1) a) By definition $\limsup_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty}(\sup\{\frac{f(m)}{g(m)} \mid m \geq n\})$. With high school math it holds for all $n \in \mathbb{N}$ that $0 \leq \frac{f(n)}{g(n)} \leq \frac{n}{n^3} + \frac{\sqrt{n}}{n^3} = \frac{1}{n^2} + \frac{1}{\sqrt{n^5}}$. Since $\lim_{n\to\infty} \frac{1}{n^2} = 0$ and $\lim_{n\to\infty} \frac{1}{\sqrt{n^5}} = 0$ it follows that $\limsup_{n\to\infty} \frac{f(n)}{g(n)} = \liminf_{n\to\infty} \frac{f(n)}{g(n)} = 0$. Thus, $f \in \mathrm{O}(g)$ and $f \notin \Omega(g)$. Hence, $f \notin \Theta(g)$.

b)
- Counterexample: Let $f(n) = g(n) = h(n) = n + 1$. We have $f(n) \geq h(n)$ and $g(n) \geq h(n)$. With the parameters $c = 1$ and $m = 1$ from the Big-Omega definition it follows that $f, g \in \Omega(h)$. But for all $n \in \mathbb{N}$ we have $\frac{f(n)}{g(n)} = 1$ and $\lim_{n\to\infty} \frac{1}{h(n)} = \liminf_{n\to\infty} \frac{1}{h(n)} = 0$, thus, $\frac{f}{g} \notin \Omega(h)$.
- Proof: By definition we have $c_f, c_g > 0$ and $m_f, m_g \in \mathbb{N}$ such that $f(n) \leq c_f \cdot g(n)$ for all $n \geq m_f$ and $g(n) \leq c_g \cdot h(n)$ for all $n \geq m_g$. Inserting the second inequality in the first gives us $f(n) \leq c_f \cdot c_g \cdot h(n)$. With $c = c_f \cdot c_g$ and $m = \max(m_f, m_g)$ from the Big-O definition it follows that $f \in \mathrm{O}(h)$.

2) a) In order to apply the master theorem, a recurrence equation of particular shape is required. However, no matter how we specify the parameters in the equation of this shape, we can always find a counterexample tree that is imbalanced in such a way that the recurrence is not satisfied. Therefore, it is not possible to apply the master theorem.

b) For balanced trees, we can specify a recurrence in the form satisfied by the master theorem. Assuming the allocation of the node takes $O(1)$ and the subtrees can be reused, the recurrence is:

$$T(n) = 2T(\frac{n}{2}) + O(1)$$

which allows the application of the master theorem.

c) From the equation:

$$T(n) = 2T(\frac{n}{2}) + O(1)$$

We specify the parameters to use in the application of the master theorem:

$$a = 2 \qquad b = 2 \qquad s = 0$$

As $a > b^s$, we are in the first case of the theorem. The theorem then states that:

$$\texttt{mirror} \in \Theta(n^{log_b a}) = \Theta(n)$$

An alternative solution might give a different assumption. For example, a solution might consider only trees that have only left children, or that are of a limited size. Then different answers are possible.

3) a) $1, 1, 4, 7, 16, 31, 64, 127, 256, 511, 1024$, e.g. by evaluating `map tp [0..10]` in Haskell for:

```
t n = if n < 4 then 1 else t (n `div` 2) + 2 * t (n `div` 4) + 1
tp k = t (2^k)
```

b) For the Master Theorem to apply to a recurrence, it must have a *single* recursive call to the function being defined. Here there are *two* such calls, namely $T(\frac{n}{2})$ and $T(\frac{n}{4})$.
Remark: If it is known that T is increasing, one can try to consider an alternative recurrence to which the Master Theorem does apply and such that T is bounded (from above) by it, e.g. here one may consider $U(n) = 3 \cdot U(\frac{n}{2}) + 1$ as $T \leq U$, using that $T(\frac{n}{4}) \leq T(\frac{n}{2})$ for all n. However, U is only an upper bound to T and typically, and also in this case as $a = 3$, $b = 2$ and $s = 0$ yielding solution $n^{\log_2 3}$, U may be *strictly* larger (also qua asymptotic complexity) than T.

c) We choose the first option.[1] First observing that the sequence in the first item progresses *almost* as powers of 2, and next that the difference is 1 for *odd* powers, i.e. $2^0, 2^1 - 1, 2^2, 2^3 - 1, \ldots$, we *guess* the solution for inputs of shape 2^k is the function f that maps 2^k to 2^k if k is even and to $2^k - 1$ otherwise.

