Problem Solving for Computer Science

IS51021C

Goldsmiths Computing

February 8, 2020

Wk 5

Problem 4:

You have been asked to organise a lottery for the national Chess Boxing and Knitting club, which has 589 members

It is decided that the lottery will be based on having a unique birthday

- You win if no one else has your birthday
- No one wins if they share birthdays

You are given a list of all members' birthdays along with their membership number as a table

Write an algorithm and/or JavaScript implementation to determine who will win the lottery

Problem 4:

B: 01/03 0123	B: 02/08 0005	B: 01/03 1001
B: 20/08 0803	B: 01/03 1111	B: 21/10 1173
B: 12/12 0667	B: 08/05 1114	B: 13/02 0051

Problem 4:

B: 20/08
0803

B: 01/03
0123

B: 02/08
0005

B: 01/03
1001

B: 13/02
0051

Number	Birthday
0803	2008
0123	103
0005	208
1001	103
0051	1302

Adapt the linear search algorithm

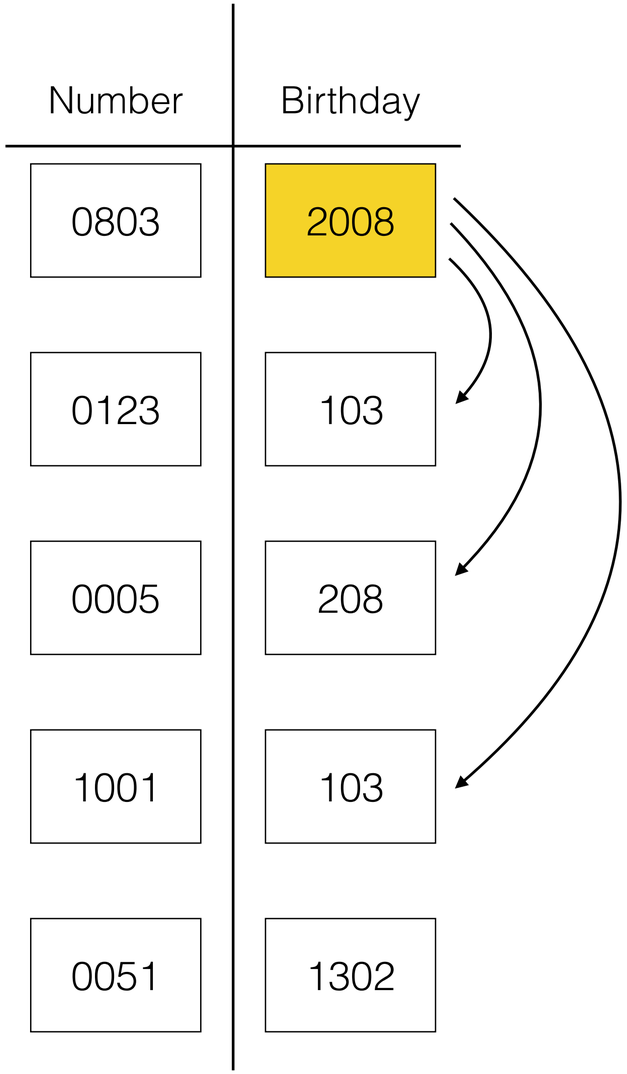
...

Problem 4:

Number	Birthday
0803	2008
0123	103
0005	208
1001	103
0051	1302

For each element, perform linear search on rest of birthdays for that value

Problem 4:

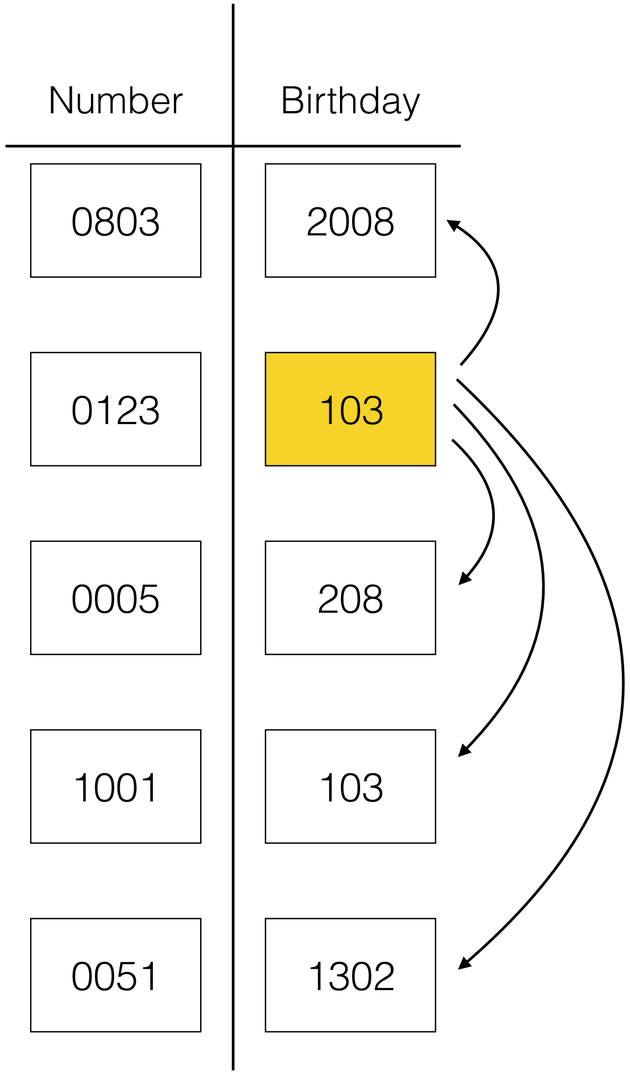


For each element, perform linear search on rest of birthdays for that value

If linear search finds value, then birthday is not unique

If value is not found, birthday is unique

Problem 4:



For each element, perform linear search on rest of birthdays for that value

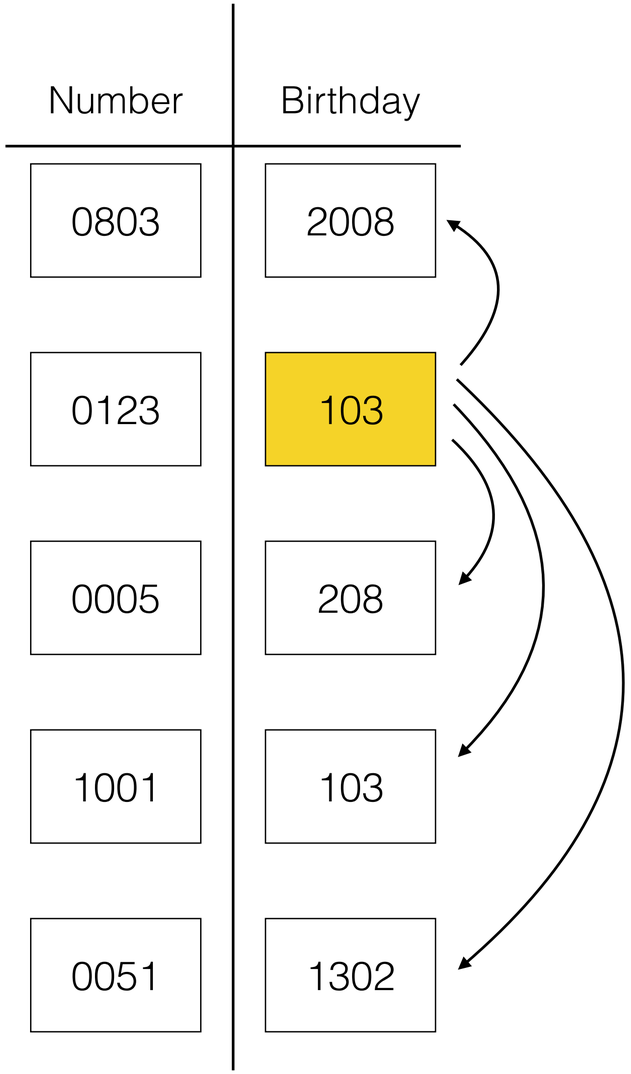
If linear search finds value, then birthday is not unique

If value is not found, birthday is unique

Repeat this for every element

Make sure we don't look in selected element!

Problem 4:



For each element, perform linear search on rest of birthdays for that value

If linear search finds value, then birthday is not unique

If value is not found, birthday is unique

Repeat this for every element

Any other ideas?

Problem 4:

Number	Birthday
0051	1302
0123	103
1001	103
0005	208
0803	2008

First sort the table according to birthday

People with same birthday will be next to each other in the table

Problem 4:

Number	Birthday
0051	1302
0123	103
1001	103
0005	208
0803	2008

First sort the table according to birthday

People with same birthday will be next to each other in the table

Only need to compare at most either side

Problem 4:

Number	Birthday
0051	1302
0123	103
1001	103
0005	208
0803	2008

First sort the table according to birthday

People with same birthday will be next to each other in the table

Only need to compare at most either side

Problem 4:

Number	Birthday
0051	1302
0123	103
1001	103
0005	208
0803	2008

First sort the table according to birthday

People with same birthday will be next to each other in the table

Only need to compare at most either side

Moral

Having sorted data can make searching easier

Today

Sorting

Two new algorithms: Bubble Sort and Insertion Sort

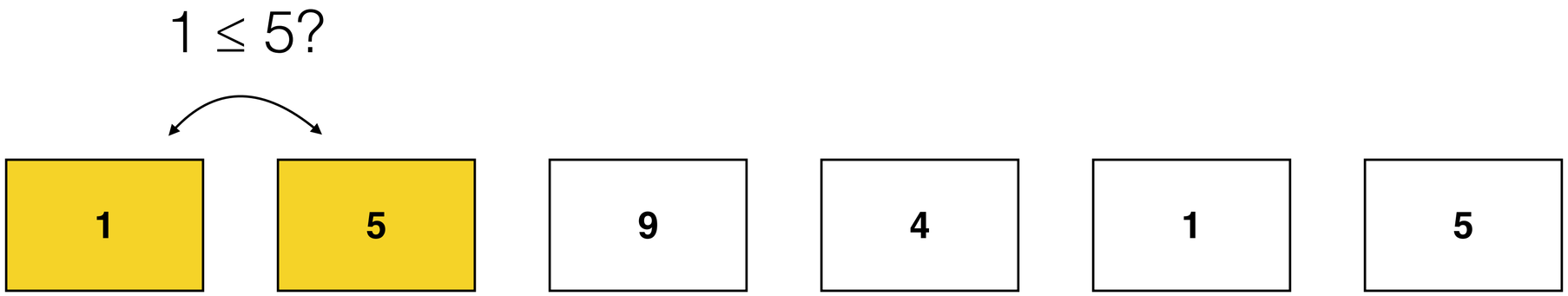
How do you check if something is sorted?

How do you check if something is sorted?

