The Master Theorem For a recurrence like $T(n) = aT(\frac{n}{b}) + f(n)$, where $a \geq 1$ and $b > 1$, the Master Theorem states three cases:

1. If $f(n) \in O(n^c)$ where $c < \log_b a$ then $T(n) \in \Theta(n^{\log_b a})$.
2. If for some constant $k \geq 0$, $f(n) \in \Theta(n^c(\log n)^k)$ where $c = \log_b a$, then $T(n) \in \Theta(n^c(\log n)^{k+1})$.
3. If $f(n) \in \Omega(n^c)$ where $c > \log_b a$ **and** $af(\frac{n}{b}) \leq kf(n)$ for some constant $k < 1$ and sufficiently large n, then $T(n) \in \Theta(f(n))$.

1 I'm a Lumberjack (And I'm Okay)

Your task is to design algorithms to solve the following problems. For full credit, your algorithm must run in logarithmic time.

1.1 Array-chopping

An *arithmetic array* is one whose elements form an arithmetic sequence, in order – i.e., they're arrays of the form

$$A = [a_1,\ a_1 + c,\ a_1 + 2c, \ldots,\ a_1 + (n-1)c],$$

where A has length n (for $n \geq 2$). You're given an arithmetic array with one element missing from somewhere in the middle (i.e., it's not the first or last element that's been removed).

For example, the missing number in $[3, 6, 12, 15, 18]$ is 9. The missing number in $[1, 15, 22, 29, 36]$ is 8.

1. Describe a way to calculate c in constant time.

2. Design an algorithm to efficiently find the missing number in the array.

3. Briefly justify a good asymptotic runtime of your algorithm.