Problem 1.

a) Let

$$f(n) = \begin{cases} 0, & n = 0 \\ n + f(n-10), & n \geq 1. \end{cases}$$

For simplicity, we assume n is a multiple of 10. Let $n = 10k$, $k \in \mathbb{N}$. In the general case, we then have

$$\begin{aligned}
f(10k) &= 10k + f(10k-10) \\
&= 10k + (10k-10) + f(10k-20) \\
&= \ldots \\
&= 10k + (10k-10) + (10k-20) + \ldots + 20 + 10 + f(0) \\
&= 10k + (10k-10) + (10k-20) + \ldots + 20 + 10 \\
&= 10k + (10k-10) + 10 + (10k-20) + 20 + \ldots \\
&= 10k + 10k + 10k + \ldots \\
&= 10(k + k + k + \ldots) \\
&= 10\sum_{i=1}^{k} i \\
&= 10\left(\frac{k(k+1)}{2}\right) \\
&= 5k(k+1).
\end{aligned}$$

Since $n = 10k$, we have $k = n/10$. It follows that

$$f(n) = \frac{n\left(\frac{n}{10}+1\right)}{2}.$$

Therefore we guess that $T(n) \in O(n^2)$. We now prove our guess by induction. Let $P(n)$ be the statement:

$$f(n) = \frac{n\left(\frac{n}{10}+1\right)}{2}, n \in \mathbb{N}.$$

Theorem: For all $n \in \mathbb{N}$, $P(n)$ is true. Proof by mathematical induction.

Basis step: For $n = 0$, $f(0) = 0$ by the recurrence relation and $f(0) = 0$ by the guessed closed form. $P(0)$ holds.

Inductive step: Assume $n = 10k$, $k \in \mathbb{N}$. Assume $P(10k)$ holds. In other words,

$$f(10k) = \frac{10k\left(\frac{10k}{10}+1\right)}{2}$$

$$= 5k(k+1).$$

We want to show $P(10(k+1))$ holds. In other words, using the recurrence relation, we want to show

$$\begin{aligned} f(10(k+1)) &= \frac{10(k+1)\left(\frac{10(k+1)}{10}+1\right)}{2} \\ &= 5(k+1)((k+1)+1) \\ &= 5(k+1)(k+2). \end{aligned}$$

By the recurrence relation,

$$\begin{aligned} f(10(k+1)) &= 10(k+1) + f(10(k+1)-10) \\ &= 10(k+1) + f(10k) \\ &= 10(k+1) + 5k(k+1) \\ &= 10k + 10 + 5k^2 + 5k \\ &= 5k^2 + 15k + 10 \\ &= 5(k^2 + 3k + 2) \\ &= 5(k+1)(k+2). \end{aligned}$$

We have shown $P(10(k+1))$ holds. Now the conclusion follows from the basis step, the inductive step, and the principle of induction. As a result, we can also guess that $T(n-100) + 100n \in O(n^2)$.

b) Let $n \in \mathbb{N}$ such that $n \geq n_1$ for some n_1 implies that $\lfloor n/2 \rfloor + 8 \leq 3n/4$. We wish to show that $T(n) \in O(n\log(n))$. We can show this by showing $T(n) \leq cn\log(n) - d$ for some c and d. Substituting this into the recurrence $T(n) = 2T(\lfloor n/2 \rfloor + 8) + n$ yields

$$\begin{aligned} T(n) &\leq 2(c(n/2+8)\log(n/2+8) - d) + n \\ &= 2c(n/2+8)\log(n/2+8) - 2d + n \\ &= (cn + 16c)\log(n/2+8) - 2d + n \\ &= cn\log(n/2+8) + 16c\log(n/2+8) - 2d + n \\ &\leq cn\log(3n/4) + 16c\log(3n/4) - 2d + n \\ &= cn(\log(n) + \log(3/4)) - d + 16c(\log(n) + \log(3/4)) - d + n \\ &= cn\log(n) - d + cn\log(3/4) + 16c\log(n) + 16c\log(3/4) - d + n. \end{aligned}$$

We now need to choose $c > 0$ such that for n sufficiently large, the above expression is

