1. (5 points) This result will be used when we get to NP-Completeness. Design an algorithm to guess an unknown positive integer $n$ using $O(\log n)$ benchmark operations, where a benchmark operation is a comparison of the form
- $k > n$?
- $k = n$?
- $k < n$?

The values of $k$ will be determined by your algorithm. Another way to view this problem is to assume that there’s an oracle who knows the value of $n$. The oracle will not tell you $n$’s value, but given any $k$ she will reply correctly to any of the benchmark operations. You may only ask the oracle at most $O(\log n)$ questions.

2. (1 point) Give a closed-form, as a function of $n$ and $m$, for $\sum_{n \leq k \leq m} k$ where $n \geq m \geq 0$.

3. (6 points) Prove or give a counter-example to each of the following:

A) **Conjecture**: If $f_1(n) \in O(g_1(n))$ and $f_2(n) \in O(g_2(n))$, then $f_1(n) + f_2(n) \in O(\max(g_1(n), g_2(n)))$.

B) **Conjecture**: If $f_1(n) \in O(g_1(n))$ and $f_2(n) \in O(g_2(n))$, then $f_1(n) + f_2(n) \in O(\min(g_1(n), g_2(n)))$.

C) **Conjecture**: If $f_1(n) \in \Theta(g_1(n))$ and $f_2(n) \in \Theta(g_2(n))$, then $f_1(n) + f_2(n) \in \Theta(\max(g_1(n), g_2(n)))$. 
4. (4 points) If a problem \( P \) has worst-case time complexity \( \Omega(n \lg n) \) and worst-case time complexity \( O(n^2) \) and algorithm \( A \) solves problem \( P \), which of the following are possible?

\[ \begin{align*}
  a & \quad \text{A has best-case time complexity } \Theta(n). \\
  b & \quad \text{A has worst-case time complexity } \Theta(n^{\sqrt{n}}). \\
  c & \quad \text{A has average-case time complexity } \Theta(n^3). \\
  d & \quad \text{A has worst-case time complexity } \Theta(n). 
\end{align*} \]

5. (16 points) Do Problem 4.1 on pg. 85 of our text.
CS525DA
HW#1 SOLUTIONS

1. \[ k \leftarrow 1 \]
   \[ \text{while } k < n \text{ do} \]
   \[ k \leftarrow 2k \]
   \[ */ k \leq n \leq k */ \]
   \[ \text{return } \text{Binary\_Search}(k/2, k) \]

   \[ \text{Binary\_Search}(lo, hi) \]
   \[ k \leftarrow \left\lfloor (lo + hi) / 2 \right\rfloor \]
   \[ \text{if } k = n \text{ then return } k \]
   \[ \text{if } n > k \text{ then return } \text{Binary\_Search}(k+1, hi) \]
   \[ \text{return } \text{Binary\_Search}(lo, k-1) \]

   For the initial loop, we note that 1 will be doubled \( \lfloor \lg n \rfloor \) times until it equals or exceeds \( n \).

   Then \( \text{Binary\_Search} \) will be executed \( O(\lg n) \) times.

2. \[ \sum_{n \geq 2 \leq m} k = \sum_{n \geq 2 \leq m} k - \sum_{m-1 \leq k \geq 0} k = \frac{n(n+1)}{2} - \frac{m(m-1)}{2} \]

3. A) The CONJECTURE is true. Because \( f_1(n) \in O\left( g_1(n) \right) \) and \( f_2(n) \in O\left( g_2(n) \right) \), there exist \( c_1, n_1, c_2 \) and \( n_2 \) such that \( f_1(n) < c_1 g_1(n) \) for all \( n > n_1 \), and \( f_2(n) < c_2 g_2(n) \) for all \( n > n_2 \).

   Choosing \( c^* = 2 \max \left( c_1, c_2 \right) \) and \( n^* = \max \left( n_1, n_2 \right) \) it follows that

   \[ f_1(n) + f_2(n) < c_1 g_1(n) + c_2 g_2(n) \leq \max \left( c_1, c_2 \right) \left( g_1(n) + g_2(n) \right) \]

   B) The CONJECTURE is false. Let \( f_1(n) = g_1(n) = n \) and \( f_2(n) = g_2(n) = n^2 \). Then

   \[ f_1(n) + f_2(n) = n^2 + n \text{ and } \min \left( g_1(n), g_2(n) \right) = \min \left( n, n^2 \right) = n. \text{ But } n^2 + n \notin O(n). \]

