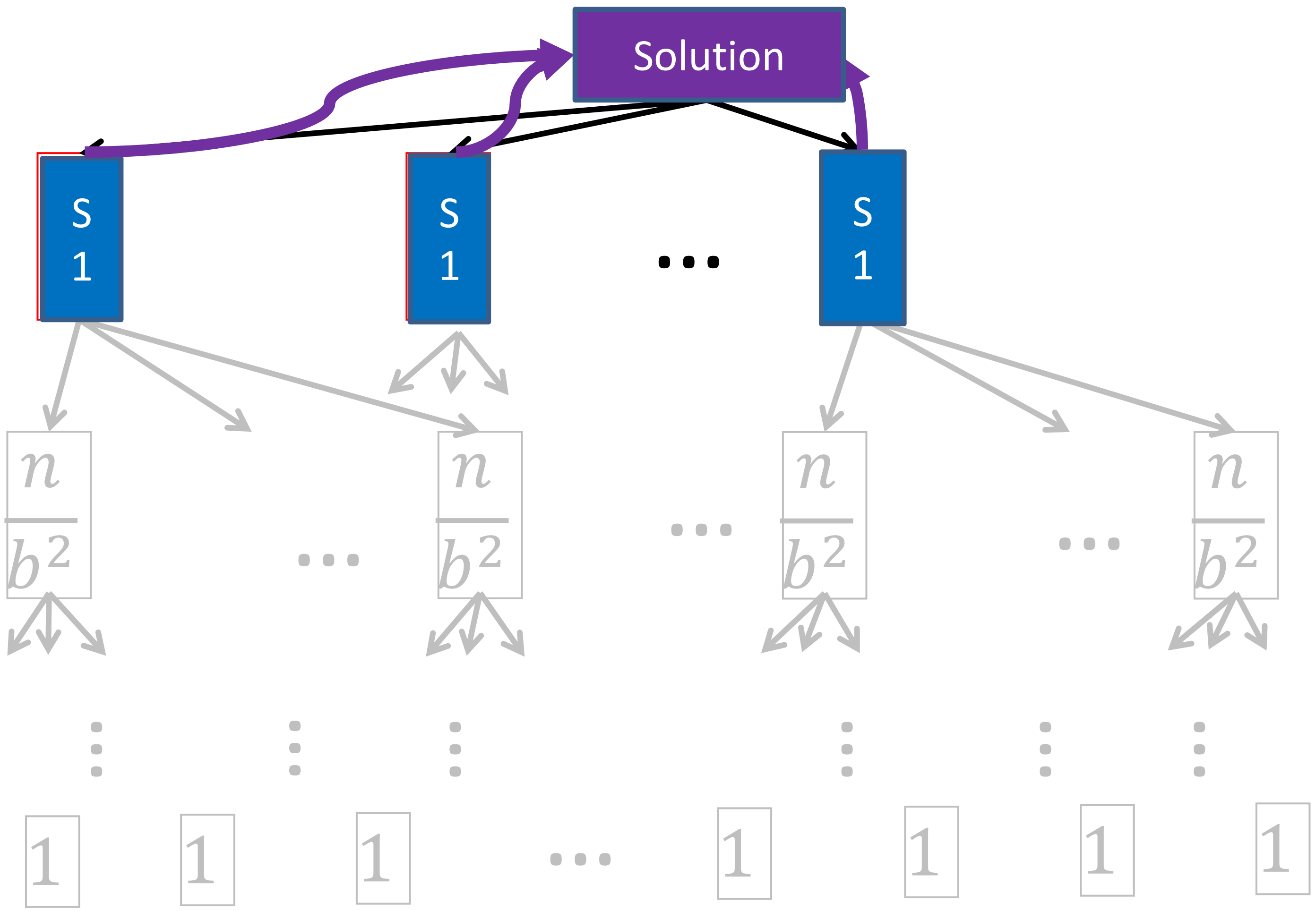
- HW4 due 11pm Thursday, February 27, 2020
  - Divide and Conquer and Sorting
  - Written (use LaTeX!)
  - Submit BOTH a pdf and a zip file (2 separate attachments)
- Regrade Office Hours
  - Fridays 2:30pm-3:30pm (Rice 210)
  - 2 weeks for HW0 regrades

