MÉTODOS DE ORDENAMIENTO

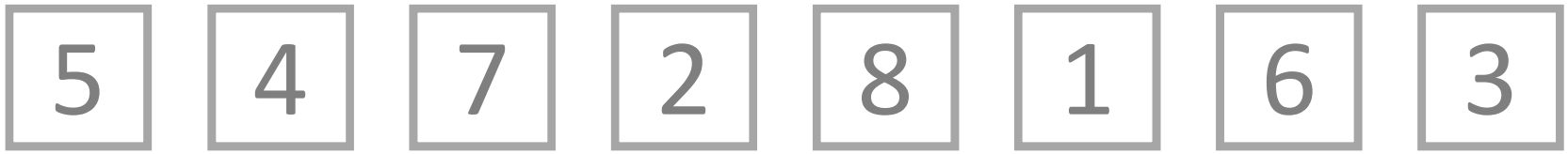
Bubble Sort, Selection Sort, Insertion Sort

Elaborado por: Prof. Kiara A. Ottogalli F.

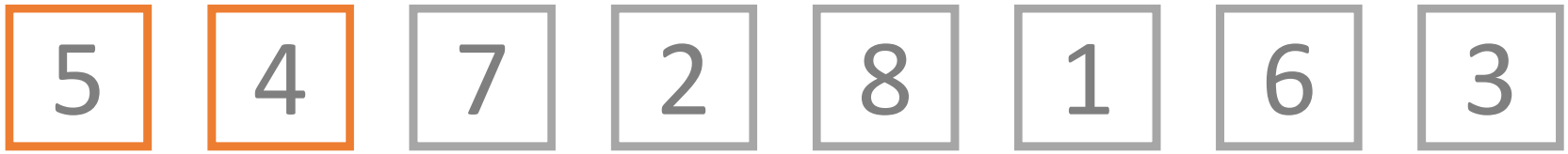
BUBBLE SORT

- Este algoritmo va moviendo los elementos mayores al final del arreglo.
- Para realizar sus movimientos se enfoca únicamente en pares de elementos adyacentes.
- Los elementos mas pequeños “burbujean” hacia el principio del arreglo.