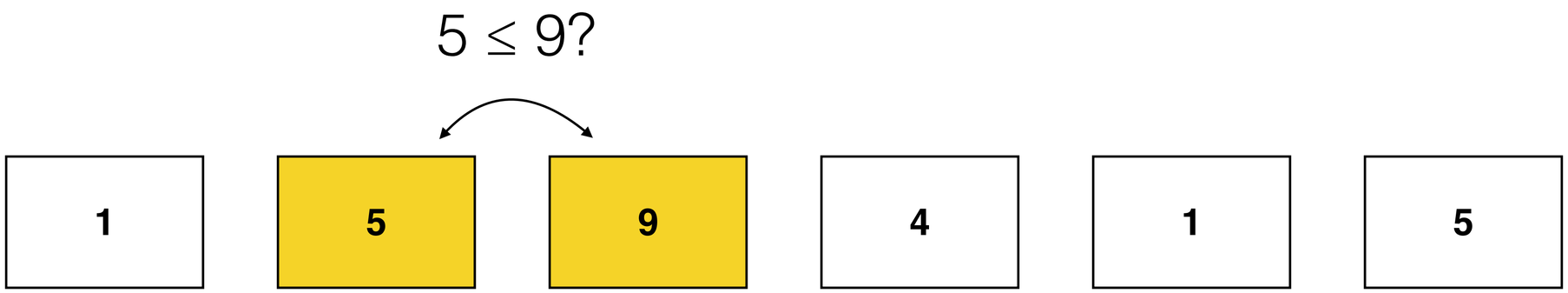
Compare pairwise

Assume ascending order: smallest value on the left

Vector of numbers

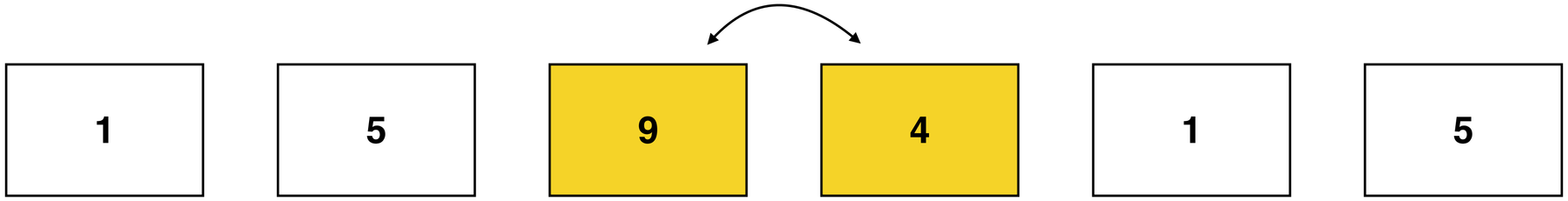


Vector of numbers

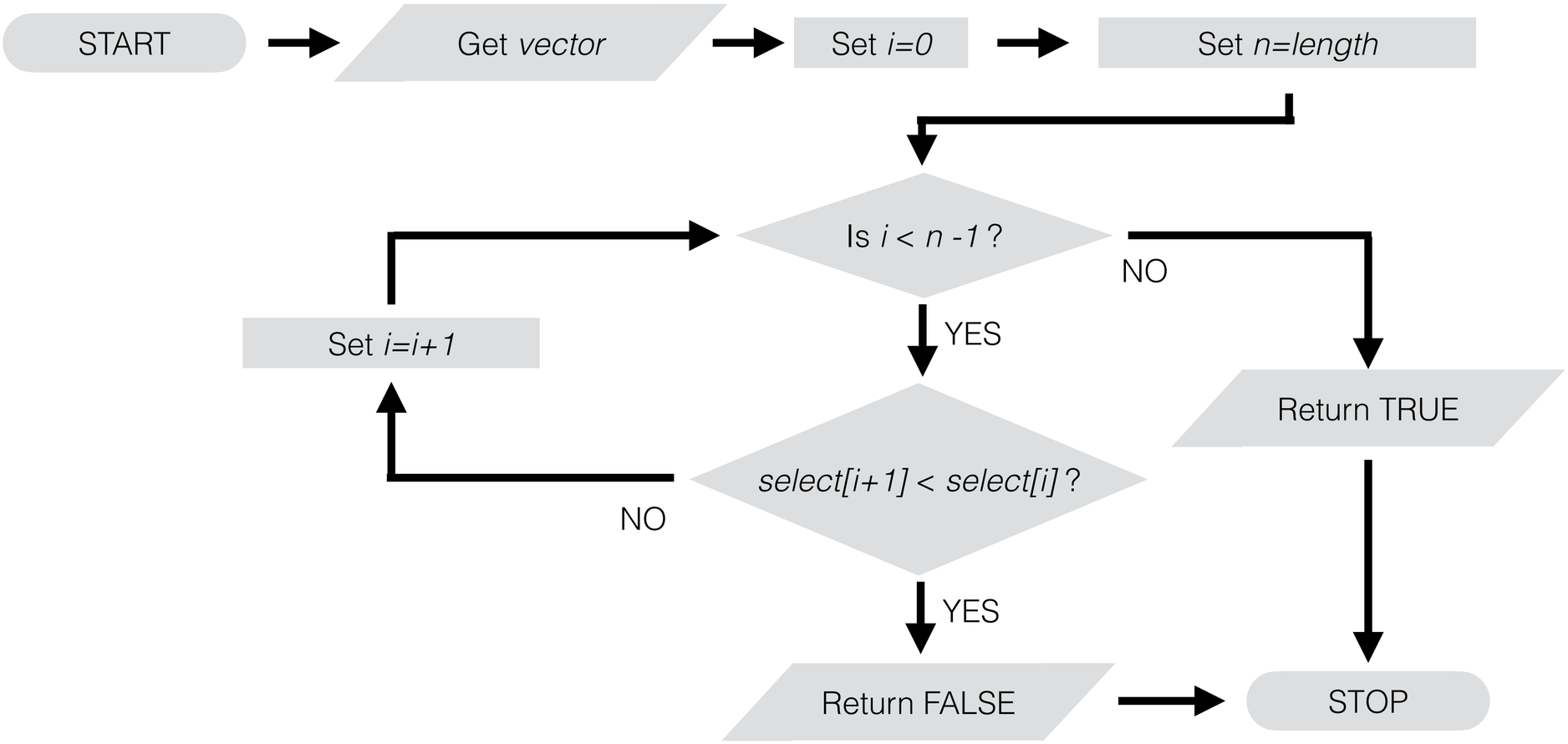


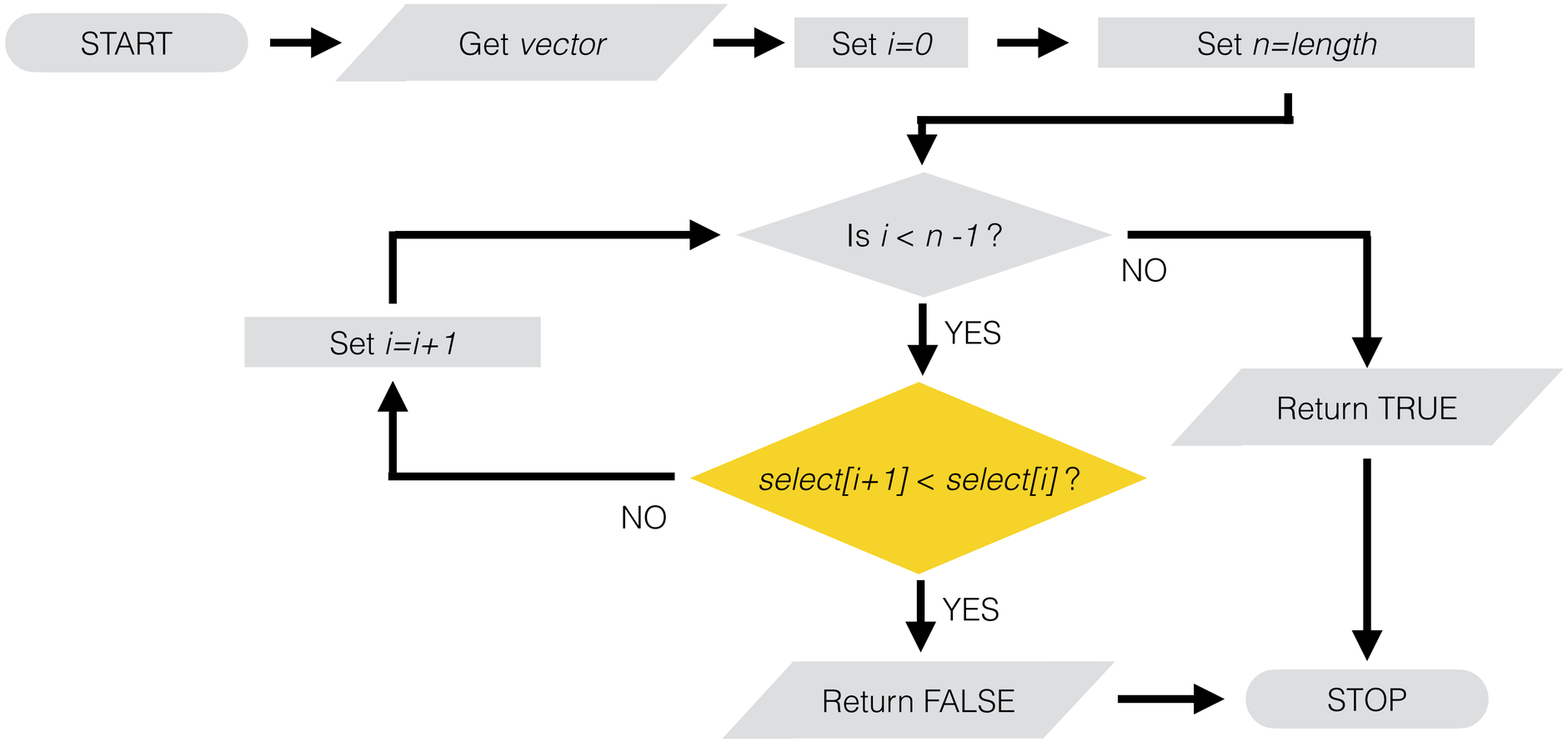
Vector of numbers

$9 \leq 4?$

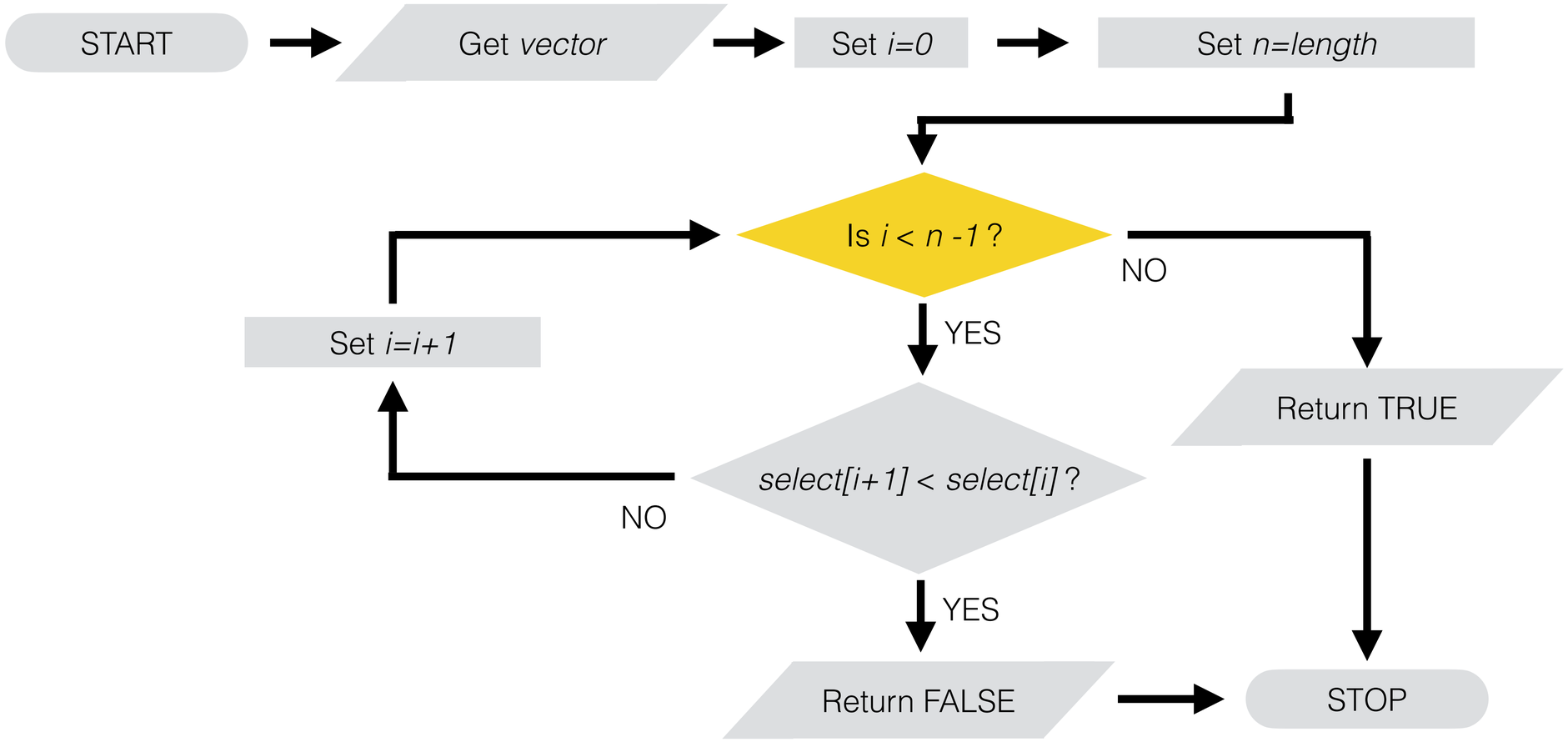


No!
The vector is not sorted


START
Get vector
Set i=0
Set n=length
Is i < n -1 ?
NO
YES
Set i=i+1
Return TRUE
select[i+1] < select[i] ?
NO
YES
Return FALSE
STOP



Compares neighbouring elements



Why this decision?

We can implement this algorithm for JavaScript arrays

```
function isSorted(array) {
    var n = array.length;
    for (var i = 0; i < n - 1; i++) {
        if (array[i + 1] < array[i]) {
            return false;
        }
    }
    return true;
}
```

How do you sort stuff?

How do you sort stuff?

Swap values if they are in the wrong order?

Vector of numbers

Smallest on the left; largest on the right

9	**5**	**1**	**4**	**1**	**5**

You want to start swapping elements? Don’t you?

What are we allowed to do?

Vector of numbers

Smallest on the left; largest on the right

9	**5**	**1**	**4**	**1**	**5**

You want to start swapping elements? Don’t you?

What are we allowed to do?

select[k] store![o,k] length

Vector of numbers

Smallest on the left; largest on the right

9	**5**	**1**	**4**	**1**	**5**

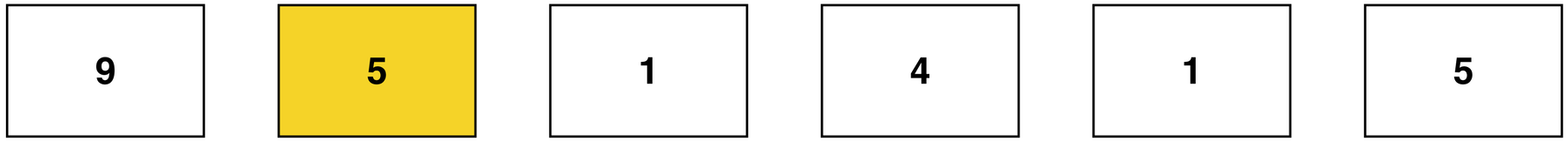
You want to start swapping elements? Don't you?

What are we allowed to do?

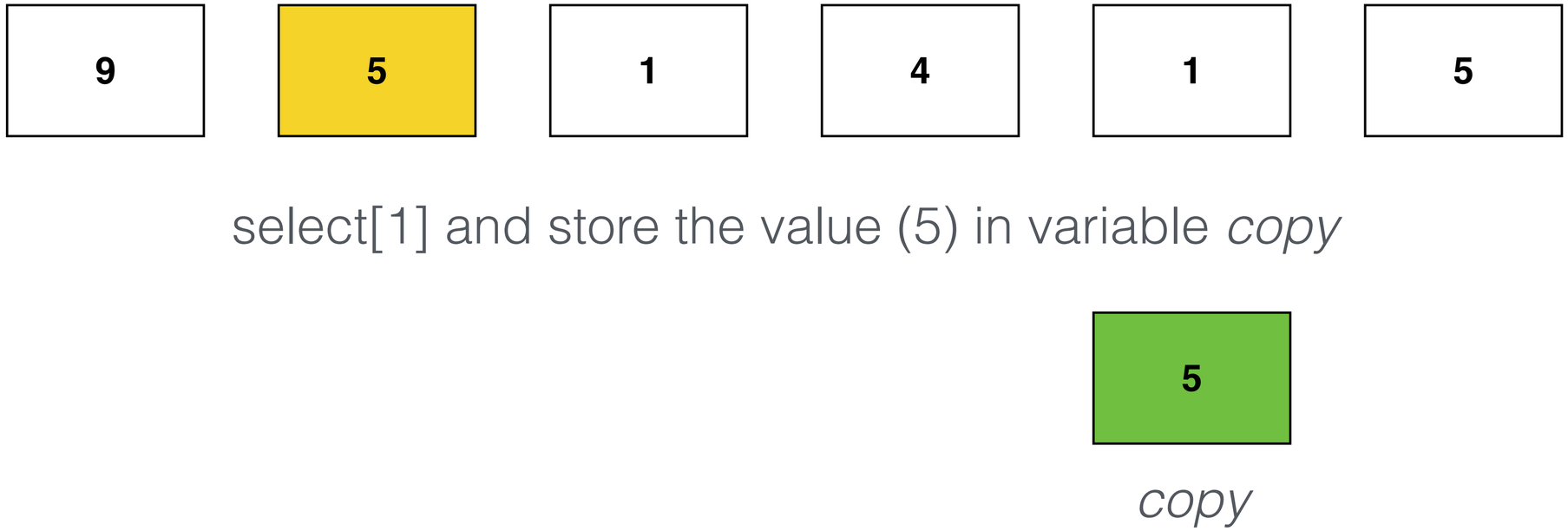
select[k] store![o,k] length

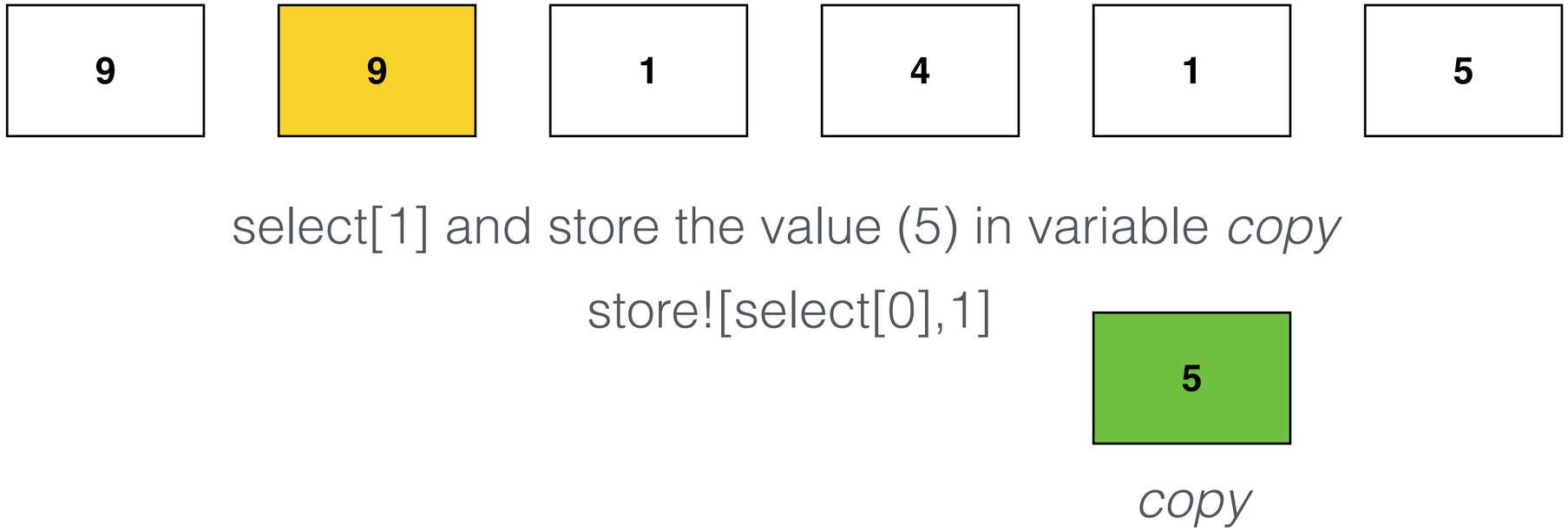
How can we *simulate* a swap?