Aside: Divide and Conquer

Generic Divide and Conquer Solution

```
def myDCalgo(problem):
        if baseCase(problem):
                solution = solve(problem) #brute force if necessary
                return solution
        subproblems[] = Divide(problem)
        for subproblem in subproblems:
                subsolutions.append(myDCalgo(subproblem))
        solution = Combine(subsolutions)
        return solution
```

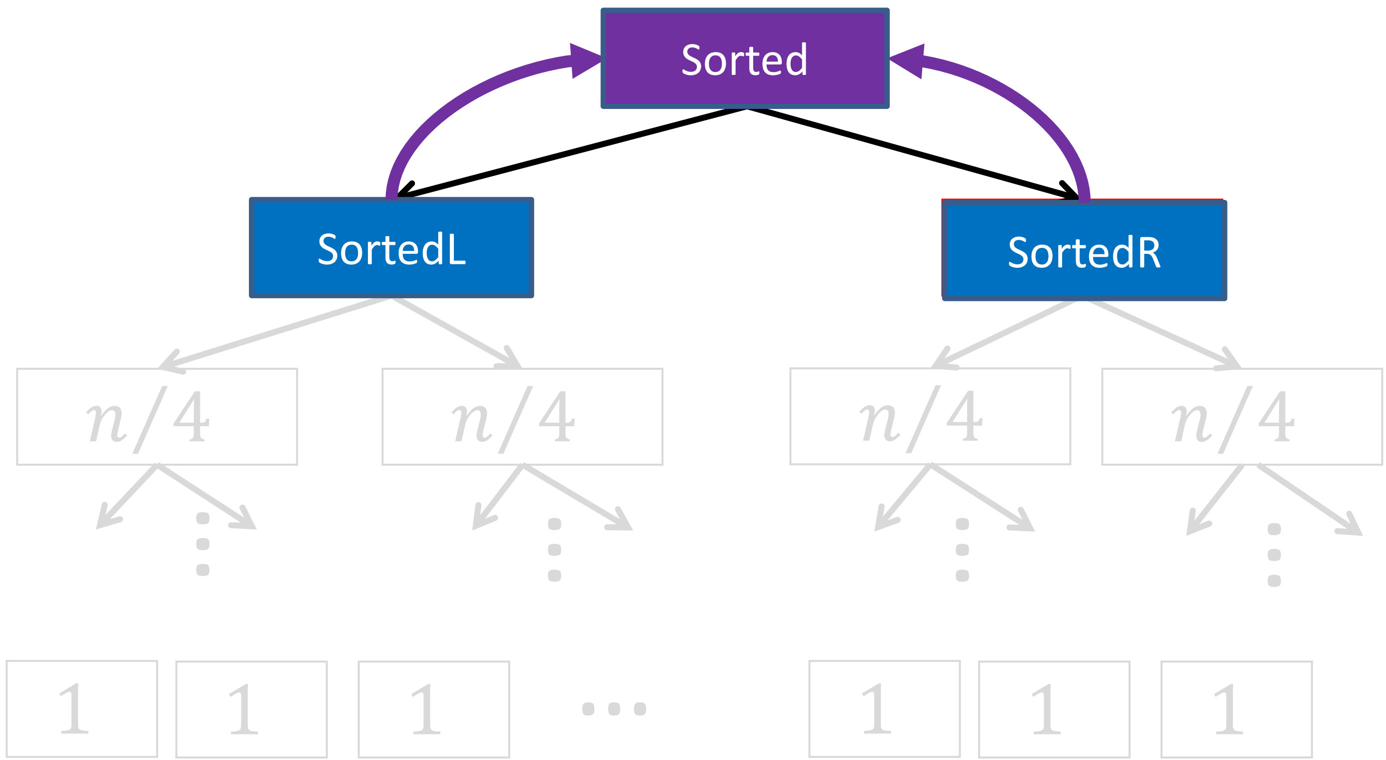
Generic Divide and Conquer Solution



MergeSort Divide and Conquer Solution

```
def mergesort(list):
    if list.length < 2:
        return list  #list of size 1 is sorted!
    {listL, listR} = Divide_by_median(list)
    for list in {listL, listR}:
        sortedSubLists.append(mergesort(list))
    solution = merge(sortedL, sortedR)
    return solution
```

MergeSort Divide and Conquer Solution



Back to Sorting!

