The following is the feature `duplicate` and some other features from class
`TWO_WAY_SORTED_SET`, which is a set containing an internal iterator. Try to devise a set of
test cases such that:
(1) All branches in `duplicate` are covered.
(2) All clauses in `duplicate` are covered.
(3) Try to devise a test case to reveal a bug in `duplicate`. Hint: analysis the preconditions of the
given features. Is this test case included in the test suite you devised in (1) or (2)? What do you
think about the used coverage criteria?

duplicate (n: INTEGER): like Current
   -- Copy of sub-set beginning at cursor position
   -- and having min (`n', `count' - `index' + 1) items
   local
      pos: CURSOR
      counter: INTEGER
   do
      pos := cursor; Result := new_chain; Result.finish; Result.forth
      from until (counter = n) or else after loop
        Result.put_left (item)
        forth
        counter := counter + 1
      end
      go_to (pos)
   end

duplicate (n: INTEGER): like Current
   -- Copy of sub-set beginning at cursor position
   -- and having min (`n', `count' - `index' + 1) items
   local
      pos: CURSOR
      counter: INTEGER
   do
      pos := cursor; Result := new_chain; Result.finish; Result.forth
      from until (counter = n) or else after loop
        Result.put_left (item)
        forth
        counter := counter + 1
      end
      go_to (pos)
   end

item: G
   -- Current item
   require
      not_off: not off

forth
   -- Move cursor to next position, if any.
   require
      not_after: not after
   ensure
      moved_forth: index = old index + 1

off: BOOLEAN
   -- Is there no current item?
   ensure
      Result = after or before
Solution

(1) There is only one branching statement, which is the loop.

s: TWO_WAY_SORTED_SET [INTEGER]
create s. make
s.extend (1)
s.start
s.duplicate (10)

(2) There are two clauses, namely, counter = n, after. We need to come up with test cases triggering both True and False for all the clauses.

TC1: counter=n: True/False

s: TWO_WAY_SORTED_SET [INTEGER]
create s. make
s.extend (1)
s.extend (2)
s.start
s.duplicate (1)

TC2: after: True/False

s: TWO_WAY_SORTED_SET [INTEGER]
create s. make
s.extend (1)
s.start
s.duplicate (10)

(3) s: TWO_WAY_SORTED_SET [INTEGER]
create s. make
s.duplicate (1) – calling duplicate when `s` is before will violates the precondition of item in the first iteration of the loop body.
Both the branch coverage and clause coverage may miss this case, thus, they are weak.