Practice problems

Analysis

• Mark T/F (no need to prove/explain)

\[ \sqrt{n} \log n \in O(n \log n) \]  □ True □ False

\[ \frac{6n \log n}{\sqrt{n}} \in \Omega(\sqrt{n}) \]  □ True □ False

\[ \log 2^{n+1} \in \Theta(\log 3^n) \]  □ True □ False
Analysis

- Mark T/F (no need to prove/explain)
  \[
  \sqrt{n \log n} \in O(n \log n) \quad \square \text{True} \quad \square \text{False}
  \]
  \[
  \frac{6n \log n}{\sqrt{n}} \in \Omega(\sqrt{n}) \quad \square \text{True} \quad \square \text{False}
  \]
  \[
  \log 2^{n+1} \in \Theta(\log 3^n) \quad \square \text{True} \quad \square \text{False}
  \]

Analysis of iterative algorithms

- Give a tight bound on the number of Hello’s produced as a function of \( n \)

\textbf{Algorithm} \textsc{Loop2} \( (n : \text{integer}) \)
  \[
  \text{for } j \leftarrow 1 \text{ to } n^2 \log n \text{ do}
  \]
  \[
  \text{for } i \leftarrow 1 \text{ to } j \text{ do}
  \]
  \[
  \text{print } \text{“Hello”}
  \]
Analysis of iterative algorithms

• Give a tight bound on the number of Hello’s produced as a function of \( n \)

\[
\text{Algorithm \ LOOP2 (} n : \text{integer) \ }
\text{for } j \leftarrow 1 \text{ to } n^2 \log n \text{ do }
\text{for } i \leftarrow 1 \text{ to } j \text{ do }
\text{print “Hello”}
\]

• Answer: \( \Theta(n^4 \log^2 n) \)

Analysis of iterative algorithms

• Give a tight bound on the number of Hi’s produced as a function of \( n \)

\[
\text{Algorithm \ LOOP1 (} n : \text{integer) \ }
\text{for } i \leftarrow 1 \text{ to } n \log^2 n \text{ do }
\quad j \leftarrow i
\quad \text{while } j \leq n \text{ do }
\quad \text{print “Hi”}
\quad j \leftarrow j + 1
\]
Analysis of iterative algorithms

• Give a tight bound on the number of Hi’s produced as a function of $n$

Algorithm $\text{LOOP1} \ (n : \text{integer})$

for $i \leftarrow 1$ to $n \log^2 n$ do

\[
j \leftarrow i
\]

while $j \leq n$ do

print “Hi”

\[
j \leftarrow j + 1
\]

• Answer: $\Theta(n^2)$

Analysis (recurrence relation)

Problem: Solve using the Master Theorem

\[
T(n) = \begin{cases} 
1 & n = 1 \\
7T\left(\frac{n}{2}\right) + n^2 & n > 1 
\end{cases}
\]

Solution: The first case applies.

$T(n) \in \Theta\left(n^{\log_2 7}\right)$
Analysis (recurrence relation)

Problem: Solve using the Master Theorem

\[ T(n) = \begin{cases} 
1 & n = 1 \\
9T\left(\frac{n}{3}\right) + n^2 \log n & n > 1 
\end{cases} \]

Solution: The second case applies \((k = 1)\).
\[ T(n) \in \Theta\left(n^2 \log^2 n\right) \]

Analysis (recurrence relation)

Problem: Solve using the Master Theorem

\[ T(n) = \begin{cases} 
1 & n = 1 \\
T\left(\frac{n}{2}\right) + n \log n & n > 1 
\end{cases} \]

Solution: The third case applies.
\[ T(n) \in \Theta\left(n \log n\right) \]
Divide & Conquer

Divide & Conquer (knowledge)

Recall that Karatsuba's algorithm computes the product $y \times z$, where $y$ and $z$ are two $n$-bits integers. By splitting $y$ in $(a, b)$, and $z$ in $(c, d)$, where $a, b, c, d$ are $n/2$-bits integers, we have that $y = a2^{n/2} + b$ and $z = c2^{n/2} + d$. Karatsuba's algorithm is based on the observation that

$$(a - b)(d - c) = (ad + bc) - (ac + bd)$$

contains two of the products we need to compute $y \times z$.

