1. For this algorithm, $n = e - b + 1$ is the number of elements in $A[b \dots e]$.

   When $n = 0$ (*i.e.*, $b = e + 1$), the algorithm executes only the first line, in constant time. So

   $$T(0) = 1.$$

   (Remember that this really represents $\Theta(1)$.)

   When $n > 0$ (*i.e.*, $b < e+1$), the algorithm makes two recursive calls on inputs of size exactly $\lfloor n/4 \rfloor$, executes a loop that iterates exactly $n - 2\lfloor n/4 \rfloor$ times (performing a constant amount of work at each iteration), and performs a constant amount of additional work. Hence,

   $$T(n) = 2T\big(\lfloor n/4 \rfloor\big) + n \qquad \text{for } n \geqslant 1.$$

   (Remember that the term "$+n$" really represents "$+\Theta(n)$", which is the amount of work performed outside the recursive calls.)

2. When performing repeated substitution, we simplify the exact recurrence by ignoring floors and ceilings:

   $$\begin{aligned} T(n) &\approx 2T(n/4) + n \\ &\approx 2\big(2T(n/16) + n/4\big) + n \\ &= 4T(n/16) + 3n/2 \\ &\approx 4\big(2T(n/64) + n/16\big) + 3n/2 \\ &= 8T(n/64) + 7n/4 \end{aligned}$$

   After k substitutions, we expect:

   $$\begin{aligned} &\approx 2^k T(n/4^k) + \frac{2^k - 1}{2^{k-1}} n \\ &= 2^k T(n/4^k) + 2n - 2n/2^k \end{aligned}$$

   The base case is reached when $n/4^k = 1$, *i.e.*, $k = \log_4 n$:

   $$\begin{aligned} &\approx 2^{\log_4 n} T(n/n) + 2n - 2n/2^{\log_4 n} \\ &= n^{\log_4 2} T(1) + 2n - 2n/n^{\log_4 2} \\ &= 3\sqrt{n} + 2n - 2\sqrt{n} \\ &= 2n + \sqrt{n} \end{aligned}$$

   Hence, $T(n) \approx 2n + \sqrt{n}$.