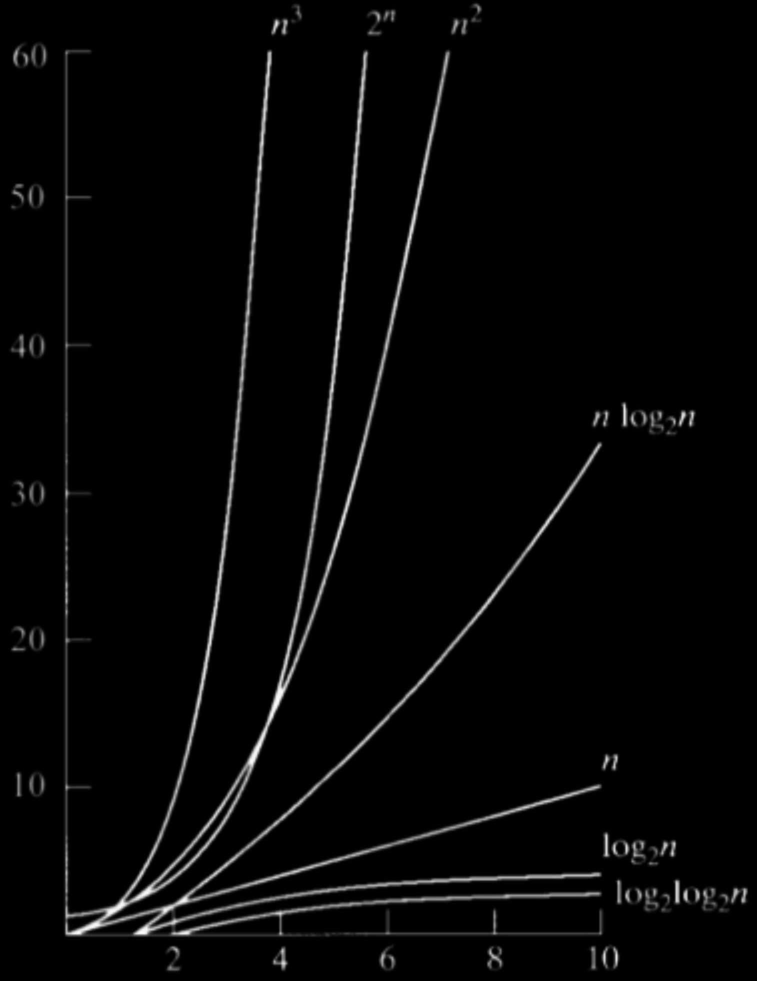

n^3
2^n
n^2
$n \log_2 n$
n
$\log_2 n$
$\log_2 \log_2 n$
60
50
40
30
20
10
2
4
6
8
10

