Assignment 1 (due Friday, January 25, 6:00pm)

Instructions:

- Hand in your assignment using Crowdmark. Detailed instructions are on the course website.
- Give complete legible solutions to all questions.
- Your answers will be marked for clarity as well as correctness.
- For any algorithm you present, you should justify its correctness (if it is not obvious) and analyze the complexity.

1. [10 marks]

(a) [6 marks] Give a proof from first principles (not using limits) that $n^3 - 100n + 1000 \in \Theta(n^3)$.

(b) [4 marks] Suppose that $f(n), g(n)$ and $h(n)$ are positive-valued functions such that $f(n) \in O(h(n))$ and $g(n) \in O(h(n))$. Prove that $2.72f(n) + 3.14g(n) \in O(h(n))$.

2. [12 marks] For each pair of functions $f(n)$ and $g(n)$, fill in the correct asymptotic notation among $\Theta$, $o$, and $\omega$ in the statement $f(n) \in \bigcap (g(n))$. Formal proofs are not necessary, but provide brief justifications for all of your answers. (The default base in logarithms is 2.)

(a) $f(n) = \sum_{i=1}^{n-1} (i + 1)/i^2$ vs. $g(n) = \log(n^{100})$

(b) $f(n) = n^{3/2}$ vs. $g(n) = (n + 1)^9/(n^3 - 1)^2$.

(c) $f(n) = (32768)^{n/3}$ vs. $g(n) = (6561)^{n/4}$

(d) $f(n) = (\log n)^{\log n}$ vs. $g(n) = n^{\log \log n}$. [Hint: take logarithms.]

3. [10 marks] Analyze the following pseudocode and give a tight $\Theta$ bound on the running time as a function of $n$. Carefully show your work.

(a) [5 marks]

1. $s = 0$
2. for $i = 1$ to $n$ do {
   3. $j = i$
   4. while $j \leq n$ do {
      5. $j = j + i$
      6. $s = s + j$
   }
(b) [5 marks]

1. \( k = 1 \)
2. \( s = 0 \)
3. for \( i = 1 \) to \( n \) do {
4. \hspace{1em} for \( j = 1 \) to \( 2k \) do
5. \hspace{2em} \( s = s + j \)
6. \hspace{1em} \( k = 2k \)
}

4. [6 marks] Consider the following problem named M3SUM: Given an array of \( n \) integers, \( S[1], \ldots, S[n] \), determine if there exist three array elements \( S[i], S[j], S[k] \) such that

\[
S[i] + S[j] = S[k]
\]

(where \( 1 \leq i, j, k \leq n \) and \( i, j, k \) are all distinct). Define \( T[\ell] = 4S[\ell] - 1 \) for \( 1 \leq \ell \leq n \) and define \( T[\ell + n] = -4S[\ell] + 2 \) for \( 1 \leq \ell \leq n \). Show that solving 3SUM on the array \( T \) (of length \( 2n \)) will solve M3SUM on the array \( S \) (so this is a reduction from M3SUM to 3SUM).

[Important: you need to show that there is a solution for M3SUM for the instance \( S \) if and only if there is a solution for 3SUM for the instance \( T \).]

5. [10 marks] Suppose Alice spends \( a_i \) dollars on the \( i \)th day and Bob spends \( b_i \) dollars on the \( i \)th day, for \( 1 \leq i \leq n \). We want to determine whether there exists some set of \( t \) consecutive days during which total amount spent by Alice is exactly the same as the total amount spent by Bob in some (possibly different) set of \( t \) consecutive days. That is, we want to determine if there exist \( i, j, t \) (with \( 0 \leq i, j \leq n - t \) and \( 1 \leq t \leq n \)) such that

\[
a_{i+1} + a_{i+2} + \cdots + a_{i+t} = b_{j+1} + b_{j+2} + \cdots + b_{j+t}.
\]

For example, for the inputs \( 10, 21, 11, 12, 19, 15 \) and \( 12, 9, 2, 31, 21, 8 \), the answer is “yes” because \( 11 + 12 + 19 = 9 + 2 + 31 \).

(a) [5 marks] First design and analyze an algorithm that solves the problem in \( \Theta(n^2) \) time by “brute force”.

(b) [5 marks] Design and analyze a better algorithm that solves the problem in \( \Theta(n^2 \log n) \) time. [Hint: use sorting.]

6. [21 marks] Suppose we are given an array of \( n \) integers, \( A[1], \ldots, A[n] \), and a positive integer \( k \). We want to find the maximum value of \( A[i] + A[j] \) subject to the condition that \( 1 \leq i < j \leq i + k \leq n \). That is, we want the maximum sum of two array elements that are at most \( k \) apart in the array. For example, for the inputs \( 10, 2, 0, 8, 1, 7, 1, 0, 11 \) and \( k = 2 \), the maximum sum is \( A[4] + A[6] = 8 + 7 = 15 \) (the two array elements are \( A[4] \) and \( A[6] \), which are two apart).
(a) [4 marks] Design and analyze a simple “brute-force” algorithm for this problem that runs in $O(kn)$ time.

(b) [8 marks] Design a divide-and-conquer algorithm for this problem in which you split the array into two equal pieces, where the “combine” operation of the algorithm runs in time $O(k)$.

(c) [4 marks] Show that this problem can be solved directly (not recursively) for an array of length $n = k$ in $O(k)$ time (this will be used as a base case for the divide-and-conquer algorithm).

(d) [5 marks] Using $n = k$ as a base case, the running time $T(n)$ for the divide-and-conquer algorithm satisfies the following recurrence which involves both $k$ and $n$:

$$T(n) = \begin{cases} 
2T(n/2) + O(k) & \text{if } n > k \\
O(k) & \text{if } n = k.
\end{cases}$$

Show that the solution to this recurrence is $T(n) \in O(n)$ if $n = k2^t$ for some integer $t$. [Hint: this can be done using either the recursion tree method or guess-and-check.]

7. [6 marks] Give a tight asymptotic (i.e., $\Theta$) bound for the solution to the following recurrence by using the recursion-tree method (you may assume that $n$ is a power of 4). Show your work.

$$T(n) = \begin{cases} 
3T(n/4) + \sqrt{n} & \text{if } n > 1 \\
5 & \text{if } n \leq 1.
\end{cases}$$