   C) The CONJECTURE is true. Because \( f_1(n) \in \Theta\left( g_1(n) \right) \) and \( f_2(n) \in \Theta\left( g_2(n) \right) \), there exist \( c_{1,1}, c_{1,2}, n_1, c_{2,1}, c_{2,2} \) and \( n_2 \) such that \( c_{1,1} g_1(n) < f_1(n) < c_{1,2} g_1(n) \) for all \( n > n_1 \), and

   \[ c_{2,1} g_2(n) < f_2(n) < c_{2,2} g_2(n) \text{ for all } n > n_2. \text{ Choosing } c_{3,0} = \min \left( c_{1,1}, c_{2,1} \right) \text{ and } \]

   \[ c_{3,1} = \max \left( c_{1,1}, c_{2,1} \right), \text{ we add the two inequalities to get } \]
So choosing \( c_1 = 2c_\omega \), \( c_2 = 2c_\omega \), and \( n^* = \max(n_i, n_j) \), it follows that
\[
 f_1(n) + f_2(n) \in \Theta(\max(g_1(n), g_2(n)))
\]

4. 

\( a \) possible
\( b \) possible
\( c \) impossible
\( d \) impossible - violation of worst-case time complexity \( \Omega(n \log n) \)

5. 

\( a = 2, b = 2, f(n) = n^3, n^{\log^a_b a} = n^{\log \log 3} = n, \quad \frac{f(n)}{n^{\log^a_b a}} = \frac{n^3}{n} = n^2 \) and case 3 of the Master Theorem applies, with \( T(n) = \Theta(n^2) \).

\( b = 1, b = 10/9, f(n) = n^2, n^{\log^a_b a} = n^{\log \log 9/10} = 1, \quad \frac{f(n)}{n^{\log^a_b a}} = \frac{n^2}{n} = n \) and case 3 of the Master Theorem applies, with \( T(n) = \Theta(n) \).

\( c = 16, b = 4, f(n) = n^2, n^{\log^a_b a} = n^{\log \log 16} = 16, \quad \frac{f(n)}{n^{\log^a_b a}} = \frac{n^2}{n^2} = 1 \) and case 2 of the Master Theorem applies, with \( T(n) = \Theta(n \log n) \).

\( d = 7, b = 3, f(n) = n^2, n^{\log^a_b a} = n^{\log \log 7} = 7, \quad \frac{f(n)}{n^{\log^a_b a}} = \frac{n^2}{n^7} = n^{2-\log_7} \). Since \( 2 > \log_7 \), case 3 of the Master Theorem applies, with \( T(n) = \Theta(n^2) \).

\( e = 7, b = 2, f(n) = n^2, n^{\log^a_b a} = n^{\log \log 7} = 7, \quad \frac{f(n)}{n^{\log^a_b a}} = \frac{n^2}{n^7} = n^{2-\log_7} \). Since \( 2 < \log_7 \), case 1 of the Master Theorem applies, with \( T(n) = \Theta(n^{1+}) \).

\( f = 2, b = 4, f(n) = \sqrt{n}, n^{\log^a_b a} = n^{\log \log 4} = \sqrt{n}, \quad \frac{f(n)}{n^{\log^a_b a}} = \frac{\sqrt{n}}{\sqrt{n}} = 1 \). Case 2 of the Master Theorem applies, with \( T(n) = \Theta(\sqrt{n \log n}) \).

\( g \). By unfolding,
\[
 T(n) = n + T(n-1) = n + (n-1) + T(n-2) = n + (n-1) + \ldots + 2 + T(1) = \sum_{k=1}^{n} k + \Theta(1)
\]
\[
 = \sum_{k=1}^{n} k - 1 + \Theta(1) = \frac{n(n+1)}{2} - 1 + \Theta(1) = \Theta(n^2).
\]
h. By unfolding, \( T(n) = 1 + T\left(n^{1/2}\right) = 1 + 1 + T\left(n^{1/4}\right) = 1 + 1 + 1 + T\left(n^{1/8}\right) = \ldots = k + T\left(n^{1/k}\right) \).

The iteration stops when \( n^{1/k} = 2 \). Taking the \( \lg \) of both sides, the iteration stops when
\[
\frac{1}{k} \lg n = \lg 2 = 1,
\]
or \( \lg n = 2^k \), or \( k = \lg \lg n \). Thus, \( T(n) = \lg \lg n \).