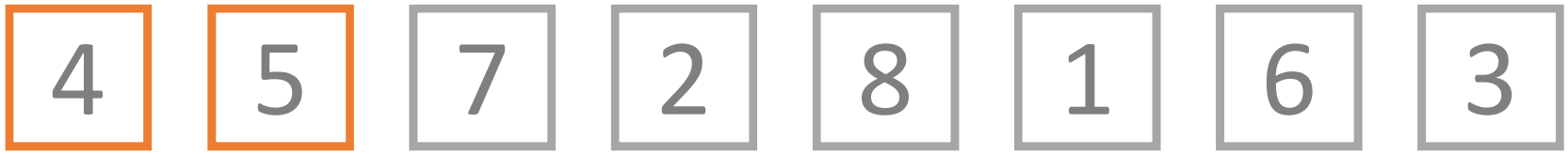
BUBBLE SORT



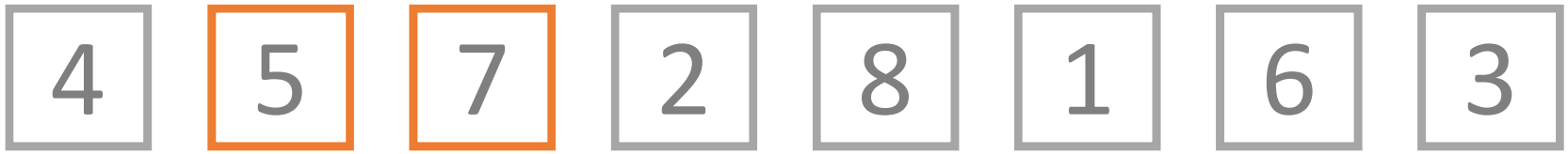
BUBBLE SORT



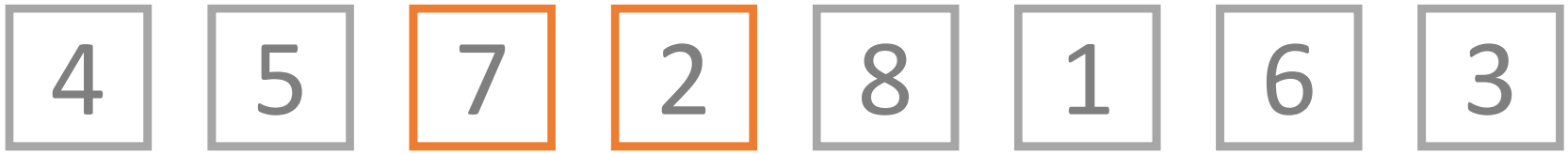
BUBBLE SORT



BUBBLE SORT



BUBBLE SORT



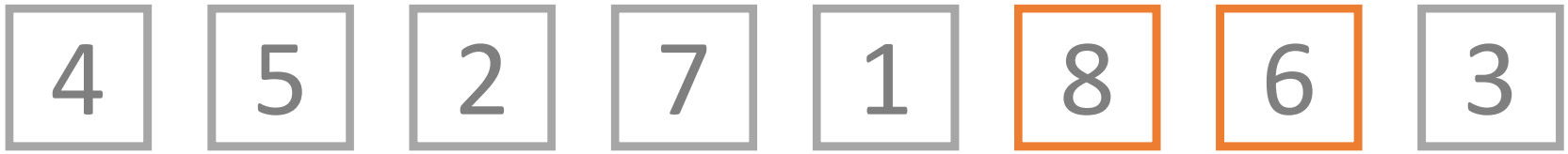
BUBBLE SORT

BUBBLE SORT

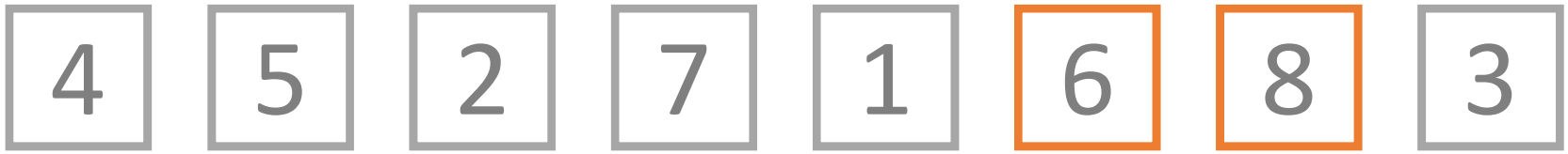
