Warm up

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

Hint: show \( n! \leq n^n \)

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

\[
\begin{align*}
\log n! &= O(n \log n) \\
n! &= n \cdot (n-1) \cdot (n-2) \cdots 2 \cdot 1 \\
n^n &= n \cdot n \cdot n \cdots \cdot n \cdot n \\
n! &\leq n^n \\
\Rightarrow &\ log(n!) \leq \log(n^n) \\
\Rightarrow &\ log(n!) \leq n \log n \\
\Rightarrow &\ log(n!) = O(n \log n)
\end{align*}
\]

\[
\begin{align*}
\log n! &= \Omega(n \log n) \\
n! &= n \cdot (n-1) \cdot (n-2) \cdots \frac{n}{2} \cdots \frac{n}{2} \cdots 2 \cdot 1 \\
\left(\frac{n}{2}\right)^n &= \frac{n}{2} \cdot \frac{n}{2} \cdot \frac{n}{2} \cdots \frac{n}{2} \cdot 1 \cdots 1 \\
n! &\geq \left(\frac{n}{2}\right)^{\frac{n}{2}} \\
\Rightarrow &\ log(n!) \geq log\left(\left(\frac{n}{2}\right)^{\frac{n}{2}}\right) \\
\Rightarrow &\ log(n!) \geq \frac{n}{2} \log \frac{n}{2} \\
\Rightarrow &\ log(n!) = \Omega(n \log n)
\end{align*}
\]
Today’s Keywords
• Divide and Conquer
• Sorting
• Quicksort
• Decision Tree
• Worst case lower bound

CLRS Readings
• Chapter 7
• Chapter 8

Homeworks
• HW2 due 11pm tonight!
  – Divide and conquer
  – Closest Pair of Points
  – Remember to submit relevant .java or .py files (no .zip!)
• HW3 due 11pm Wednesday Feb. 20
  – Divide and conquer
  – Written (use LaTeX)
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!

Partition (Divide step)

- Given: a list, a pivot value p
- Start: unordered list

Start: unordered list
5 7 3 1 2 4 6 8 12 10 9 11

Goal: All elements < p on left, all > p on right
5 3 1 2 4 6 8 12 10 9 11

Is it worth it?

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

- If the partition is always $\frac{n}{10}$ 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 partition is always $d$th order statistic:

Then we shorten by $d$ each time.

If the partition is always $d$th order statistic:

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

$T(n) = \Theta(n^2)$

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 pivot are among $d$ smallest:

$\frac{d}{n} \cdot \frac{d}{n-d} \cdot \frac{d}{n-2d} \cdots \frac{d}{2d} \cdot \frac{1}{\binom{n}{d}}$

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
Formal Argument for $n \log n$ Average

- Remember, run time counts comparisons!
- Quicksort only compares against the *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

\[
\begin{array}{cccccccccccc}
5 & 7 & 3 & 1 & 2 & 4 & 6 & 8 & 12 & 10 & 9 & 11
\end{array}
\]

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

\[
\begin{array}{cccccccccccc}
5 & 7 & 3 & 1 & 2 & 4 & 6 & 8 & 12 & 10 & 9 & 11
\end{array}
\]

---

Formal Argument for $n \log n$ Average

- What is the probability of comparing two given elements?

\[
\begin{array}{cccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12
\end{array}
\]

- (Probability of comparing 3 and 4) = 1
  - Why?
    - Otherwise I wouldn't know which came first
    - ANY sorting algorithm must compare adjacent elements
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

• (Probability of comparing 1 and 12) = $\frac{1}{12}$
  - Why?
    • We only compare 1 with 12 if either was chosen as the first pivot
    • Otherwise they would be divided into opposite sublists

Formal Argument for $n \log n$ Average

• Probability of comparing $i$ and $j$ (where $j > i$):
  - inversely proportional to the number of elements between $i$ and $j$
    \[ \frac{2}{j-i+1} \]
  • Expected (average) number of comparisons:
    \[ \sum_{j<i} \frac{2}{j-i+1} \]

Expected number of Comparisons

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
(they will always be compared)

Sum so far: $\frac{2}{2}$
Consider when \( i = 1 \)

\[
\sum_{|S| = i+1} \frac{1}{|S| + 1}
\]

Compared if 1 or 3 are chosen as pivot
(but not if 2 is ever chosen)

\[
\sum \text{ so far: } \frac{2}{3} + \frac{2}{3}
\]

Expected number of Comparisons

Consider when \( i = 1 \)

\[
\sum_{|S| = i+1} \frac{1}{|S| + 1}
\]

Compared if 1 or 4 are chosen as pivot
(but not if 2 or 3 are chosen)

\[
\sum \text{ so far: } \frac{2}{3} + \frac{2}{3} + \frac{2}{4}
\]

Expected number of Comparisons

Consider when \( i = 1 \)

\[
\sum_{|S| = i+1} \frac{1}{|S| + 1}
\]

Compared if 1 or 12 are chosen as pivot
(but not if 2 \( \rightarrow 11 \) are chosen)

\[
\text{Overall sum: } \frac{2}{3} + \frac{2}{3} + \frac{2}{4} + \frac{2}{5} + \cdots + \frac{2}{n}
\]
Expected number of Comparisons
\[
\sum_{i=1}^{n} \frac{2}{i} \\
\text{When } i = 1: \ 2 \left( \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots + \frac{1}{2^n} \right)
\]
\[\leq 2 \left( \frac{\frac{1}{2} - \frac{1}{i-1}}{\frac{1}{2}} \right) \Theta(\log n)
\]
Quicksort overall: expected \( \Theta(n \log n) \)

Sorting, so far
- Sorting algorithms we have discussed:
  - Mergesort \( \Theta(n \log n) \)
  - Quicksort \( \Theta(n \log n) \)
- Other sorting algorithms (will discuss):
  - Bubblesort \( \Theta(n^2) \)
  - Insertionsort \( \Theta(n^2) \)
  - Heapsort \( \Theta(n \log n) \)
Can we do better than \( \Theta(n \log n) \)?

Worst Case Lower Bounds
- Prove that there is no algorithm which can sort faster than \( \Theta(n \log n) \)
- Non-existence proof!
  - Very hard to do
• Sorting algorithms use comparisons to figure out the order of input elements
• Draw tree to illustrate all possible execution paths

### Worst case run time is the longest execution path
• i.e., “height” of the 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 $o(n \log n)$
Sorting, so far

- Sorting algorithms we have discussed:
  - Mergesort \(O(n \log n)\) Optimal!
  - Quicksort \(O(n \log n)\) Optimal!

- Other sorting algorithms
  - Bubblesort \(O(n^2)\)
  - Insertionsort \(O(n^2)\)
  - Heapsort \(O(n \log n)\) Optimal!

---

Speed Isn’t Everything

- Important properties of sorting algorithms:
  - **Run Time**
    - Asymptotic Complexity
    - Constants
  - **In Place (or In-Situ)**
    - Done with only constant additional space
  - **Adaptive**
    - Faster if list is nearly sorted
  - **Stable**
    - Equal elements remain in original order
  - **Parallelizable**
    - Runs faster with many computers