1. Answer the following questions (circle True or False and fill in the blank). *Explain your answers*

(a) **True or False**: False
The average-case behavior of an algorithm is always asymptotically better than its worst-case behavior. The worst- and average-cases are usually asymptotically the same.

(b) **True or False**: True
The amount of work done by an algorithm usually depends on the size of the input.
*For any algorithm that has to process every element in an array, e.g.*

(c) **True or False**: False. It may correspond to, e.g., the number of digits in a binary number

(d) Suppose the running time of a divide-and-conquer algorithm is given by the following recurrence:

$$T(n) = 3T\left(\frac{n}{4}\right) + \Theta(n)$$

This algorithm partitions the original problem into 3 subproblems, each of size $\frac{n}{4}$. It spends $\Theta(n)$ time to divide and/or combine the output.

(e) **True or False**: Note: In each of questions 1(e), 1(f), and 1(g), the highest degree polynomial of $n$ is $n^4$.

$$f(n) = 1000n \log^9 n + n^3 + n^2 \log^5 n + \frac{1}{1000} n^4 \log^2 n$$
is $\Omega(n^3)$.

True because $n^4 \geq n^3$

(f) **True or False**: $f(n) = 1000n \log^9 n + n^3 + n^2 \log^5 n + \frac{1}{1000} n^4 \log^2 n$ is $O(n^3)$.

False because $n^4 \not\leq n^3$

(g) **True or False**: $f(n) = 1000n \log^9 n + n^3 + n^2 \log^5 n + \frac{1}{1000} n^4 \log^2 n$ is $O(3^n)$.

True because $n^4 \leq 3^n$

(h) **True or False**: Every comparison-based sorting algorithm must take at least $\Omega(n \log n)$ time in the best-case.

False. *It is true for the worst-case, but the best case may have a lower running time.*

(i) **True or False**: If the input is a sorted array of $n$ elements, the selection problem can be solved in $O(\log n)$ time.

We did not cover this chapter. This is true, because in a sorted array, we can select the item at any position (the selection problem) in constant time.

2. For each part, circle asymptotic bounds that apply.

(a) What is the **worst-case** time for MAX-HEAPIFY on an array of $n$ distinct elements?

$O(\log n)$

(b) What is the **worst-case** time for Counting sort on $n$ elements, each in the range $1$ to $\log n$?

We did not cover this chapter.

$O(n)$

(c) What is the **best-case** time for Counting sort on $n$ distinct elements, each in the range $1$ to $n^2$?

We did not cover this chapter.

$O(n^2)$
(d) What is the amount of extra storage used by Heapsort on \( n \) distinct elements?

\( O(1) \)

(e) What is the worst-case time for \textsc{Build-Max-Heap} on an unordered array of \( n \) distinct elements?

\( O(n \log n) \) Actually, this algorithm can be shown to run in \( O(n) \) time, but we did not cover that proof.

3. Consider the following comparison-based sorting algorithm.

\[
\text{Fast-Sort}(\text{array } L \text{ of integers})
\]

\[
\begin{align*}
\text{if } & (L \text{ has less than 3 elements}) \\
& \text{sort } L \text{ with Insertion-Sort and return } L \\
\text{else} \\
& L1 = \text{Fast-Sort}(1\text{st quarter of } L) \\
& L2 = \text{Fast-Sort}(2\text{nd quarter of } L) \\
& L3 = \text{Fast-Sort}(3\text{rd quarter of } L) \\
& L4 = \text{Fast-Sort}(4\text{th quarter of } L) \\
& L = \text{4-way-Merge}(L1, L2, L3, L4) \\
& \text{return } L
\end{align*}
\]

Assume the complexity of \( 4\text{-way-merge}(L1, L2, L3, L4) \) is \( O(n_1 + n_2 + n_3 + n_4) \), where \( n_i \) is the number of elements in \( L_i \), \( i = 1, 2, 3, 4 \). Write down the the recurrence relation for the asymptotic running time of \textsc{Fast-Sort} and solve the recurrence using the Master Theorem:

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

\( a = 4, b = 4, f(n) = n, \) and \( n^{\log_4 a} = n \). Case 2 of the Master Theorem holds and

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

4. Consider the following \textbf{comparison-based} algorithm for the selection problem.

\[
\text{New-Select}(A, p, r, i)
\]

\[
\begin{align*}
1 & \text{ if } p == r \\
2 & \quad \text{return } A[p] \\
3 & \quad m = \text{Magic-Median}(A, p, r) \quad // \text{ returns median of } A, m \\
4 & \quad q = \text{Magic-Partition}(A, p, r, m) \quad // \text{ partitions } A \text{ around } m \\
5 & \quad k = q - p + 1 \\
6 & \quad \text{if } i \leq k \\
7 & \quad \quad \text{return } \text{New-Select}(A, p, q, i) \\
8 & \quad \quad \text{else return } \text{New-Select}(A, q + 1, r, i - k)
\end{align*}
\]

(a) Assume the running time of \( \text{Magic-Median}(L) \) is \( O(1) \) and the running time of \( \text{Magic-Partition}(L) \) is \( O(\lg n) \). State the recurrence relation for the asymptotic running time of \textsc{New-Select}.

