Exercise 1 (Divide and Conquer) due September 7

• Written (use LaTeX!)
• Asymptotic notation
• Recurrences
• Divide and conquer
Can you cover an $8 \times 8$ grid with 1 square missing using “trominoes?”
Can you cover an $8 \times 8$ grid with 1 square missing using “trominoes?”
Trominoes Puzzle Solution

What about larger boards?

\[ 2^n \]
Trominoes Puzzle Solution

Divide the board into quadrants
Place a tromino to occupy the three quadrants without the missing piece
Place a tromino to occupy the three quadrants without the missing piece
Observe: Each quadrant is now a smaller subproblem!
Trominoes Puzzle Solution

Solve **Recursively**
Trominoes Puzzle Solution

Solve Recursively
Our first algorithmic technique!
Divide and Conquer

**Divide:**
- Break the problem into multiple subproblems, each smaller instances of the original

**Conquer:**
- If the subproblems are “large”:
  - Solve each subproblem recursively
- If the subproblems are “small”:
  - Solve them directly (base case)

**Combine:**
- Merge solutions to subproblems to obtain solution for original problem

When is this an effective strategy?

[CLRS Chapter 4]
Analyzing Divide and Conquer

1. Break into smaller subproblems
2. Use recurrence relation to express recursive running time
3. Use asymptotic notation to simplify

**Divide:** $D(n)$ time

**Conquer:** Recurse on smaller problems of size $s_1, \ldots, s_k$

**Combine:** $C(n)$ time

**Recurrence:**
- $T(n) = D(n) + \sum_{i \in [k]} T(s_i) + C(n)$
Recurrence Solving Techniques

- **Tree**
  - get a picture of recursion

- **Guess/Check**
  - guess and use induction to prove

- **“Cookbook”**
  - MAGIC!

- **Substitution**
  - substitute in to simplify
Merge Sort

**Divide:**
- Break $n$-element list into two lists of $n/2$ elements

**Conquer:**
- If $n > 1$:
  - Sort each sublist recursively
- If $n = 1$:
  - List is already sorted (base case)

**Combine:**
- Merge together sorted sublists into one sorted list
**Merge**

**Combine**: Merge sorted sublists into one sorted list

**Inputs**:
- 2 sorted lists ($L_1, L_2$)
- 1 output list ($L_{out}$)

While ($L_1$ and $L_2$ not empty):
- If $L_1[0] \leq L_2[0]$:
  - $L_{out}.append(L_1.pop())$
- Else:
  - $L_{out}.append(L_2.pop())$

$L_{out}.append(L_1)$
$L_{out}.append(L_2)$
Analyzing Merge Sort

1. Break into smaller subproblems
2. Use recurrence relation to express recursive running time
3. Use asymptotic notation to simplify

**Divide:** 0 comparisons

**Conquer:** recurse on 2 small problems, size \( \frac{n}{2} \)

**Combine:** \( n \) comparisons

**Recurrence:**
- \( T(n) = 2T(n/2) + n \)
Recurrence Solving Techniques

- Tree
- Guess/Check
- “Cookbook”
- Substitution
Tree Method

\[ T(n) = 2T \left( \frac{n}{2} \right) + n \]
Tree Method

\[ T(n) = 2T\left(\frac{n}{2}\right) + n \]
Tree Method

\[ T(n) = 2T\left(\frac{n}{2}\right) + n \]

Number of subproblems | Cost to combine
--- | ---
1 | \(n\)
2 | \(\frac{n}{2}\)
4 | \(\frac{n}{4}\)
\(2^k\) | \(\frac{n}{2^k} = 1\)
Tree Method

3. Use asymptotic notation to simplify

\[ T(n) = 2T(n/2) + n \]

How many levels?

Problem size at \( k^{th} \) level: \( \frac{n}{2^k} \)

Base case: \( n = 1 \)

At level \( k \), it should be the case that \( \frac{n}{2^k} = 1 \)

\[ n = 2^k \Rightarrow k = \log_2 n \]
Tree Method

3. Use asymptotic notation to simplify

\[ T(n) = 2T(n/2) + n \]

\[ k = \log_2 n \]

What is the cost?

Cost at level \( i \):

\[ 2^i \cdot \frac{n}{2^i} = n \]

Total cost:

\[ T(n) = \sum_{i=0}^{\log_2 n} n = n \sum_{i=0}^{\log_2 n} 1 = n \log_2 n \]

\[ = \Theta(n \log n) \]
Multiplication

Want to multiply large numbers together

\[
\begin{array}{c}
4102 \\
\times 1819 \\
\hline
\end{array}
\]

\(n\)-digit numbers

How do we measure input size?

What do we “count” for run time?

number of digits

number of elementary operations

(single-digit multiplications)
“Schoolbook” Multiplication

How many multiplications?

\[
\begin{array}{c}
4102 \\
\times 1819 \\
\hline
36918 \\
4102 \\
32816 \\
+ 4102 \\
\hline
7461538
\end{array}
\]

\(n\)-digit numbers

\(n\) mults

\(n\) mults

\(n\) mults

\(n\) levels

\(\Theta(n^2)\)
"Schoolbook" Multiplication

Can we do better?

