find

`find.c`

- prompts for numbers to fill the haystack
- searches the haystack for a needle
  - calls `sort` and `search`,
    functions defined in `helpers.c`

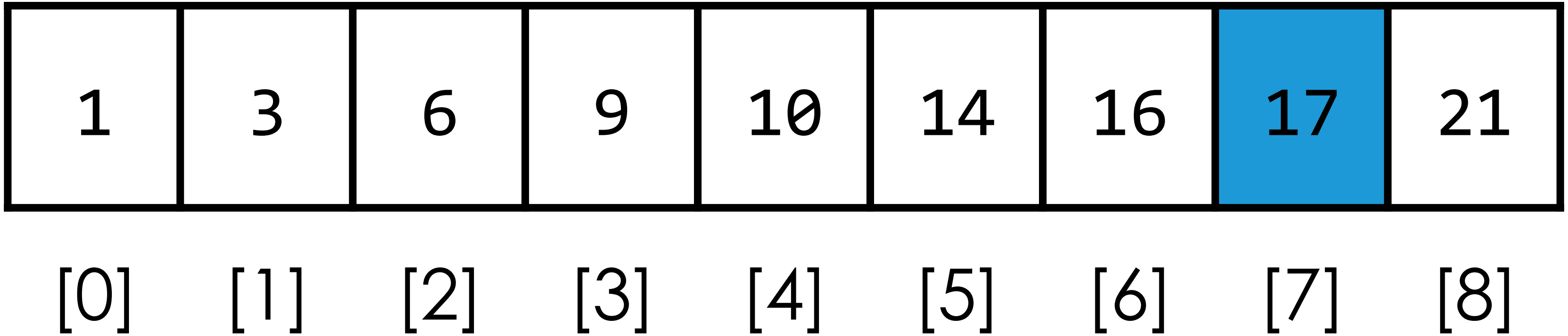
TODO

- □ `search`
  - ▣ return `true` if `value` is found in `haystack`
  - ▣ return `false` if `value` is not in `haystack`
- □ `sort`
  - ▣ sort the `values[]` array

search

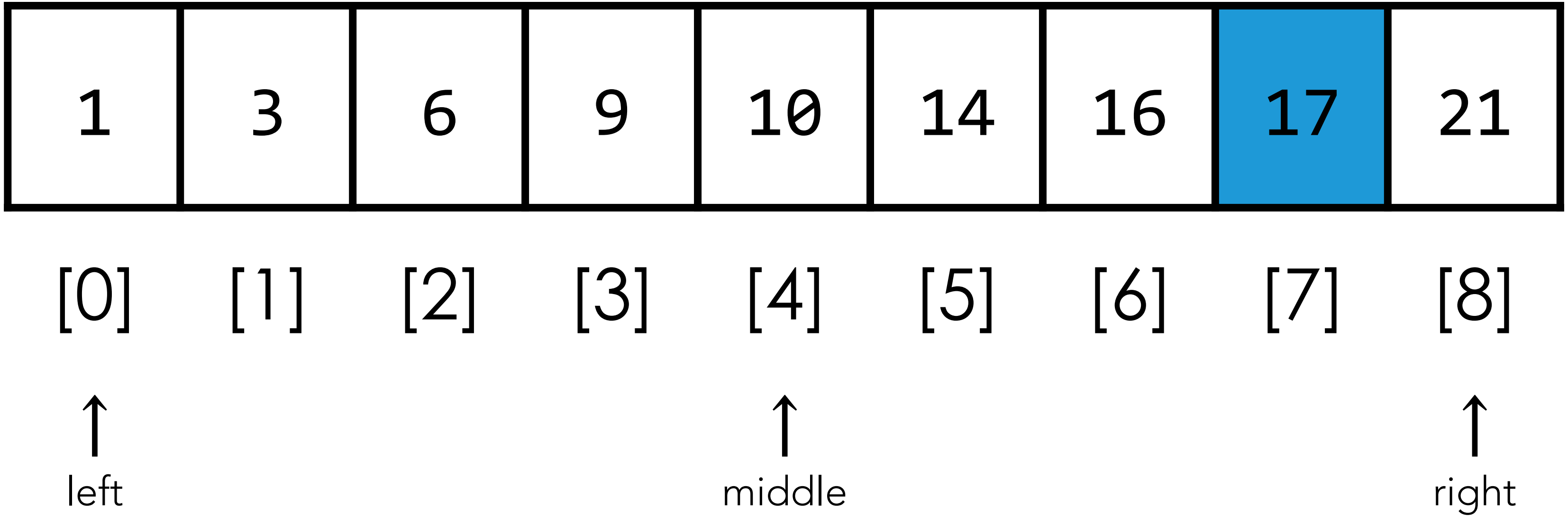
- linear search: you can do better!
  - *O*(*n*) → slow
  - Can search any list
- binary search
  - *O*(log *n*) → fast
  - can only search sorted lists

binary search



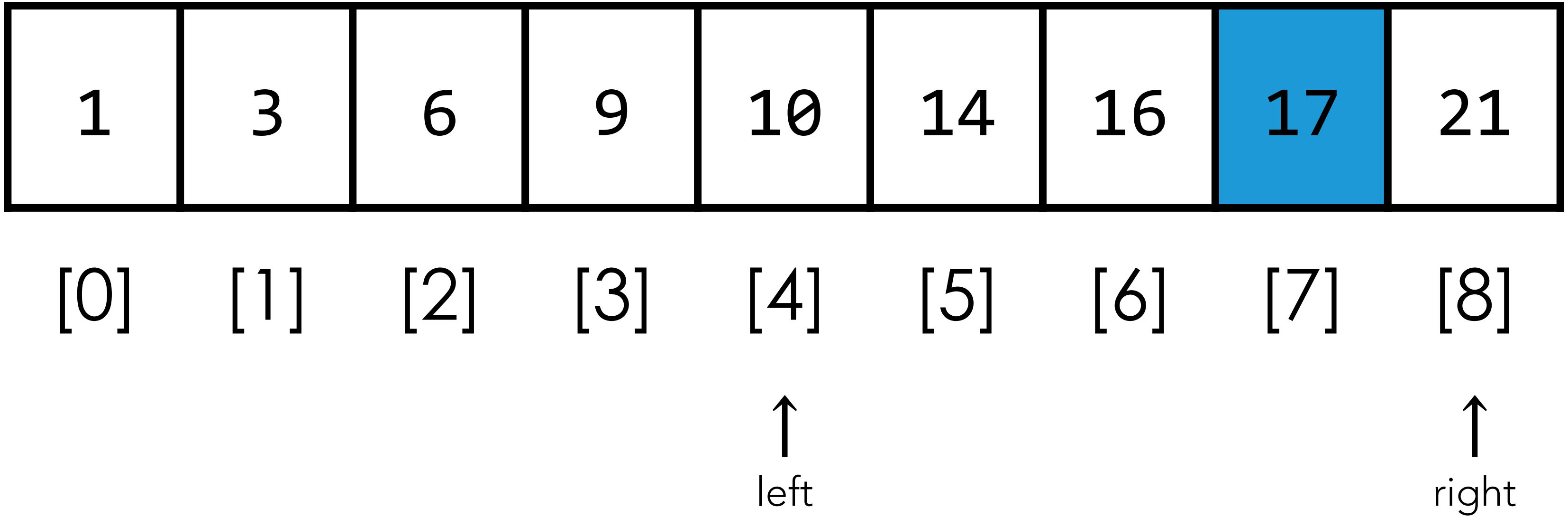
1	3	6	9	10	14	16	17	21
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]

binary search



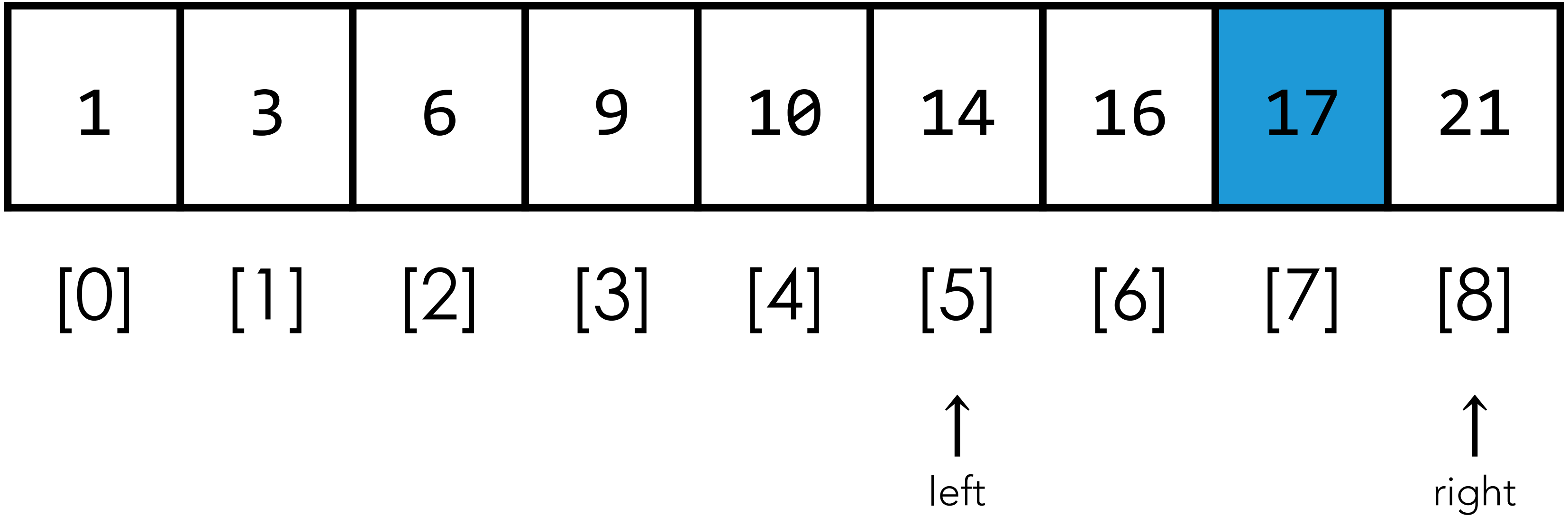
1	3	6	9	10	14	16	17	21
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]

binary search



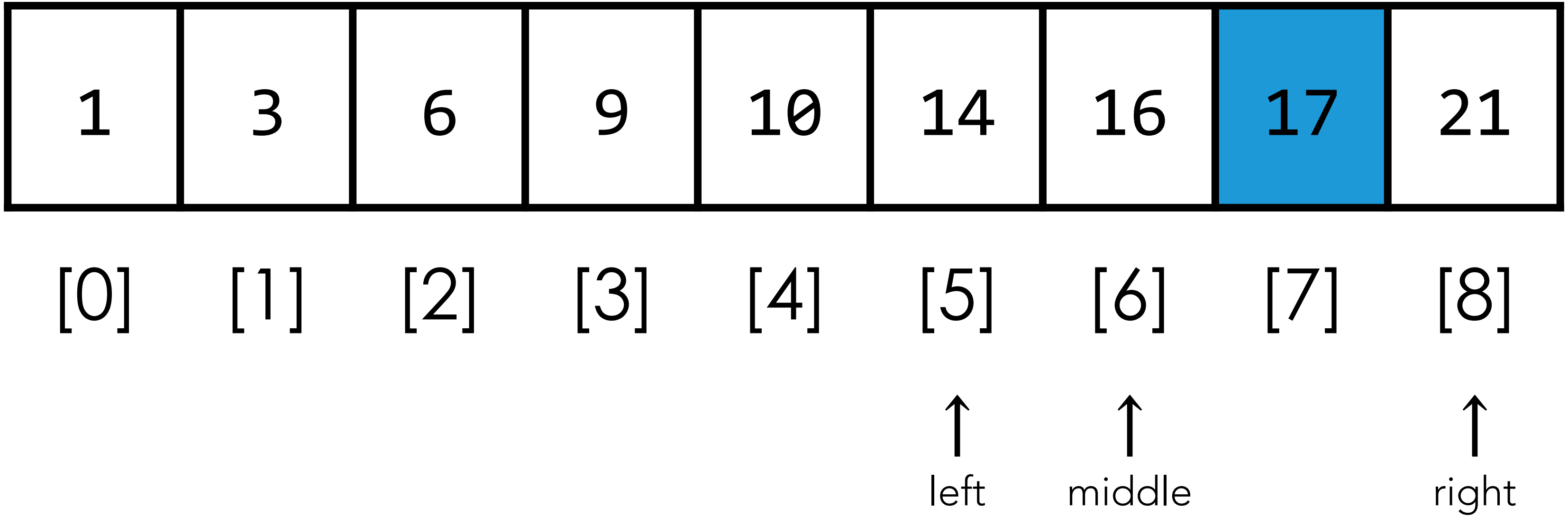
1	3	6	9	10	14	16	17	21
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
				left				right