Quicksort

- Idea: pick a pivot element, recursively sort two sublists around that element
- Divide: select an element p, Partition(p)
- Conquer: recursively sort left and right sublists
- Combine: Nothing!

Quicksort

- Idea: pick a pivot element using Quickselect + Median of Medians, recursively sort two sublists around that element
- Divide: select an element p, Partition(p)
- Conquer: recursively sort left and right sublists
- Combine: Nothing!

Guaranteed Quicksort

Using Quickselect, with a Median-of-Medians partition:

2	5	1	3	6	4	7	8	10	9	11	12

2	1	3	5	6	4	7	8	9	10	11	12

Then we divide in half each time

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$$

$$T(n) = \Theta(n \log n)$$

Is it worth it?

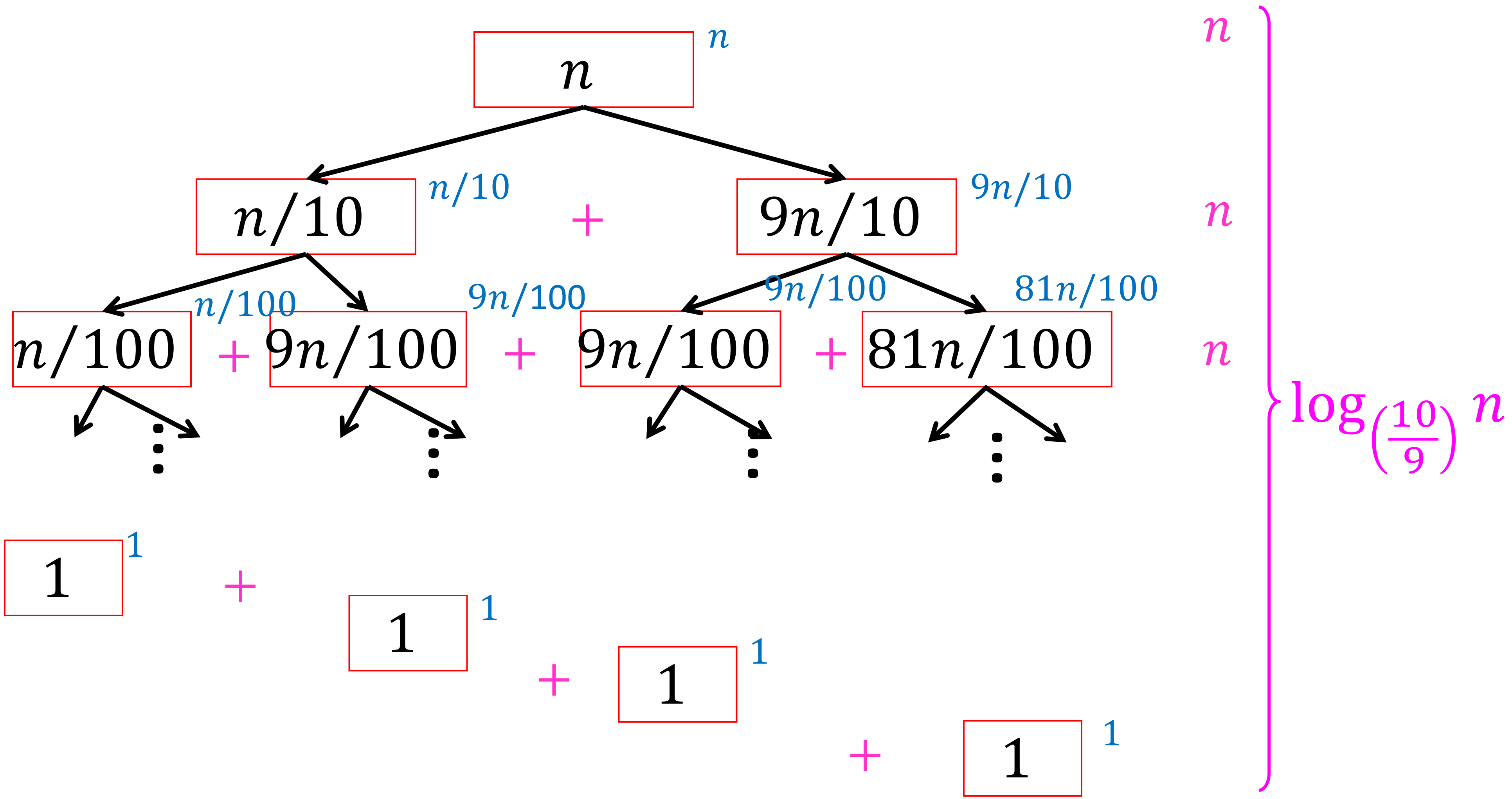
- Using Quickselect to pick median guarantees $\Theta(n \log n)$ run time
- Approach has very large constants
  - If you really want $\Theta(n \log n)$, better off using MergeSort
- Better approach: Random pivot
  - Very small constant (very fast algorithm)
  - Expected to run in $\Theta(n \log n)$ time
    - Why? Unbalanced partitions are very unlikely

