Practice problems

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

### D&C algorithms covered in lectures

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Recurrence</th>
<th>Time Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary Search</td>
<td>$T(n)=T(n/2)+O(1)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>Merge Sort</td>
<td>$T(n)=2T(n/2)+O(n)$</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>Towers of Hanoi</td>
<td>$T(n)=2T(n-1)+O(1)$</td>
<td>$O(2^n)$</td>
</tr>
<tr>
<td>Integer Multiplication (Karatsuba)</td>
<td>$T(n)=3T(n/2)+O(n)$</td>
<td>$O(n^{\log_2 3})$</td>
</tr>
<tr>
<td>Matrix Multiplication (Strassen)</td>
<td>$T(n)=7T(n/2)+O(n^2)$</td>
<td>$O(n^{\log_2 7})$</td>
</tr>
<tr>
<td>Closest Pair</td>
<td>$T(n)=2T(n/2)+O(n)$</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>Selection (k-th smallest)</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>Recurrence</th>
<th>Time Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merge $k$ sorted array</td>
<td>$T(k)=2T(k/2)+O(kn)$</td>
<td>$O(kn \log k)$</td>
</tr>
<tr>
<td>Majority Element</td>
<td>$T(n)=2T(n/2)+O(n)$</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>Polynomial Multiplication</td>
<td>$T(n)=3T(n/2)+O(n)$</td>
<td>$O(n^{\log_3 5})$</td>
</tr>
<tr>
<td>Hadamard matrices Multiplication</td>
<td>$T(n)=2T(n/2)+O(n)$</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>Find $k$-smallest of two sorted lists</td>
<td>$T(k)=T(k/2)+O(1)$</td>
<td>$O(\log k)$</td>
</tr>
<tr>
<td>Count number of exchanged pairs</td>
<td>$T(n)=2T(n/2)+O(n)$</td>
<td>$O(n \log n)$</td>
</tr>
</tbody>
</table>
Divide & Conquer ("black box")

Assume you are given the procedure \textsc{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 \textsc{Strassen}(A, B, n) as a subroutine, show how can you multiply a $kn \times n$ matrix by a $n \times kn$ matrix ($k > 1$).

Briefly describe your algorithm, and analyze its time complexity as a function of $n$ and $k$. 

Divide & Conquer ("black box")

Assume you are given the procedure \textsc{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 \textsc{Strassen}(A, B, n) as a subroutine, show how can you multiply a $n \times kn$ matrix by a $kn \times n$ matrix ($k > 1$).

Briefly describe your algorithm in pseudo-code, and analyze its time complexity as a function of $n$ and $k$. 

Stefano Lonardi, UCR
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 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 \( i = j = 0 \).