binary search



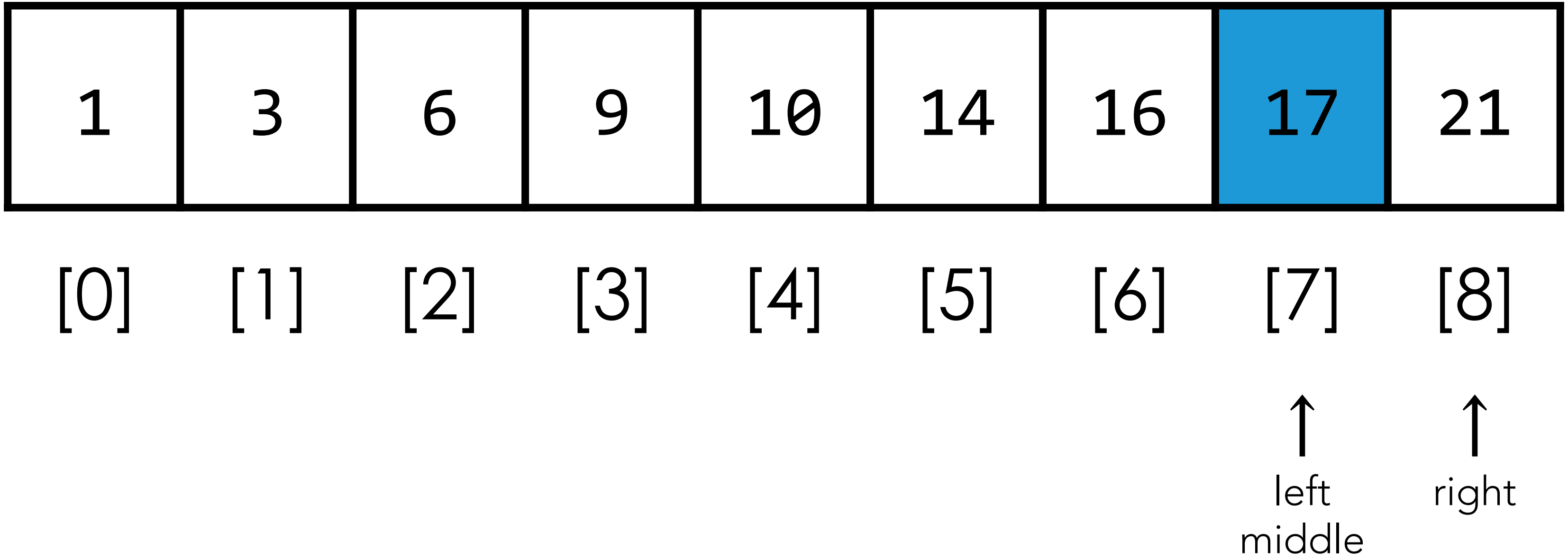
1	3	6	9	10	14	16	17	21
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]

binary search



1	3	6	9	10	14	16	17	21
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
					↑ left	↑ middle		↑ right

binary search



1	3	6	9	10	14	16	17	21
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]

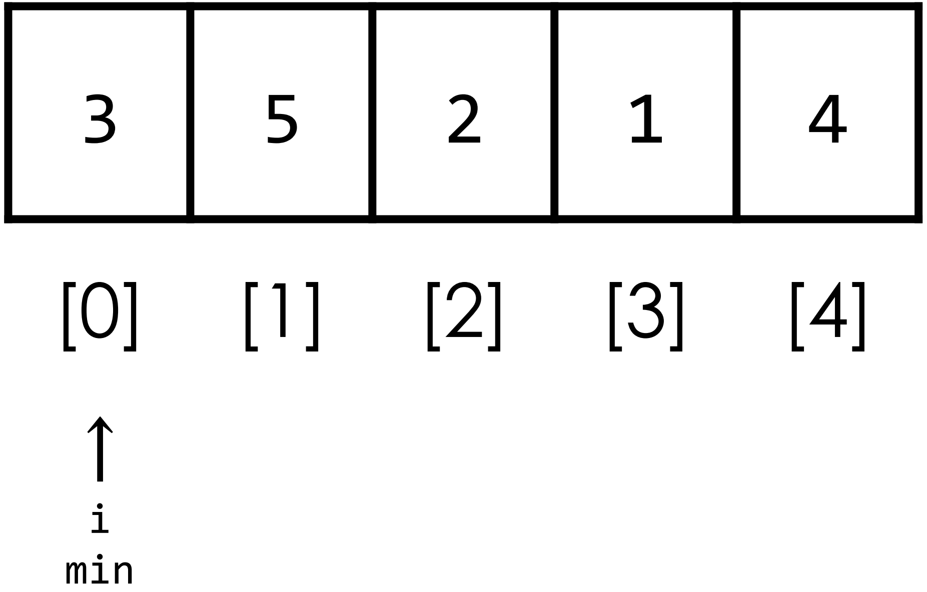
binary search: pseudocode

```
while length of list > 0
    look at middle of list
    if number found, return true
    else if number higher, search left
    else if number lower, search right
return false
```

TODO

- ☑ search
- □ sort

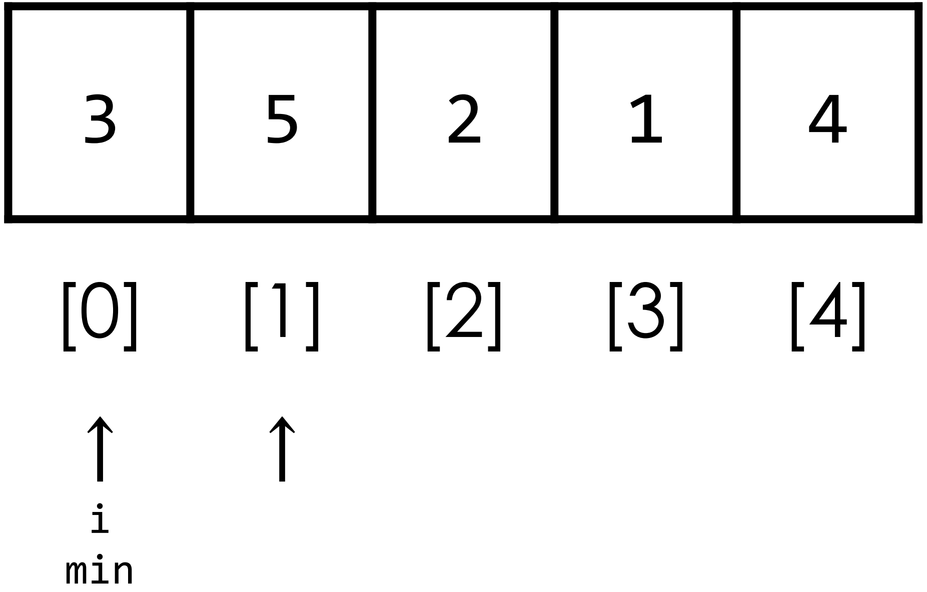
selection sort



3	5	2	1	4
[0]	[1]	[2]	[3]	[4]

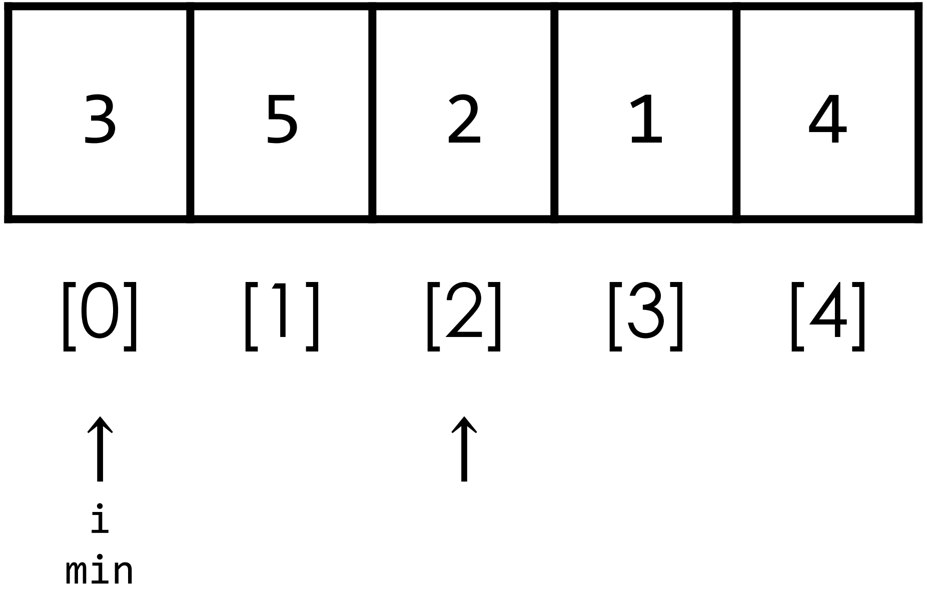
↑
i
min

selection sort



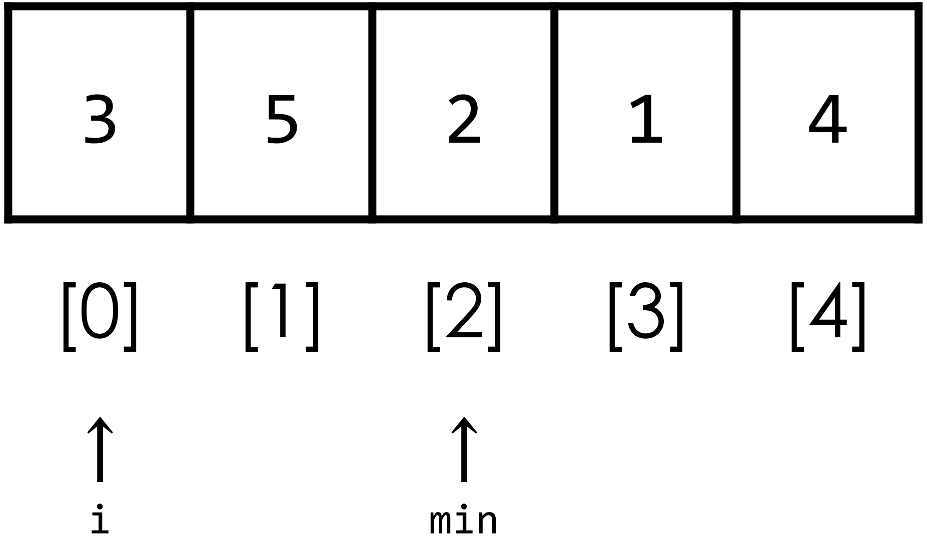
3	5	2	1	4
[0]	[1]	[2]	[3]	[4]