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$

Sorting Algorithms

Bubble Sort

Insertion Sort

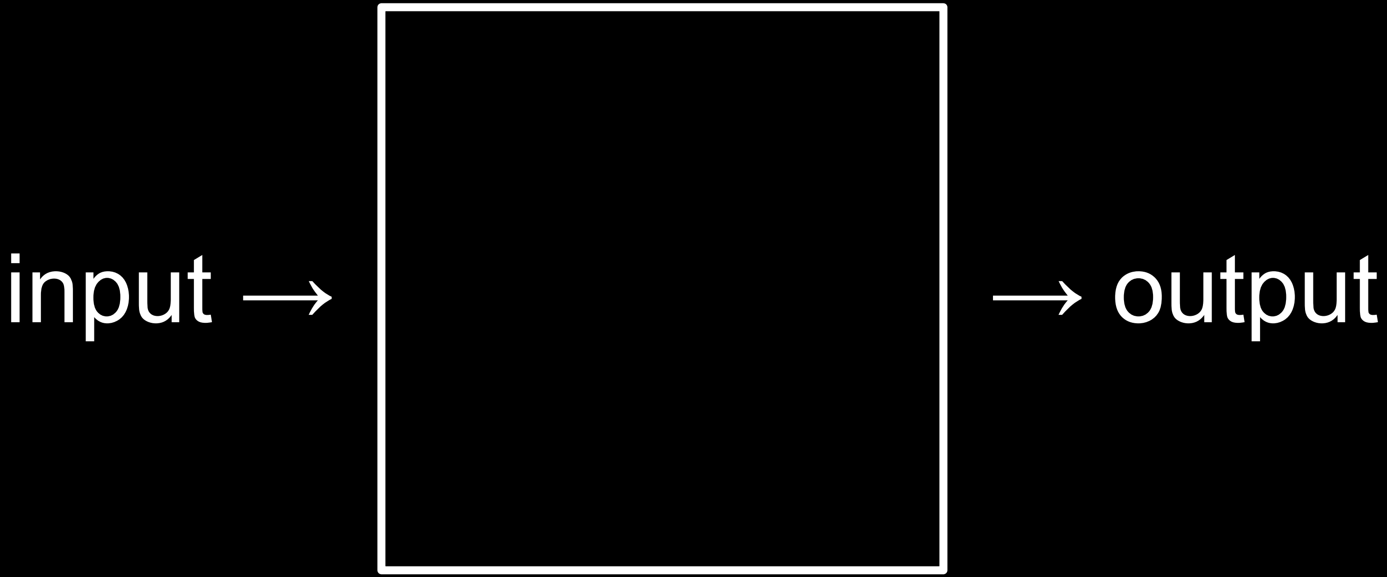
Shell Sort

Quick Sort

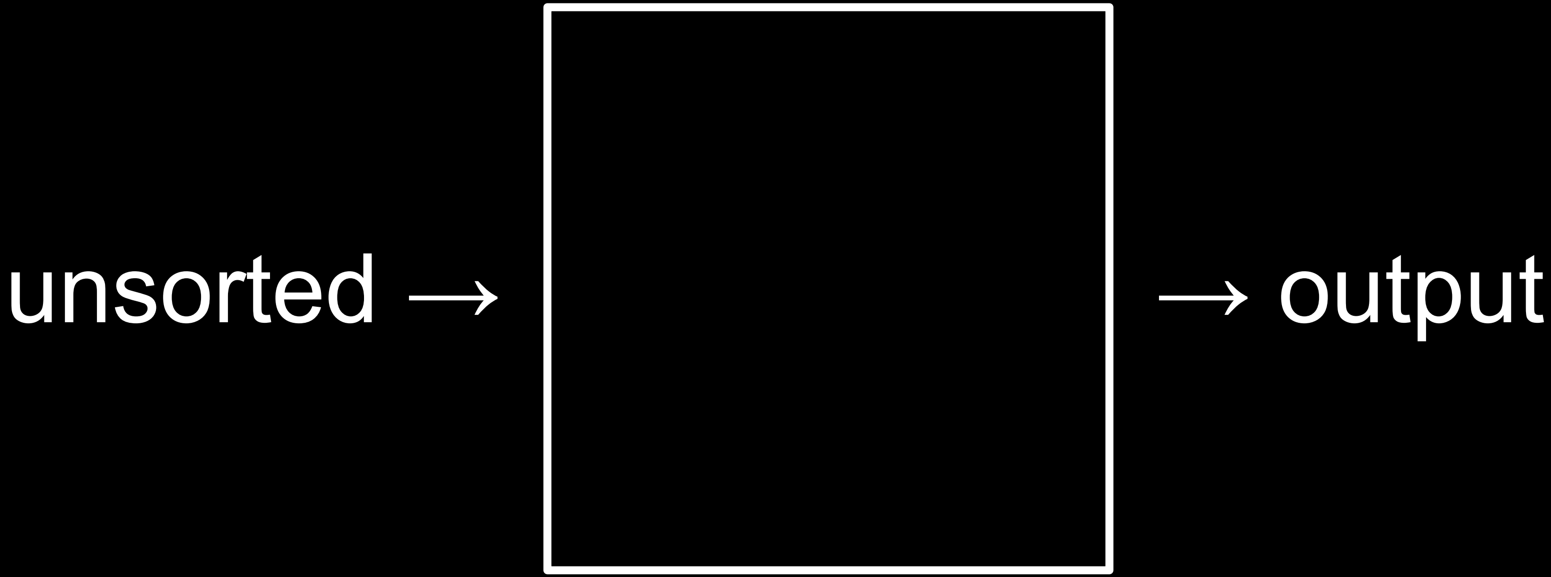
Merge Sort