$\leq cn\log(n) - d$. Let us choose $c = -2/\log(3/4)$ and $d = 32$. The above expression becomes

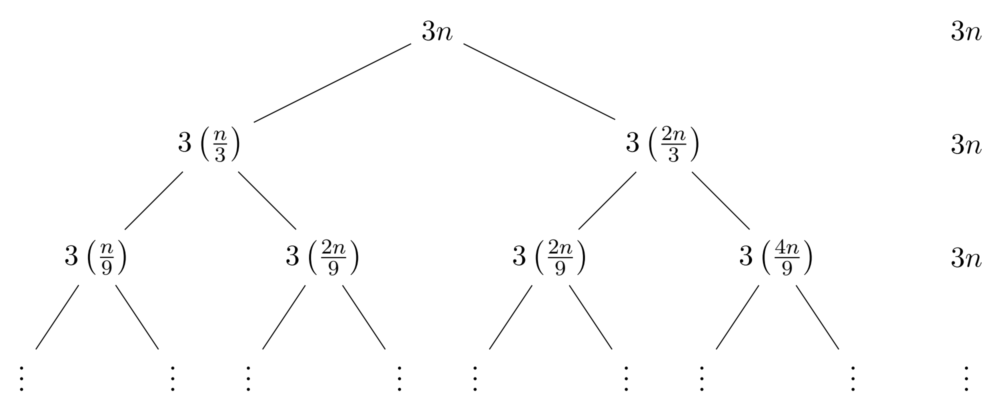
$$\begin{aligned} T(n) &\leq cn\log(n) - d - 2n + 16c\log(n) - 32 - 32 + n \\ &= cn\log(n) - d + 16c\log(n) - n - 64 \\ &\leq cn\log(n) - d + 16c\log(n) - n. \end{aligned}$$

Since $\log(n) \in o(n)$, then $n \geq n_2$ for some n_2 implies that $n \geq 16c\log(n)$. It follows that

$$T(n) \leq cn\log(n) - d$$

for sufficiently large n. In fact, by letting $n_0 = \max\{n_1, n_2\}$, $n \geq n_0$ implies that $T(n) \leq cn\log(n) - d$. Hence we conclude $T(n) \in O(n\log(n))$.

c) The recurrence tree of $T(n) = T(n/3) + T(2n/3) + 3n$ is shown below:



The longest path from the root to a leaf is $n \to (2/3)n \to (2/3)^2 n \to \ldots \to 1$. Since $(2/3)^k n = 1$ when $k = \log_{3/2}(n)$, we conclude the height of the recurrence tree will be $\log_{3/2}(n)$. Hence, since each level in the tree sums to $3n$, we conclude $T(n) = 3n\log_{3/2}(n)$. We guess that the tight upper bound and tight lower bound is $O(n\log(n))$; in other words, $T(n) \in O(n\log(n))$ and $T(n) \in \Omega(n\log(n))$.

We can show $T(n) \in O(n\log(n))$. First, we note that $T(n)$ can be simplified to

$$\begin{aligned} T(n) &= 3n\log_{3/2}(n) \\ &= 3n\frac{\log(n)}{\log(3/2)} \\ &= \frac{3}{\log(3/2)}n\log(n). \end{aligned}$$

By definition, $h(n) \in O(f(n))$ if there exists $c > 0$, $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$ we have $h(n) \leq cf(n)$. For $T(n)$, we see that if we choose $c = 3/\log(3/2)$, then $T(n) \leq cn\log(n)$ for all n_0. Hence we conclude $T(n) \in O(n\log(n))$.

d) For simplicity, we assume $n = 2^m$ ($m = \log(n)$). Then the recurrence becomes

$$T(2^m) = 2T(2^{m/2}) + \log(m).$$

Now assume $S(m) = T(2^m)$. Then the recurrence becomes

$$S(m) = 2S(m/2) + \log(m).$$

We see that the recurrence is now in the form $T(n) = aT(n/b) + f(n)$. As a result, we can apply the master method. Let's consider the first case of the master theorem. We have $a = 2$, $b = 2$, and $f(m) = \log(m)$. We have that

