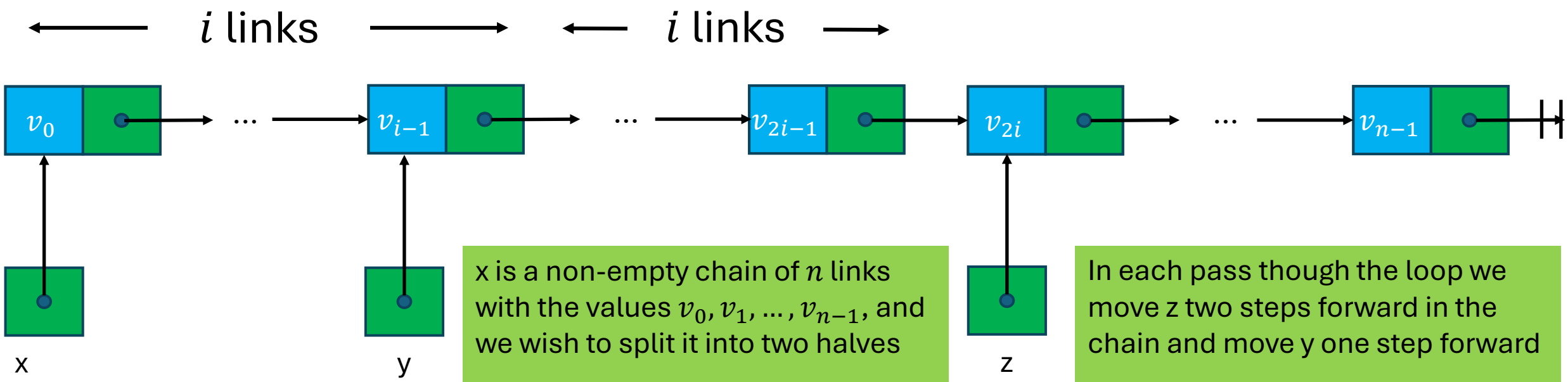
Sorting mutable lists

Merge sort
Insertion sort
Selection sort

Loop invariants
Recursive methods

Splitting a list destructively in a loop



x is a non-empty chain of n links with the values $v_0, v_1, \ldots, v_{n-1}$, and we wish to split it into two halves

y is some non-empty suffix of x with *i-1* links of x in front of the suffix

z is some suffix of x, maybe empty, with *2i* links of x in front of it

The case when x contains less than two links must be handled separately

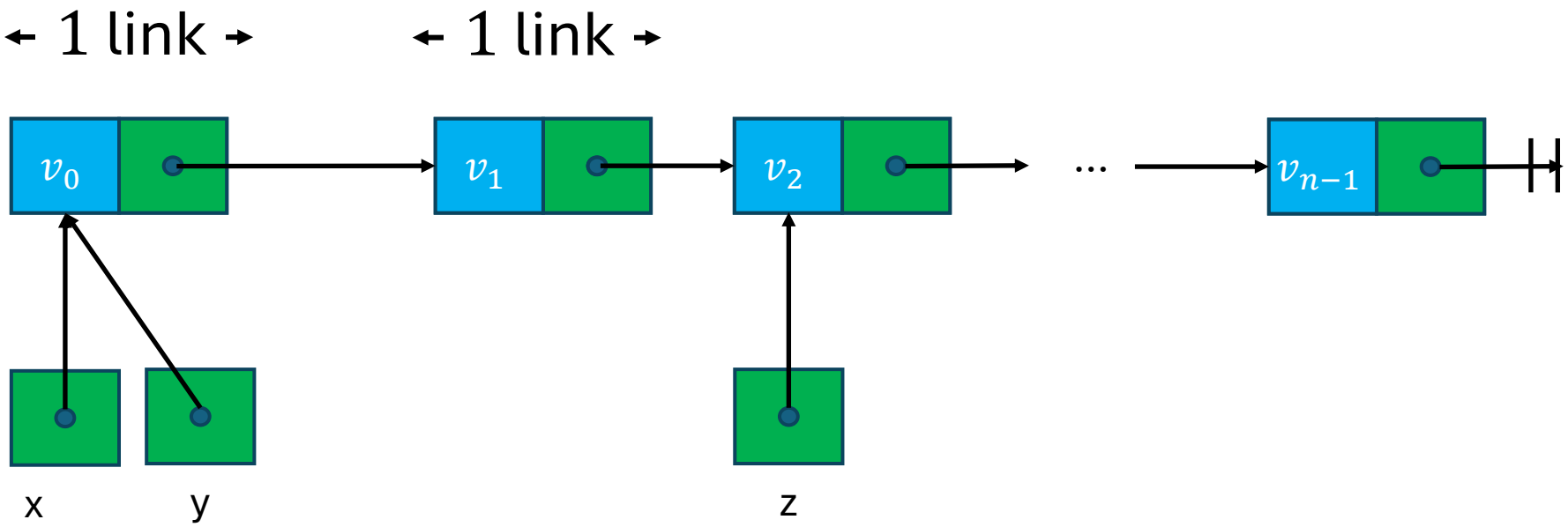
In each pass though the loop we move z two steps forward in the chain and move y one step forward