Heap Sort

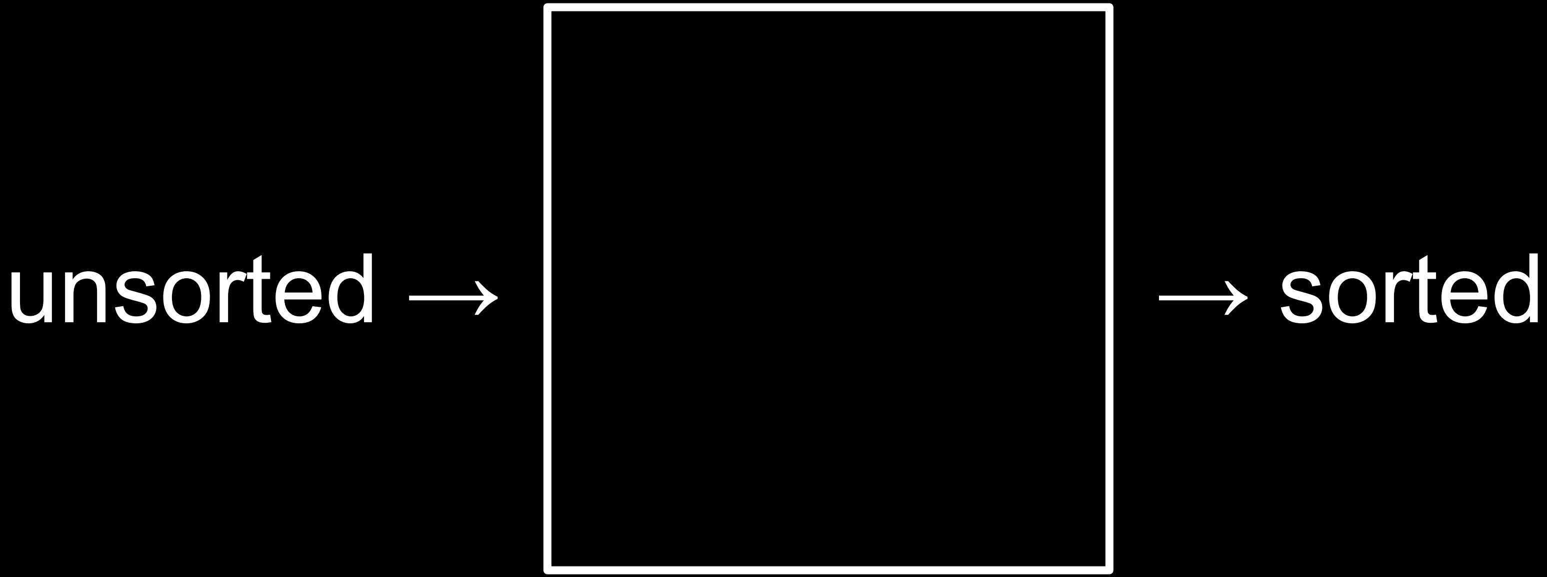
Selection Sort



input → → output

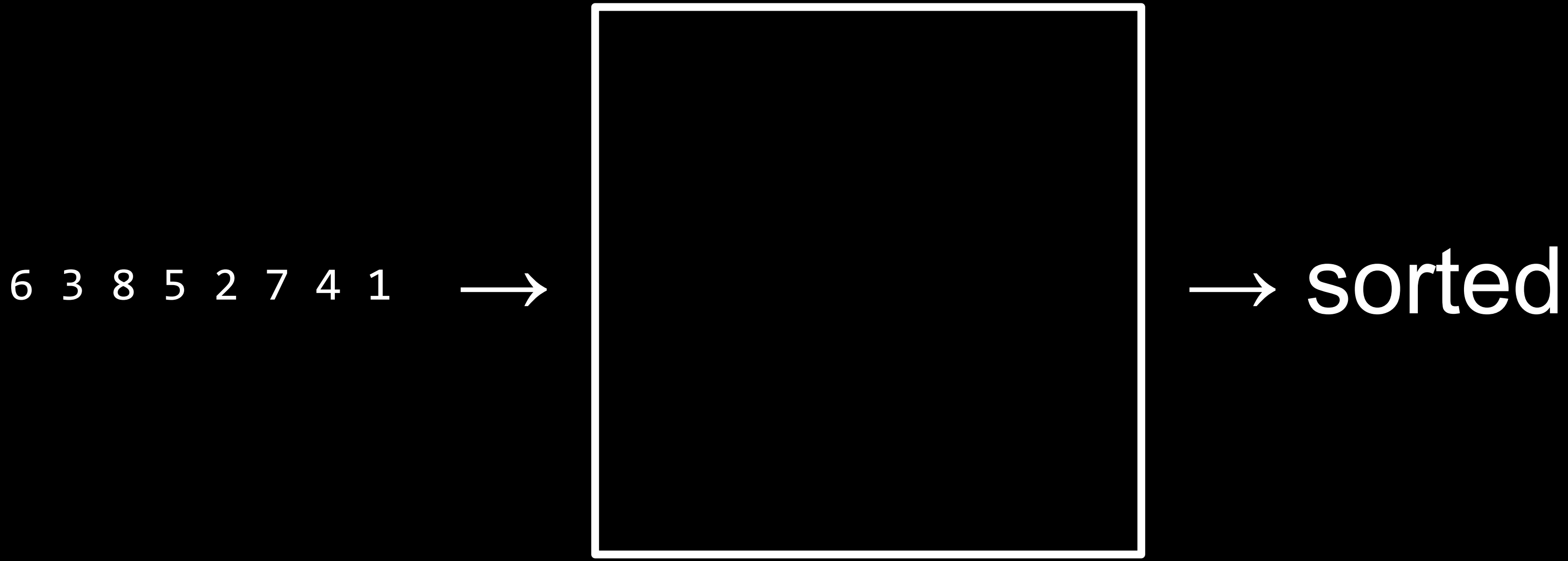


unsorted → → output



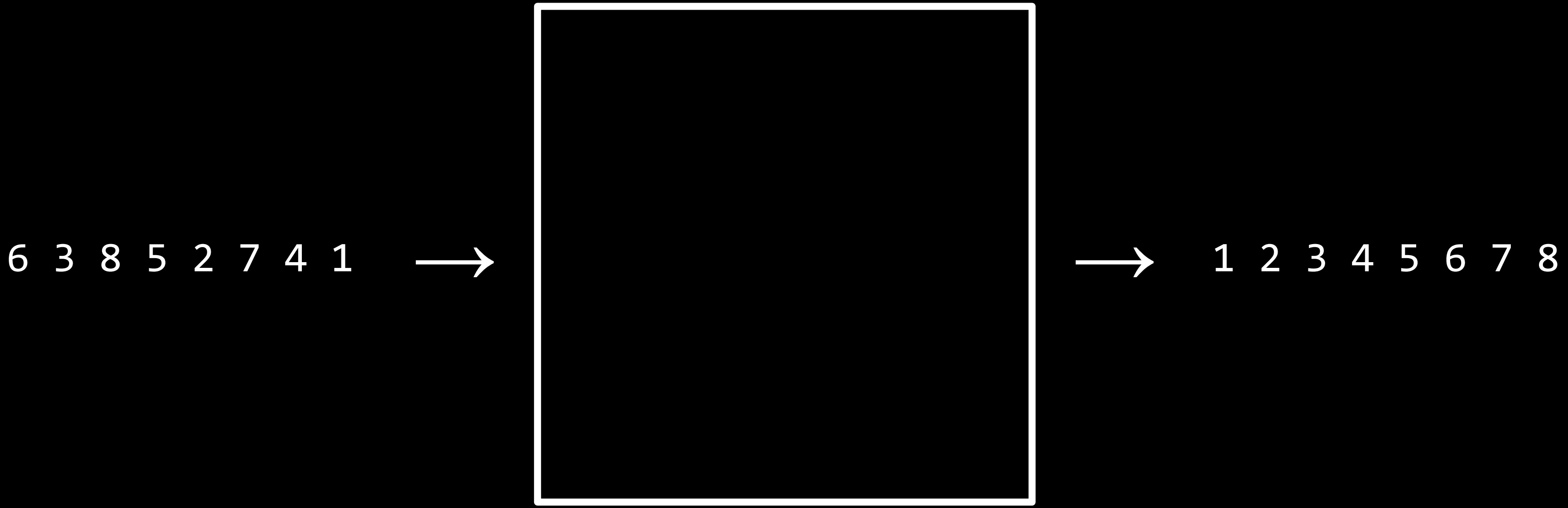
unsorted →

→ sorted



6 3 8 5 2 7 4 1 →

→ sorted



6 3 8 5 2 7 4 1 → → 1 2 3 4 5 6 7 8

selection sort

6 3 8 5 2 7 4 1

```
For i from 0 to n-1
    Find smallest item between i'th item and last item
    Swap smallest item with i'th item
```