Quicksort Run Time

If the pivot is always $\frac{n}{10}$th order statistic:

$$T(n) = T\left(\frac{n}{10}\right) + T\left(\frac{9n}{10}\right) + n$$

$$T(n) = T\left(\frac{n}{10}\right) + T\left(\frac{9n}{10}\right) + n$$



Quicksort Run Time

If the pivot is always $\frac{n}{10}$th order statistic:

$$T(n) = T\left(\frac{n}{10}\right) + T\left(\frac{9n}{10}\right) + n$$

$$T(n) = \Theta(n \log n)$$

Quicksort Run Time

If the pivot is always d^{th} order statistic:

1	5	2	3	6	4	7	8	10	9	11	12

1	2	3	5	6	4	7	8	10	9	11	12

Then we shorten by d each time

d is not a fraction of n

$$T(n) = T(n-d) + n$$

$$T(n) = O(n^2)$$

What's the probability of this occurring?

Probability of n^2 run time

We must consistently select pivot from within the first d terms

Probability first pivot is among d smallest: $\frac{d}{n}$

Probability second pivot is among d smallest: $\frac{d}{n-d}$

Probability all pivots are among d smallest:

$$\frac{d}{n} \cdot \frac{d}{n-d} \cdot \frac{d}{n-2d} \cdot \ldots \cdot \frac{d}{2d} \cdot 1 = \frac{1}{\left(\frac{n}{d}\right)!}$$

Random Pivot

- Using Quickselect to pick median guarantees $\Theta(n \log n)$ run time
  - Approach has very large constants
  - If you really want $\Theta(n \log n)$, better off using MergeSort
- Better approach: **Random pivot**
  - Very small constant (very fast algorithm)
  - Expected to run in $\Theta(n \log n)$ time
    - Why? Unbalanced partitions are very unlikely
  - Other options: Median of 5

Formal Argument for $n \log n$ Average

- Remember, run time counts comparisons!
- Quicksort only compares against a pivot
  - Element i only compared to element j if one of them was the pivot

Partition (Divide step)

Given: a list, a pivot value p

Start: unordered list

8	5	7	3	12	10	1	2	4	9	6	11

Goal: All elements $< p$ on left, all $> p$ on right

5	7	3	1	2	4	6	8	12	10	9	11

Formal Argument for $n \log n$ Average

What is the probability of comparing two given elements?

1	2	3	4	5	6	7	8	9	10	11	12

Consider the sorted version of the list

Observation: Adjacent elements must be compared

- **Why?** Otherwise I would not know which came first
- **Every** sorting algorithm **must** compare adjacent elements

In quicksort: adjacent elements <u>always</u> end up in same sublist, unless one is the pivot

Formal Argument for $n \log n$ Average

What is the probability of comparing two given elements?