Write the pseudocode of Karatsuba's algorithm and write the recurrence relation associated with its time complexity (no need to solve it).
**D&C algorithms covered in lectures**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>( T(n) = T(n/2) + O(1) )</th>
<th>( O(\log n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Binary Search</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Merge Sort</strong></td>
<td>( T(n) = 2T(n/2) + O(n) )</td>
<td>( O(n \log n) )</td>
</tr>
<tr>
<td><strong>Towers of Hanoi</strong></td>
<td>( T(n) = 2T(n-1) + O(1) )</td>
<td>( O(2^n) )</td>
</tr>
<tr>
<td><strong>Integer Multiplication</strong></td>
<td>( T(n) = 3T(n/2) + O(n) )</td>
<td>( O(n^{\log_3 3}) )</td>
</tr>
<tr>
<td><strong>Matrix Multiplication</strong></td>
<td>( T(n) = 7T(n/2) + O(n^2) )</td>
<td>( O(n^{\log_7 7}) )</td>
</tr>
<tr>
<td><strong>Closest Pair</strong></td>
<td>( T(n) = 2T(n/2) + O(n) )</td>
<td>( O(n \log n) )</td>
</tr>
<tr>
<td><strong>Selection (k-th smallest)</strong></td>
<td>( T(n) = T(n/5) + T(7n/10 + 6) + O(n) )</td>
<td>( O(n) )</td>
</tr>
</tbody>
</table>

**D&C algorithms covered in homework**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>( T(n) = T(n/2) + O(1) )</th>
<th>( O(\log n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Median of two sorted arrays</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Polynomial Multiplication</strong></td>
<td>( T(n) = 3T(n/2) + O(n) )</td>
<td>( O(n^{\log_3 3}) )</td>
</tr>
<tr>
<td><strong>Grouping into ( k ) groups</strong></td>
<td>(linear-time selection ( k ) times)</td>
<td>( O(nk) )</td>
</tr>
</tbody>
</table>

**Today**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>( T(n) = T(n/2) + O(1) )</th>
<th>( O(\log n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Peak finding (unimodal)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Max subsequence</strong></td>
<td>( T(n) = 2T(n/2) + O(n) )</td>
<td>( O(n \log n) )</td>
</tr>
</tbody>
</table>

---

**Divide & Conquer (\( \log n \) design)**

Suppose you are given an array \( A = \{a_1, a_2, \ldots, a_n\} \) of \( n \) distinct integers. You are told that the sequence of values \( a_1, a_2, \ldots, a_n \) is unimodal, that is, for some index \( p \in [1, n] \), the values in the array increase up to position \( p \) in \( A \), and then decrease the remainder of the way until position \( n \). Give an algorithm to find the position \( p \) in \( O(\log n) \) time. You can assume \( n \) to be a power of 2.
Divide & Conquer (n log n design)

Given an array $A$ of $n$ (possibly negative) integers, find two indices $1 \leq i \leq n$ and $1 \leq j \leq n$ such that the value of $\sum_{k=i}^{j} a_k$ is maximized.

Here are some examples (the solution is underlined):

- $A = [-2, 11, -4, 13, -5, 2]$ which has answer 20.
- $A = [1, -3, 4, -2, -1, 6]$ which has answer 7.
- $A = [-1, 4, -3, 5, -2, -1, 2, 6, -21]$ which has answer 11.

Write an $O(n \log n)$ time algorithm for the problem described above. The algorithm should return $i$ and $j$. If all elements of the array are negative, the algorithm should return $i = j = 0$.

Divide & Conquer ("black box")

Assume you are given the procedure $\text{STRASSEN}(A, B, n)$ which implements Strassen algorithm. Recall that the procedure computes the product of two squared matrices $A$ and $B$ of size $n \times n$.

Using $\text{STRASSEN}(A, B, n)$ as a subroutine, show how you multiply an $n \times n$ matrix by an $n \times kn$ matrix ($k > 1$).

Briefly describe your algorithm, and analyze its time complexity as a function of $n$ and $k$. 
Divide & Conquer ("black box")

Given an unsorted array $A$ of $n$ distinct floating point numbers we want to print the smallest $\lfloor \sqrt{n} \rfloor$ numbers of $A$ in sorted order. For instance given $A = \{3.1, 4.2, 1.013, 2.12, 5.50, 6.12, 0.15, 8.2, 9.1\}$ containing 9 numbers, the algorithm is supposed to print 0.15, 1.013, 2.12 in sorted order. Give a $O(n)$-time algorithm for this problem. **Hint:** Use linear-time SELECT (as a black box) to solve this problem.