selection sort



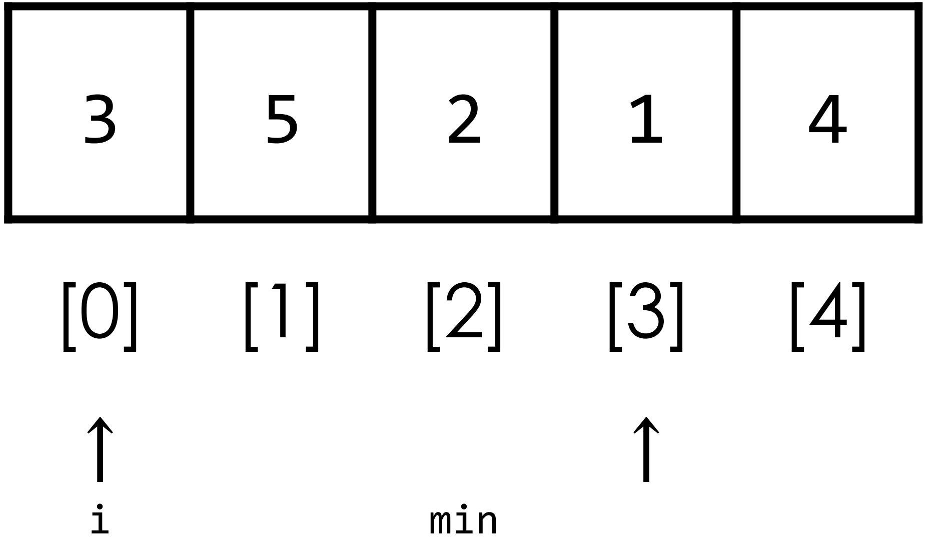
3	5	2	1	4
[0]	[1]	[2]	[3]	[4]

selection sort



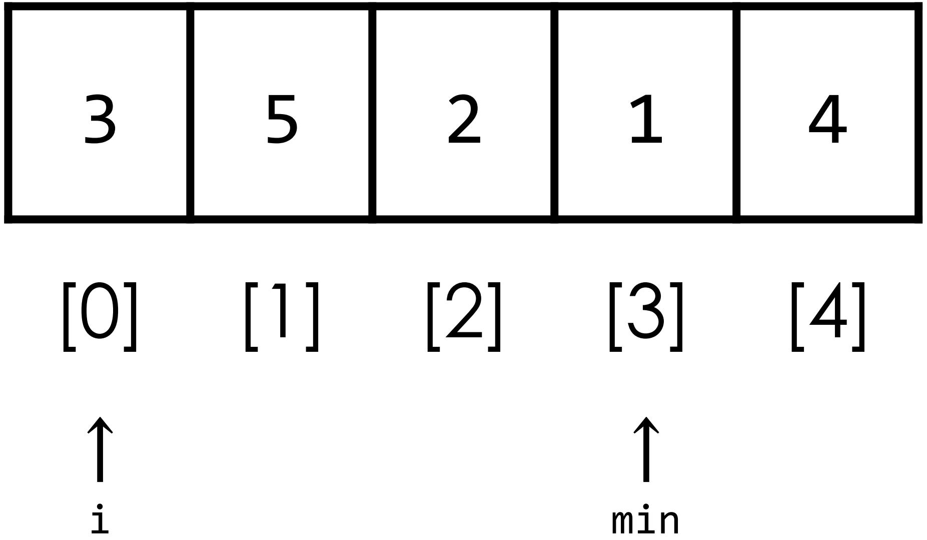
3	5	2	1	4
[0]	[1]	[2]	[3]	[4]

selection sort



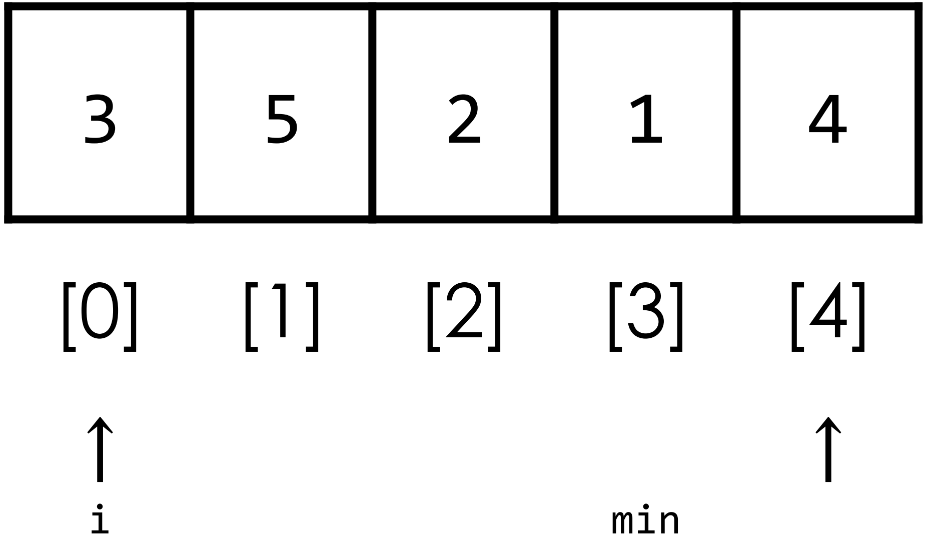
3	5	2	1	4
[0]	[1]	[2]	[3]	[4]

selection sort



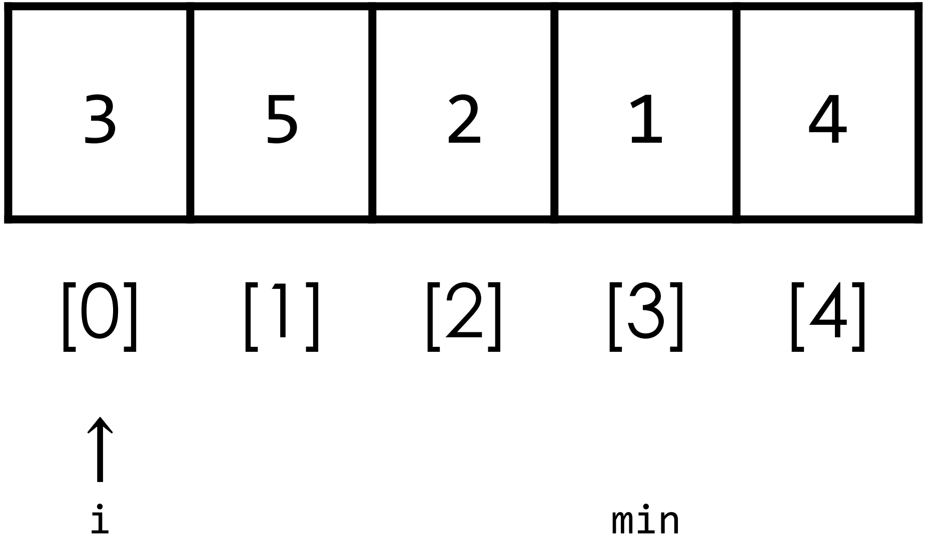
3	5	2	1	4
[0]	[1]	[2]	[3]	[4]
↑			↑	
i			min	

selection sort



3	5	2	1	4
[0]	[1]	[2]	[3]	[4]

selection sort



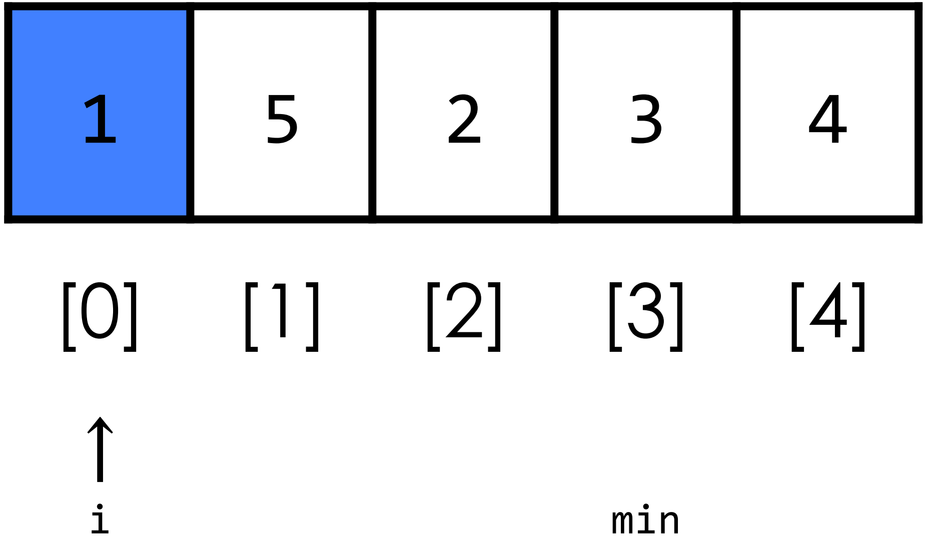
3	5	2	1	4
[0]	[1]	[2]	[3]	[4]

↑

i

min

selection sort



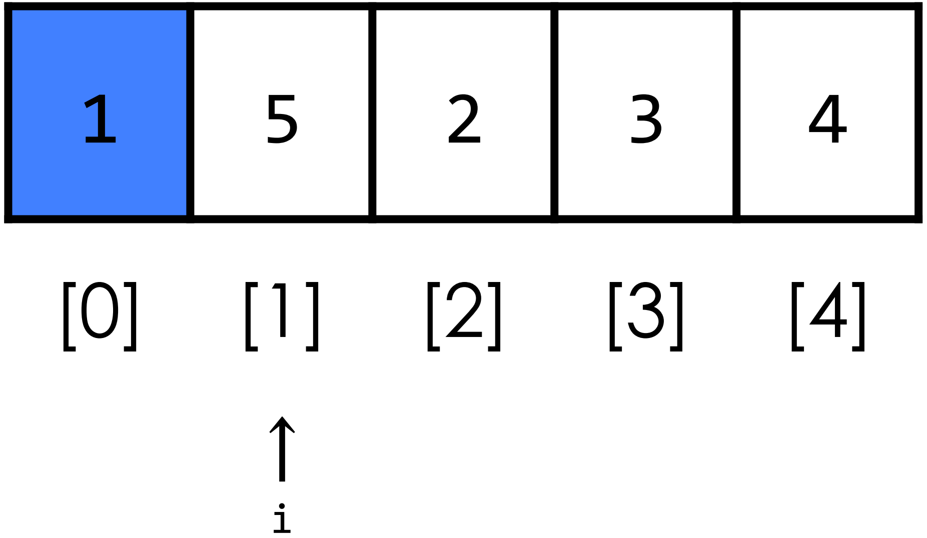
1	5	2	3	4
[0]	[1]	[2]	[3]	[4]