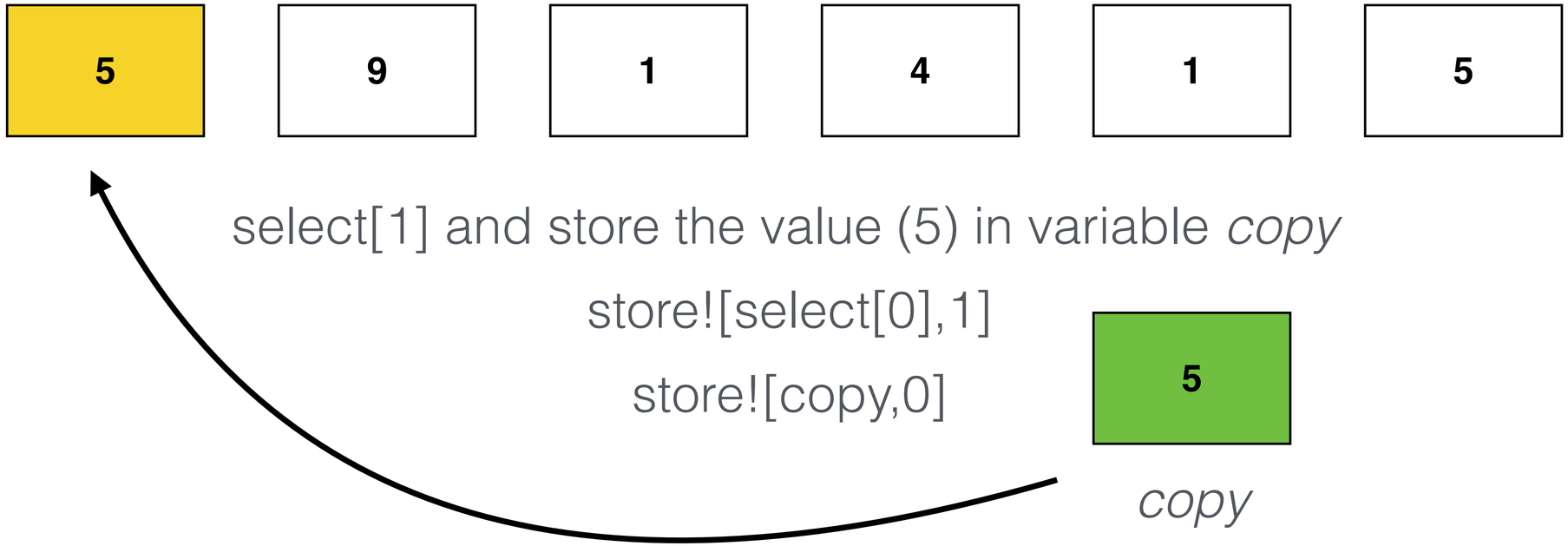
Swap first two values

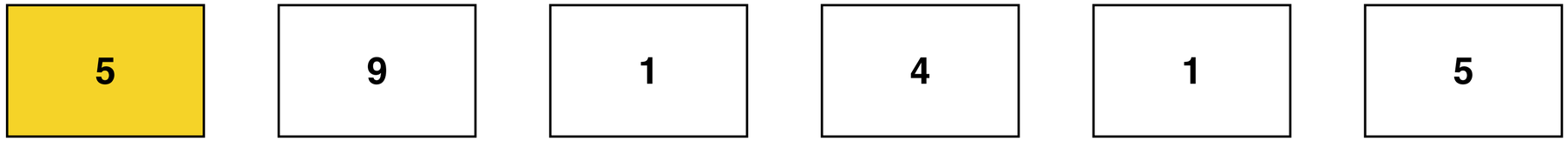


select[1] and store the value (5) in variable *copy*


9
5
1
4
1
5
select[1] and store the value (5) in variable *copy*
5
copy


9
9
1
4
1
5
select[1] and store the value (5) in variable *copy*
store![select[0],1]
5
copy

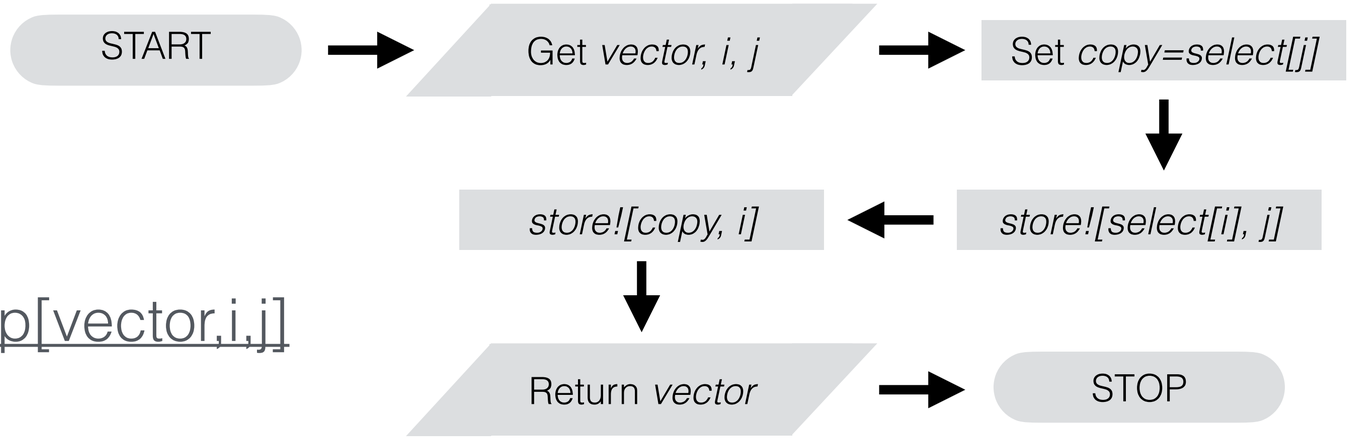

5
9
1
4
1
5
select[1] and store the value (5) in variable *copy*
store![select[0],1]
store![copy,0]
5
copy



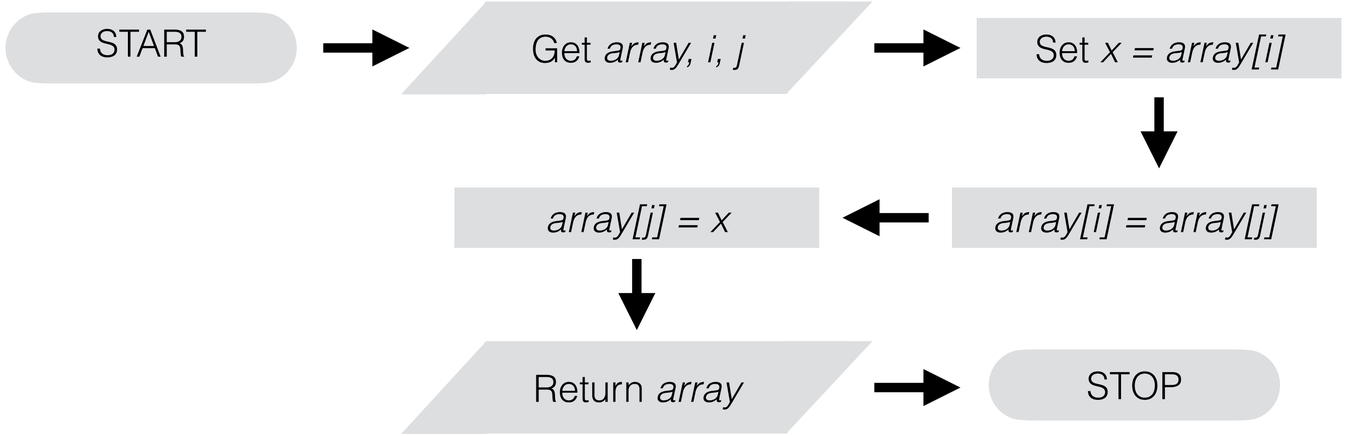
select[1] and store the value (5) in variable *copy*

store![select[0],1]

store![copy,0]



swap[vector,i,j]

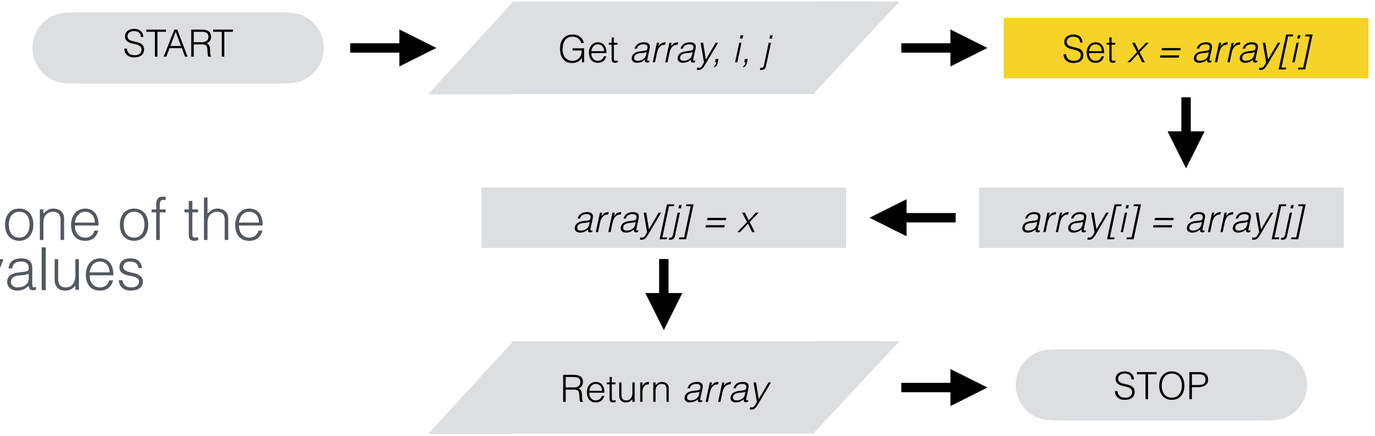


Worksheet 1

Swap function applied to arrays

```
function swap(array, i, j) {
    var x = array[i];
    array[i] = array[j];
    array[j] = x;
    return array;
}
```

Save one of the values



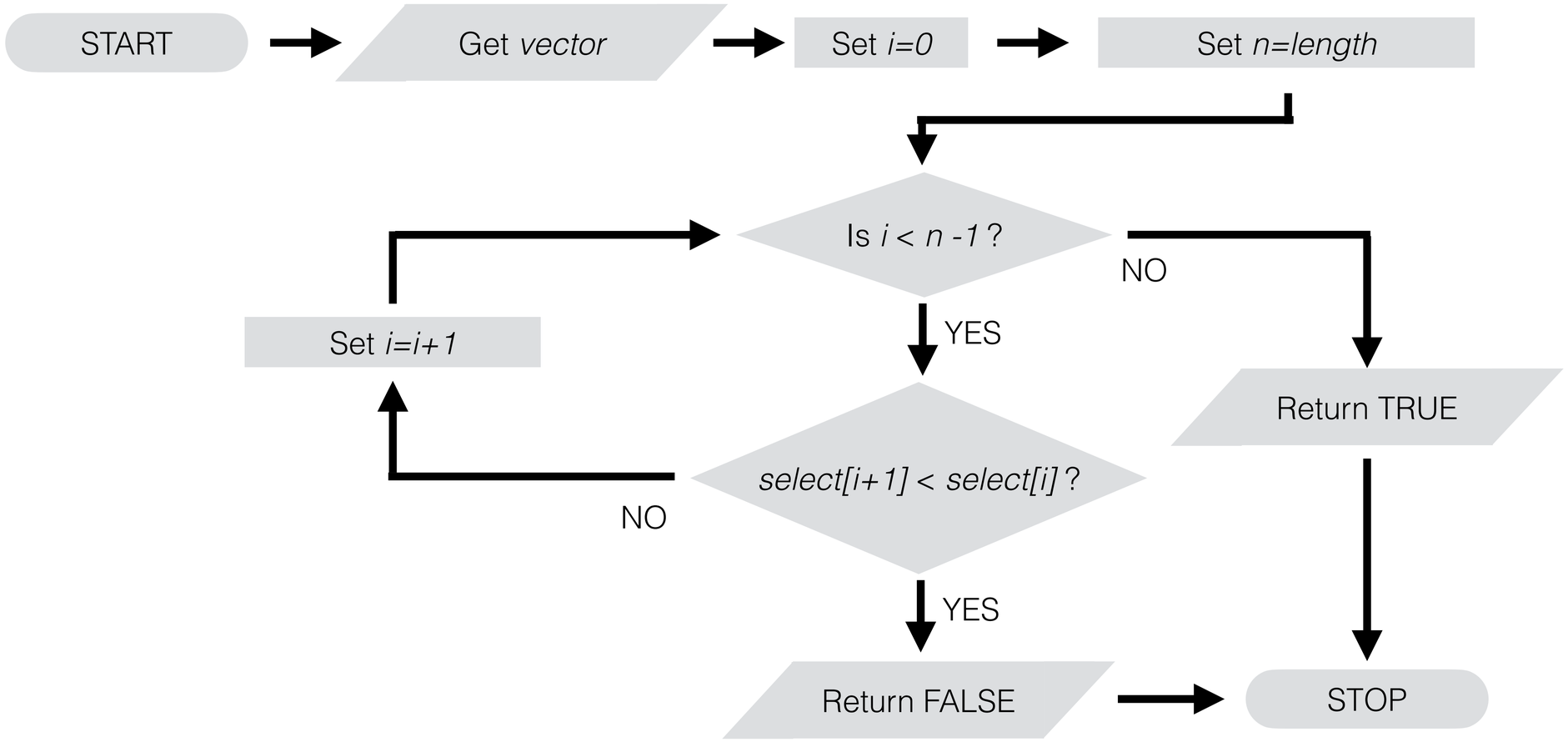
Worksheet 1

Swap function applied to arrays

```
function swap(array, i, j) {
    var x = array[i];
    array[i] = array[j];
    array[j] = x;
    return array;
}
```