BUBBLE SORT

BUBBLE SORT

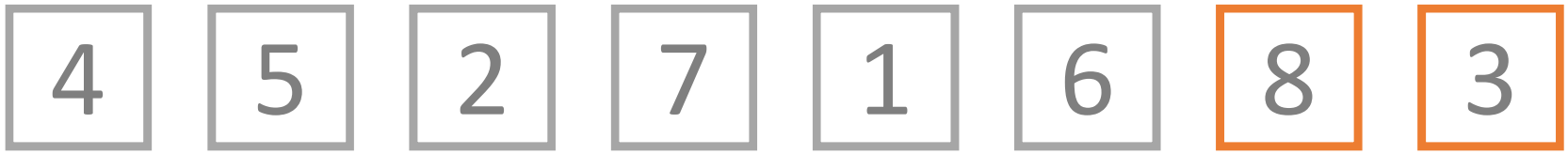
BUBBLE SORT



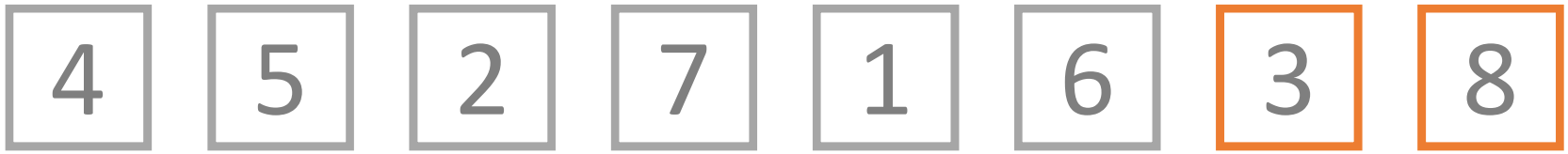
BUBBLE SORT



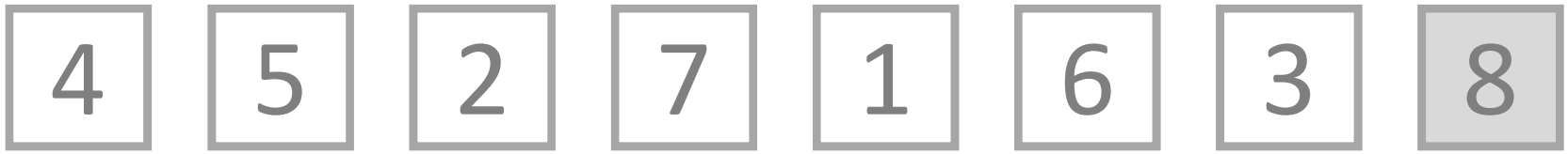
BUBBLE SORT



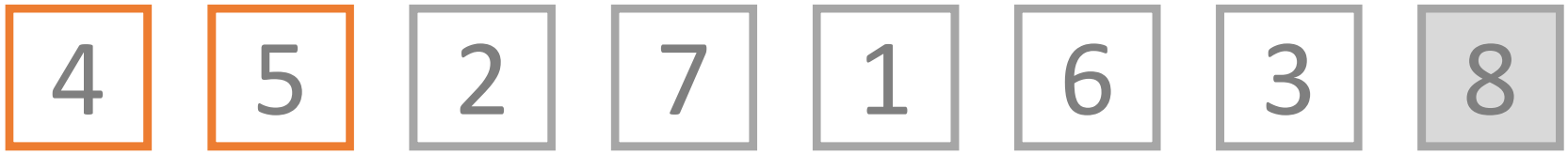
BUBBLE SORT



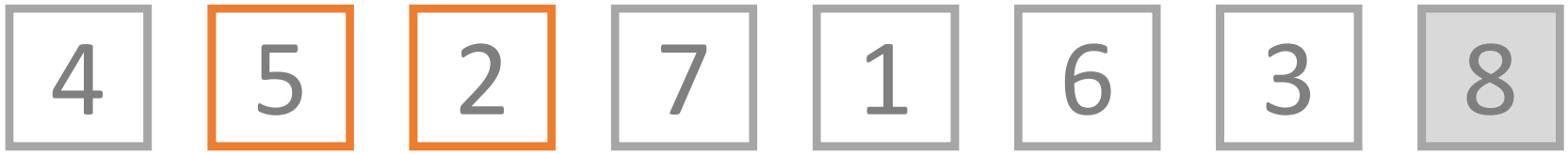
BUBBLE SORT