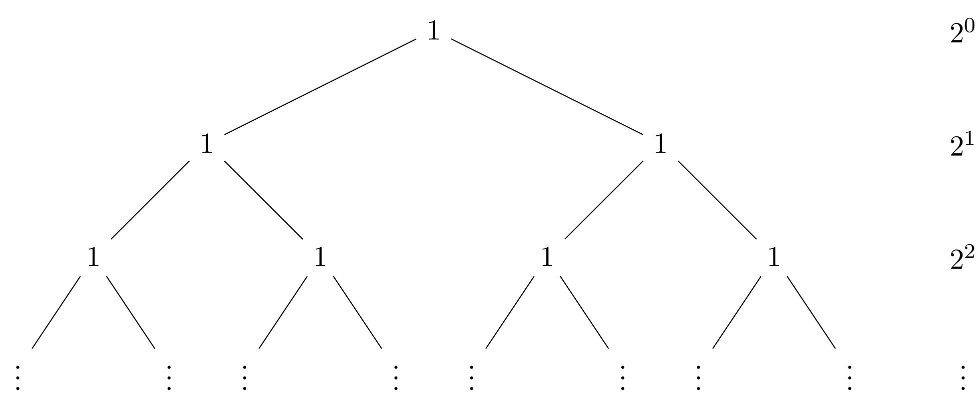
$$f(m) = \log(m) \in O(m^{\log_b(a)-\epsilon}) = O(m^{\log_2(2)-\epsilon}) = O(m^{1-\epsilon})$$

for any $0 < \epsilon < 1$ meaning $S(m) \in \Theta(m)$. Changing our variable back to n, we have

$$S(m) = T(2^m) = T(n) \in \Theta(m) = \Theta(\log(n)).$$

Conclusion: $T(n) \in \Theta(\log(n))$.

e) Assume n is even for simplicity. Assume $T(0) = 1$. The recurrence tree of $T(n) = 2T(n-2)+1$ is shown below:



The tree has $n/2$ levels and the cost of each level sums to 2^k where k is the level. The cost of the tree is therefore

$$\begin{aligned} T(n) &= \sum_{i=0}^{n/2} 2^i \\ &= 2^{n/2+1} - 1 \\ &= 2 \cdot 2^{n/2} - 1 \\ &= 2\sqrt{2^n} - 1. \end{aligned}$$

We can confirm this using induction.

Theorem: For all $n \in \mathbb{N}$, $T(n) = 2\sqrt{2^n} - 1$.

Basis step: For $n = 0$, $T(0) = 1$ by definition and $2\sqrt{2^0} - 1 = 1$. Hence $T(n) = 2\sqrt{2^n} - 1$ holds for $n = 0$.

Inductive step: Assume $T(k) = 2\sqrt{2^k} - 1$. We need to prove $T(k+2) = 2\sqrt{2^{k+2}} - 1$. From the recurrence, we have

$$\begin{aligned} T(k+2) &= 2T(k) + 1 \\ &= 2(2\sqrt{2^k} - 1) + 1 \\ &= 2\sqrt{2^2}\sqrt{2^k} - 2 + 1 \\ &= 2\sqrt{2^2 \cdot 2^k} - 1 \\ &= 2\sqrt{2^{k+2}} - 1. \end{aligned}$$

We have shown $T(k+2) = 2\sqrt{2^{k+2}} - 1$. Now the conclusion follows from the basis step, the inductive step, and the principle of induction. Hence, we conclude $T(n) \in \Theta(\sqrt{2^n})$. To verify this, we will verify $T(n) \in O(\sqrt{2^n})$ and $T(n) \in \Omega(\sqrt{2^n})$.

- $T(n) \in O(\sqrt{2^n})$: we need to show $T(n) \leq c\sqrt{2^n}$ for $n \geq n_0$ where $c, n_0 \in \mathbb{N}$. Let us choose $c = 2$. Then we have

  $$T(n) = 2\sqrt{2^n} - 1 \leq c\sqrt{2^n} = 2\sqrt{2^n}.$$

  This is true for all n_0, hence we have shown $T(n) \in O(\sqrt{2^n})$.