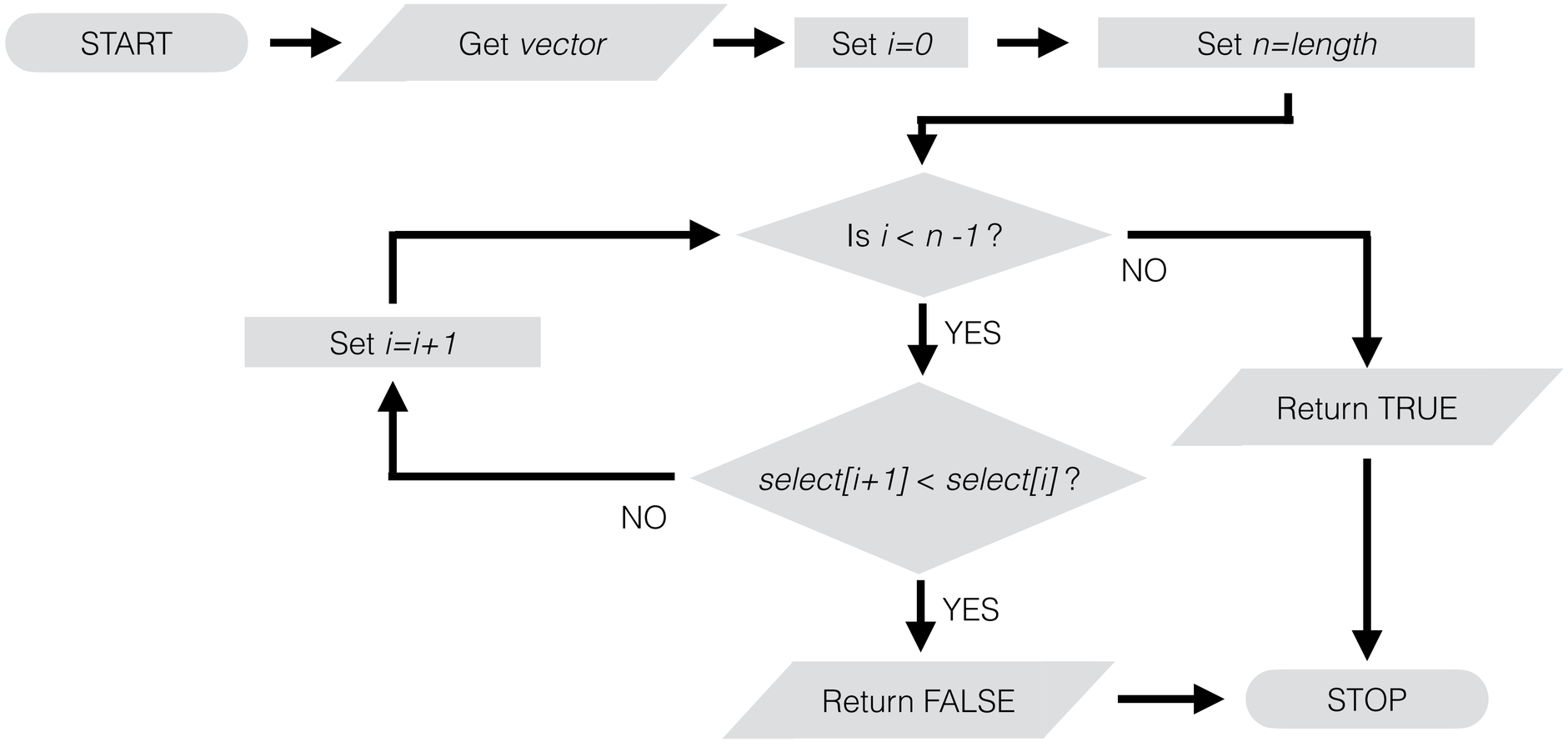
Bubble Sort

Starting point

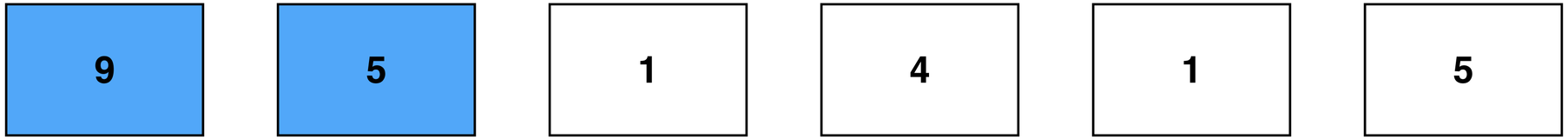


To see if sorted, we pairwise compare

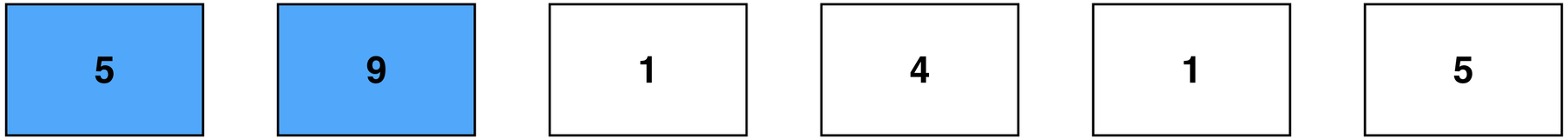
Starting point

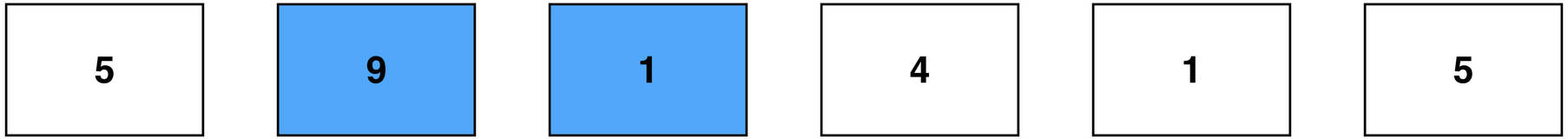


If pairwise elements in **wrong** order: *swap values*
If pairwise elements in **right** order: *do not swap values*

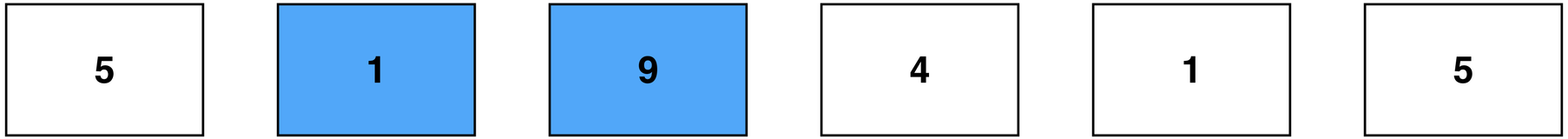


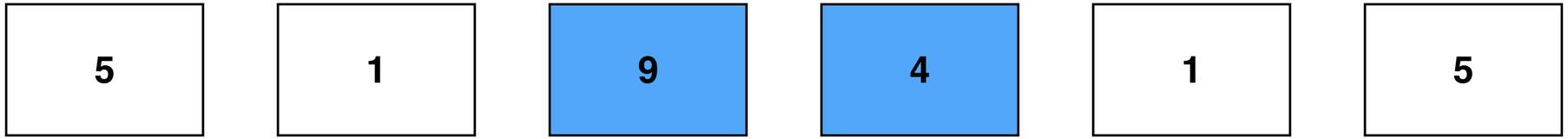
$9 < 5?$


5
9
1
4
1
5

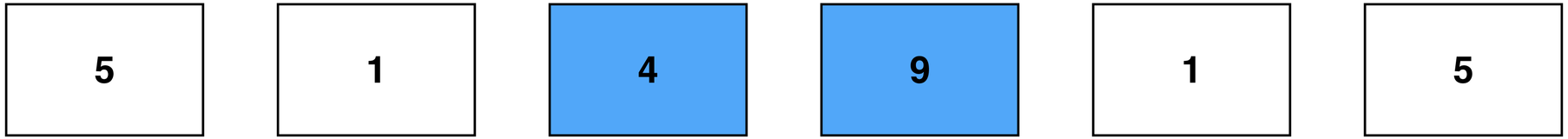


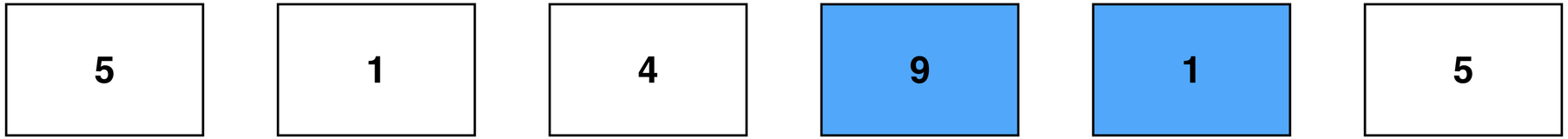
$9 < 1?$


5
1
9
4
1
5

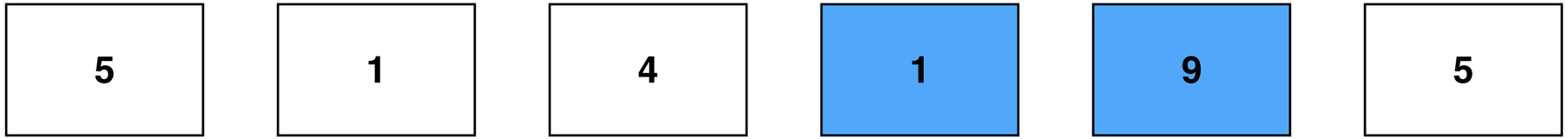


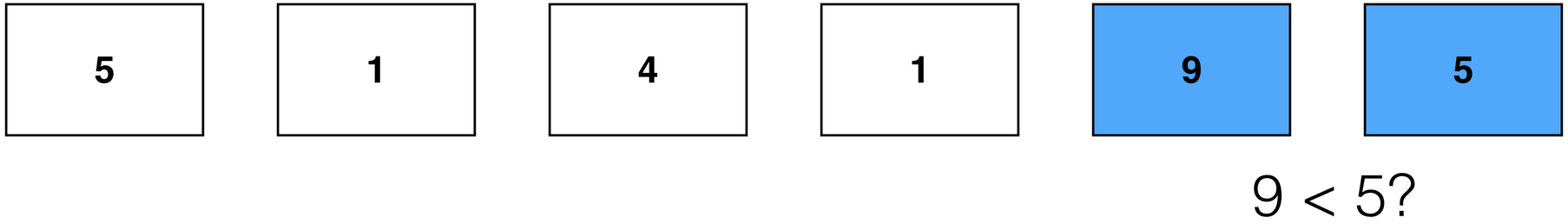
$9 < 4?$

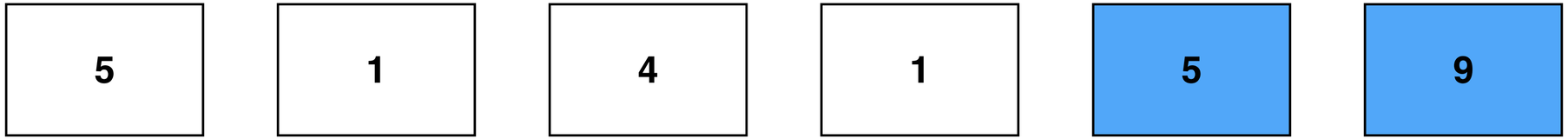

5
1
4
9
1
5

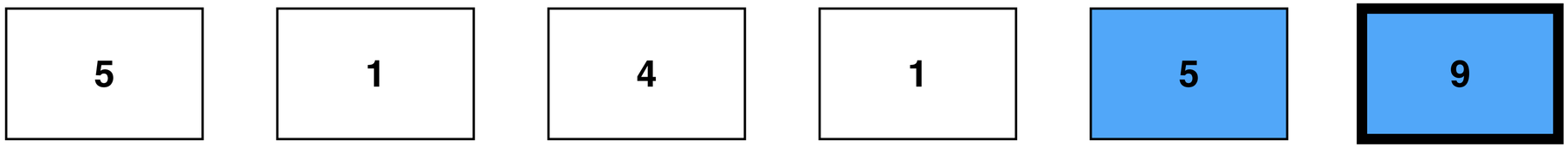


$9 < 1?$


5
1
4
1
9
5

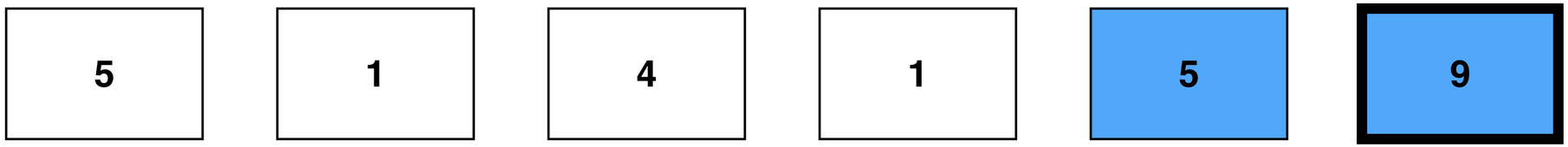

5
1
4
1
9
5
9 < 5?


5
1
4
1
5
9



Is it sorted?

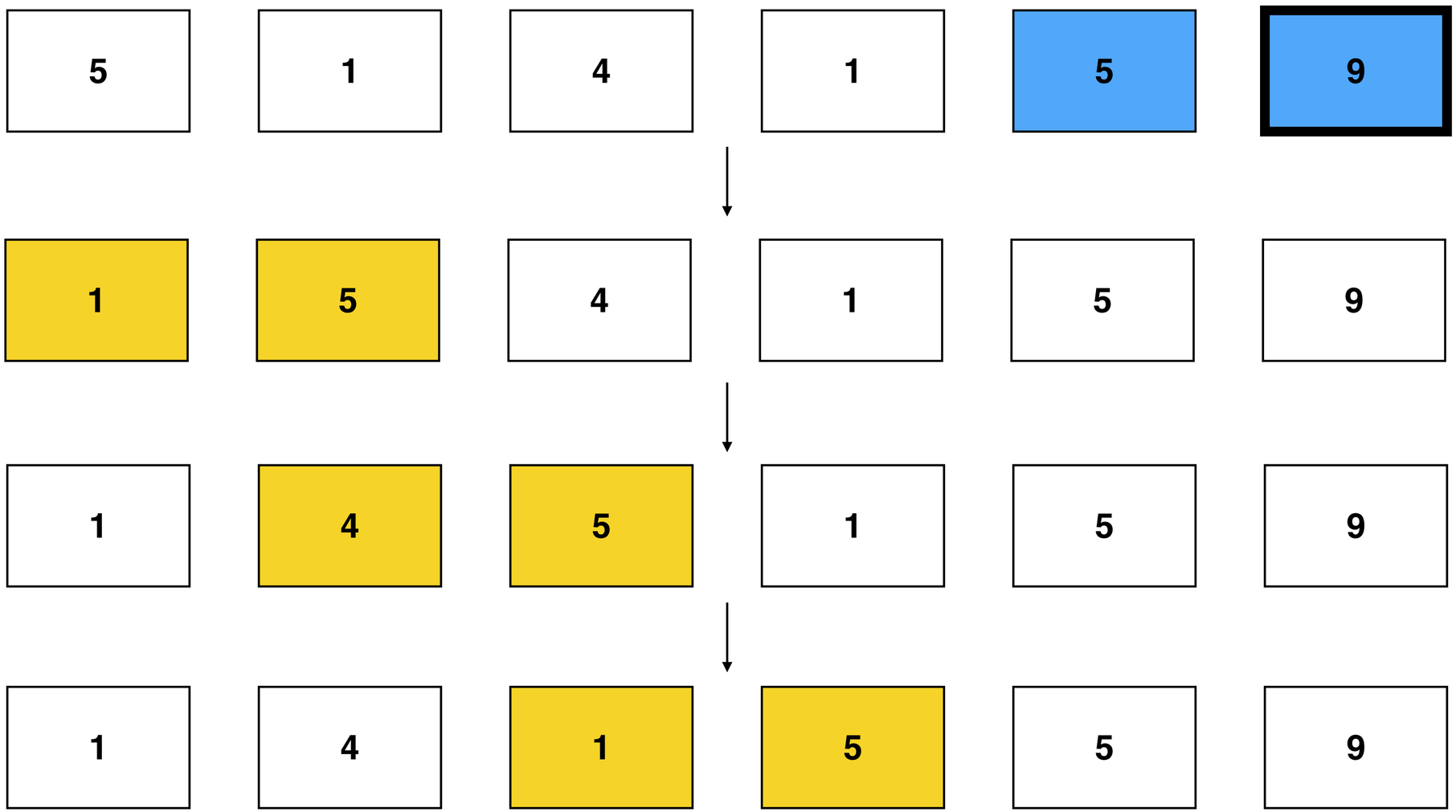
Largest number gets pushed to the end



This is the first ***pass*** of the algorithm

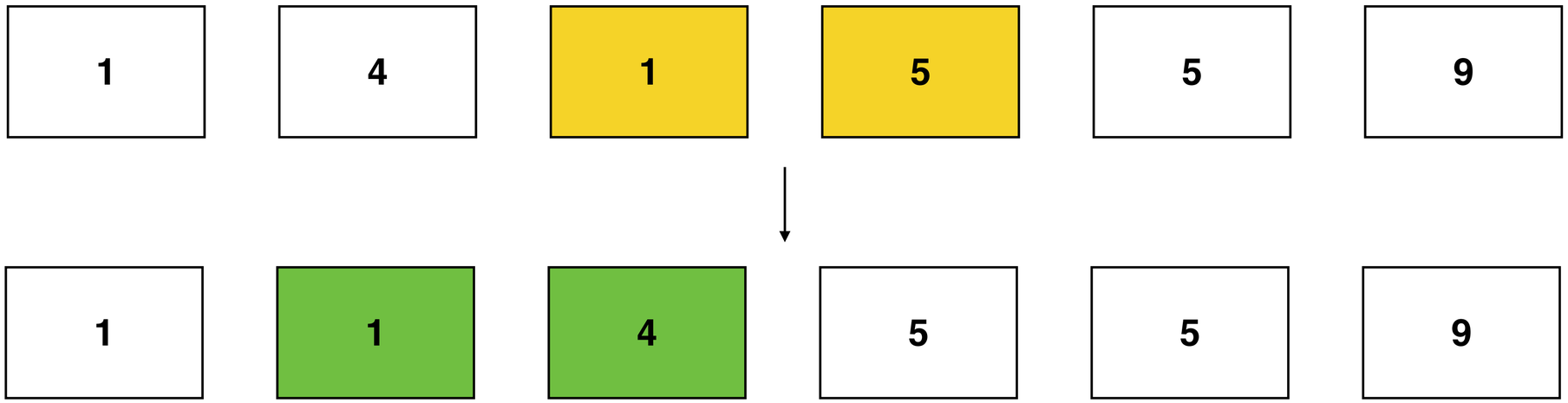
A pass is one whole sequence of comparing all pairwise neighbours

We need to repeat the previous swaps



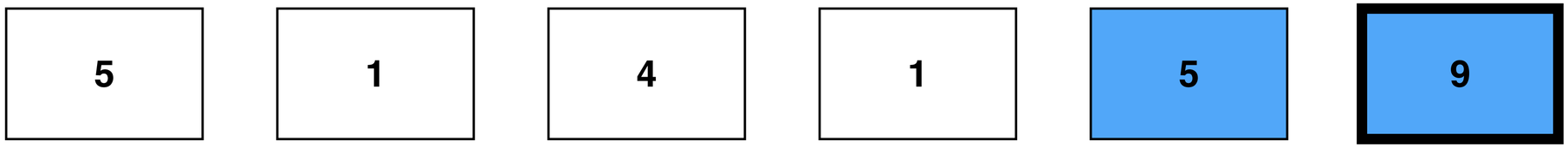
This is the second *pass* of the algorithm

Still not sorted!