n

$n + (n - 1)$

$n + (n - 1) + (n - 2)$

$n + (n - 1) + (n - 2) + ... + 1$

$n + (n - 1) + (n - 2) + ... + 1$

$n(n + 1)/2$

$n + (n – 1) + (n – 2) + ... + 1$

$n(n + 1)/2$

$(n^2 + \mathrm{n})/2$

$n + (n - 1) + (n - 2) + ... + 1$

$n(n + 1)/2$

$(n^2 + \mathrm{n})/2$

$n^2/2 + n/2$

$n + (n - 1) + (n - 2) + ... + 1$

$n(n + 1)/2$

$(n^2 + \mathrm{n})/2$

$n^2/2 + n/2$

$O(n^2)$

$O(n^2)$	
$O(n \log n)$	
$O(n)$	linear search
$O(\log n)$	binary search
$O(1)$	

$O(n^2)$	selection sort
$O(n \log n)$	
$O(n)$	linear search
$O(\log n)$	binary search
$O(1)$	

```
For i from 0 to n-1
    Find smallest item between i'th item and last item
    Swap smallest item with i'th item
```

$\Omega(n^2)$	
$\Omega(n \log n)$	
$\Omega(n)$	
$\Omega(\log n)$	
$\Omega(1)$	linear search, binary search

$\Omega(n^2)$	selection sort
$\Omega(n \log n)$	
$\Omega(n)$	
$\Omega(\log n)$	
$\Omega(1)$	linear search, binary search

bubble sort

6 3 8 5 2 7 4 1

Bubble Sort

An idea of this algorithm is to move higher valued elements generally towards the right and lower elements generally towards the left.

In pseudocode:

```
Set swap counter to non-zero value
Repeat until the swap counter is 0:
    Reset swap counter to 0
    Look at each adjacent pair
        If the adjacent elements are not in order, swap them and add one to swap counter.
```

```
Repeat until sorted
    For i from 0 to n-2
        If i'th and i+1'th elements out of order
            Swap them
```

```
Repeat n-1 times
    For i from 0 to n-2
        If i'th and i+1'th elements out of order
            Swap them
```

$(n - 1) \times (n - 1)$

$$(n - 1) \times (n - 1)$$

$$n^2 - 1n - 1n + 1$$

$(n - 1) \times (n - 1)$

$n^2 - 1n - 1n + 1$

$n^2 - 2n + 1$

$(n - 1) \times (n - 1)$

$n^2 - 1n - 1n + 1$

$n^2 - 2n + 1$

$O(n^2)$

$O(n^2)$	selection sort
$O(n \log n)$	
$O(n)$	linear search
$O(\log n)$	binary search
$O(1)$	

$O(n^2)$	selection sort, bubble sort
$O(n \log n)$	
$O(n)$	linear search
$O(\log n)$	binary search
$O(1)$	

```
Repeat n-1 times
    For i from 0 to n-2
        If i'th and i+1'th elements out of order
            Swap them
    If no swaps
        Quit
```