↑

i

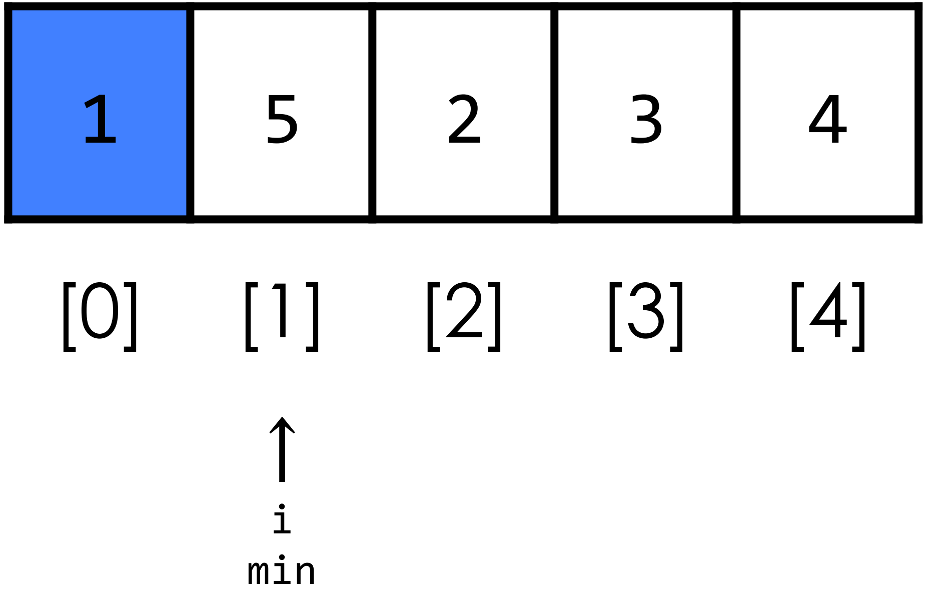
min

selection sort



1	5	2	3	4
[0]	[1]	[2]	[3]	[4]
	↑ i			

selection sort



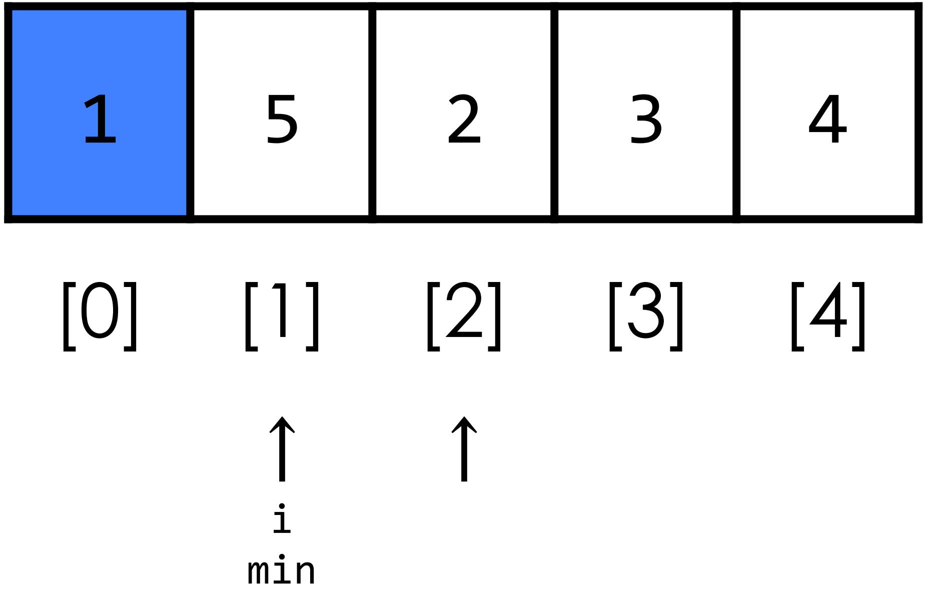
1	5	2	3	4
[0]	[1]	[2]	[3]	[4]

↑

i

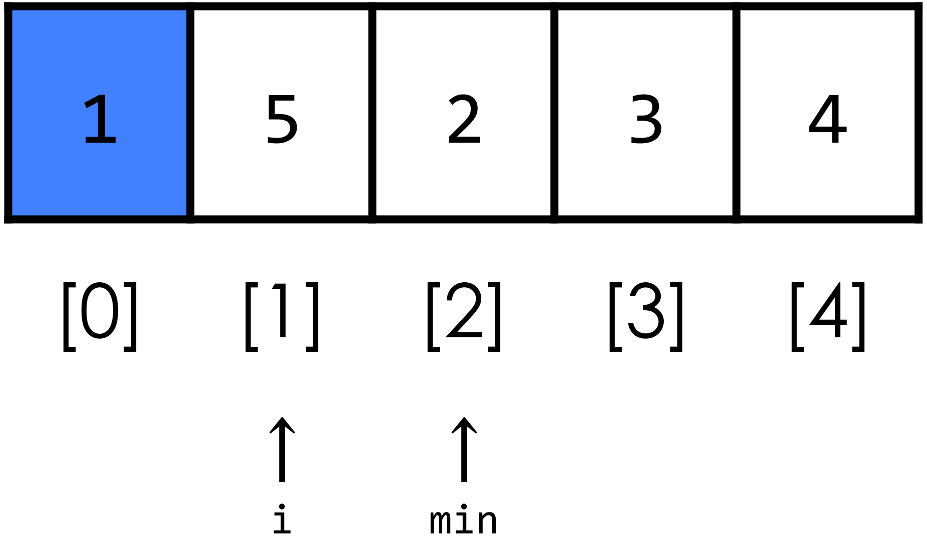
min

selection sort



1	5	2	3	4
[0]	[1]	[2]	[3]	[4]

selection sort



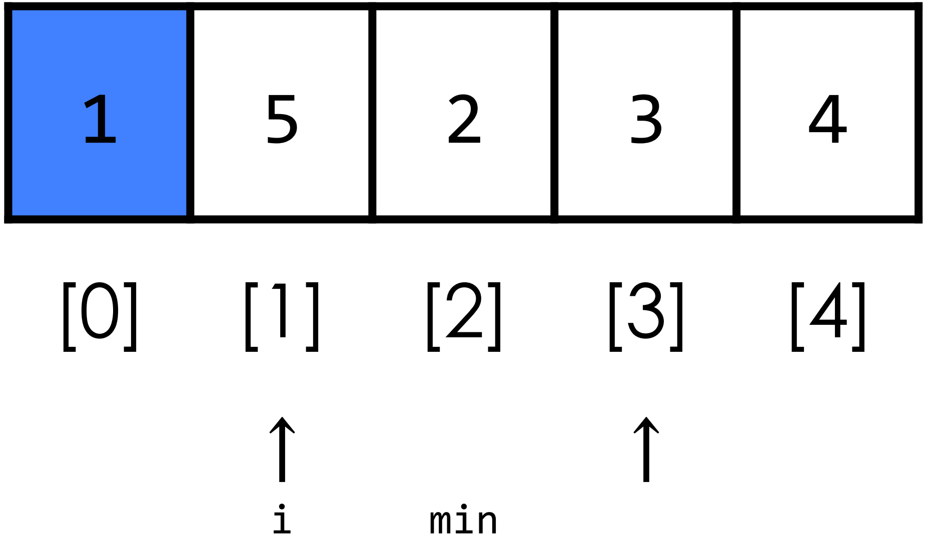
1 5 2 3 4

[0] [1] [2] [3] [4]

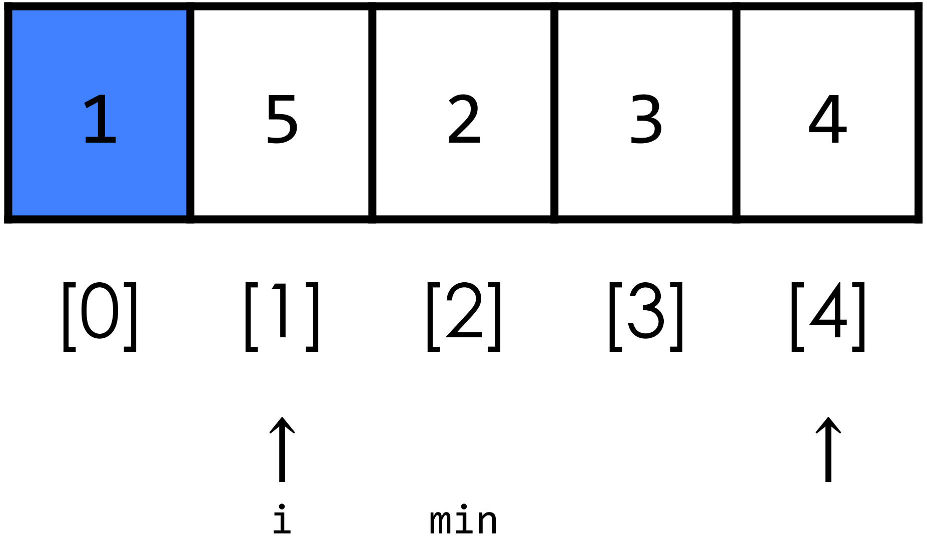
i

min

selection sort

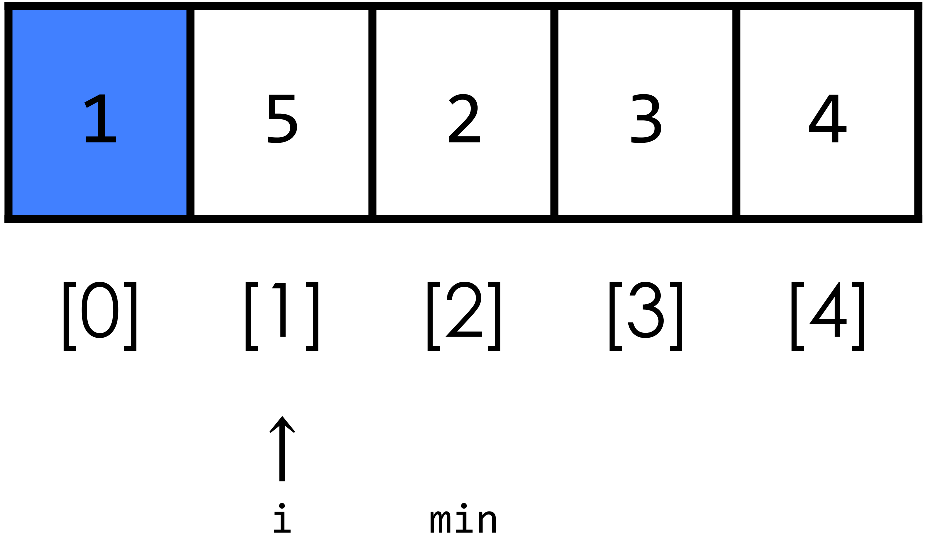


selection sort

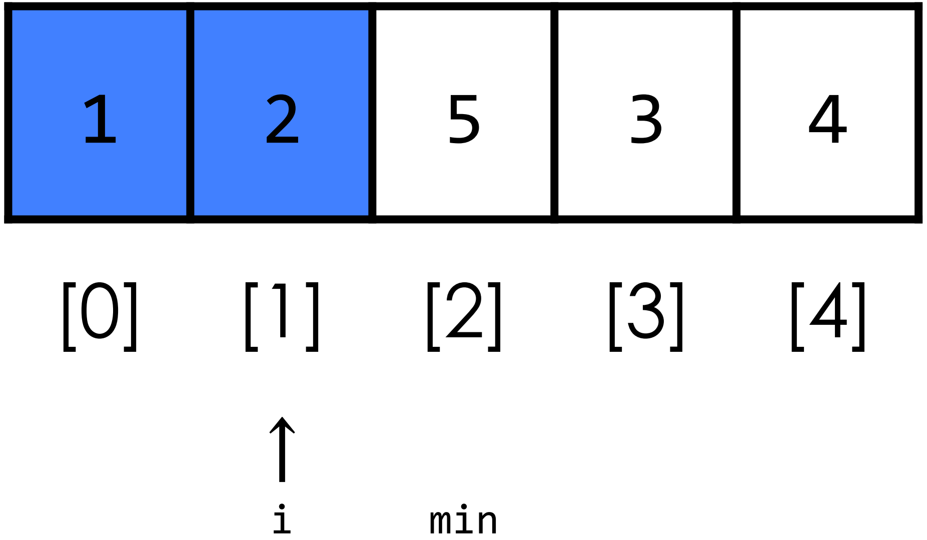


1	5	2	3	4
[0]	[1]	[2]	[3]	[4]
	i	min		

selection sort



selection sort

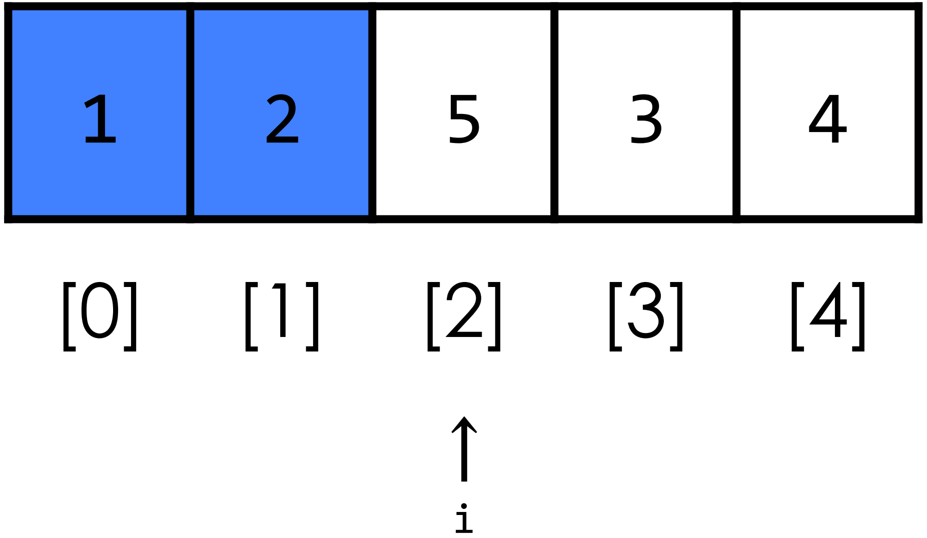


1	2	5	3	4
[0]	[1]	[2]	[3]	[4]