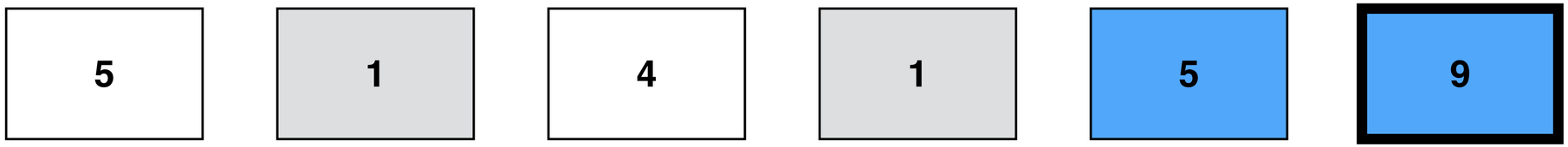
Will be sorted after a *third pass*

For a general vector, how many passes are needed?



Next largest number always gets pushed towards end after each pass

What happens to the smallest number(s)?



Next largest number always gets pushed towards end after each pass

What happens to the smallest number(s)?

What happens to the smallest number(s)?

9	5	1	4	1	5

After first *pass:*

5	1	4	1	5	9

Second *pass:*

1	4	1	5	5	9

Third *pass:*

1	1	4	5	5	9

9	5	1	4	1	5

After first *pass:*

5	1	4	1	5	9

Second *pass:*

1	4	1	5	5	9

Third *pass:*

1	1	4	5	5	9

Crawls one element at a time to the left

What happens to the smallest number(s)?

Crawls one element at a time to the left

5	3	4	9	5	1
3	4	5	5	1	9
3	4	5	1	5	9
3	4	1	5	5	9
3	1	4	5	5	9
1	3	4	5	5	9

What happens to the smallest number(s)?

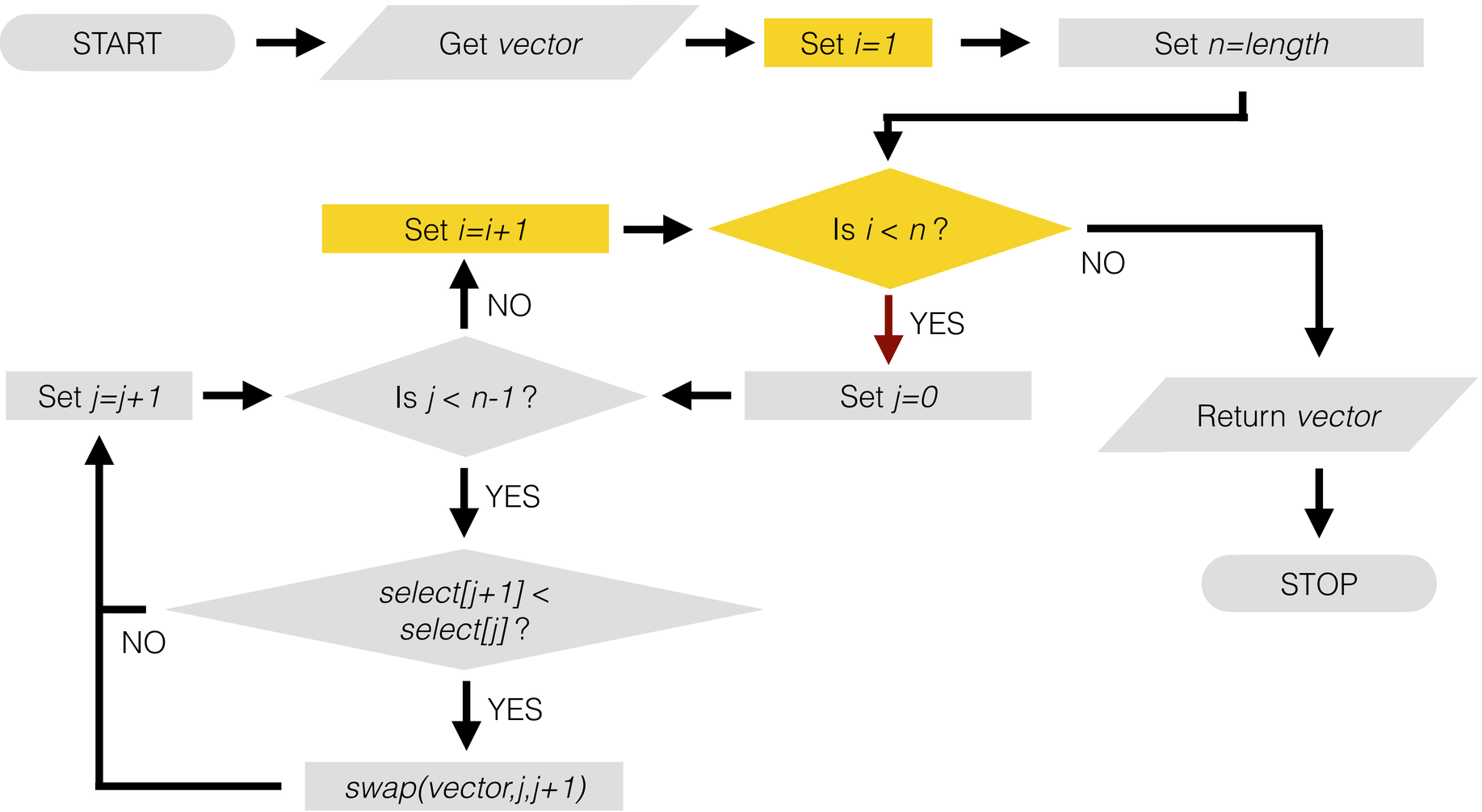
Crawls one element at a time to the left

5	3	4	9	5	1
3	4	5	5	1	9
3	4			5	9
3	4			5	9
3	1	4	5	5	9
1	3	4	5	5	9

For vector of length *n*

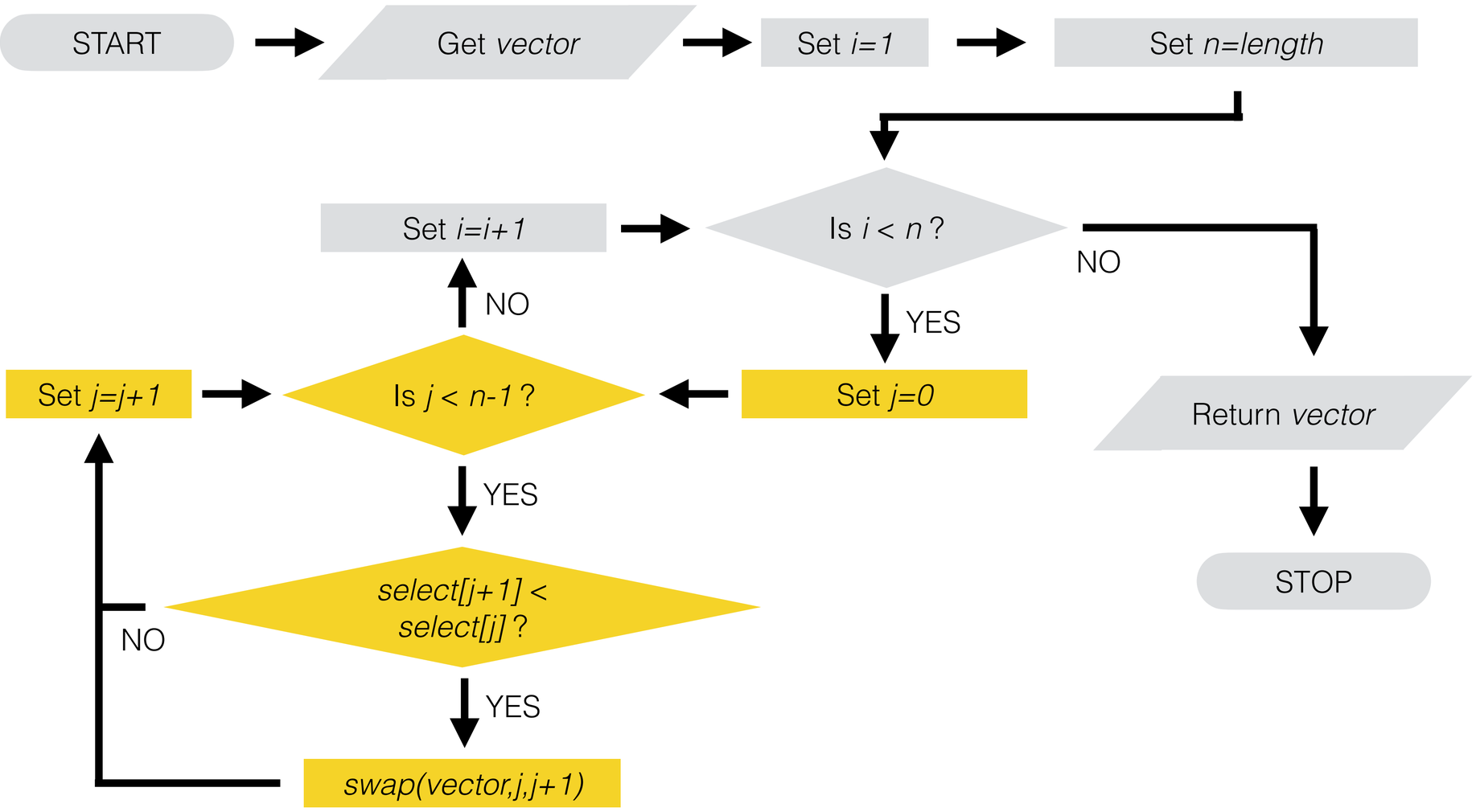
Need *n-1* passes

Bubble Sort



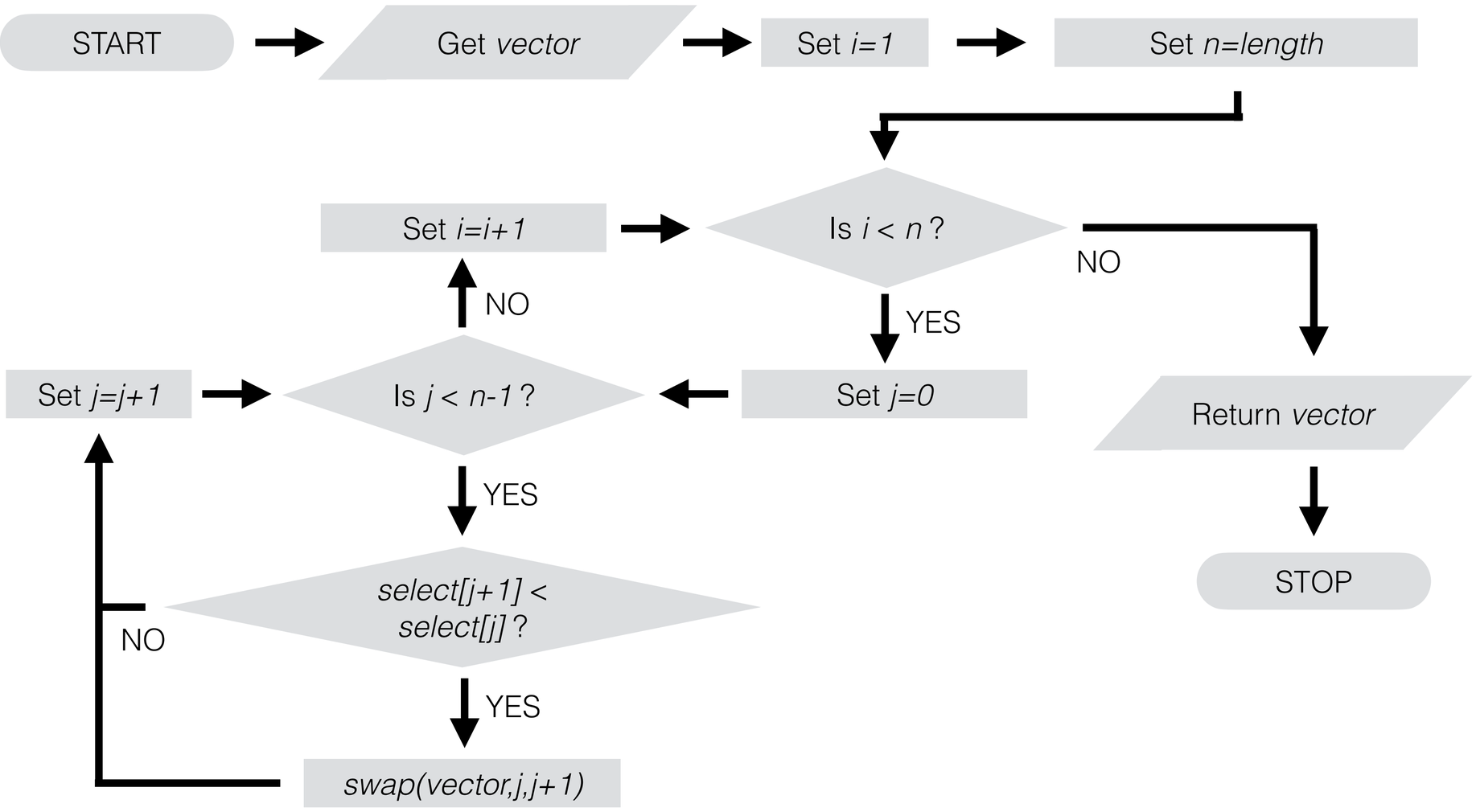
i iterates over passes

Bubble Sort



j iterates over neighbouring elements

Bubble Sort



Also works for Dynamic Arrays
Implement on JavaScript Arrays

Admin

- Fourth quiz available today from 4pm
  - Second quiz **closes today at 4pm**

- Sudoku assignment available as Worksheet 4 (Week 5) **today from 11am**
  - Deadline **1st March at 4pm**
  - 10 tasks: 8 programming, 2 written
  - Submit separately js file only
  - Written work in a *separate* text file, e.g. doc, txt, rtf, pdf
  - Get help in Virtual Contact Hours through Classmates
  - Go through Worksheets 1 to 3 if you have not done so already - the resources are there to help you