Bubble Sort Demo

Swap Counter

-1

6	5	8	2	4	1

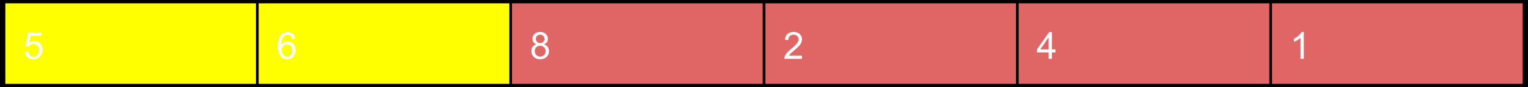
Swap Counter

0

6	5	8	2	4	1

Swap Counter

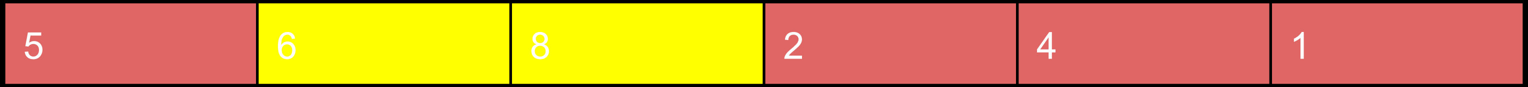
1



5	6	8	2	4	1

Swap Counter

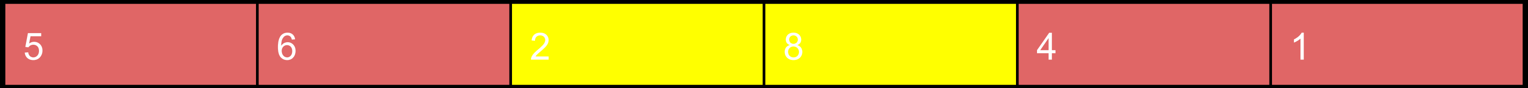
1



5	6	8	2	4	1

Swap Counter

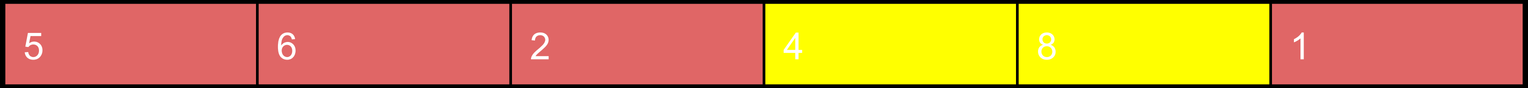
2



5	6	2	8	4	1

Swap Counter

3



5	6	2	4	8	1

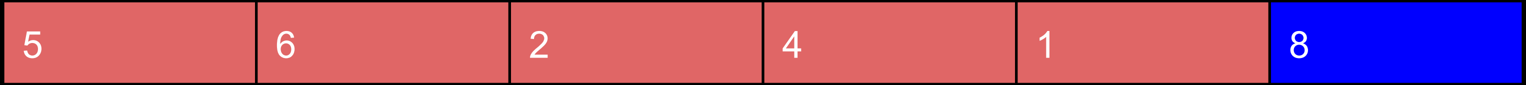
Swap Counter

4

5	6	2	4	1	8

Swap Counter

4



5	6	2	4	1	8

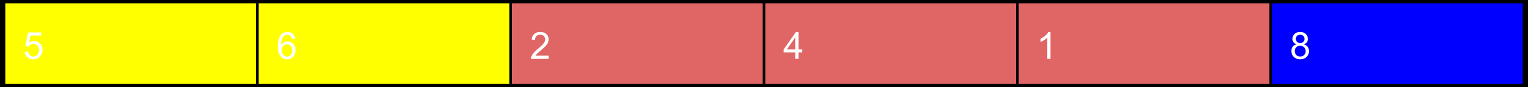
Swap Counter

0

5	6	2	4	1	8

Swap Counter

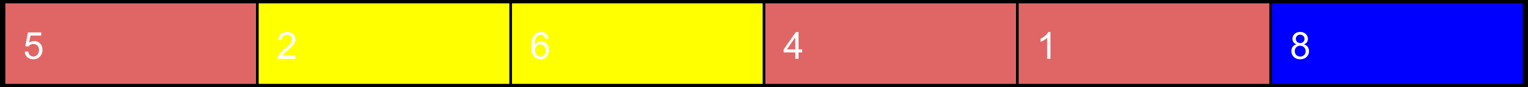
0



5	6	2	4	1	8

Swap Counter

1



5	2	6	4	1	8

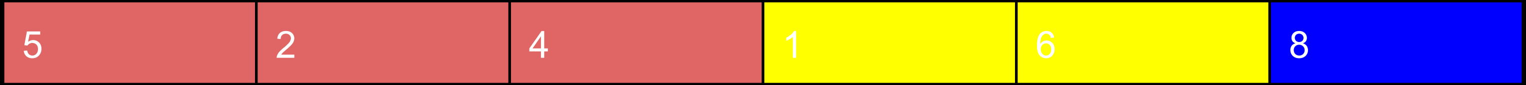
Swap Counter

2

5	2	4	6	1	8

Swap Counter

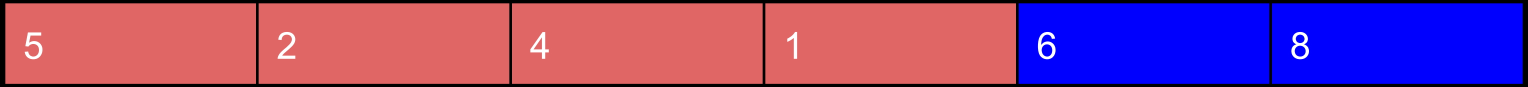
3



5	2	4	1	6	8

Swap Counter

3



5 2 4 1 6 8

Swap Counter

0

5	2	4	1	6	8

Swap Counter

1

