Design and Analysis of Algorithms

Homework 1
Outline

• Question 1  Basic
• Question 2  Basic
• Question 3  Basic
• Question 4  Moderate
• Question 5  Moderate
• Question 6(a) and 6(b)  Challenge
Question 1

• Practice to solve the following recurrences:
  
• (a) \( T(n) = 9T(n/2) + n^3 \)
  
• (b) \( T(n) = 7T(n/2) + n^3 \)
  
• (c) \( T(n) = T(\sqrt{n}) + \log n \)
  
• (d) \( T(n) = 0.5T(n/2) + n \)
  
• (e) \( T(n) = 3T(n/3) + n/3 \)
Question 1

• You can solve by substitution method, recursion tree method or master theorem
• \( T(1) = 1 \) (if necessary)
• Do your best to use \( \Theta \)-notation. If you can not use \( \Theta \)-notation, then use \( O \)-notation.
Question 2

• Prove that if \( f(n) \in \omega(g(n)) \), then \( f(n) \notin O(g(n)) \)

• Hint: \( \lim_{n \to \infty} \frac{f(n)}{g(n)} \)
Question 3

Merge Sort

• Recall that it first divides the list into two parts

\[ 3 \ 1 \ 8 \ 7 \ 6 \ 5 \ 4 \ 2 \]

• Then sorts each part recursively

\[ 3 \ 1 \ 8 \ 7 \ 6 \ 5 \ 4 \ 2 \]  

• Merges the two sorted parts

• Running time: $\Theta(n \log n)$
Question 3

Consider another merge sort

• It first divides the list into three parts

3 1 8 7 6 5 4 2 9

• Then sorts each part recursively

3 1 8 7 6 5 4 2 9

• Merges the three sorted parts

• Running time: ?
Question 3

• How to merge the three sorted parts together?

• Is the three-part merge sort faster than the two-part merge sort? Why?
  – compare by asymptotic notation
Question 4

• Analyze the running time of the following code

```cpp
for (i = 1; i <= n; i++) {
    for (j = 1; j <= n; j += i)
        x = x + 1;
}
```

• Please use \( \Theta \)-notation
Question 5

• Your smart friend discovers a new sorting algorithm based on a function called **Greedy Pick** described as follows:

  **Input:** an integer sequence $A$
  **Method:** Pick an increasing sequence start from the first element of $A$
  **Output:** the increasing sequence
Question 5

• Greedy pick example:

<table>
<thead>
<tr>
<th>A:</th>
<th>3</th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>6</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pick:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Question 5

• Greedy pick example:

A: \[\boxed{1 \ 2 \ 5 \ 6 \ 4}\]

Pick: 3
Question 5

• Greedy pick example:

A: \[1 \boxed{2} 5 6 4\]

Pick: \[3\]
Question 5

• Greedy pick example:

<table>
<thead>
<tr>
<th>A:</th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>6</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pick:</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Question 5

• Greedy pick example:

<table>
<thead>
<tr>
<th>A:</th>
<th>1</th>
<th>2</th>
<th>6</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pick:</td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Question 5

- Greedy pick example:

A:  

\[ \begin{array}{c|c|c|c} 
& 1 & 2 & 4 \\
\hline 
\end{array} \]

Pick:  

\[ \begin{array}{c|c|c|c} 
3 & 5 & 6 & \text{ } \\
\hline 
\end{array} \]
Question 5

- Greedy pick example:

<table>
<thead>
<tr>
<th>A:</th>
<th>1</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pick:</td>
<td>3</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>
Question 5

- Greedy pick example:

  Input: \[3 \ 1 \ 2 \ 5 \ 6 \ 4\]

  Output: \[3 \ 5 \ 6\]

- Note: Not the longest increasing sequence 1, 2, 5, 6. And can not select 4 although it is after 3.
Question 5

• Sorting by greedy pick

A: 3 1 5 4 2

Pick:  

B:  

Input
Question 5

- Sorting by greedy pick

A:  
\[1 \ 4 \ 2\]  

Pick:  
\[3 \ 5\]  

B:  

Pick
Question 5

- Sorting by greedy pick

A: 1 4 2

Pick: 

B: 3 5

Merge
Question 5

- Sorting by greedy pick

A: 2  
Pick: 1 4  
B: 3 5

Pick
Question 5

• Sorting by greedy pick
Question 5

- Sorting by greedy pick

A:  

Pick: 2

B: 1 3 4 5

Pick
Question 5

- Sorting by greedy pick

A: [ ]  Pick: [ ]  B: [1, 2, 3, 4, 5]

Merge

Output
Question 5

• Show the sorting algorithm is correct

• Describe a worst-case input such that the above algorithm will run in $\Theta(n^2)$ time, and show your analysis
Question 6(a)

- Your friend is holding a sorted integer array $A$ which contains $n$ integer.
- The integers in $A$ are from 0 to $n$ and they are distinct
  - So there's an integer missing!
- Try to find the missing integer by asking $O(\log^2 n)$ questions in the form:
  "What is the $j$th bit of $A[i]$?"
Question 6(a)

• If we can ask "What is A[i]?", then we can find the number easily by binary search

• But now we can only see a bit at a time

• Note: n is the number of integers, not the number of bits
  – If we see all the bits, it's not $O(n)$
Question 6(b)

• Just like 6(a), if A is unsorted, can you find the missing integer in the running time of $O(n)$ questions?

• Hint: If you have seen the last bits of every $A[i]$, can you determine if the missing number is odd or even?

• Hint: What is $n + n/2 + n/4 + n/8 + \ldots$?
Score

<table>
<thead>
<tr>
<th>Question</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 1</td>
<td>15%</td>
</tr>
<tr>
<td>Question 2</td>
<td>15%</td>
</tr>
<tr>
<td>Question 3</td>
<td>15%</td>
</tr>
<tr>
<td>Question 4</td>
<td>15%</td>
</tr>
<tr>
<td>Question 5</td>
<td>15%</td>
</tr>
<tr>
<td>Question 6(a)</td>
<td>15%</td>
</tr>
<tr>
<td>6(b)</td>
<td>15%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>105%</strong></td>
</tr>
</tbody>
</table>

*Question 7* 10%

105% + 105% * 10%