1	2	3	4	5	6	7	8	9	10	11	12

Consider the sorted version of the list

$$\Pr[\text{we compare } 1 \text{ and } 12] = \frac{2}{12}$$

Assuming pivot is chosen uniformly at random

Only compared if 1 or 12 was chosen as the first pivot since otherwise they are in <u>different</u> sublists

Formal Argument for $n \log n$ Average

What is the probability of comparing two given elements?

1	2	3	4	5	6	7	8	9	10	11	12
				i				j			

Case 1: Pivot contained in $[i+1, \ldots, j-1]$

Then i and j are in different sublists and will <u>never</u> be compared

$$\Pr[\text{we compare } i \text{ and } j] = 0$$

Formal Argument for $n \log n$ Average

What is the probability of comparing two given elements?

1	2	3	4	5	6	7	8	9	10	11	12
				i				j			

Case 2: Pivot is either i or j

Then we will always compare i and j

$$\Pr[\text{we compare } i \text{ and } j] = 1$$

Formal Argument for $n \log n$ Average

Probability of comparing i with j ($j > i$):

- dependent on the number of elements between (and including) i and j

$$\frac{2}{j - i + 1}$$

Expected number of comparisons for Quicksort:

$$\sum_{i<j} \frac{2}{j - i + 1}$$

$$\boxed{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j - i + 1}}$$

Formal Argument for $n \log n$ Average

Expected number of comparisons:

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} < 2 \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{1}{k} < 2 \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{1}{k}$$

Substitution:
$k = j - i$

$$\frac{1}{k+1} < \frac{1}{k}$$

Harmonic series:

$$\sum_{k=1}^{n} \frac{1}{k} = \Theta(\log n)$$

$$= 2 \sum_{i=1}^{n-1} \Theta(\log n) = \Theta(n \log n)$$

Quicksort overall: expected $\Theta(n \log n)$

Expected number of Comparisons

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

Consider when $i = 1$

1	2	3	4	5	6	7	8	9	10	11	12

Compared if 1 or 2 are chosen as pivot
(these will always be compared)

Sum so far: $\frac{2}{2}$

Expected number of Comparisons

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

Consider when $i = 1$

1	2	3	4	5	6	7	8	9	10	11	12

Compared if 1 or 3 are chosen as pivot
(but never if 2 is ever chosen)

Sum so far: $\frac{2}{2} + \frac{2}{3}$

Expected number of Comparisons

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

Consider when $i = 1$

1	2	3	4	5	6	7	8	9	10	11	12

Compared if 1 or 4 are chosen as pivot
(but never if 2 or 3 are chosen)

Sum so far: $\frac{2}{2} + \frac{2}{3} + \frac{2}{4}$

Expected number of Comparisons

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

Consider when $i = 1$

1	2	3	4	5	6	7	8	9	10	11	12

Compared if 1 or 12 are chosen as pivot
(but never if 2 -> 11 are chosen)

Overall sum: $\frac{2}{2} + \frac{2}{3} + \frac{2}{4} + \frac{2}{5} + \cdots + \frac{2}{n}$

Expected number of Comparisons

$$\boxed{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}}$$

When $i = 1$:

$$2\left(\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n}\right) < 2\boxed{\sum_{x=1}^{n} \frac{1}{x}} \qquad O(\log n)$$

n terms overall in the outer sum

Quicksort overall: expected $O(n \log n)$

Sorting, so far

- Sorting algorithms we have discussed:
  - Mergesort $O(n \log n)$
  - Quicksort $O(n \log n)$
- Other sorting algorithms (will discuss):
  - Bubblesort $O(n^2)$
  - Insertionsort $O(n^2)$
  - Heapsort $O(n \log n)$

