Solution to Written Assignment 1

1. (a) Let $n_0 = 1$ and $C = 1$. We have $f(n) \le Cn^2$ for every $n \ge n_0$, so $f(n) = O(n^2)$.

   (b) Observe that $\log n = o(n^{1/16}) = O(n^{1/16})$. Here is why.

$$\begin{aligned} \lim_{n\to\infty} \frac{\log n}{n^{1/16}} &= \lim_{n\to\infty} \frac{16}{n^{1/16}} \\ &= 0 \end{aligned}$$

   Combining 8 copies of the above equality, one has $\log^8 n = O(\sqrt{n})$.

   (c) There exist $n_1, C_1 > 0$ so that $f(n) \le C_1 n^2$ for every $n \ge n_1$. There also exist $n_2, C_2 > 0$ so that $g(n) \le C_2\sqrt{n}$ for every $n \ge n_2$. Hence, $f(n) \cdot g(n) \le C_1 C_2 n^{2.5}$ for every $n \ge \max\{n_1, n_2\}$. By the definition of big-O, $f(n) \cdot g(n) = O(n^{2.5})$.

2. (a) Let

$$S(n) = \begin{cases} S(\lceil n/2 \rceil) + dn & \text{if } n > 1 \\ d & \text{if } n = 1 \end{cases}$$

   If d is picked as a sufficiently large constant, then $T(n) \le S(n)$ for every $n \ge 1$. Observe that $dn = \Omega(n^{\log_2 1+\epsilon})$ and $1 \cdot dn/2 \le 1/2 \cdot dn$. The third case of Master Theorem applies. We get $S(n) = O(n)$, yielding that $T(n) = O(n)$.

   (b) Let

$$S(n) = \begin{cases} S(\lfloor n/2 \rfloor) + S(\lceil n/2 \rceil) + dn & \text{if } n > 1 \\ d & \text{if } n = 1 \end{cases}$$

   If d is picked as a sufficiently large constant, then $T(n) \le S(n)$ for every $n \ge 1$. Observe that $dn = \Theta(n^{\log_2 2})$. The second case of Master Theorem applies. We get $S(n) = O(n \log n)$, yielding that $T(n) = O(n \log n)$.

   (c) Let

$$S(n) = \begin{cases} 2T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + dn & \text{if } n > 1 \\ d & \text{if } n = 1 \end{cases}$$

   If d is picked as a sufficiently large constant, then $T(n) \le S(n)$ for every $n \ge 1$. Observe that $dn = \Theta(n^{\log_2 3-\epsilon})$. The first case of Master Theorem applies. We get $S(n) = O(n^{\log_2 3})$, yielding that $T(n) = O(n^{\log_2 3})$.

   (d) Let

$$T(n = 2^k) = S(k) = \begin{cases} 2S(\approx k/2) + d & \text{if } n > 1 \\ d & \text{if } n = 1 \end{cases}$$

   By Master Theorem, one has a rough guess that $T(n = 2^k) = S(k) = O(k) = O(\log n)$.

   We use the substitution method to prove the guess $T(n) \le c \log n - c$ for some $c > 0$, for every $n \ge 3$. The induction base $n = 3$ holds by setting c sufficiently large (w.r.t. d). Assume that for every $n < t$, the guess holds. For $n = t$, $T(t) \le 2c \log \sqrt{t} - 2c + d \le c \log t$ if $c > d$. By induction, $T(n) = O(\log n)$.

3. We assume that all numbers are distinct or break ties arbitrarily.

   (a)

   if $\mathcal{H}[2] < \mathcal{H}[3]$ **then**
       return $\min\{\mathcal{H}[4], \mathcal{H}[5], \mathcal{H}[3]\}$
   else
       return $\min\{\mathcal{H}[6], \mathcal{H}[7], \mathcal{H}[2]\}$
   end

   (b) All elements in $\mathcal{H}$ except $\mathcal{H}[1]$.

4. (a) One can solve this problem by binary search, detailed as follows. The initial call is $\text{find}(1, n)$.