- $T(n) \in \Omega(\sqrt{2^n})$: we need to show $T(n) \geq c\sqrt{2^n}$ for $n \geq n_0$ where $c, n_0 \in \mathbb{N}$. If we select $n_0 \geq 2$, then we have $\sqrt{2^n} \geq 2$ for all $n \geq n_0$. If we also select $c = 1$, then it then follows that the difference between $2\sqrt{2^n}$ and $c\sqrt{2^n} = \sqrt{2^n}$ is at least 2 for all $n \geq n_0$. Then it follows that

  $$T(n) = 2\sqrt{2^n} - 1 \geq c\sqrt{2^n} = \sqrt{2^n}$$

  since the difference between $2\sqrt{2^n} - 1$ and $\sqrt{2^n}$ is always at least 1. Hence, $T(n) \in \Omega(\sqrt{2^n})$.

We conclude $T(n) \in \Theta(\sqrt{2^n})$.

Problem 2.

a) We have $a = 8$, $b = 4$, and $f(n) = n^2$. We have that $\log_b(a) = \log_4(8) = 3/2$, so $n^{\log_b(a)} = n^{3/2}$. We see that $f(n) \in \Omega(n^{3/2+\epsilon})$ for $\epsilon = 1/2$, so case 3 applies. We can also confirm $af(n/b) \leq \delta f(n)$ for some constant $\delta < 1$ for all sufficiently large n. We have that

$$\begin{aligned} af(n/b) &= 8\left(\frac{n}{4}\right)^2 \\ &= \frac{8}{16}n^2 \\ &= \frac{n^2}{2} \\ &\leq \delta f(n) \end{aligned}$$

which holds for $\delta = 1/2$. As a result, we conclude $T(n) \in \Theta(n^2)$.

b) We have $a = 5$, $b = 9$, and $f(n) = \sqrt{n}$. We have that $\log_b(a) = \log_9(5) \approx 0.732$, so $n^{\log_b(a)} = n^{\log_9(5)} \approx n^{0.732}$. We see that $f(n) \in O(n^{\log_9(5)-\epsilon})$ for $\epsilon = 0.1$, so case 1 applies. We therefore conclude $T(n) \in \Theta(n^{\log_9(5)})$.

c) We have $a = 2$, $b = 4$, and $f(n) = 1$. We have that $\log_b(a) = \log_4(2) = 1/2$, so $n^{\log_b(a)} = n^{1/2}$. We see that $f(n) \in O(n^{1/2-\epsilon})$ for $\epsilon = 0.1$, so case 1 applies. We therefore conclude $T(n) \in \Theta(\sqrt{n})$.

d) We have $a = 9$, $b = 8$, and $f(n) = n^2 + n$. We have that $\log_b(a) = \log_8 9 \approx 1.057$, so $n^{\log_b(a)} = n^{\log_8(9)} \approx n^{1.057}$. We see that $f(n) \in \Omega(n^{\log_8(9)+\epsilon})$ for $\epsilon = 0.9$, so case 3 applies. We can also confirm $af(n/b) \leq \delta f(n)$ for some constant $\delta < 1$ for all sufficiently large n. We have that

$$\begin{aligned} af(n/b) &= 9\left(\frac{n}{8}\right)^2 + 9\left(\frac{n}{8}\right) \\ &= \frac{9}{64}n^2 + \frac{9}{8}n. \end{aligned}$$

Let us select $\delta = 1/2$. Subtracting $af(n/b)$ from $\delta f(n) = f(n)/2$ yields

$$\begin{aligned} \frac{n^2}{2} + \frac{n}{2} - \left(\frac{9}{64}n^2 + \frac{9}{8}n\right) &= \frac{23}{64}n^2 - \frac{5}{8}n \\ &= \frac{23}{64}n\left(n - \frac{64}{23}\cdot\frac{5}{8}\right) \\ &= \frac{23}{64}n\left(n - \frac{40}{23}\right) \end{aligned}$$

Since the above result is at least zero for all $n \geq 40/23$, we conclude that $af(n/b) \leq \delta f(n)$ for $\delta = 1/2$ when $n \geq 40/23$. As a result, we conclude $T(n) \in \Theta(n^2)$.