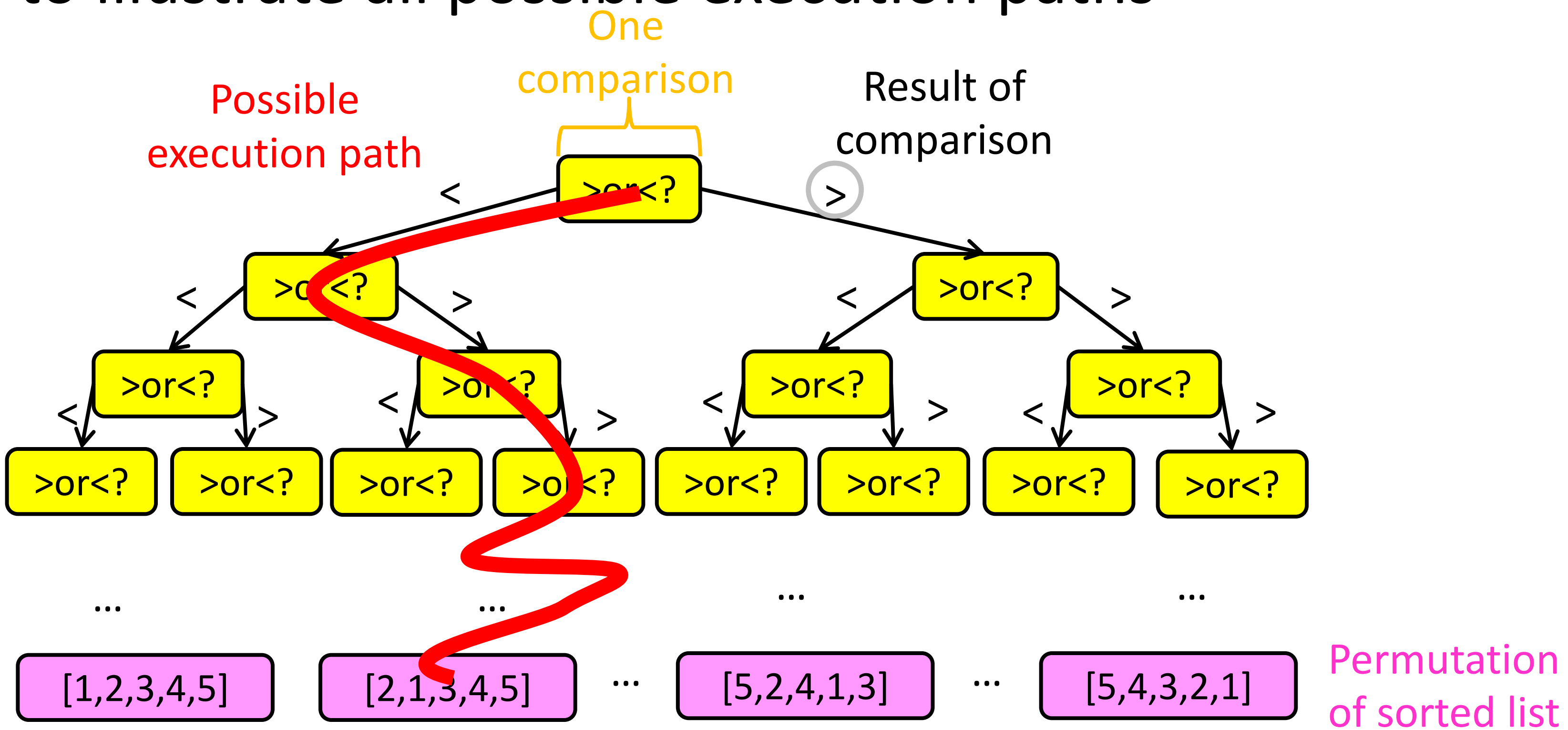
Can we do better than $O(n \log n)$?

