Program testing can be used very effectively to show the presence of bugs but never to show their absence.

E.W. Dijkstra, EWD 303

All by itself, a program is no more than half a conjecture. The other half of the conjecture is the functional specification the program is supposed to satisfy. The programmer's task is to present such complete conjectures as proven theorems.

E.W. Dijkstra, EWD 1036

Bounded linear search returns

$$\begin{array}{ll} -1 & \text{if } \forall\ i \in [0, n), \sim P(\texttt{sequence}[i]) \\ k & \text{otherwise, where } P(\texttt{sequence}[k]) \\ & \qquad \text{and } \forall\ i \in [0, k), \sim P(\texttt{sequence}[i]) \end{array}$$

Invariant (Loop of `bounded_linear_search`.)

(a) $\forall\ j \in [0, i-1), \sim P(\texttt{sequence}[j])$

(b) `found` *iff* $P(\texttt{sequence}[i-1])$

(c) *i is the number of iterations completed.*

Initialization.

(a) Since i is initially 0, the range $[0, i) = [0, 0)$ which is empty. Hence the proposition is vacuously true.

(b) Assuming $\sim P(\mathit{undef})$ makes this hold.

(c) There have been 0 iterations, and $i = 0$.

Maintenance. Distinguish i_{pre} and i_{post}, namely $i_{\text{post}} = i_{\text{pre}} + 1$. Similarly distinguish $\texttt{found}_{\text{pre}}$ and $\texttt{found}_{\text{post}}$

(a) It must be that $\sim \texttt{found}_{\text{pre}}$ or else the guard would have failed. Thus $\sim P(\texttt{sequence}[i_{\text{pre}} - 1])$, by *inductive hypothesis*, part b. Together with the fact that that $\forall\, j \in [0, i_{\text{pre}} - 1), \sim P(\texttt{sequence}[j])$, we now have $\forall\, j \in [0, i_{\text{pre}}), \sim P(\texttt{sequence}[j])$, that is $\forall\, j \in [0, i_{\text{post}} - 1), \sim P(\texttt{sequence}[j])$.

(b) Immediate from the assignment to `found`.

(c) Immediate from the update to i. □

Correctness claim (`bounded_linear_search`.)

After at most n iterations, `bounded_linear_search` *will return as specified.*

Proof. By Invariant 1.c, after at most n iterations, $i = n$ and the guard will fail. Moreover, when the guard fails, either `found` or $i = n$.

Case 1. Suppose `found`. Then we return $i - 1$. Invariant 1.a tells us that nothing in $[0, i - 1)$ satisfies P. Invariant 1.b tells us that $i - 1$ does. Together these fulfill the second part of the specification: $i - 1$ is the first item satisfying P, and we return it.

Case 2. Suppose $\sim$ `found`. By elimination $i = n$. Invariant 1.a tells us that nothing in $[0, n - 1)$ satisfies P. Invariant 1.b tells us that $i - 1$, that is, $n - 1$, also does not satisfy P. We return -1, fulfilling the first part of the specification. $\square$

Binary search returns

$$\begin{array}{ll} -1 & \text{if } \forall\ i \in [0, n), \texttt{sequence}[i] \neq \texttt{item} \\ k & \text{otherwise, where } \texttt{sequence}[k] = \texttt{item} \end{array}$$

Invariant (Loop of `binary_search`.)

(a) *If* $\exists\ j \in [0, n)$ *such that* $\texttt{item} = \texttt{sequence}[j]$, *then* $\exists\ j \in [\texttt{low}, \texttt{high})$ *such that* $\texttt{item} = \texttt{sequence}[j]$.

(b) *After* i *iterations,* $\texttt{high} - \texttt{low} \leq \frac{n}{2^i}$.

Initialization.

(a) Initially $\texttt{low} = 0$ and $\texttt{high} = n$, so the hypothesis and conclusion are identical.