If that is not possible then the z chain contains at most one link

We then make a note of the tail of the link that y refers to and then change that tail to null

We then return x plus the old tail of y

Splitting a list destructively in a loop
Starting state



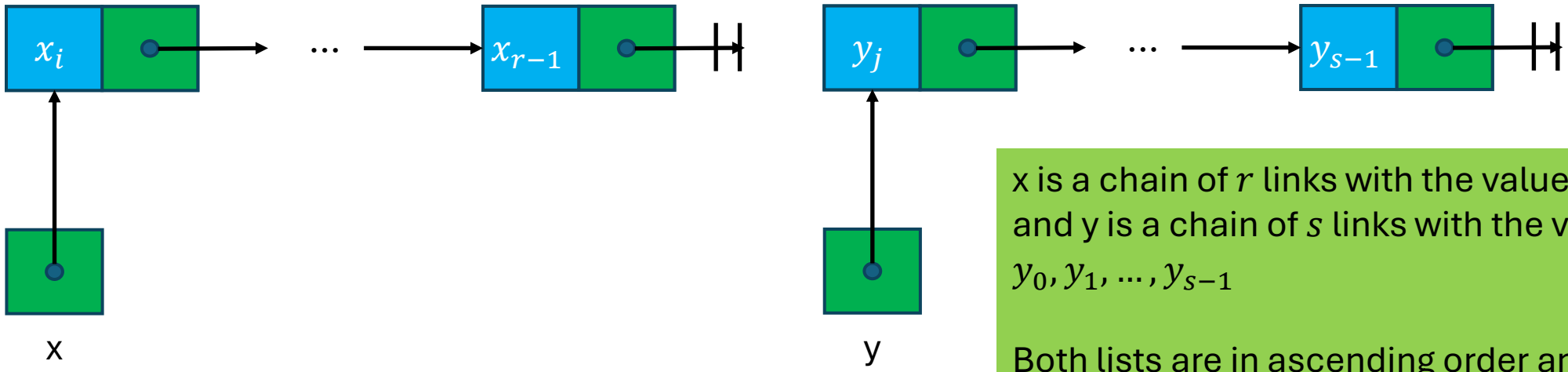
Splitting a list destructively using recursion

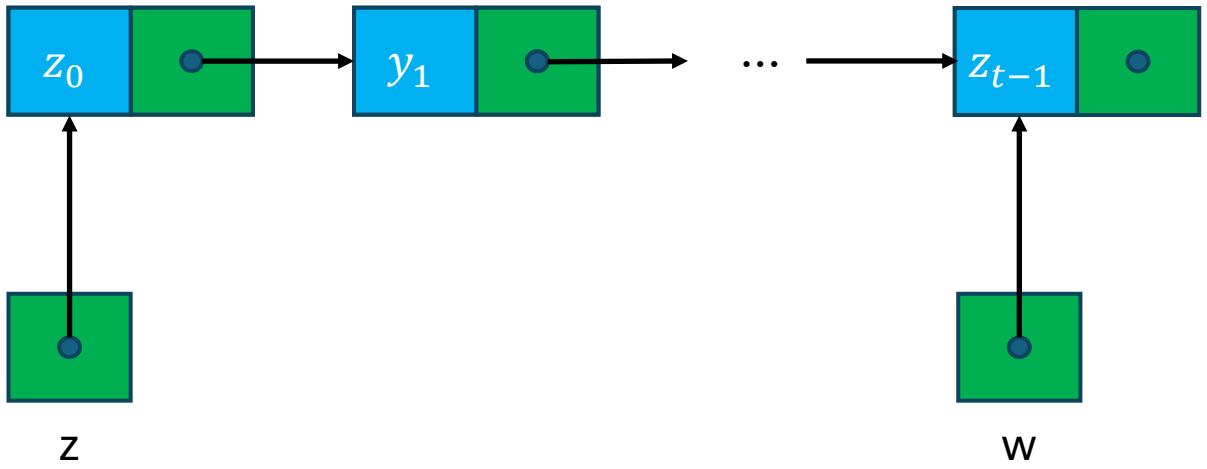
- If the list contains less than two links return the list unchanged, plus an empty list
- Otherwise recursively split the tail of the tail of the list into two halves and then add each of the two original frontmost links to the two halves, one link to each half
- This will reverse the values, but the order of the links is often irrelevant, for example when we are doing merge sort of random values