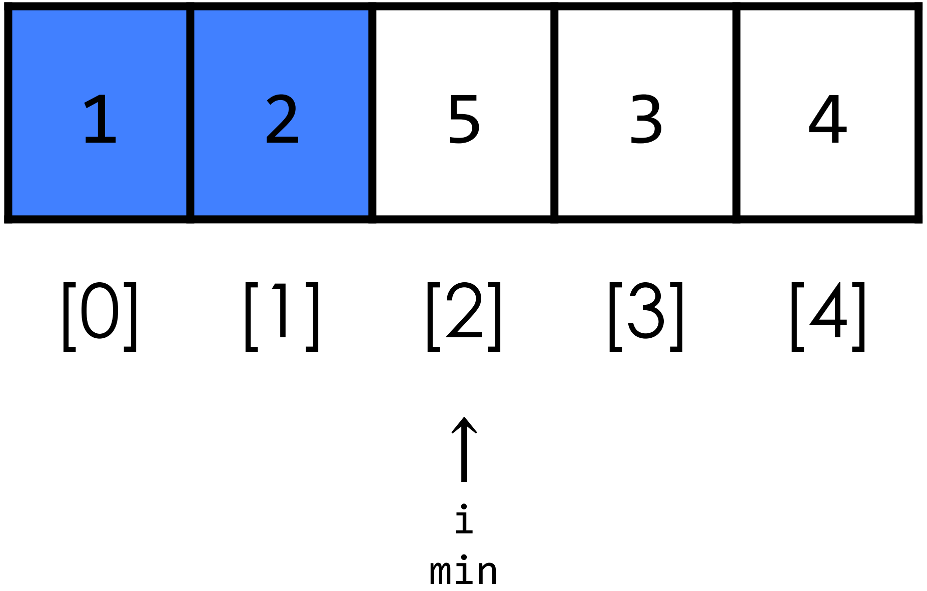
i

min

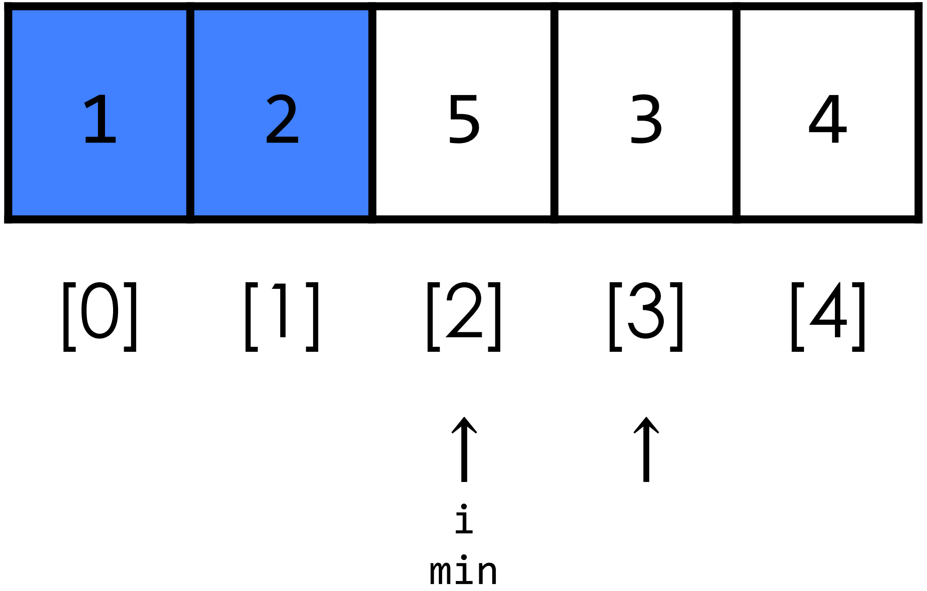
selection sort



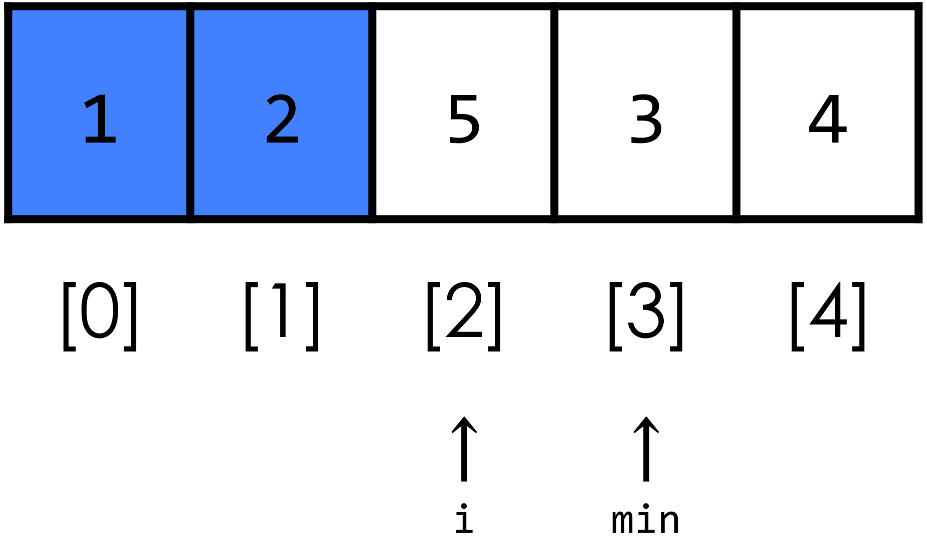
selection sort



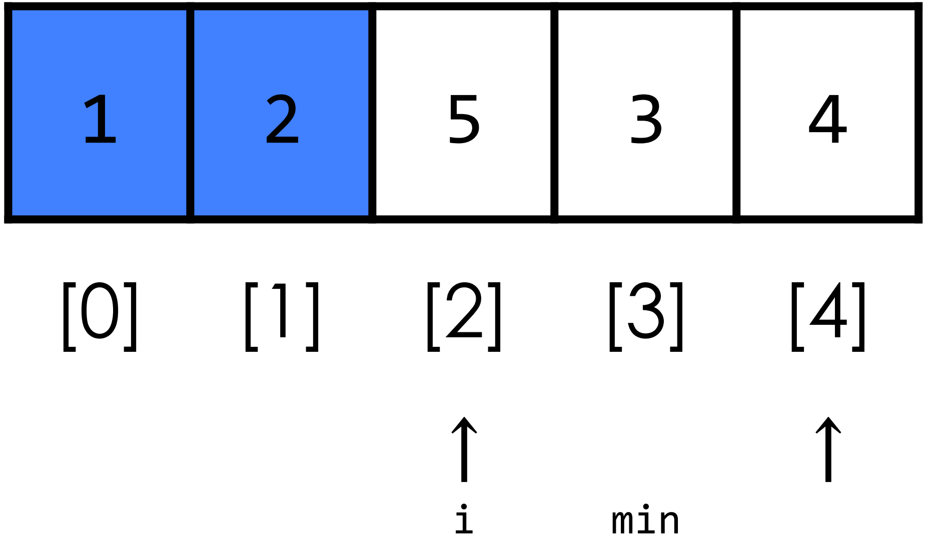
selection sort



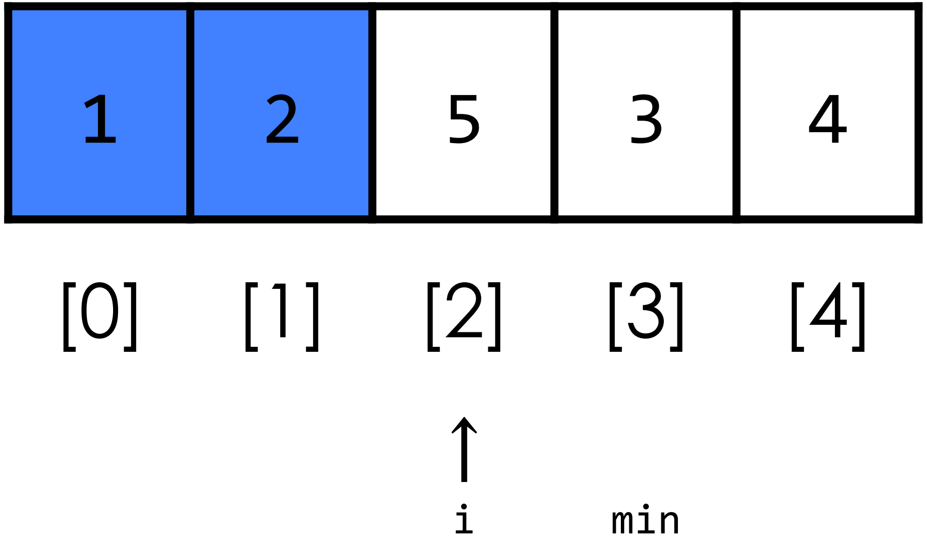
selection sort



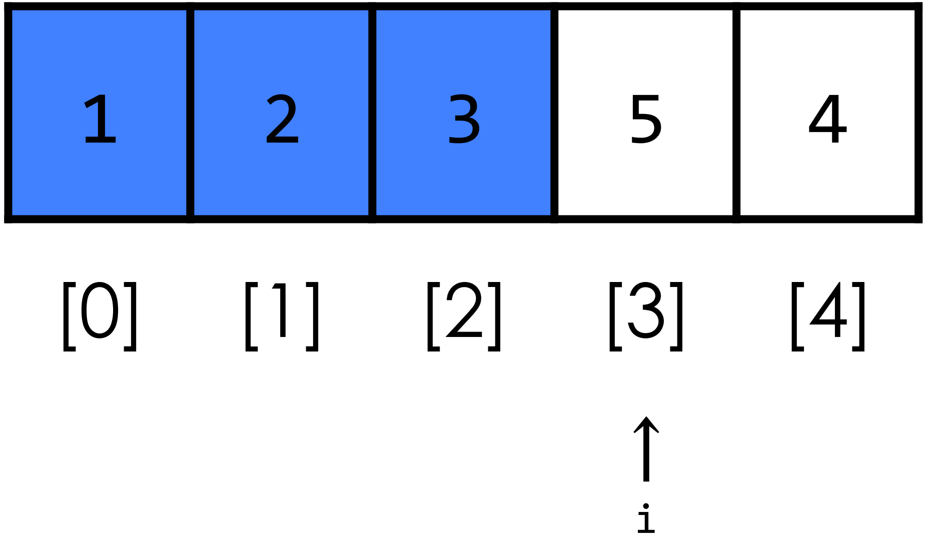
selection sort



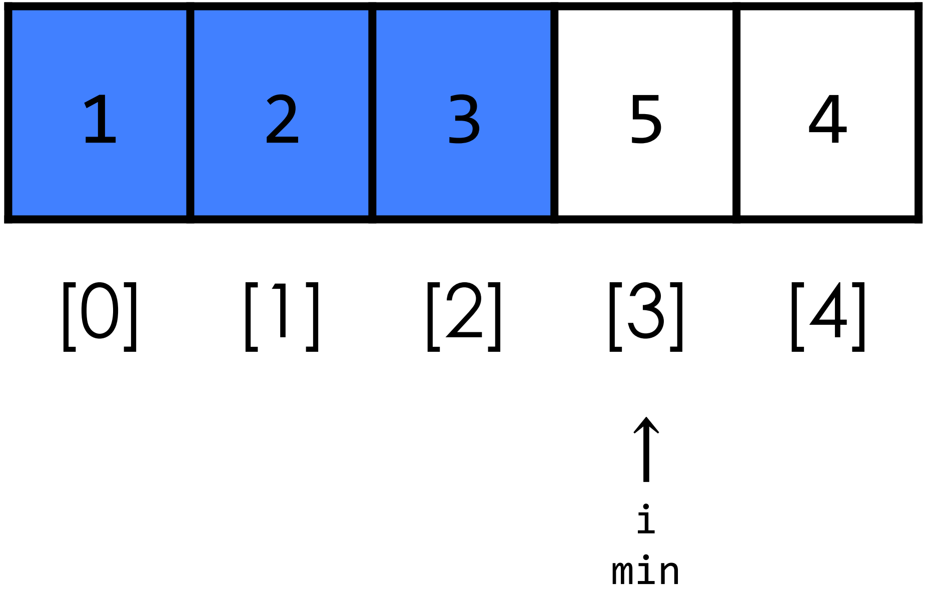
selection sort



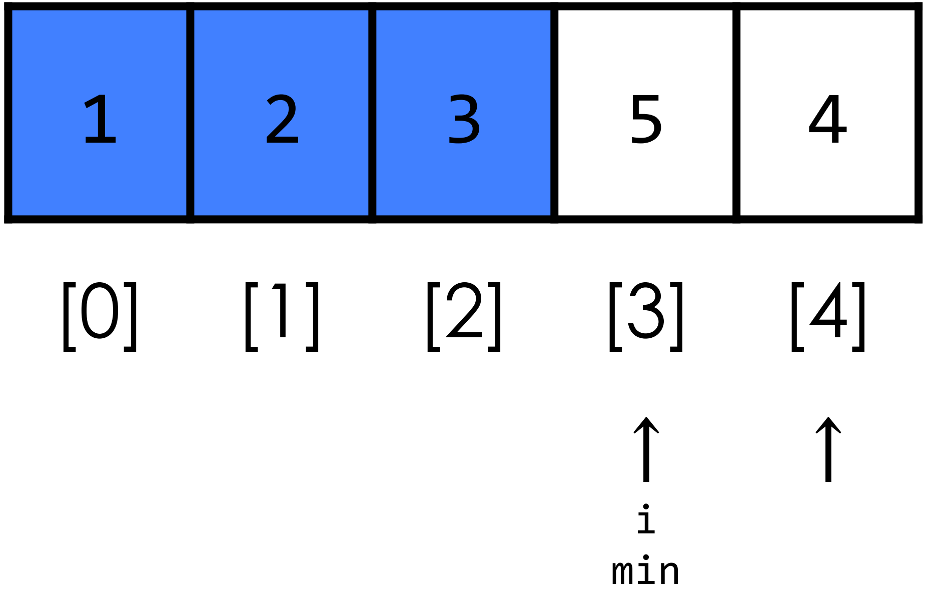
selection sort



selection sort



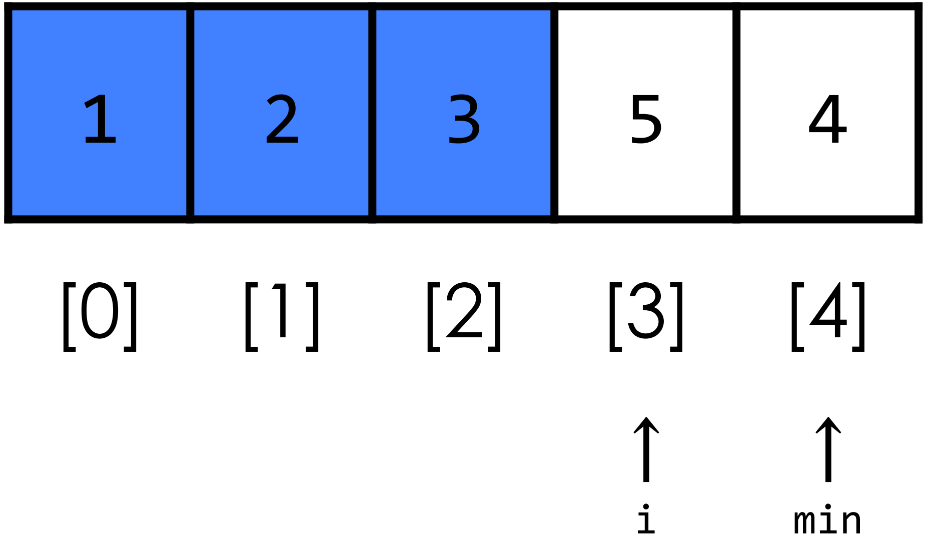
selection sort



1	2	3	5	4
[0]	[1]	[2]	[3]	[4]

i
min

selection sort

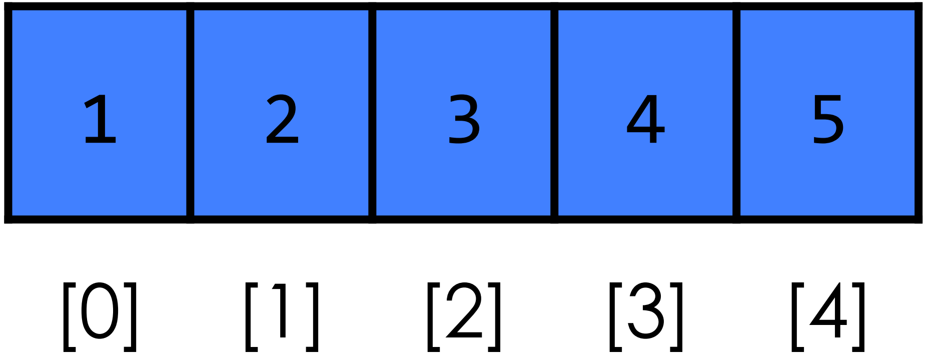


1 2 3 5 4