(b) No iterations yet, so $\texttt{high} - \texttt{low} = n - 0 = n = \frac{n}{1} = \frac{n}{2^0}$

Maintenance. Distinguish `pre` and `post` variables. Let i be the the number of iterations completed. The inductive hypothesis says:

(a) If $\exists\, j \in [0, n)$ such that $\mathit{item} = \texttt{sequence}[j]$, then $\exists\, j \in [\texttt{low}_{\text{pre}}, \texttt{high}_{\text{pre}})$ such that $\mathit{item} = \texttt{sequence}[j]$

(b) $\texttt{high}_{\text{pre}} - \texttt{low}_{\text{pre}} \leq \frac{n}{2^{i-1}}$

The guard also assures us that $\texttt{high}_{\text{pre}} - \texttt{low}_{\text{pre}} > 1$.
We have three possibilities, corresponding to the if-elif-else:

Case 1: Suppose $\texttt{item} < \texttt{sequence}[\texttt{mid}]$.

(a) Since sequence is sorted, $\forall\, j \in [\texttt{mid}, \texttt{high}_{\text{pre}})$, $\texttt{item} < \texttt{sequence}[j]$. Thus if $\exists\, j \in [\texttt{low}_{\text{pre}}, \texttt{high}_{\text{pre}})$, then $\exists\, j \in [\texttt{low}_{\text{pre}}, \texttt{mid})$, that is (with the update to high but not to low), $\exists\, j \in [\texttt{low}_{\text{post}}, \texttt{high}_{\text{post}})$
Now, by transitivity of the conditional, if $\exists\, j \in [0, n)$ such that $\mathit{item} = \texttt{sequence}[j]$, then $\exists\, j \in [\texttt{low}_{\text{post}}, \texttt{high}_{\text{post}})$ such that $\mathit{item} = \texttt{sequence}[j]$.

(b) If the length of the range is odd, then the sub-ranges above and below mid are of equal size, each half of the range length minus one. If the range length is even, then the lower subrange is half that size and the upper subrange is one less than half. Either way we throw away at least half and keep no more than half. So,

$$\texttt{high}_{\text{post}} - \texttt{low}_{\text{post}} \leq \frac{1}{2} \cdot (\texttt{high}_{\text{pre}} - \texttt{low}_{\text{pre}}) \leq \frac{1}{2} \cdot \frac{n}{2^{i-1}} \leq \frac{n}{2^i}$$

Case 2: Suppose $\texttt{item} = \texttt{sequence}[\texttt{mid}]$.

(a) Immediately we have $\exists\, j \in [\texttt{mid}, \texttt{mid} + 1)$, and, with the update to $\texttt{high}$ and $\texttt{low}$, that means $\exists\, j \in [\texttt{low}_{\text{post}}, \texttt{high}_{\text{post}})$. Moreover, the conditional is $T \to T \equiv T$.

(b) Note $\texttt{high}_{\text{post}} - \texttt{low}_{\text{post}} = 1$. Earlier we said $1 < \texttt{high}_{\text{pre}} - \texttt{low}_{\text{pre}} \leq \frac{n}{2^{i-1}}$. Since $\texttt{high}_{\text{pre}} - \texttt{low}_{\text{pre}}$ must be a whole number, $2 \leq \frac{n}{2^{i-1}}$, and so $1 \leq \frac{n}{2^i}$. Finally $\texttt{high}_{\text{post}} - \texttt{low}_{\text{post}} \leq \frac{n}{2^i}$.

Case 3: Suppose $\texttt{item} > \texttt{sequence}[\texttt{mid}]$. Similar to Case 1. □

Correctness claim (`binary_search`.)

After at most lg n *iterations,* `binary_search` *returns as specified.*

Proof. Suppose $i \geq \lg\ n$. Then $2^i \geq n$ and $\frac{n}{2^i} \leq 1$. Hence $\texttt{high} - \texttt{low} \leq 1$ and the guard fails.

Part a of the invariant means that if the item is anywhere, it's in the range. By the guard, on loop exit the range has size 0 or 1.

Case 1. Suppose the range has size 0. Then the item isn't in the range, and thus it isn't anywhere. Since $\texttt{high} = \texttt{low}$, the first part of the conditional fails and and -1 is returned, as specified.

Case 2. Suppose the range has size 1. We still don't know if the item is in the range, but we have only one location to check. If it's in `sequence[low]`, then we return `low`, which meets the specification. Otherwise the second part of the condition fails and -1 is returned, as specified. □

Invariant (Class `Exercise2`.)

(a) *`head == null` iff `tail == null` iff `size == 0`.*

(b) *If `tail != null` then `tail.next == null`.*

(c) *If `head != null` then `tail` is reached by following `size - 1` `next` links from `head`.*