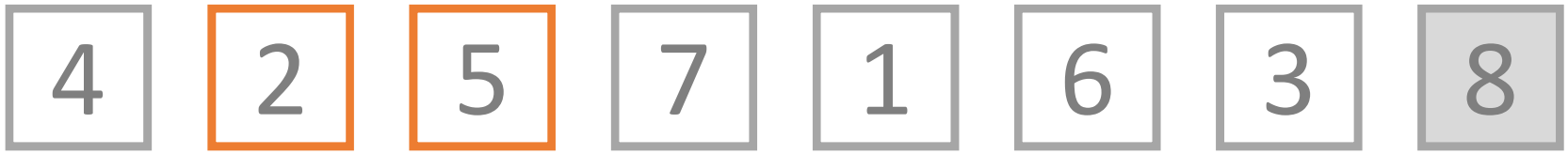
BUBBLE SORT



BUBBLE SORT



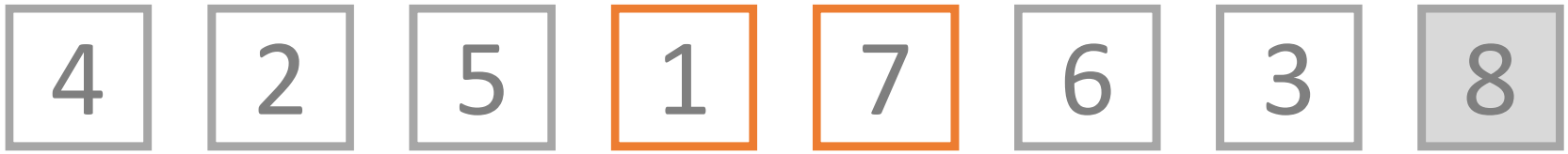
BUBBLE SORT



BUBBLE SORT

BUBBLE SORT

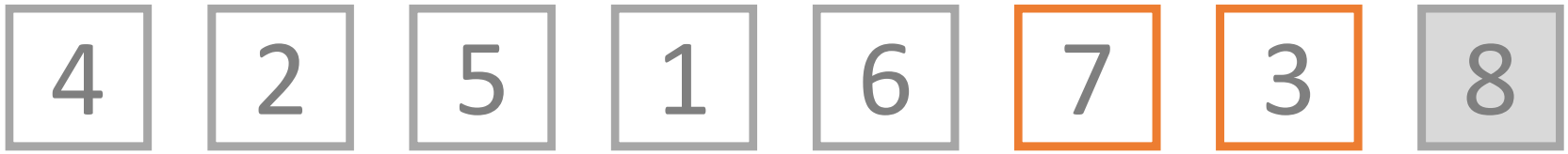
BUBBLE SORT



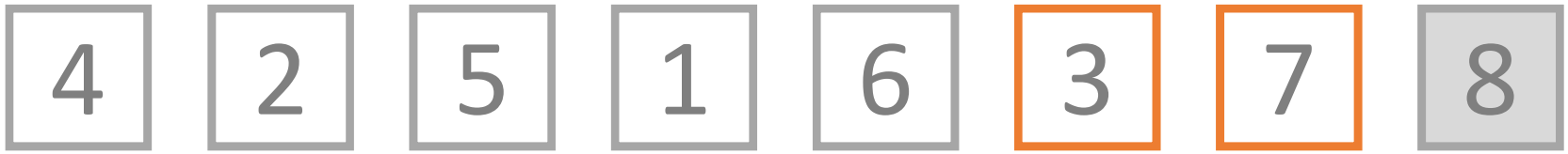
BUBBLE SORT

BUBBLE SORT

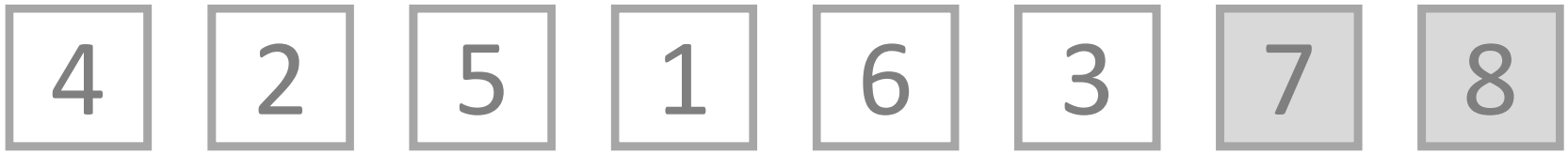