Problem 3.

a) **Claim:** For any array A and some natural number n such that A has at least $n = e - b + 1$ cells, subarray $A[b..e]$ is sorted when `SomeSort`(A, b, e) terminates for any b, e. Proof by strong induction.

Basis step: We have two base cases:

- $n = 1$: the algorithm terminates immediately. There is only one element in array A so A is sorted by definition.
- $n = 2$: this implies $e = b + 1$. We enter the first `if` statement. If $A[b] > A[e]$, then we simply swap $A[b]$ and $A[e]$ which leads to subarray $A[e..b]$ being sorted. Otherwise, if $A[b] \leq A[e]$, subarray $A[e..b]$ is already sorted.

Inductive step: Let $n \geq 2$ be an arbitrary integer. We assume the claim holds for all integers from 1 to n, now we show the claim holds for $n + 1$.

We call `SomeSort` where $n + 1$ is the length array A. We calculate $p = \left\lfloor \frac{n+1}{3} \right\rfloor$. Since $n \geq 2$, then $n + 1 \geq 3$ and hence $p \geq 1$. Using p, we will be able to divide A into three equal parts. Let us call these sections S_1, S_2, and S_3.

We call `SomeSort` on the first two sections of A; namely, S_1 and S_2. The length of the subarray that makes up the first two sections has length $(n+1)-p$. Since $p \geq 1$, the subarray has length of at most n. By our assumption, `SomeSort` on an array of length n or less will terminate and result in a correctly sorted subarray. Hence, after this call to `SomeSort`, the first two sections of A will be sorted. We can also deduce that all the elements in S_2 will be greater than or equal to any element in S_1.

Next, we call `SomeSort` on the last two sections of A; namely, S_2 and S_3. For the same reasoning as the previous call, the last two sections of A will be sorted after this call to `SomeSort` terminates. This means all elements in S_3 will be greater than or equal to any element in S_2. We can further deduce that all elements in S_3 will be greater than or equal to any element in S_1 because all elements in S_2 were greater than or equal to any element in S_1. As such, we can conclude that all elements in S_3 are in the correct place since sections S_2 and S_3 combined are sorted. This implies all the remaining elements that need to be sorted are in the first two sections of A.

Finally, we call `SomeSort` on the first two sections of A again; namely, S_1 and S_2. For the same reasoning as the previous calls, the first two sections of A will be sorted after this call to `SomeSort` terminates. Now we can conclude that all elements in sections S_1, S_2, and S_3 are sorted.

It then follows that A is sorted and that the claim holds for $n+1$. Now the conclusion follows from the basis step, the inductive step, and the principle of strong induction.

b) Five valid states are:

- 1, 8, 4, 9, 7, 3, 2, 6, 5
- 1, 4, 8, 7, 9, 3, 2, 6, 5
- 1, 4, 7, 8, 3, 9, 2, 6, 5
- 1, 3, 4, 7, 8, 2, 9, 6, 5
- 1, 3, 4, 2, 7, 8, 6, 9, 5

c) The recurrence is given by $T(n) = 3T(2n/3) + c$ where c is some constant. We now apply the master method to solve the recurrence. We have $a = 3$, $b = 3/2$, and $f(n) = c$. We have that $\log_b(a) = \log_{3/2}(3) \approx 2.71$, so $n^{\log_b(a)} = n^{\log_{3/2}(3)} \approx n^{2.71}$. We see that $f(n) \in O(n^{\log_{3/2}(3)-\epsilon})$ for $\epsilon = 1$ (although any $0 < \epsilon < \log_{3/2}(3)$ would work), so case 1 applies. We therefore conclude $T(n) \in \Theta(n^{\log_{3/2}(3)})$.

d) `SomeSort` is very inefficient compared to insertion sort and merge sort. The worst-case running time of `SomeSort` is $\Theta(n^{\log_{3/2}(3)})$ or about $\Theta(n^{2.71})$. On the other hand, the worst-case running time of insertion sort is $\Theta(n^2)$ and the worst-case running time of merge sort is $\Theta(n\log(n))$. When n is large, `SomeSort` will run incredibly slowly compared to either insertion sort or merge sort.