Destructively sort a list using merge sort

- If the list contains less than two links then return the list unchanged
- Otherwise split the list destructively into two halves that differ by at most one link in lengths
- Then sort the two smaller lists destructively yielding lists in ascending order
- Then merge the two sorted lists destructively yielding a list containing all the original links, but now in ascending order

Merging two sorted lists destructively in a loop





x is a chain of r links with the values $x_0, x_1, \ldots, x_{r-1}$ and y is a chain of s links with the values $y_0, y_1, \ldots, y_{s-1}$

Both lists are in ascending order and we wish to merge then destructively into an ascending chain of the same links

Loop invariant

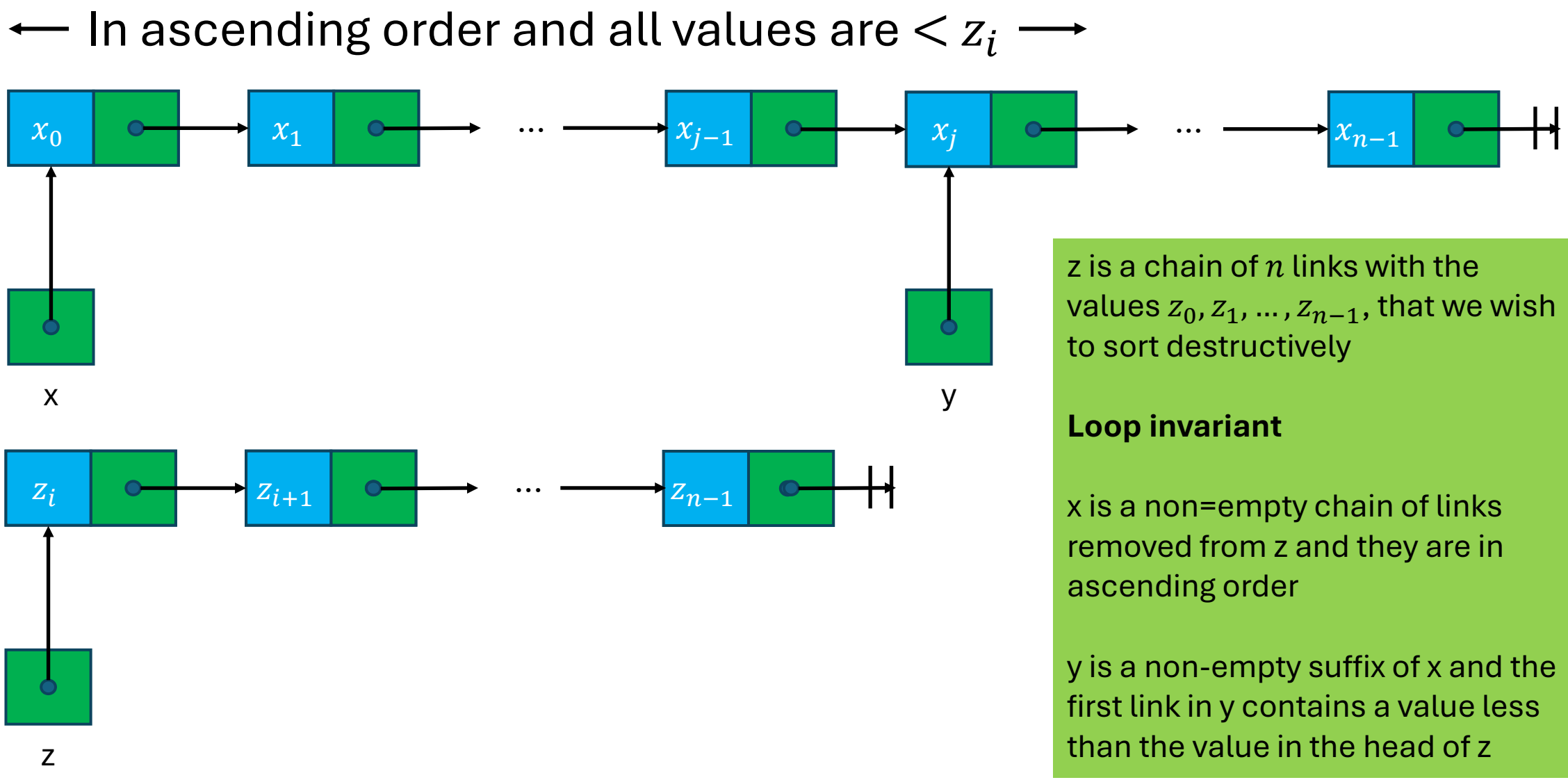
z is a non-empty chain containing links removed from the fronts of x and y, and w refers to the last link in that chain

All the values in z are $\leq$ all the values remaining in x and y and are in ascending order