2	5	4	1	6	8

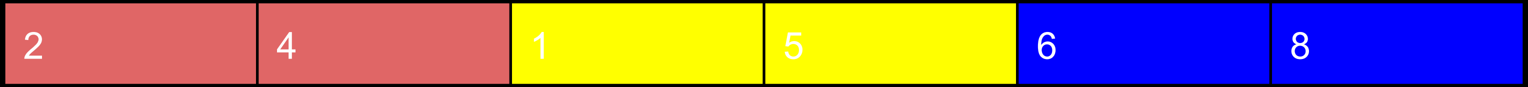
Swap Counter

2

2	4	5	1	6	8

Swap Counter

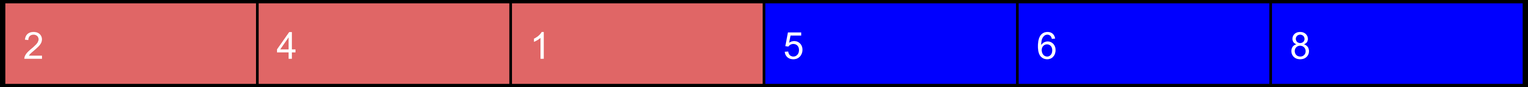
3



2	4	1	5	6	8

Swap Counter

3



2	4	1	5	6	8

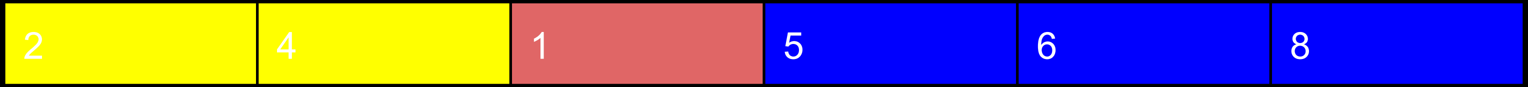
Swap Counter

0

2	4	1	5	6	8

Swap Counter

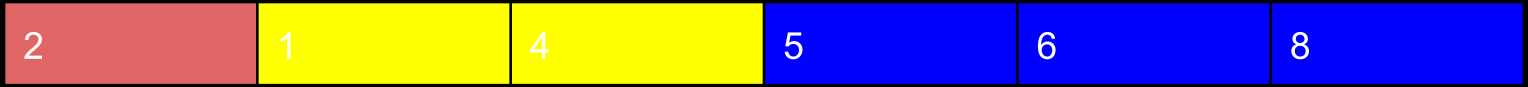
0



2	4	1	5	6	8

Swap Counter

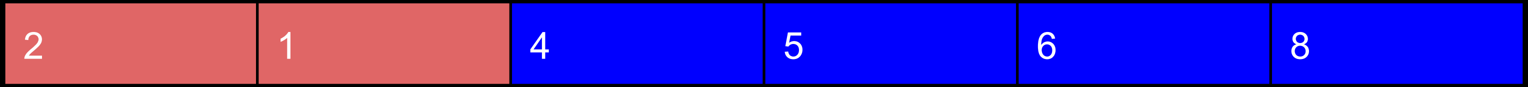
1



2	1	4	5	6	8

Swap Counter

1



2	1	4	5	6	8

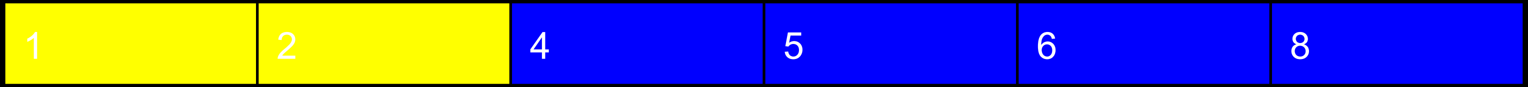
Swap Counter

0

2	1	4	5	6	8

Swap Counter

1



1	2	4	5	6	8

Swap Counter

1

1	2	4	5	6	8

Swap Counter

0

1	2	4	5	6	8

$\Omega(n^2)$	selection sort
$\Omega(n \log n)$	
$\Omega(n)$	
$\Omega(\log n)$	
$\Omega(1)$	linear search, binary search

$\Omega(n^2)$	selection sort
$\Omega(n \log n)$	
$\Omega(n)$	bubble sort(When array is sorted)
$\Omega(\log n)$	
$\Omega(1)$	linear search, binary search