[0] [1] [2] [3] [4]

i min

selection sort



1	2	3	4	5
[0]	[1]	[2]	[3]	[4]

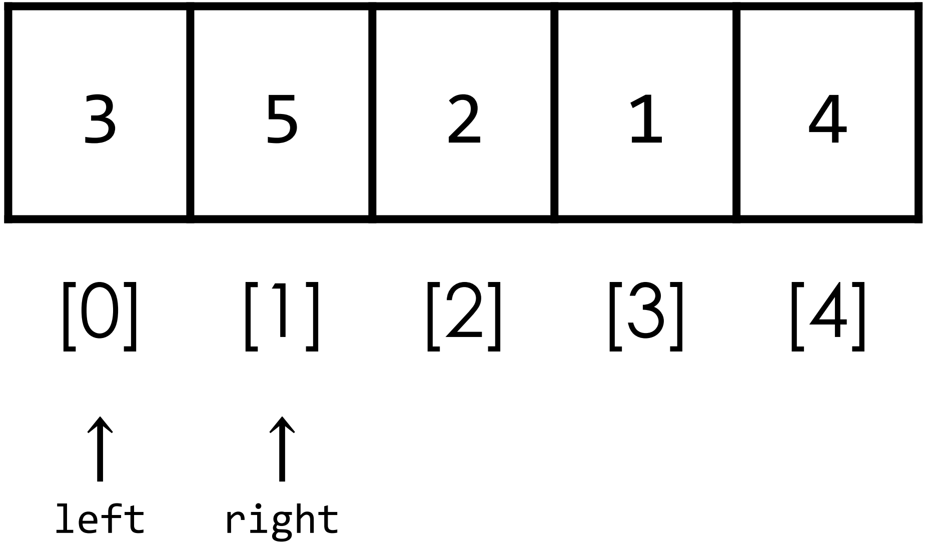
selection sort pseudocode

```
for i = 0 to n – 2
    min = i
    find smallest element from i to n – 1
    if min != i
        exchange smallest element with element at i
```

bubble sort

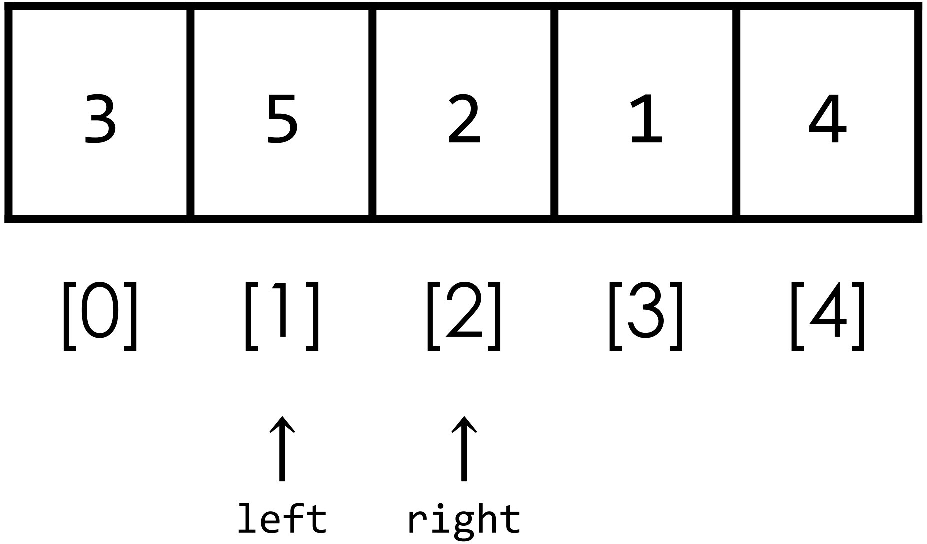
- iterate over list
- compare adjacent elements
- swap elements that are in the wrong order
- largest element will 'bubble' to the end
- the list is sorted once no elements have been swapped

bubble sort



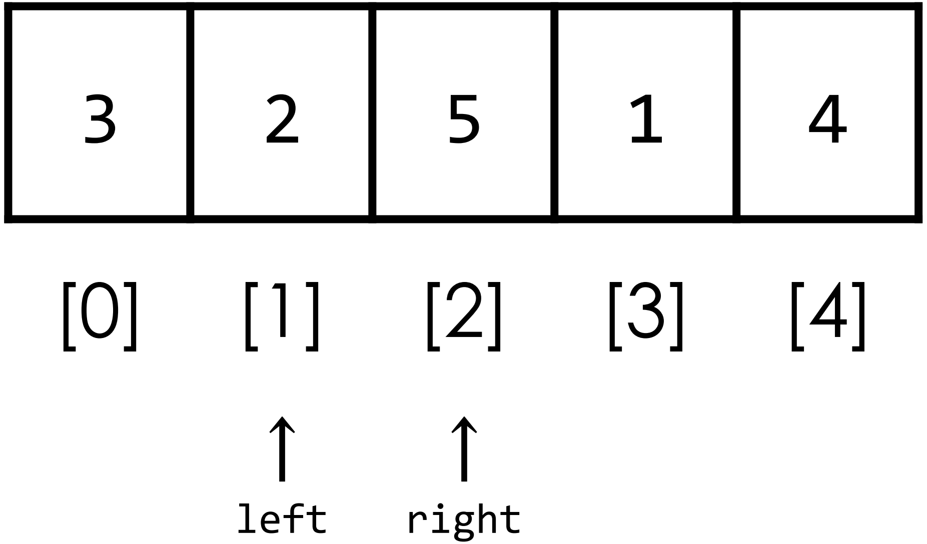
3	5	2	1	4
[0]	[1]	[2]	[3]	[4]
↑	↑			
`left`	`right`			