4 1 0 2
× 1 8 1 9

3 6 9 1 8
4 1 0 2
3 2 8 1 6
+ 4 1 0 2

7 4 6 1 5 3 8

How many multiplications?

\[ \nu \text{-digit numbers} \]

\[ \nu \text{ mults} \]

\[ \nu \text{ mults} \]

\[ \nu \text{ mults} \]

\[ \nu \text{ mults} \]

\[ \nu \text{ levels} \]

\[ \Theta(\nu^2) \]

What about cost of additions?

\[ \Theta(\nu^2) \]

\[ \Rightarrow \Theta(\nu^2) \]
Divide and Conquer Multiplication

1. Break into smaller subproblems

\[
\begin{align*}
\begin{array}{c}
\begin{array}{c}
\text{a} \\
\text{b}
\end{array}
\end{array}
\times
\begin{array}{c}
\begin{array}{c}
\text{c} \\
\text{d}
\end{array}
\end{array}
\end{align*}
\]

\[
= 10^n \left( \begin{array}{c}
\begin{array}{c}
\text{a} \\
\times
\end{array}
\end{array} \begin{array}{c}
\text{c}
\end{array} \right) + 10^n \left( \begin{array}{c}
\begin{array}{c}
\text{a} \\
\times
\end{array} \begin{array}{c}
\text{d}
\end{array} + \begin{array}{c}
\begin{array}{c}
\text{b} \\
\times
\end{array} \begin{array}{c}
\text{c}
\end{array}
\end{array} \right) + (b \times (b \times d))
\]
\]
Divide and Conquer Multiplication

Divide:
• Break $n$-digit numbers into four numbers of $n/2$ digits each (call them $a, b, c, d$)

Conquer:
• If $n > 1$:
  • Recursively compute $ac$, $ad$, $bc$, $bd$
• If $n = 1$: (i.e. one digit each)
  • Compute $ac$, $ad$, $bc$, $bd$ directly (base case)

Combine:
• $10^n(ac) + 10^{n/2}(ad + bc) + bd$

For simplicity, assume that $n = 2^k$ is a power of 2
Divide and Conquer Multiplication

2. Use recurrence relation to express recursive running time

\[ 10^n (ac) + 10^{n/2} (ad + bc) + bd \]

Recursively solve

\[ T(n) \]
Divide and Conquer Multiplication

2. Use recurrence relation to express recursive running time

$$10^n(ac) + 10^{n/2}(ad + bc) + bd$$

Recursively solve

$$T(n) = 4T\left(\frac{n}{2}\right)$$

Need to compute 4 multiplications, each of size $n/2$
2. Use recurrence relation to express recursive running time

\[ 10^n(ac) + 10^{n/2}(ad + bc) + bd \]

Recursively solve

\[ T(n) = 4T\left(\frac{n}{2}\right) + 5n \]

Need to compute 4 multiplications, each of size \( n/2 \)

2 shifts and 3 additions on \( n \)-bit values
Divide and Conquer Multiplication

3. Use asymptotic notation to simplify

\[ T(n) = 4T(n/2) + 5n \]

<table>
<thead>
<tr>
<th>Number of subproblems</th>
<th>Cost to combine</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5n</td>
</tr>
<tr>
<td>4</td>
<td>( \frac{5n}{2} )</td>
</tr>
<tr>
<td>16</td>
<td>( \frac{5n}{4} )</td>
</tr>
<tr>
<td>( 4^k )</td>
<td>( \frac{5n}{2^k} = 33 )</td>
</tr>
</tbody>
</table>
3. Use asymptotic notation to simplify \( T(n) = 4T(n/2) + 5n \)

<table>
<thead>
<tr>
<th>Number of subproblems</th>
<th>Cost to combine</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5n</td>
</tr>
<tr>
<td>4</td>
<td>( \frac{5n}{2} )</td>
</tr>
<tr>
<td>16</td>
<td>( \frac{5n}{4} )</td>
</tr>
<tr>
<td>( 4^k )</td>
<td>( \frac{5n}{2^k} = 5 )</td>
</tr>
</tbody>
</table>

How many levels?

Problem size at \( k \)th level: \( \frac{n}{2^k} \)

Base case: \( n = 1 \)

At level \( k \), it should be the case that \( \frac{n}{2^k} = 1 \)

\( n = 2^k \Rightarrow k = \log_2 n \)
3. Use asymptotic notation to simplify

\[ T(n) = 4T(n/2) + 5n \]

\[ k = \log_2 n \]

What is the cost?

Cost at level \( i \):

\[ 4^i \cdot \frac{5n}{2^i} = 2^i \cdot 5n \]

Total cost:

\[ T(n) = \sum_{i=0}^{\log_2 n} 2^i \cdot 5n = 5n \sum_{i=0}^{\log_2 n} 2^i \]

\[ 4^k \]

\[ \frac{5n}{2^k} = 5 \]
Divide and Conquer Multiplication

3. Use asymptotic notation to simplify

\[ T(n) = 4T(n/2) + 5n \]

\[ = 5n \sum_{i=0}^{\log_2 n} 2^i \]

\[ = 5n \cdot \frac{2^{\log_2 n+1} - 1}{2 - 1} \]

\[ = 5n(2n - 1) = \Theta(n^2) \]

No better than the schoolbook method!
3. Use asymptotic notation to simplify

\[ T(n) = 4T(n/2) + 5n \]

\[ = 5n \sum_{i=0}^{\log_2 n} 2^i \]

\[ = 5n \cdot \frac{2^{\log_2 n + 1} - 1}{2 - 1} \]

\[ = 5n(2n - 1) = \Theta(n^2) \]

\[ \sum_{i=0}^{L} a^i = \frac{a^{L+1} - 1}{a - 1} \]

Is there a \( o(n^2) \) algorithm for multiplication?