BUBBLE SORT



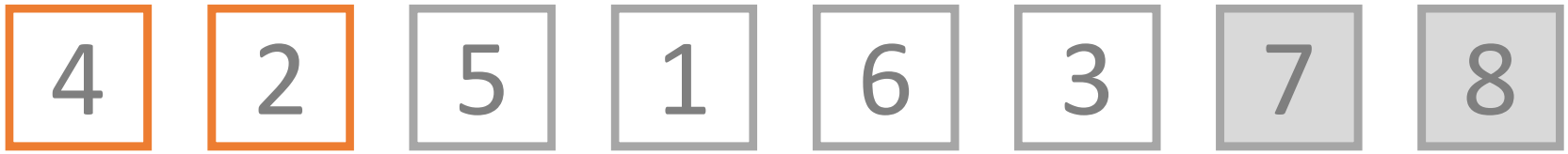
BUBBLE SORT



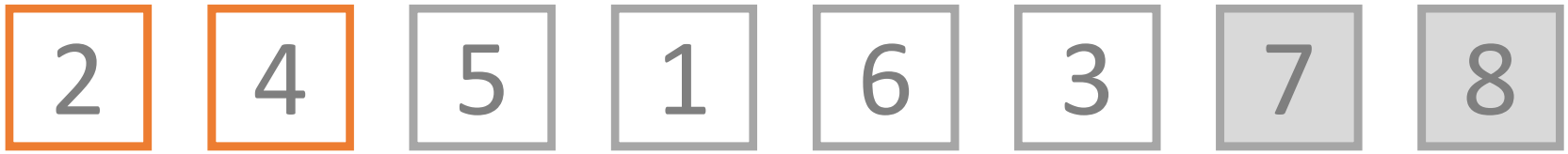
BUBBLE SORT



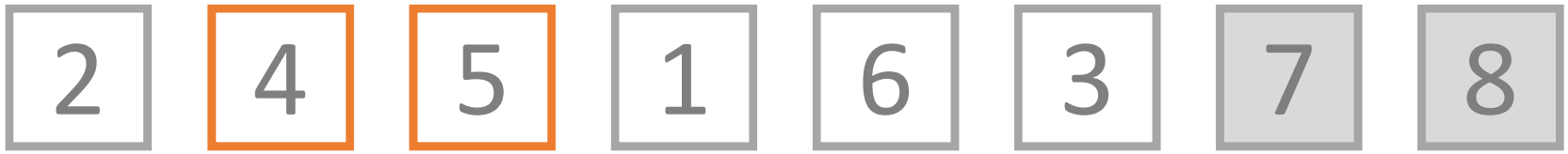
BUBBLE SORT



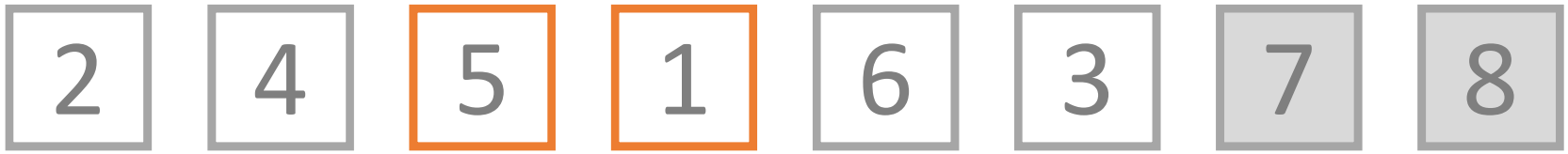
BUBBLE SORT



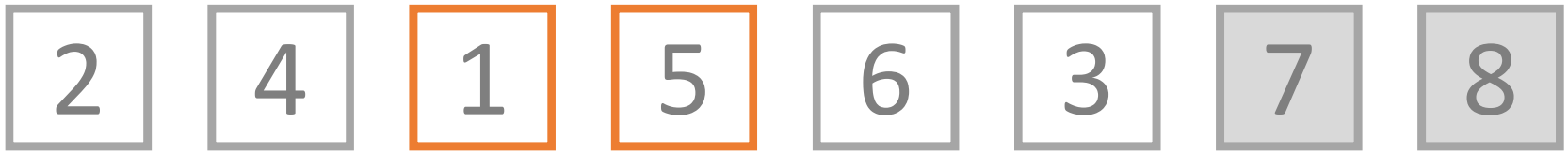
BUBBLE SORT



