3.4.0.25)

Consider Equation (3.3). An alternative loop invariant can be derived as

{*Pinv* : a ≤ i ≤ b ∧ α = α-hat + ∑ j | i ≤ j ≤ n : x(j) \* y(j) }

1. Determine the bounds *a* and *b*.

If *i* = *b* = *n*+1, the sum is over the empty range.

If *i* = *a* = 1, then the sum equals the desired result.

Hence we take *a* = 1 and *b* = *n* + 1.

2. Annotate the algorithm in Figure 3.7 as much as you can without determining *G*, e1, and e2.

{α = α-hat ∧ 1 ≤ n}

{a ≤ e1 ≤ b ∧ α = α-hat + ∑ j | e1 ≤ j ≤ n : x(j) \* y(j) }

*i* := (e1)

{a ≤ i ≤ b ∧ α = α-hat + ∑ j | i ≤ j ≤ n : x(j) \* y(j) }

while (*G*)

{*Pinv* ∧ *G*: a ≤ i ≤ b ∧ α = α-hat + ∑ j | i ≤ j ≤ n : x(j) \* y(j) ∧ *G*}

a := (e2)

{a ≤ i - 1 ≤ b ∧ e2 = α-hat + ∑ j | i - 1 ≤ j ≤ n : x(j) \* y(j) }

i := i+1

{*Pinv* :{a ≤ i - 1 ≤ b ∧ e2 = α-hat + ∑ j | i - 1 ≤ j ≤ n : x(j) \* y(j) }}

endwhile

{*Pinv* ∧ ¬*G*: a ≤ i ≤ b ∧ α = α-hat + ∑ j | i ≤ j ≤ n : x(j) \* y(j) ∧ ¬*G*}

{R: α = α-hat + ∑ j | 1 ≤ j ≤ n : x(j) \* y(j) }

3. Determine *G*, e1, and e2.

G: i = 1

e1: e1 = n + 1

e2: e2 = α + x(i - 1) \* y(i - 1)

4. Fill Figure 3.7 with the final result.

{α = α-hat ∧ 1 ≤ n}

{a ≤ n+1 ≤ b ∧ α = α-hat + ∑ j | n+1 ≤ j ≤ n : x(j) \* y(j) }

*i* := n+1

{a ≤ i ≤ b ∧ α = α-hat + ∑ j | i ≤ j ≤ n : x(j) \* y(j) }

while (1 ≤ i)

{*Pinv* ∧ 1 ≤ i : a ≤ i ≤ b ∧ α = α-hat + ∑ j | i ≤ j ≤ n : x(j) \* y(j) ∧ 1 ≤ i}

a := α + x(i - 1) \* y(i - 1)

{a ≤ i - 1 ≤ b ∧ α + x(i - 1) \* y(i - 1) = α-hat + ∑ j | i - 1 ≤ j ≤ n : x(j) \* y(j) }

i := i+1

{*Pinv* :{a ≤ i - 1 ≤ b ∧ α + x(i - 1) \* y(i - 1) = α-hat + ∑ j | i - 1 ≤ j ≤ n : x(j) \* y(j) }}

endwhile

{*Pinv* ∧ i < 1: a ≤ i ≤ b ∧ α = α-hat + ∑ j | i ≤ j ≤ n : x(j) \* y(j) ∧ i < 1}

{R: α = α-hat + ∑ j | 1 ≤ j ≤ n : x(j) \* y(j) }