bubble sort



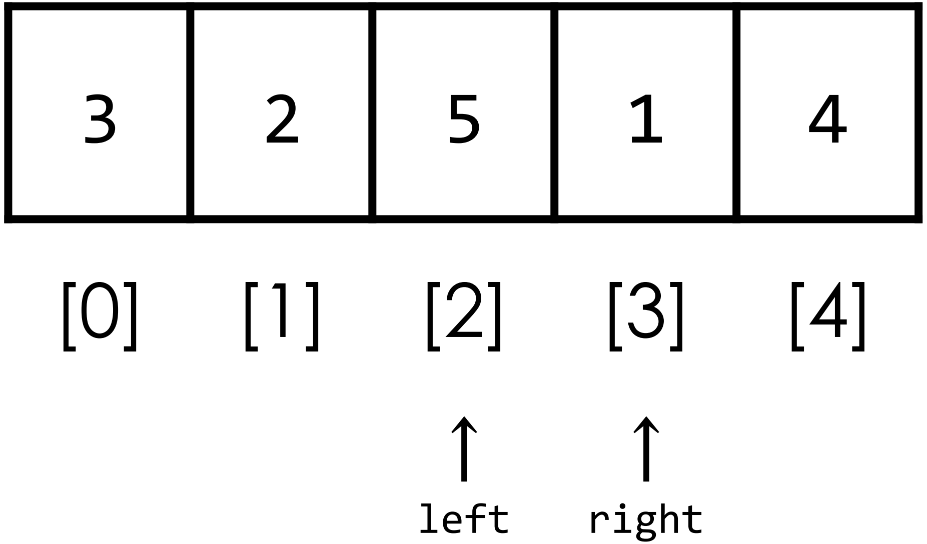
3	5	2	1	4
[0]	[1]	[2]	[3]	[4]
	↑	↑		
	left	right		

bubble sort



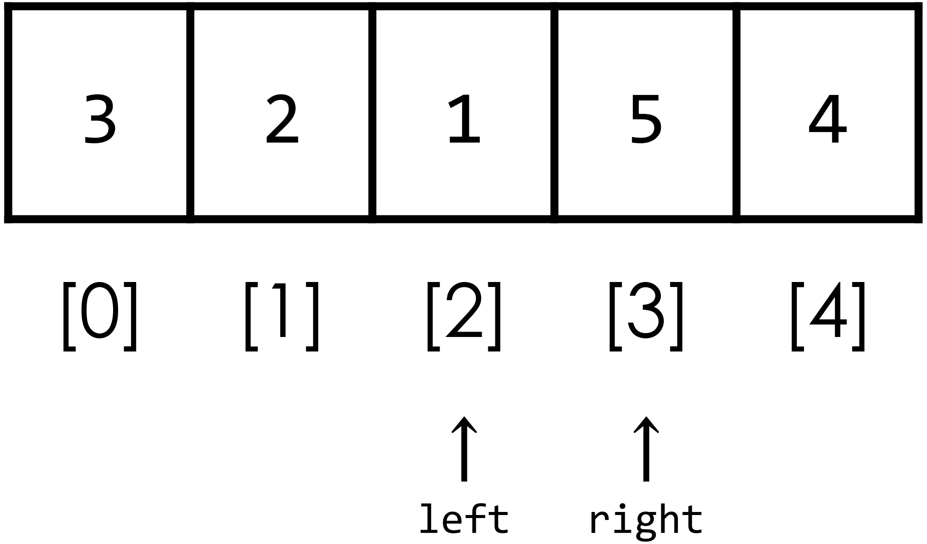
3	2	5	1	4
[0]	[1]	[2]	[3]	[4]
	↑	↑		
	left	right		

bubble sort



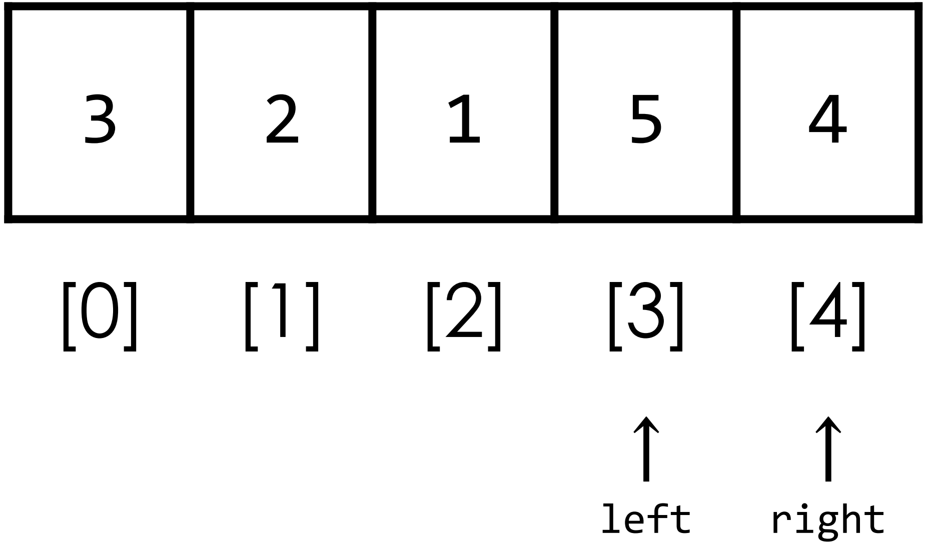
[0]	[1]	[2]	[3]	[4]
3	2	5	1	4

bubble sort



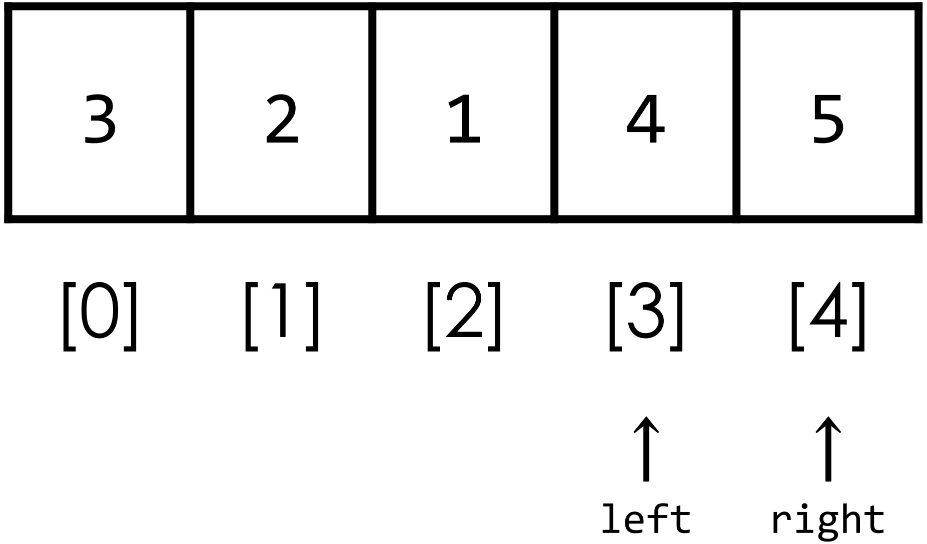
[0]	[1]	[2]	[3]	[4]
3	2	1	5	4

bubble sort



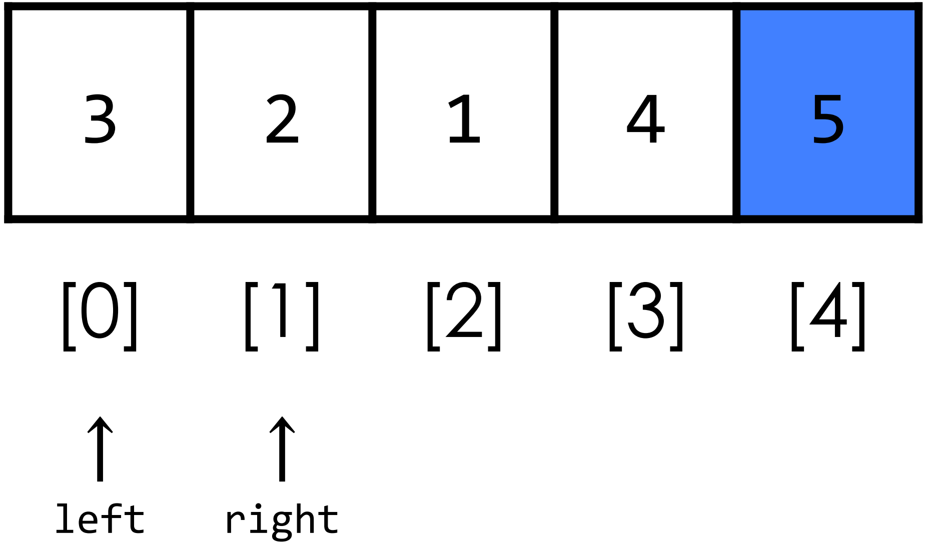
3	2	1	5	4
[0]	[1]	[2]	[3]	[4]
			↑	↑
			left	right

bubble sort



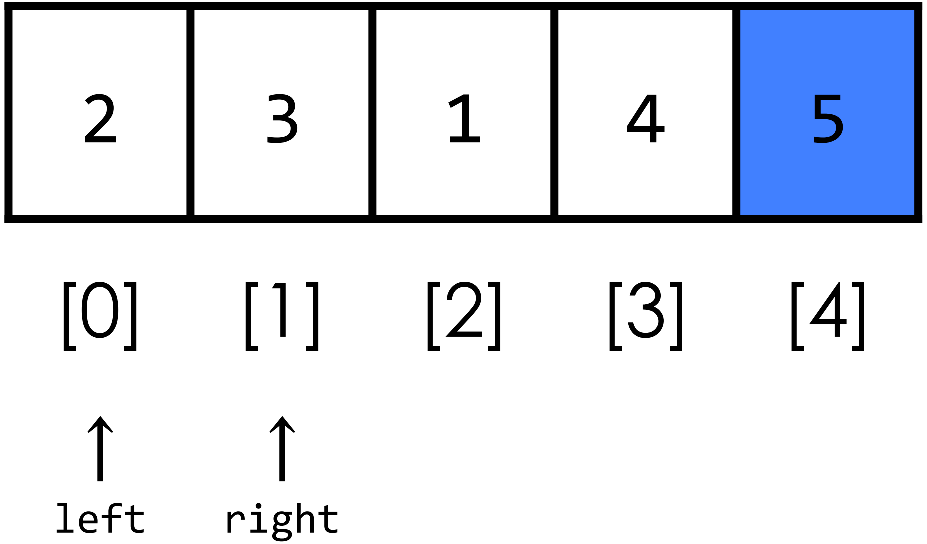
3	2	1	4	5
[0]	[1]	[2]	[3]	[4]
			↑	↑
			left	right

bubble sort



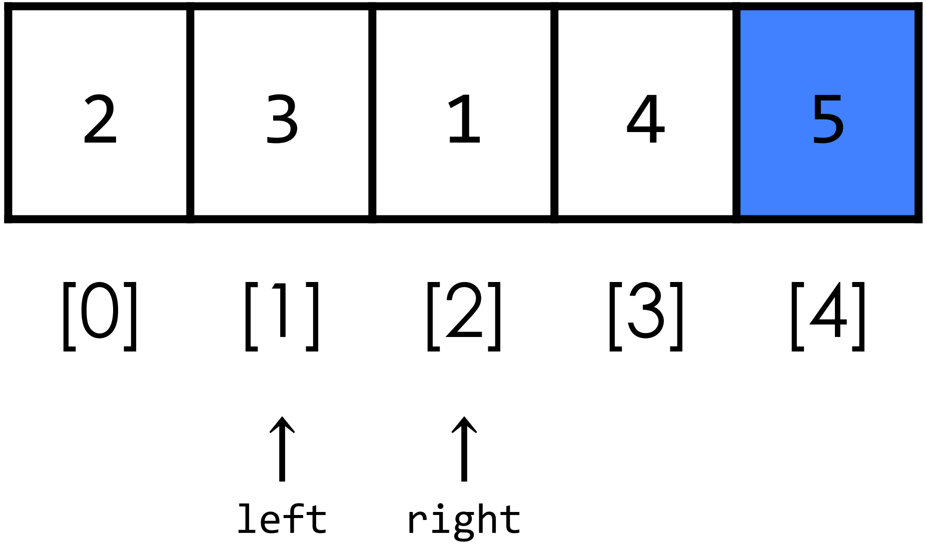
3	2	1	4	5
[0]	[1]	[2]	[3]	[4]
↑	↑			
left	right			