It remains to show that f is indeed the solution for the recurrence, i.e. that $f(2^k) = f(2^{k-1}) + 2 \cdot f(2^{k-2}) + 1$ if $k \geq 2$, and $f(2^k) = 1$ for $k = 0$ and $k = 1$. This we prove by universal generalisation and case distinction on k. If $k = 0$ then $f(2^k) = f(1) = 1$ as desired, since k is even. If $k = 1$ then $f(2^k) = f(2) = 2 - 1 = 1$ as desired, since k is odd. If $k \geq 2$ and k is even, then $k - 1$ is odd and $k - 2$ is even, so

$$f(2^k) = 2^k = 2 \cdot 2^{k-1} = (2^{k-1} - 1) + 2 \cdot (2^{k-2}) + 1 = f(2^{k-1}) + 2 \cdot f(2^{k-2}) + 1$$

If $k \geq 2$ and k is odd, then $k - 1$ is even and $k - 2$ is odd, so

$$f(2^k) = 2^k - 1 = 2 \cdot 2^{k-1} - 1 = (2^{k-1}) + 2 \cdot (2^{k-2} - 1) + 1 = f(2^{k-1}) + 2 \cdot f(2^{k-2}) + 1$$

By f being (almost) the identity function we conclude the asymptotic complexity is in $\Theta(n)$

4*) We choose the first option and define

```
f 0 = 1
f n = if n `mod` 2 == 0 then f (n-1) + 1 else (n+1) * g n
g n = if n `mod` 2 == 0 then (n+1) * f n else g (n-1) + 1
```

with the idea (as given in the lecture) that if for some input n $f(n) < g(n)$, then $f(n+1) > g(n+1)$ and vice versa, and moreover the factor by which the one is larger than the other is ever increasing (here we have taken it to be $n+1$), so neither is bounded by a *constant* factor times the other. Listing the first few values for f yields $[1, 4, 5, 64, 65, 1956, 1957, 109600, 109601, 9864100, 9864101]$ and for g $[1, 2, 15, 16, 325, 326, 13699, 13700, 986409, 986410, 108505111]$. Note the alternation between f and g (for the same input).

That f and g defined in this way are monotonically increasing follows from that both are defined recursively, with the value for some input n based on the value for $n - 1$ by adding 1 possibly combined with multiplications by positive numbers. We show $f \notin O(g)$ by a proof by contradiction. Suppose $f \in O(g)$, .i.e. suppose $f(n) \leq c \cdot g(n)$ for all $n \geq m$, for some c and m. Take n' to be some odd natural number greater than or equal to both c and m. Then $f(n') = (n' + 1) \cdot g(n') > n' \cdot g(n') \geq c \cdot g(n')$, contradicting the assumption. That $g \notin O(f)$ follows analogously.

The idea of two functions 'taking turns' in becoming either just incremented or multiplied by an arbitrary factor may be generalised to infinitely many functions. To see this, assume

[1] The second option boils down to showing the same, but by well-founded induction instead of by universal generalisation and case distinction.

to have a function e on the natural numbers such that for all i, $e^{-1}(i)$ is infinite (intuition: $e(n)$ determines whose turn it is to be multiplied for input n, with the other functions only being incremented; the condition expresses that each i gets infinitely many turns; a possibility for e is to produce ever increasing initial segments $[0, 0, 1, 0, 1, 2, 0, 1, 2, 3, \ldots]$). Then define $f_i(0) = 1$ and for $n > 0$, $f_i(n) = n \cdot x$ if $e(n) = i$ and x otherwise, where $x = 1 + f_{e(n-1)}(n-1)$. Using that $f_{e(k)}(k) = \max_j f_j(k)$ (the maximum of all $f_j(k)$ is attained by the function 'whose turn it is' $f_{e(k)}$), it follows that this collection of functions satisfies the conditions as above. In particular, to show $f_i \notin O(f_j)$ we may take for n' a number such that $e(n') = i$ with n' greater than or equal to both constants witnessing the purported $f_i \in O(f_j)$ (such an n' exists by the assumption on e).