Make your code individual - there will be plagiarism checks

Virtual Contact Hours

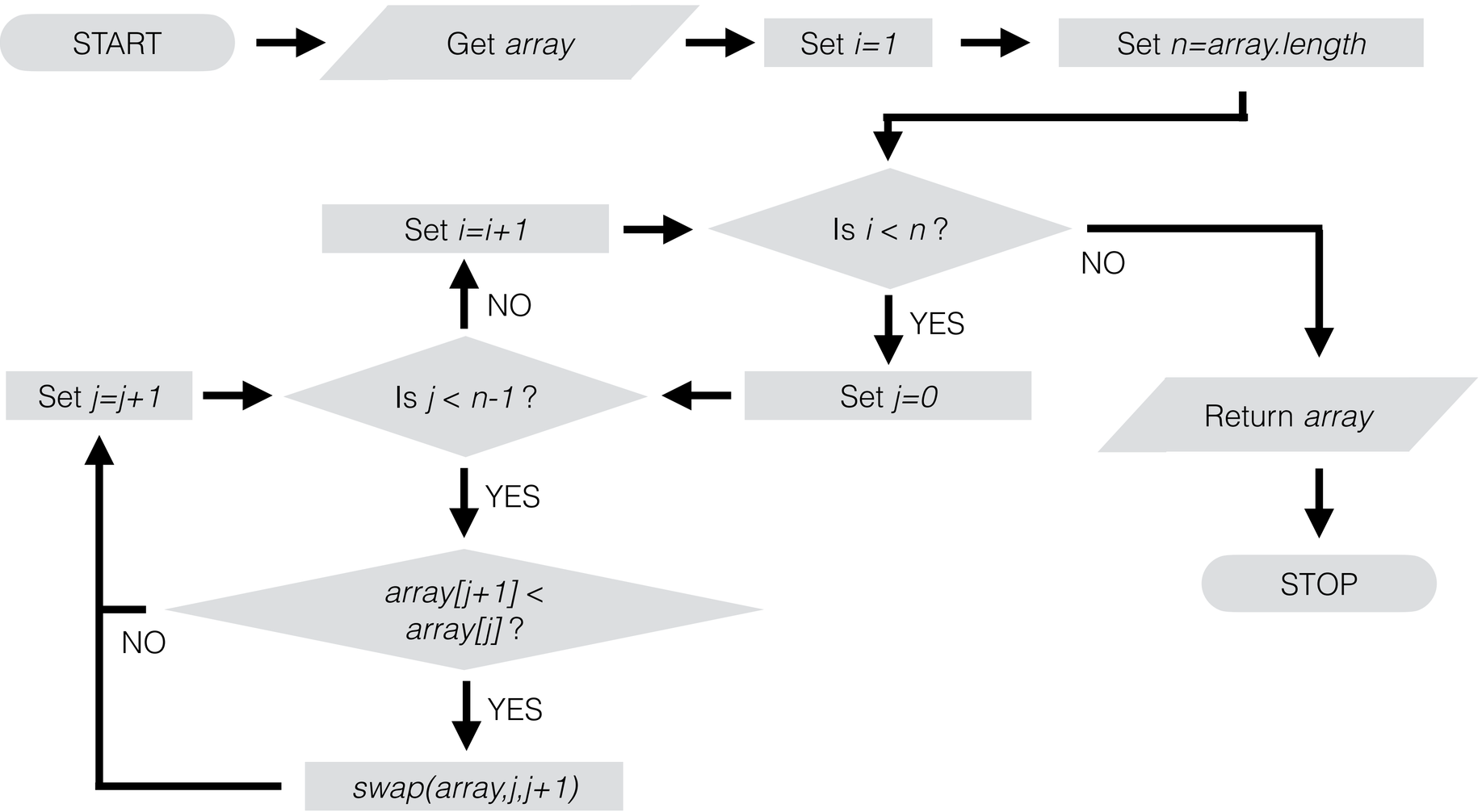
We will have a brief meeting at the beginning of the VCH this week, as normal

After this meeting for the rest of the session it will be Classmates only

There will be no new worksheets until 1st March

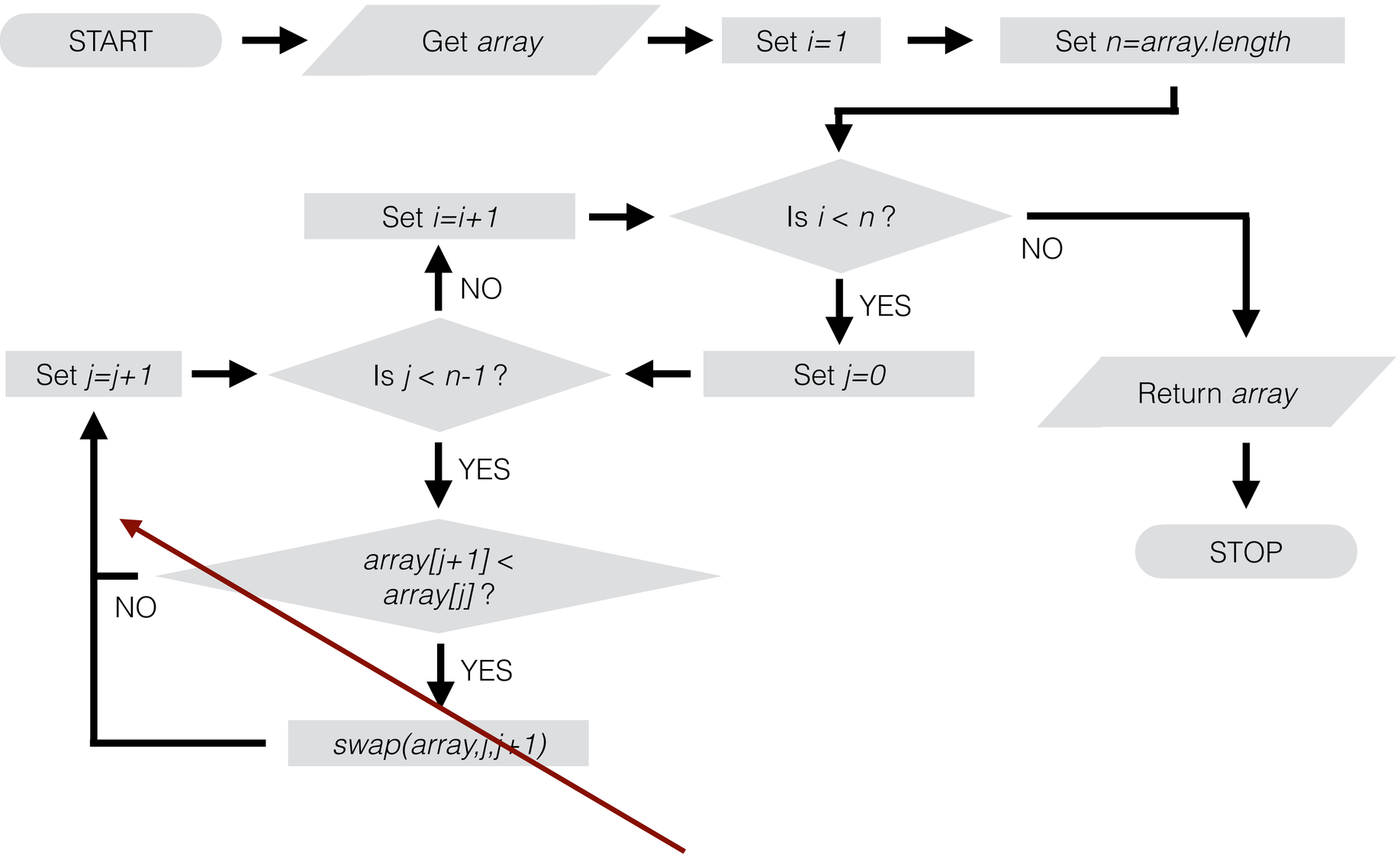
First VCH after Reading Week for help with coursework

Bubble Sort for arrays



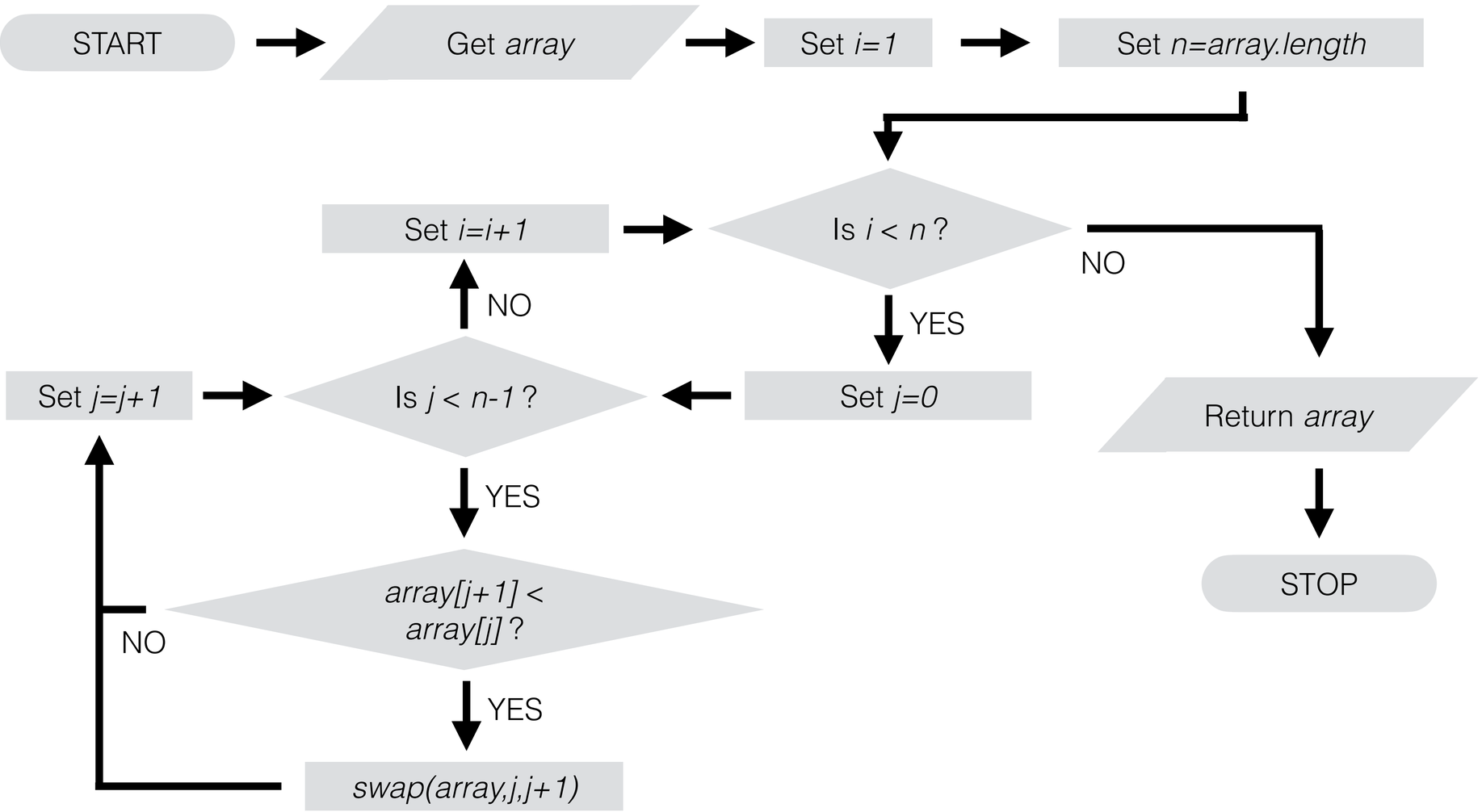
Complete the bubbleSort function for arrays in *sort.js*

Bubble Sort for arrays



Print array to console here

Bubble Sort for arrays

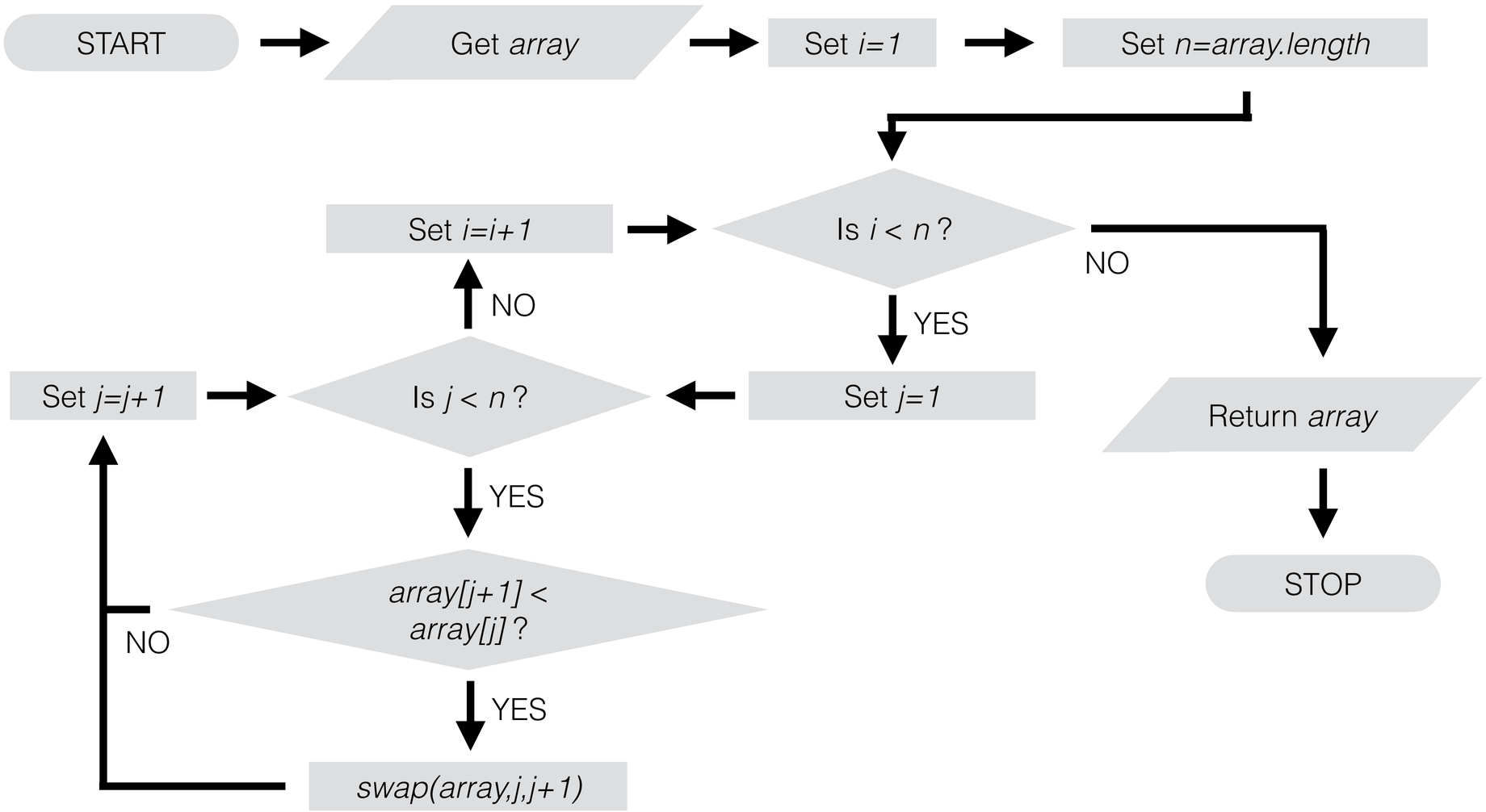


[2,1,3,4,5,6]