bubble sort



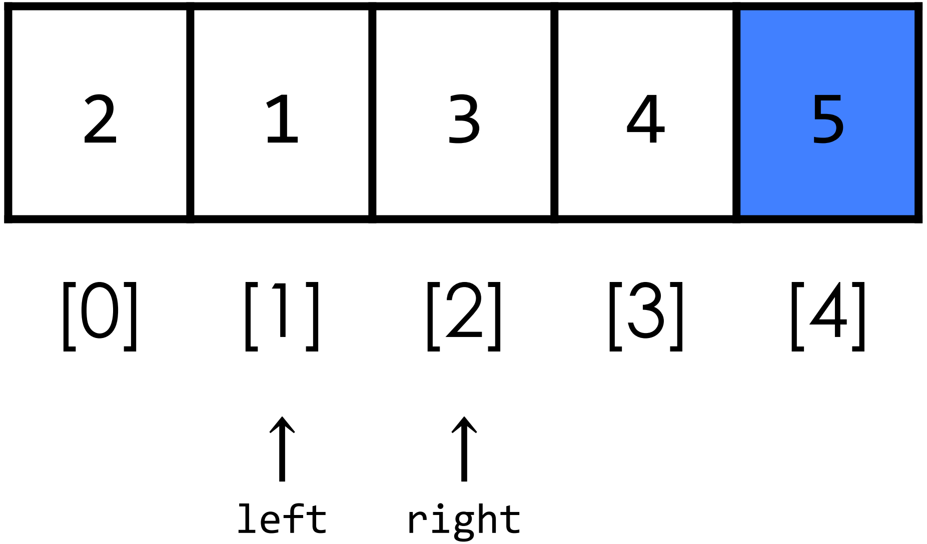
2	3	1	4	5
[0]	[1]	[2]	[3]	[4]
↑ left	↑ right			

bubble sort



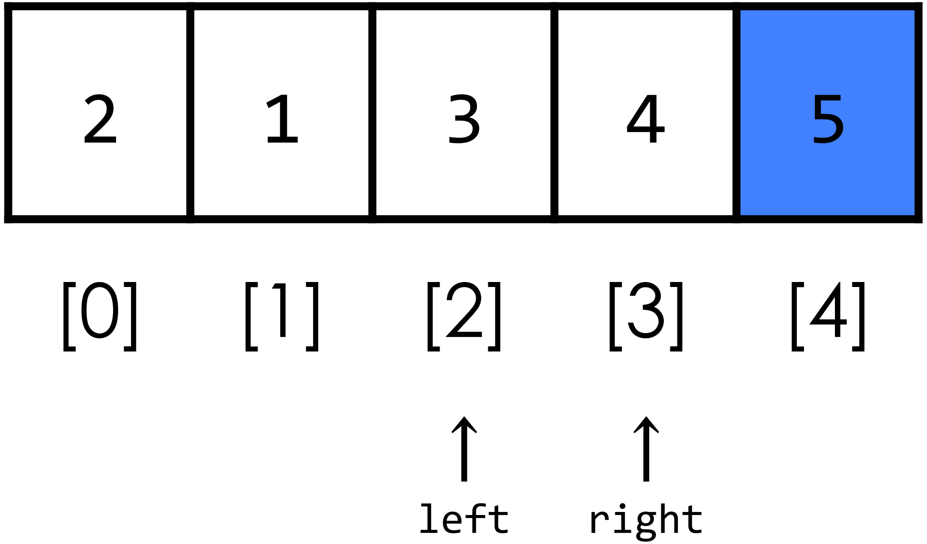
2	3	1	4	5
[0]	[1]	[2]	[3]	[4]

bubble sort



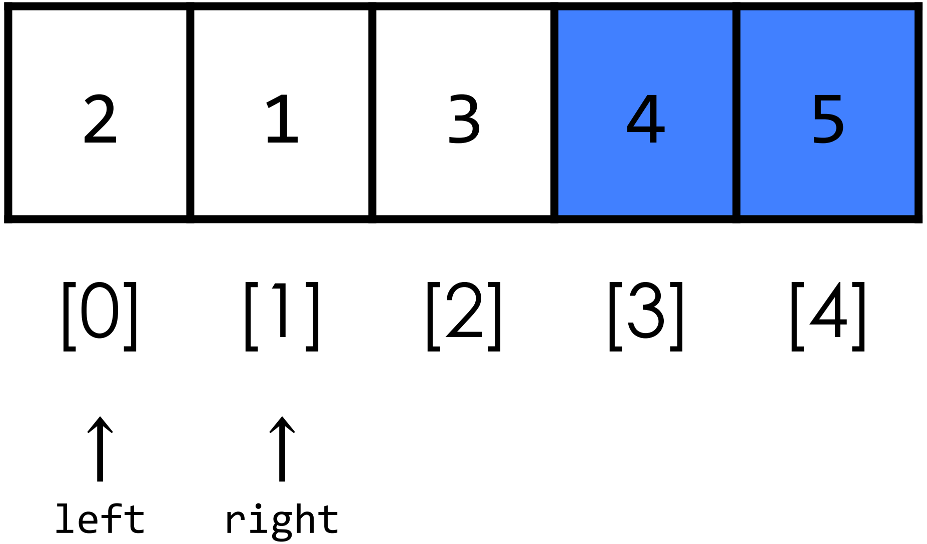
2	1	3	4	5
[0]	[1]	[2]	[3]	[4]
	↑	↑		
	left	right		

bubble sort



2	1	3	4	5
[0]	[1]	[2]	[3]	[4]
		left	right	

bubble sort

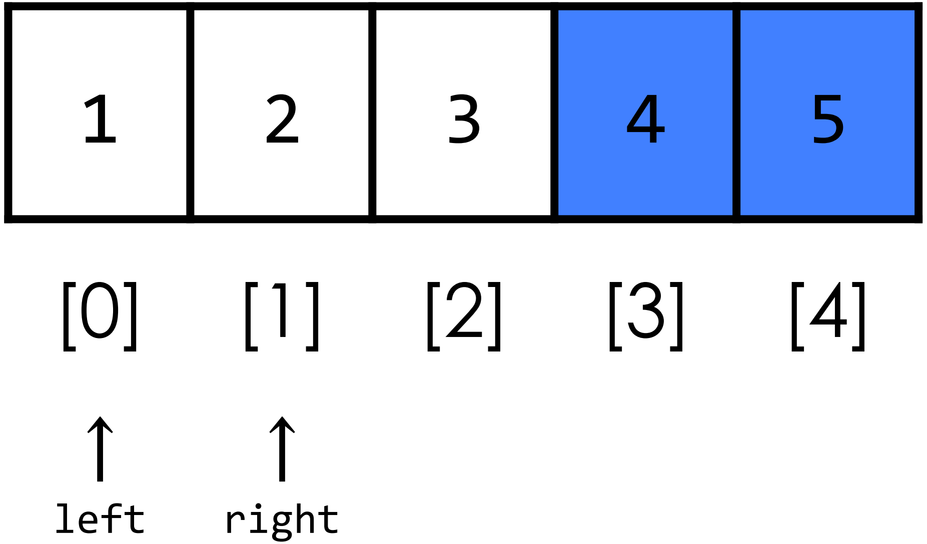


2 1 3 4 5

[0] [1] [2] [3] [4]

left right

bubble sort

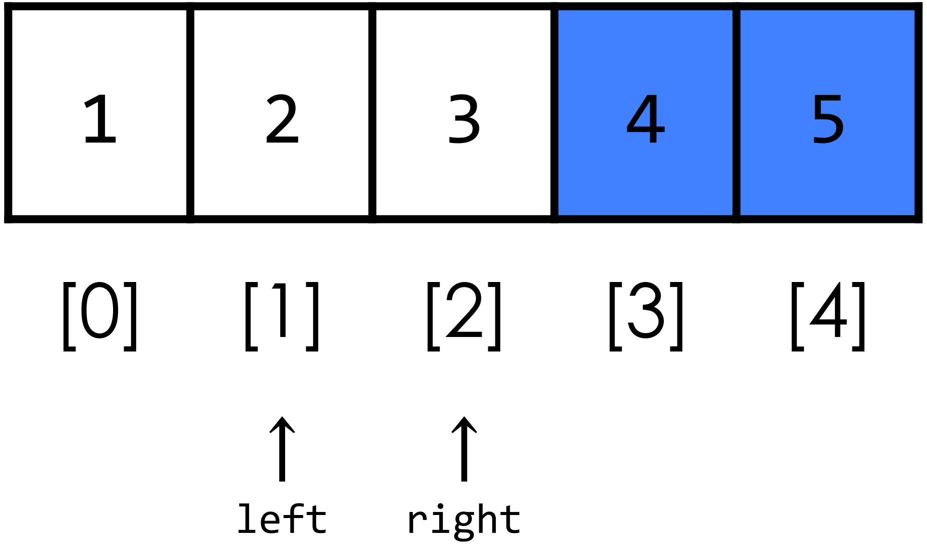


1 2 3 4 5

[0] [1] [2] [3] [4]

left right

bubble sort

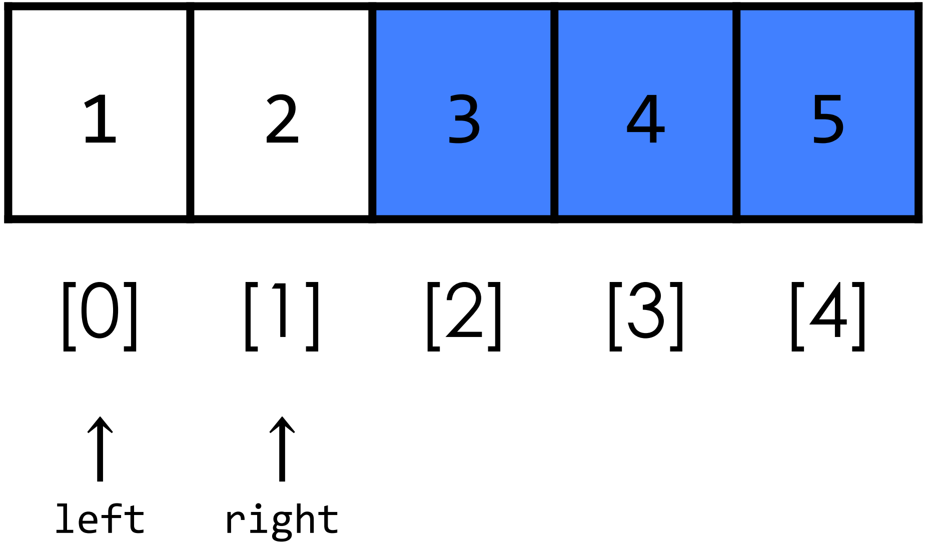


1 2 3 4 5

[0] [1] [2] [3] [4]

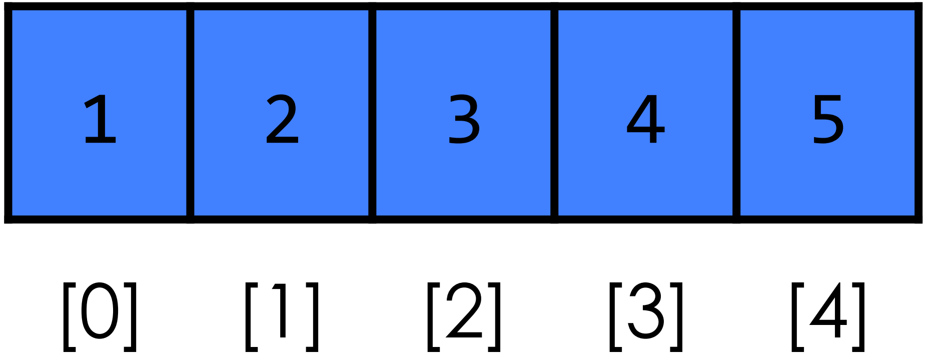
↑ left ↑ right

bubble sort



1	2	3	4	5
[0]	[1]	[2]	[3]	[4]
left	right			

bubble sort



1	2	3	4	5
[0]	[1]	[2]	[3]	[4]

TODO

- ☑ `search`
- ☑ `sort`

this was find