   Function `find` (ℓ, r) **:**
       if $r - \ell$ *is small* **then**
           solve by a linear scan
       end
       $p \leftarrow (\ell + r)/2$
       if $S[p]$ *equals* $-p$ **then**
           return Yes
       else
           if $S[p] < -p$ **then**
               return `find` $(\ell, p - 1)$
           else
               return `find` $(p + 1, r)$
           end
       end

   (b) We prove this by constructing an adversary game. For any algorithm $\mathcal{A}$ that uses at most $n - 1$ probes, if $\mathcal{A}$ probes $S[k]$, then Alice always claims that $S[k] = -(k + 1)$. Since there exists an $S[i]$ for some $i \in [1, n]$ that $\mathcal{A}$ does not know the value, Alice has the freedom to claim $S[i] = -i$ or $S[i] = -(i + 1)$. Hence, $\mathcal{A}$ has no way to answer correctly. Any algorithm that can solve this problem requires $\Omega(n)$ probes (time).

5. (a) We prove this problem by reduction. In what follows, we devise an $o(n \log n)$-time algorithm for the element uniqueness problem using an $o(n \log n)$-time algorithm for the second mode. However, any algorithm in the comparison-based model requires $\Omega(n \log n)$ time to solve the element uniqueness problem. Hence, the $o(n \log n)$-time algorithm for the second mode does not exist in the comparison-based model.

Function `uniqueness` $(a_1, a_2, \ldots, a_n)$ **:**
  $a_0 \leftarrow \min\{a_1, a_2, \ldots, a_n\} - 1$
  $\mu \leftarrow$ `2ndMode` $(\underbrace{a_0, a_0, \ldots, a_0}_{n+1 \textit{ copies}}, a_1, a_2, \ldots, a_n)$
  if $\mathrm{freq}(\mu)$ *equals* 1 **then**
    return $a_1, a_2, \ldots, a_n$ are all distinct
  else
    return Some of $a_1, a_2, \ldots, a_n$ repeats
  end

(b) Represent each a_i in base n, so each a_i has 4 n-ary digits. By RADIXSORT, one can sort a's in $O(4n)$ time. Followed by a linear scan, one can compute the frequency of a_i for each $i \in [1, n]$. Given the frequencies, output the second mode can be done in linear time. In total, we use only $O(n)$ time.

6. Let $x_1 = \arg\min_{x \in S} \mathrm{freq}(x)$ and $x_2 = \arg\max_{x \in S} \mathrm{freq}(x)$. Let further $\Delta = n(1 - 1/\log n)$.

If $\mathrm{freq}(x_2) \geq \Delta/2$, then at least one of the $(n/4)$-th, $(2n/4)$-th, $(3n/4)$-th order statistics equals x_2. Hence, one can decide whether $\mathrm{freq}(x_2) \geq \Delta/2$ in $O(n)$ time by 3 selections and 1 linear scan. If the above procedure succeeds, then we receive the exact value $\mathrm{freq}(x_2)$. If the above procedure fails in any way, then $\mathrm{freq}(x_2) < \Delta/2$, implying that

$$\mathrm{freq}(x_1) + \mathrm{freq}(x_2) \leq 2\mathrm{freq}(x_2) < \Delta,$$

so output "No."

Given $\mathrm{freq}(x_2)$, if $\mathrm{freq}(x_2) > n(1 - 2/\log n)$, then there are $O(n/\log n)$ values different from x_2. In this case, one can sort the numbers not equal to x_2 in $O(n)$ time, so $\mathrm{freq}(x_1)$ is obtained. Otherwise, $\mathrm{freq}(x_2) = n(1 - 2/\log n) - \delta$ for some $\delta \geq 0$. Then the only possibility to output "Yes" is $\mathrm{freq}(x_1) \geq n/\log n + \delta$. In this case, the n given numbers have at most

$$2 + \frac{n - \mathrm{freq}(x_1) - \mathrm{freq}(x_2)}{\mathrm{freq}(x_1)} \leq 3.$$

distinct values. Hence, use 3 linear scan to figure out whether there are only 3 distinct values in the input. If yes, then $\mathrm{freq}(x_1)$ can be computed in linear time. Otherwise, output "No.".

Conclusion: this problem can be solved in $O(n)$ time.