Merging two sorted lists destructively using recursion

- If one of the two lists is empty, then return the other list
- Otherwise remove the smaller of the two frontmost links
  - That link will contain the smallest value in the two lists
- Merge the two remaining lists recursively and destructively
- Prepend the removed link to the result of the merge and return the resulting list
- The result of the merge contains all the links in the original two lists, but chained together differently
- The big disadvantage of using recursion is that the depth of recursion becomes excessive for large lists, potentially making the run-time stack blow up

Insertion sort inner loop



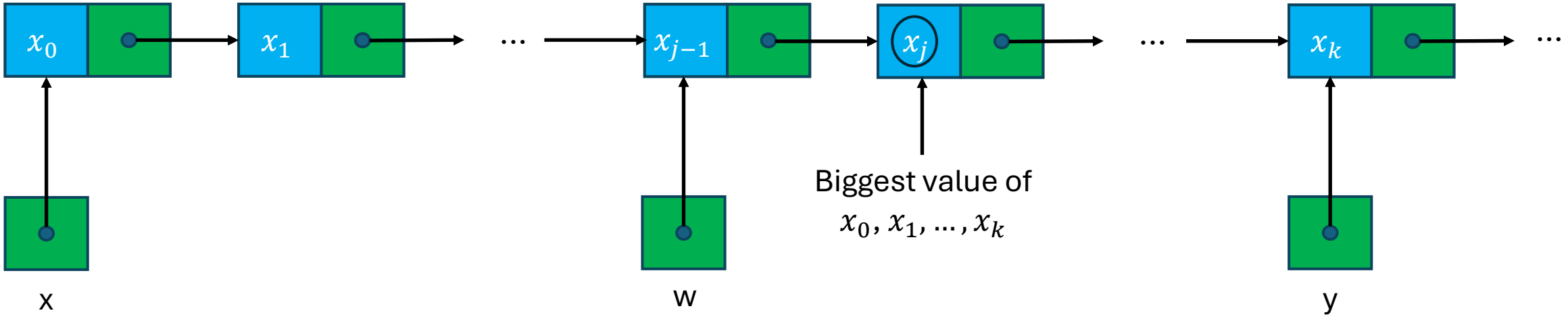
z is a chain of n links with the values $z_0, z_1, \ldots, z_{n-1}$, that we wish to sort destructively

Loop invariant

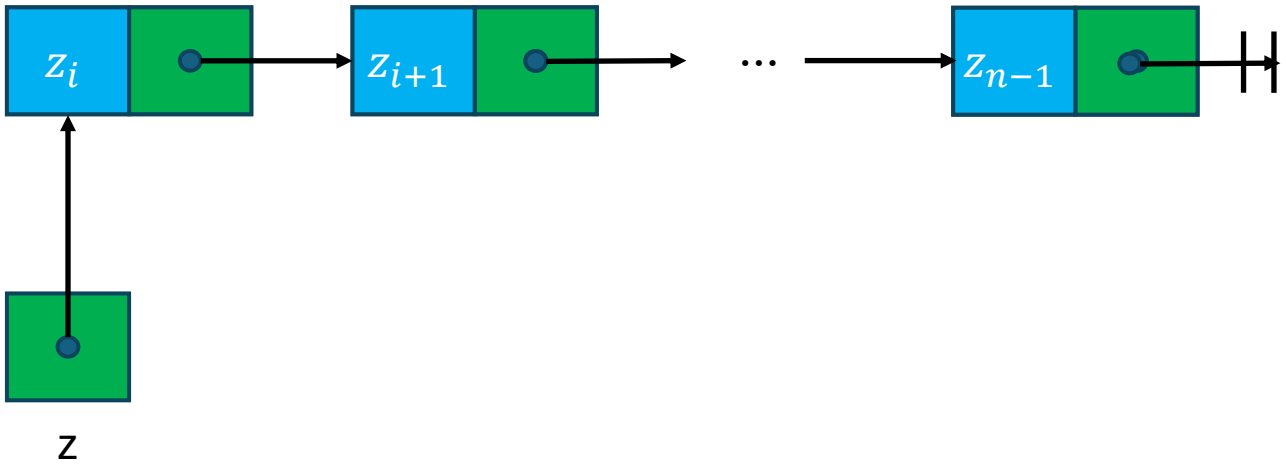
x is a non=empty chain of links removed from z and they are in ascending order

y is a non-empty suffix of x and the first link in y contains a value less than the value in the head of z

Selection sort inner loop



← Biggest in ascending order →



x is a chain of n links with the values $x_0, x_1, \ldots, x_{n-1}$, that we wish to sort destructively

Loop invariant

z is a chain of links removed from x containing the biggest values and they are in ascending order

y is a proper suffix of x and w points to the link in x in front of y that is immediately in front of the link that contains the biggest value among the links in front of y and including y, or w contains null if the first link in x has the biggest value