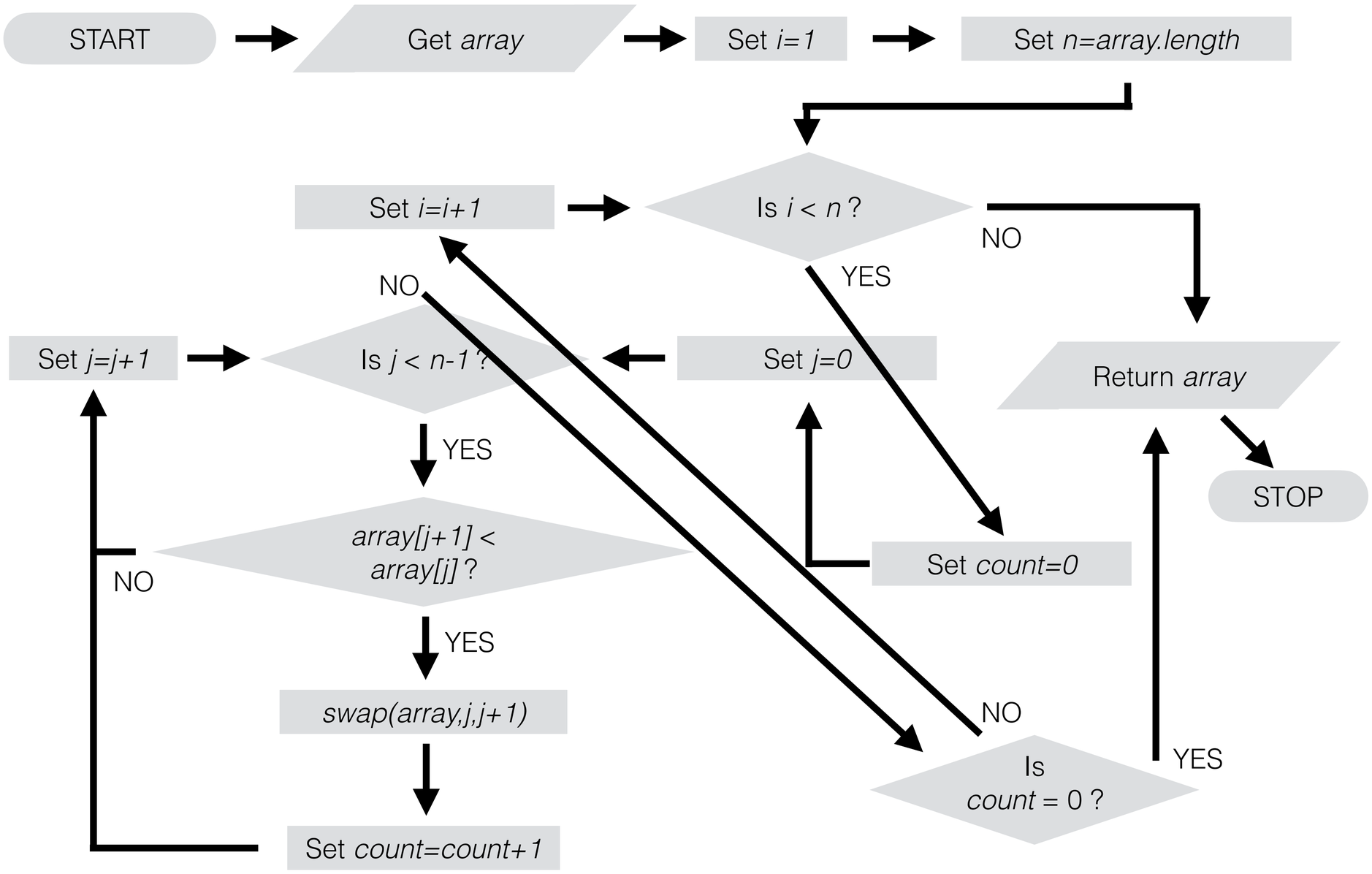
Apply sort to this array

How could you improve this implementation?

Bubble Sort


START
Get array
Set i=1
Set n=array.length
Is i < n ?
NO
YES
Return array
STOP
Set i=i+1
NO
Set j=1
Set j=j+1
Is j < n ?
YES
array[j+1] < array[j] ?
NO
YES
swap(array,j,j+1)

Bubble Sort

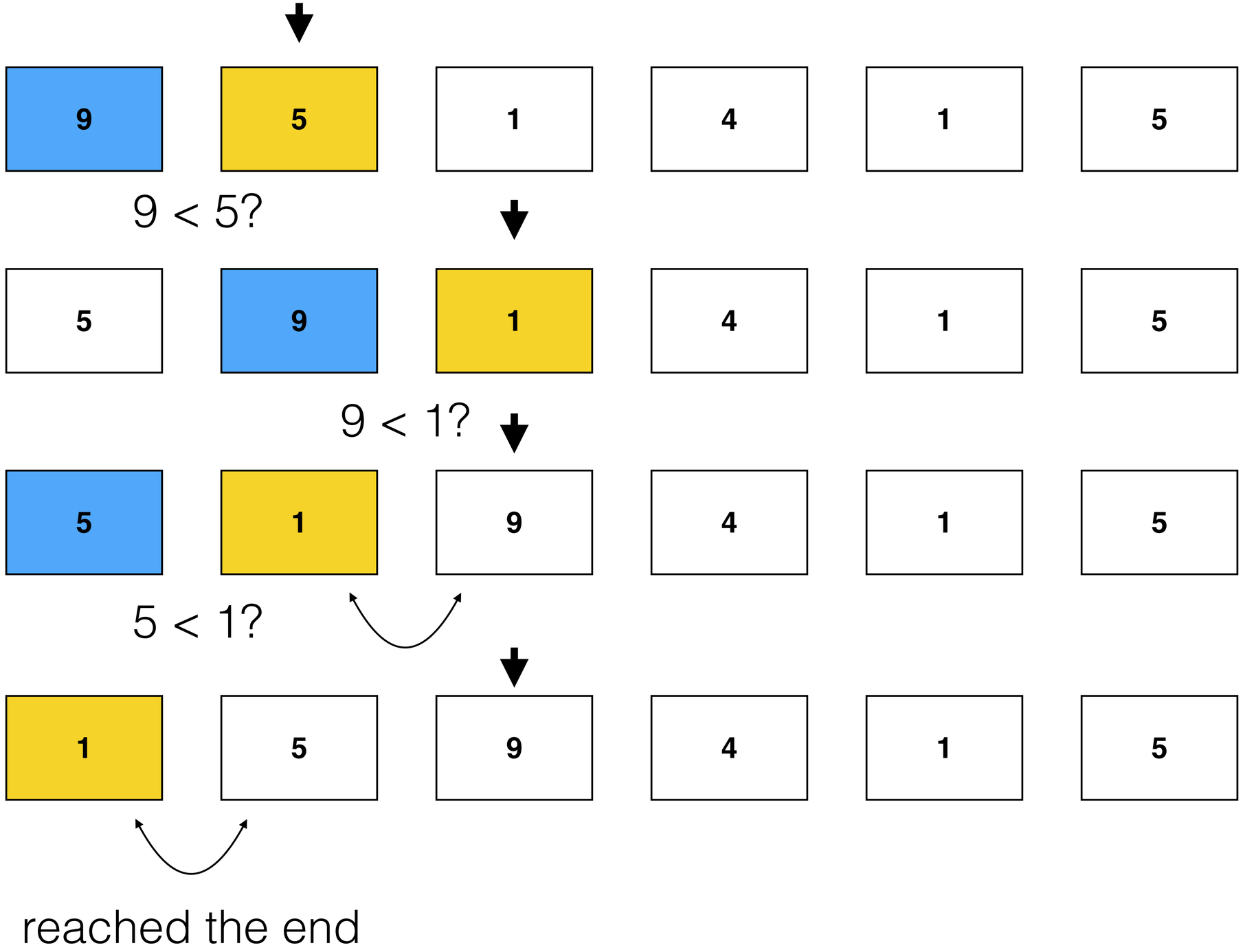

START
Get array
Set i=1
Set n=array.length
Is i < n ?
NO
YES
Set i=i+1
NO
Set j=j+1
Is j < n-1 ?
Set j=0
Return array
YES
STOP
array[j+1] < array[j] ?
Set count=0
NO
YES
swap(array,j,j+1)
NO
Is count = 0 ?
YES
Set count=count+1

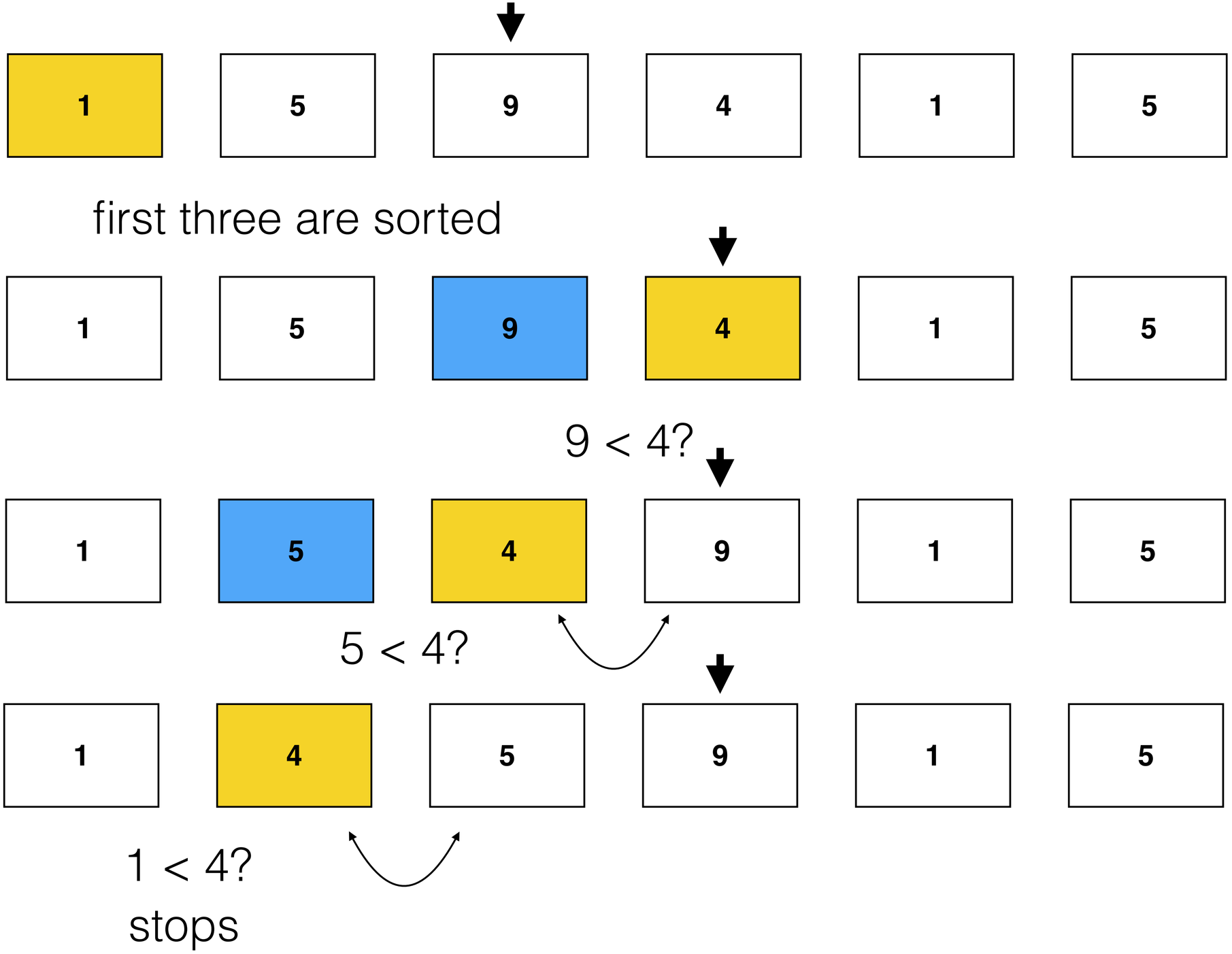
Standard Bubble Sort is this second version