Worst Case Lower Bounds

- Prove that there is no algorithm which can sort faster than $O(n \log n)$
  - Every algorithm, in the worst case, must have a certain lower bound
- Non-existence proof!
  - Very hard to do

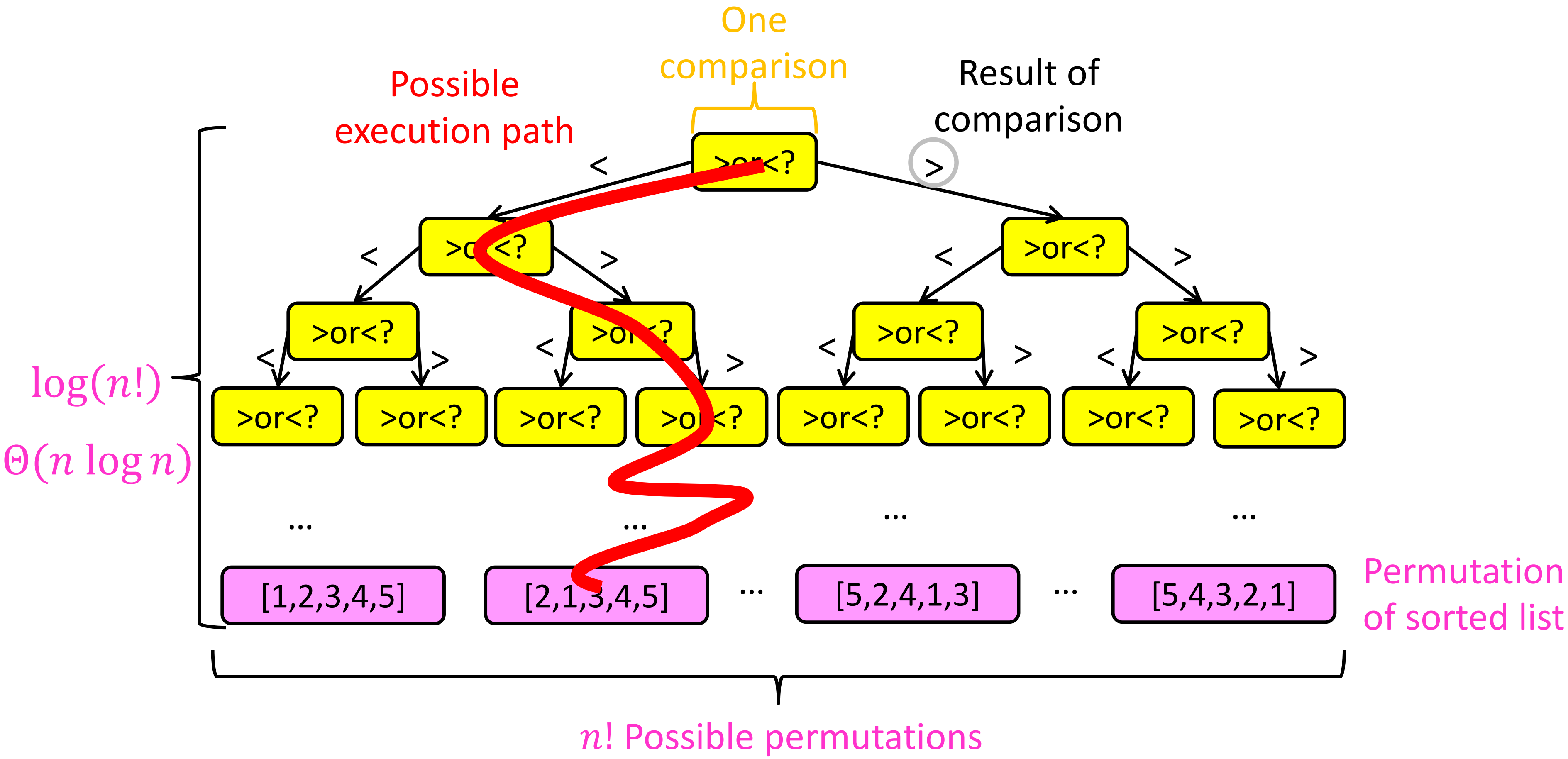
Strategy: Decision Tree

- Sorting algorithms use comparisons to figure out the order of input elements
- Draw tree to illustrate all possible execution paths



Strategy: Decision Tree

- Worst case run time is the longest execution path
- i.e., "height" of the decision tree



Strategy: Decision Tree

- Conclusion: Worst Case Optimal run time of sorting is $\Theta(n \log n)$
  - There is no (comparison-based) sorting algorithm with run time $o(n \log n)$