(P1)

from

i := 1; j := n

until i = j loop

m := (i + j) // 2

if t [m] <= x then

i := m

else

j := m -- j := m - 1

end

end

Result := (x = t[i])

The algorithm is incorrect.

Suppose we have an array [1,2,3] and we want to check if there’s a number 2 in it

Let’s run the algorithm step by step

i = 1, j = 3 🡪 m = 2, t[m] = 2 <= 2 🡪 i = 2, j = 3 🡪 m = 2, t[m] = 2 <= 2 🡪 i = 2, j = 3 🡪 …

We’ve stuck in infinite loop.

The changes that fix the algorithm are written after “--“ symbol

(P2)

from

i := 1; j := n; Result := False

until i = j and not Result loop

m := (i + j) // 2

if t [m] < x then

i := m + 1

elseif t [m] = x then

Result := True

else

j := m - 1

end

end

The algorithm is incorrect.

Suppose we have an array [1,2,3] and we want to check if there’s a number 2 in it.

Let’s run the algorithm step by step

i = 1, j = 3, Result = false 🡪 m = 2, t[m] = 2 = 2 🡪 Result = true 🡪 …

So in until statement we have (i = j and not Result) which is always false because (not Result) is always false as (Result) is always true. To fix this problem we may change until statement in following way:

until i = j or Result loop

(P3)

from

i := 0; j := n; -- i := 1; j := n

until i = j loop

m := (i + j + 1) // 2 -- m := (i + j ) // 2

if t [m] <= x then -- if t [m] < x then

i := m + 1

else

j := m

end

end

if i >= 1 and i <= n then

Result := (x = t [i])

else

Result := False

End

The algorithm is incorrect.

Suppose we have an array [1,2,3] and we want to check if there’s a number 1 in it.

Let’s run the algorithm step by step

i = 0, j = 3 🡪 m = 2, t[m] = 2 > 1 🡪 i = 0, j = 2 🡪 m = 1, t[m] = 1 <= 1 🡪 i = m + 1 = 2 = j 🡪 end of the loop 🡪 i >= 1 and i <= 3 🡪 Result = (1 = 2) = false, though we have number 1 in the array.

The changes that fix the algorithm are written after “--“ symbol

(P4)

from

i := 0; j := n + 1; -- i = 1; j = n

until i = j loop

m := (i + j) // 2

if t [m] <= x then -- if t [m] < x then

i := m + 1

else

j := m

end

end

if i >= 1 and i <= n then

Result := (x = t [i])

else Result := False

end

The algorithm is incorrect.

Suppose we have an array [1,2,3] and we want to check if there’s a number 3 in it.

Let’s run the algorithm step by step

i = 0, j = 4 🡪 m = 2, t[m] = 2 <= 3 🡪 i = 3, j = 4 🡪 m = 3, t[m] = 3 <= 3 🡪 i = 4 = j 🡪 end of the loop 🡪 --> (i >= 1 and i <= 3) = false 🡪 Result = false, though there’s a number 3 in the array

The changes that fix the algorithm are written after “--“ symbol