Insertion Sort

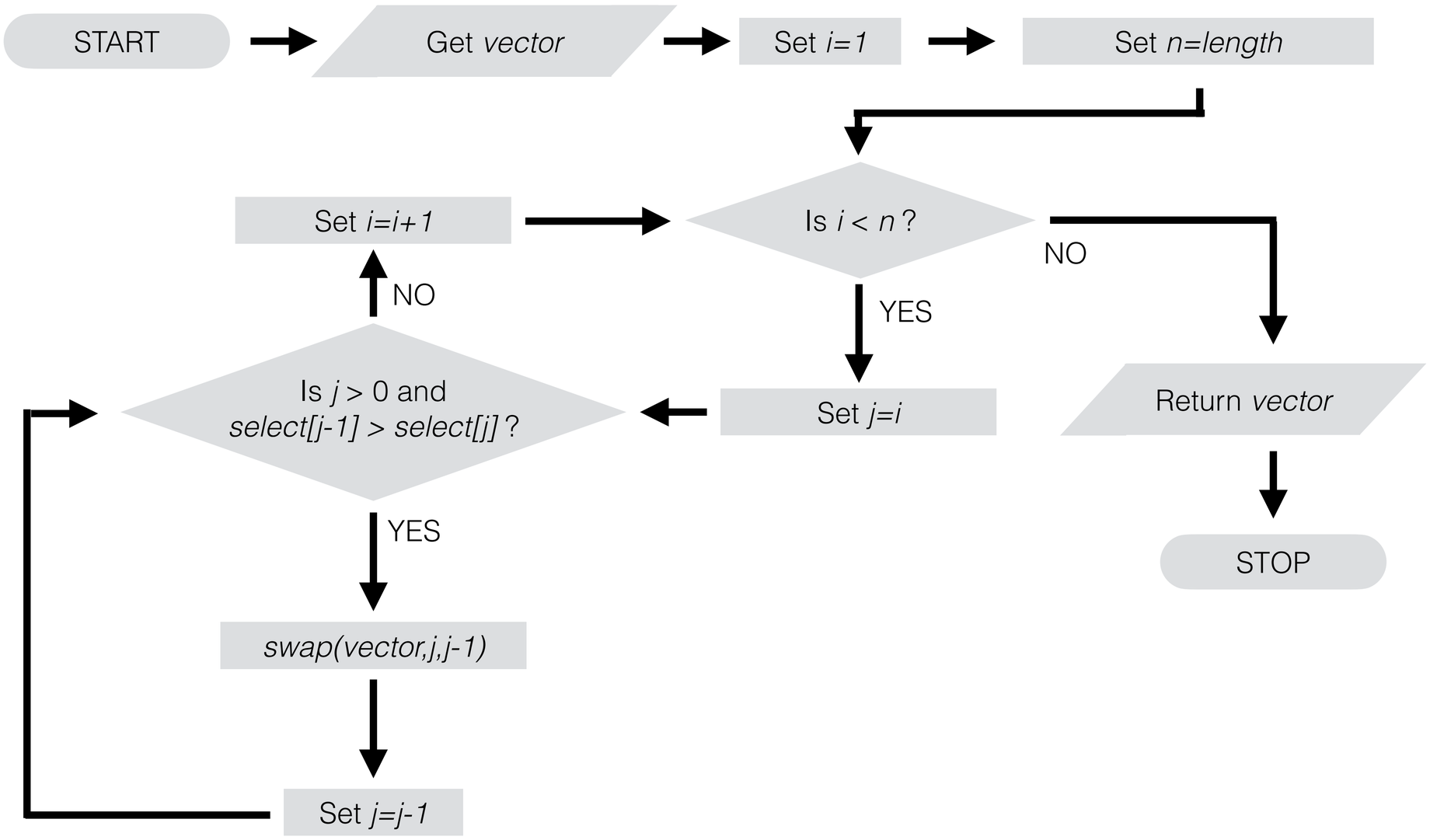
Instead of comparing neighbours

Compare pairwise to the left

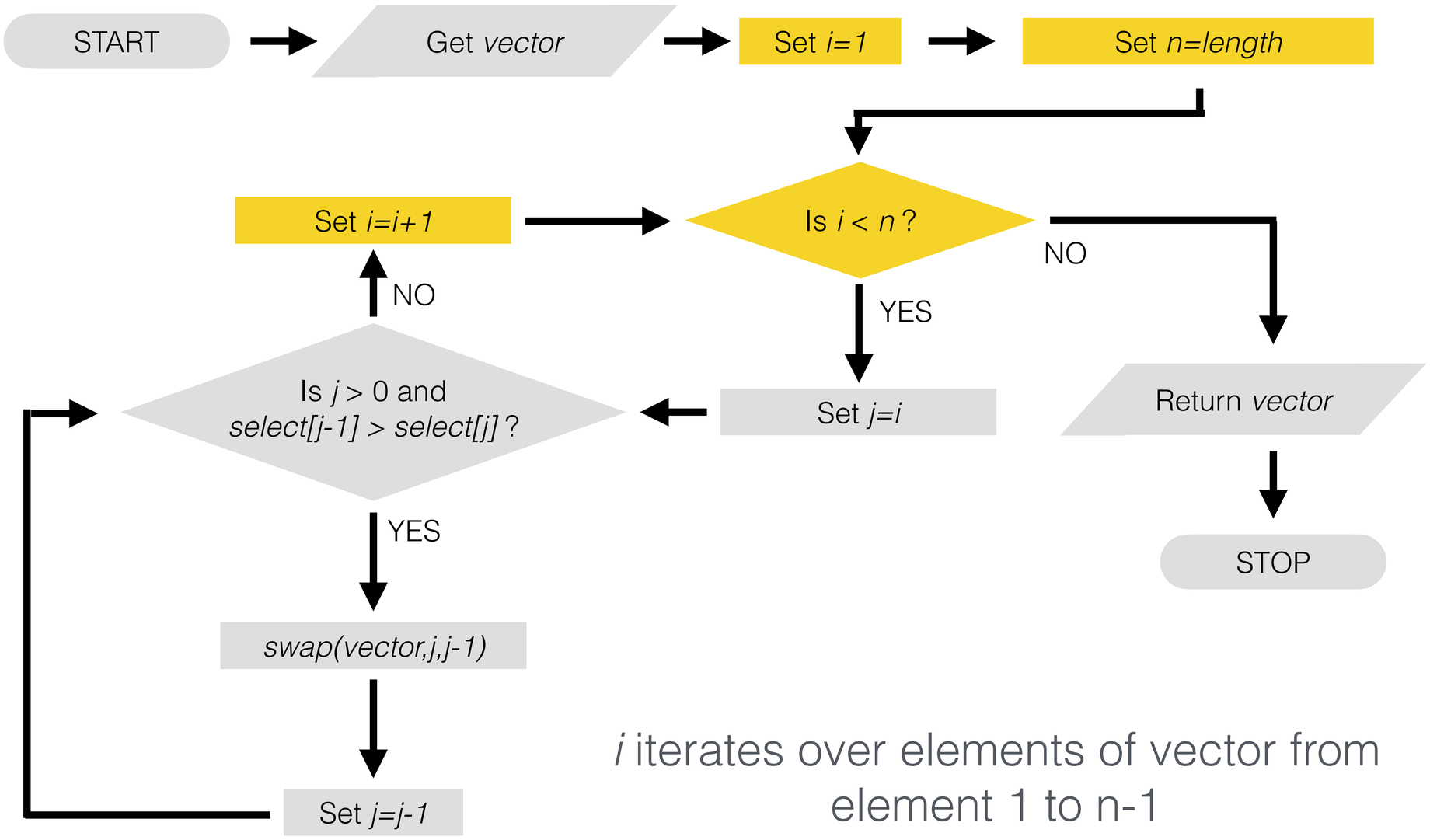

9 5 1 4 1 5
9 < 5?
5 9 1 4 1 5
9 < 1?
5 1 9 4 1 5
5 < 1?
1 5 9 4 1 5
reached the end


1
5
9
4
1
5
first three are sorted
1
5
9
4
1
5
9 < 4?
1
5
4
9
1
5
5 < 4?
1
4
5
9
1
5
1 < 4?
stops

Insertion Sort

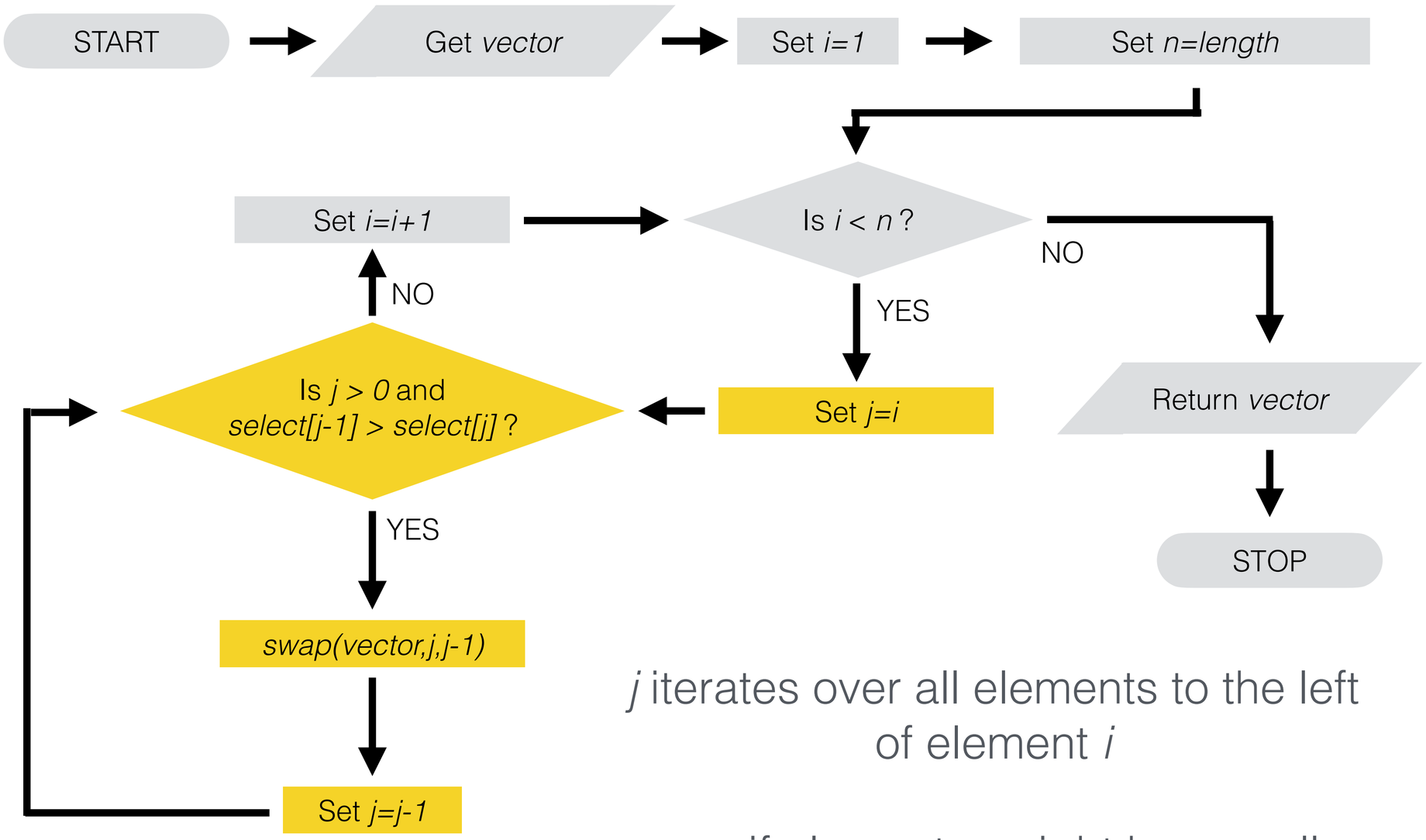


Insertion Sort



i iterates over elements of vector from element 1 to n-1

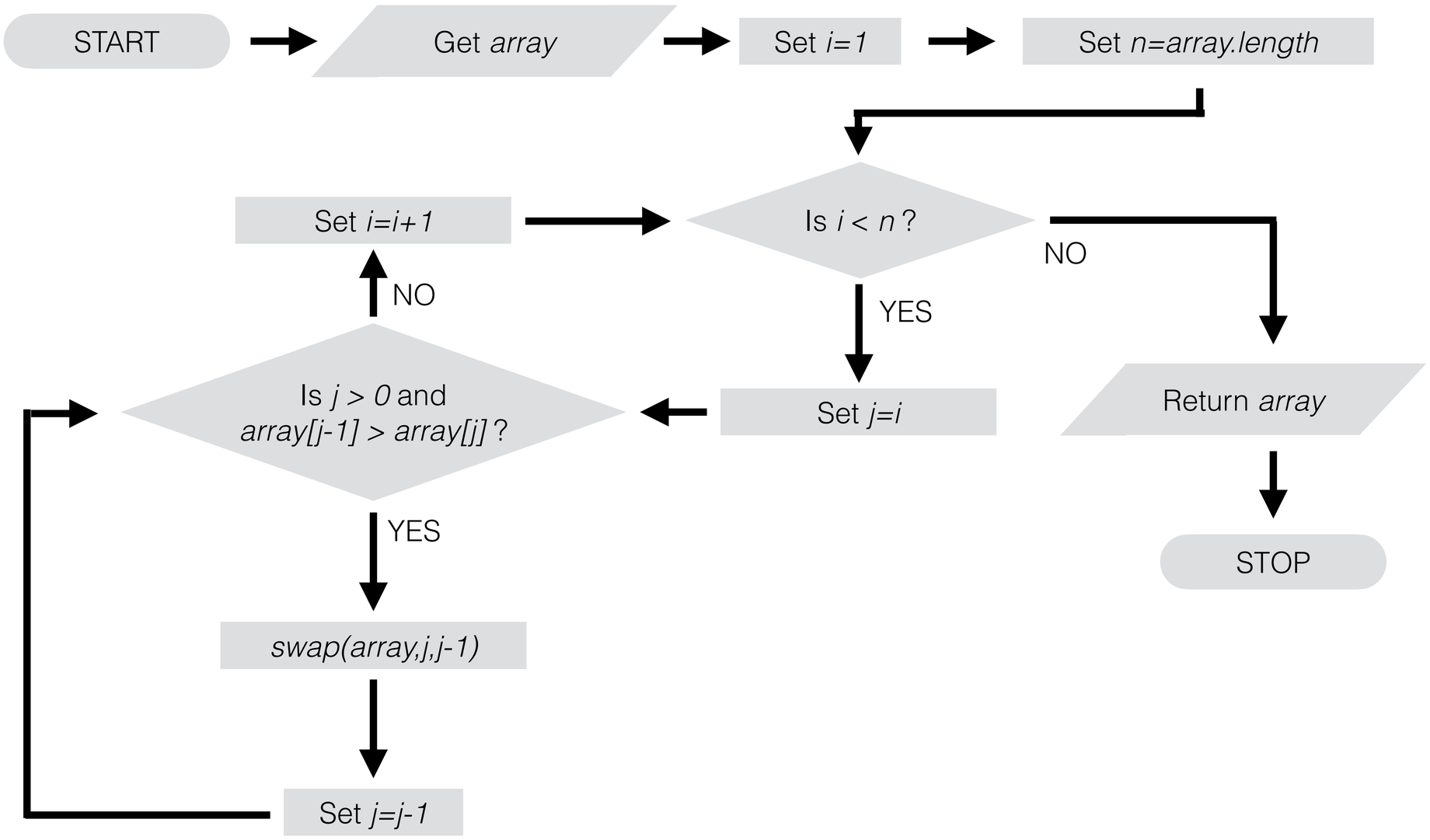
Insertion Sort



j iterates over all elements to the left of element *i*

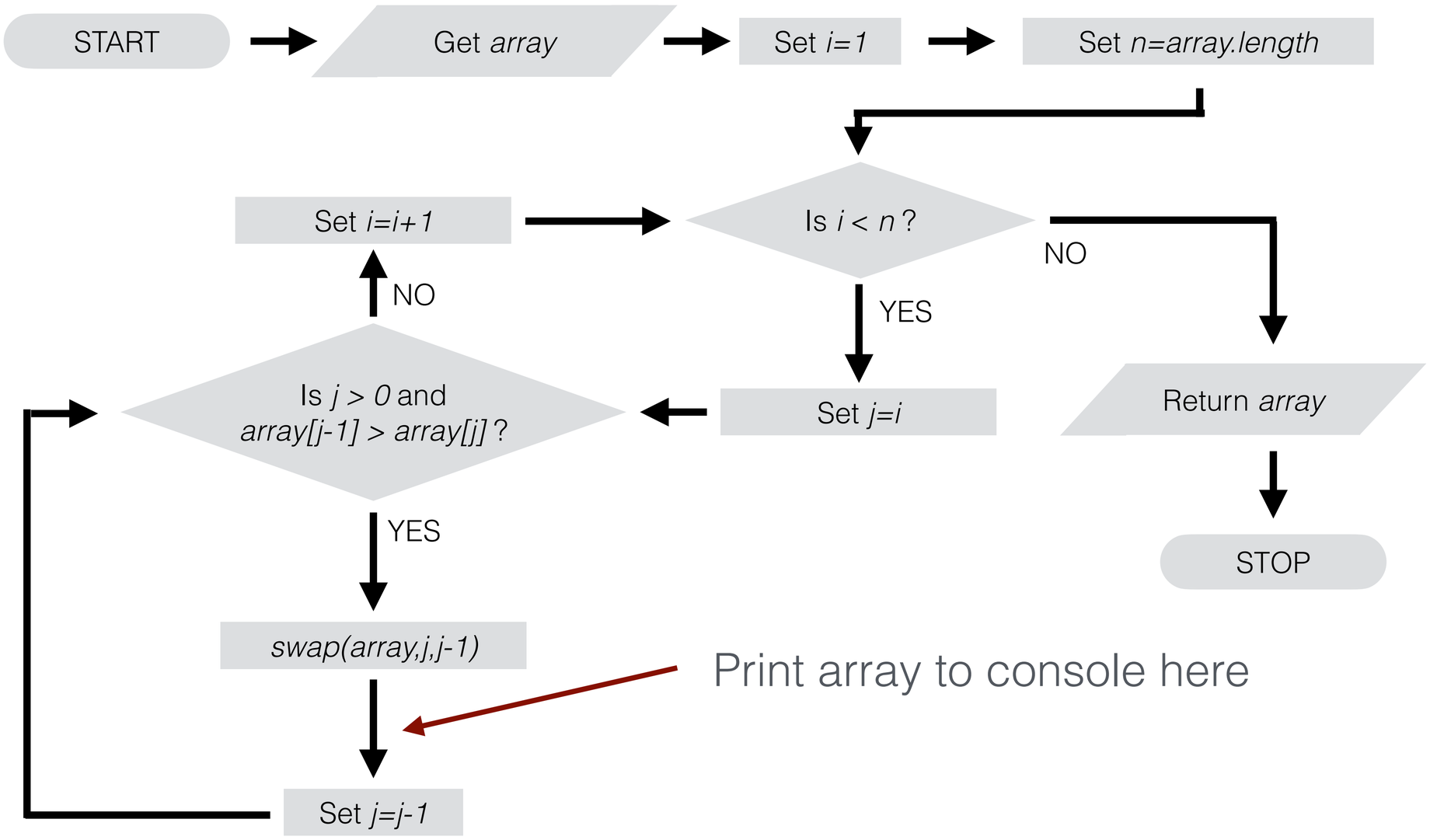
swap if element on right has smaller value than element on left

Insertion Sort for arrays



Complete the insertionSort function for arrays in *sort.js*

Insertion Sort for arrays



Complete the insertionSort function for arrays in *sort.js*

There is an alternative implementation of Insertion Sort without reference to swap function

See Quiz 4

For the Review Seminar:

“pass by value” vs. “pass by reference”