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

(b) Solve the recurrence you derived in part (a); show the the tightest bound you can. Show your work and clearly state any assumptions you make.

The Master Theorem cannot be used for this recurrence because \( \lg n \) is not a polynomial.
By the Alternate Master Theorem, we have that \( a = 1, b = 2, k = 0, \) and \( p = 1 \). \( a = 1 = b^0, \) so case 2 holds and

\[
T(n) = \Theta(n^{0 \log^{p+1} n}) = \Theta(\lg^2 n).
\]
(c) Is it possible for the algorithm to run in the amount of time you determined in (b)? Why or Why not?

No. If partition could be done in $O(\lg^2 n)$ time, then quicksort could run in $T(n) = 2T(n/2) + \lg^2 n$ time. By the Alternate Master Theorem, for this recurrence, $a = 2, b = 2, k = 0, p = 2,$ and $n^{\log_2 a} = n^{\log_2 2} = n$. $a = 2 > b^0 = 1$, so case 1 holds and $T(n) = \Theta(n^{\log_2 2}) = \Theta(n^1) = \Theta(n)$ time. This is impossible by the proof of the lower bound on comparison based sorting algorithms, which says that any comparison-based sorting algorithm has a worst-case input that causes it to run in $\Omega(n \lg n)$ time.

5. Consider inserting the following keys, in this order, into a hash table of length $m = 11$. For each part, state the number of collisions.

keys to insert (in this order): 3, 4, 14, 15, 25, 26

(a) Suppose you use chaining with the hash function $h(k) = k \mod 11$. Illustrate the result of inserting the keys above using chaining.

Every slot of the hash table would be empty except position 3, which would contain the list $25 \rightarrow 14 \rightarrow 3$ and position 4, which would contain the list $26 \rightarrow 15 \rightarrow 4$.

(b) Suppose you use open addressing with the primary hash function $h_1(k) = k \mod 11$. Illustrate the result of inserting the keys above using linear probing, i.e., $h(k, i) = (h_1(k) + i) \mod 11$.

<table>
<thead>
<tr>
<th>index</th>
<th>content</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>25</td>
</tr>
<tr>
<td>8</td>
<td>26</td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

3 % 11 = 3
4 % 11 = 4
14 % 11 = 3 collision 1
    rehash to 4 collision 2
    rehash to 5
15 % 11 = 4 collision 3
    rehash to 5 collision 4
    rehash to 6
25 % 11 = 3 collision 5
    rehash to 4 collision 6
    rehash to 5 collision 7
    rehash to 6 collision 8
    rehash to 7
26 % 11 = 4 collision 9
    rehash to 5 collision 10
    rehash to 6 collision 11
    rehash to 7 collision 12
    rehash to 8

DONE

There would be 12 collisions.
(c) Suppose you use open addressing with hash functions $h_1(k) = k \mod 11$ and $h_2(k) = (k \mod 10)$. Illustrate the result of inserting the keys above using double hashing, i.e., $h(k, i) = (h_1(k) + (h_2(k) \cdot i)) \mod 11$.

<table>
<thead>
<tr>
<th>index no.</th>
<th>content</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>25</td>
</tr>
<tr>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>10</td>
<td>26</td>
</tr>
</tbody>
</table>

3 % 11 = 3
4 % 11 = 4
14 % 11 = 3 collision 1
   rehash to $(3 + ((14 \mod 10) \times 1)) \mod 11 = 7 \mod 11 = 7$
15 % 11 = 4 collision 2
   rehash to $(4 + ((15 \mod 10) \times 1)) \mod 11 = 9 \mod 11 = 9$
25 % 11 = 3 collision 3
   rehash to $(3 + ((25 \mod 10) \times 1)) \mod 11 = 8 \mod 11 = 8$
26 % 11 = 4 collision 4
   rehash to $(4 + ((26 \mod 10) \times 1)) \mod 11 = 10 \mod 11 = 10$

DONE

There would be 4 collisions.