COMP 550-001. Fall 2016
Homework assignment 3 - solutions

---

1. Exercise 9.3-8

```
TWO-ARRAY-MEDIAN(X, Y)
  n = X.length                    // n also equals Y.length
  median = FIND-MEDIAN(X, Y, n, 1, n)
  if median == NOT-FOUND
      median = FIND-MEDIAN(Y, X, n, 1, n)
  return median

FIND-MEDIAN(A, B, n, low, high)
  if low > high
      return NOT-FOUND
  else k = ⌊(low + high)/2⌋
      if k == n and A[n] ≤ B[1]
          return A[n]
      elseif k < n and B[n − k] ≤ A[k] ≤ B[n − k + 1]
          return A[k]
      elseif A[k] > B[n − k + 1]
          return FIND-MEDIAN(A, B, n, low, k − 1)
      else return FIND-MEDIAN(A, B, n, k + 1, high)
```

2. Exercise 8.2-4

   Compute the C array as is done in counting sort. The number of integers in the range $[a \,..\, b]$ is $C[b] - C[a-1]$, where we interpret $C[-1]$ as 0.

3. Exercise 8.3-4

   Treat the numbers as 3-digit numbers in radix n. Each digit ranges from 0 to $n-1$. Sort these 3-digit numbers with radix sort.

   There are 3 calls to counting sort, each taking $\Theta(n+n) = \Theta(n)$ time, so that the total time is $\Theta(n)$.

4. Problem 8-3 (a)

   The usual, unadorned radix sort algorithm will not solve this problem in the required time bound. The number of passes, d, would have to be the number of digits in the largest integer. Suppose that there are m integers; we always have $m \le n$. In the worst case, we would have one integer with $n/2$ digits and $n/2$ integers with one digit each. We assume that the range of a single digit is constant. Therefore, we would have $d = n/2$ and $m = n/2 + 1$, and so the running time would be $\Theta(dm) = \Theta(n^2)$.

   Let us assume without loss of generality that all the integers are positive and have no leading zeros. (If there are negative integers or 0, deal with the positive numbers, negative numbers, and 0 separately.) Under this assumption, we can observe that integers with more digits are always greater than integers with fewer digits. Thus, we can first sort the integers by number of digits (using counting sort), and then use radix sort to sort each group of integers with the same length. Noting that each integer has between 1 and n digits, let m_i be the number of integers with i digits, for $i = 1, 2, \ldots, n$. Since there are n digits altogether, we have $\sum_{i=1}^{n} i \cdot m_i = n$.

   It takes $O(n)$ time to compute how many digits all the integers have and, once the numbers of digits have been computed, it takes $O(m+n) = O(n)$ time to group the integers by number of digits. To sort the group with m_i digits by radix sort takes $\Theta(i \cdot m_i)$ time. The time to sort all groups, therefore, is

$$\begin{aligned}\sum_{i=1}^{n} \Theta(i \cdot m_i) &= \Theta\left(\sum_{i=1}^{n} i \cdot m_i\right) \\ &= \Theta(n)\,.\end{aligned}$$