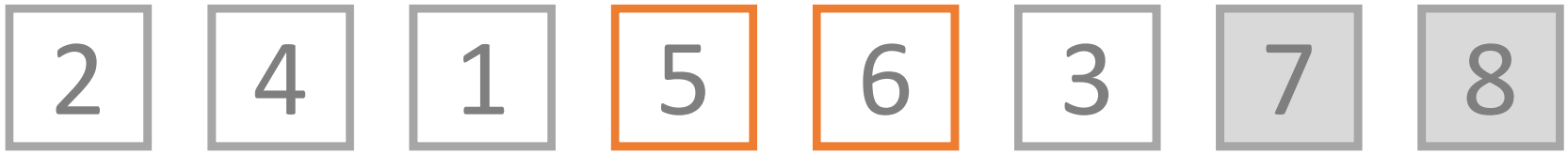
BUBBLE SORT



BUBBLE SORT



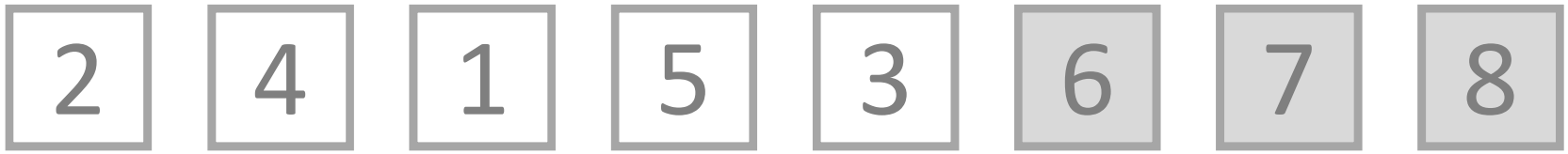
BUBBLE SORT



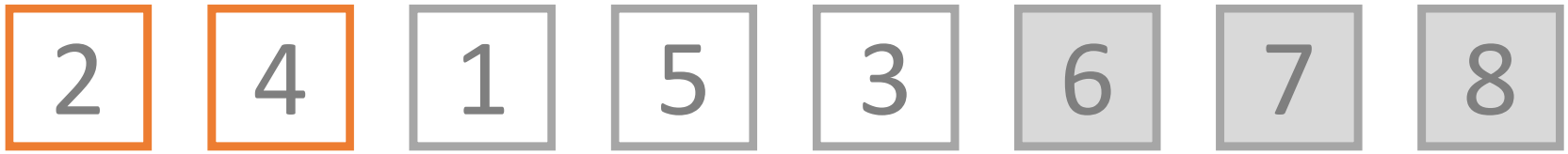
BUBBLE SORT

BUBBLE SORT

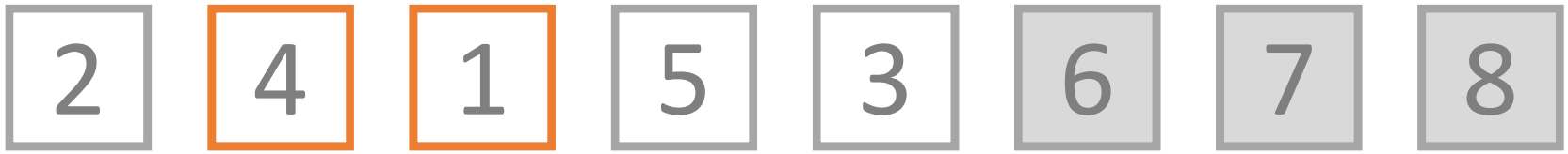
BUBBLE SORT



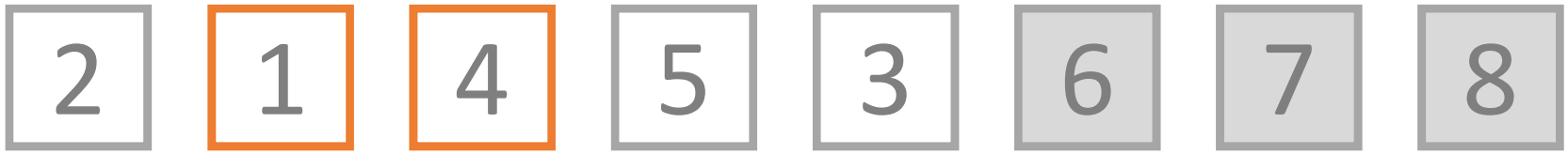
BUBBLE SORT



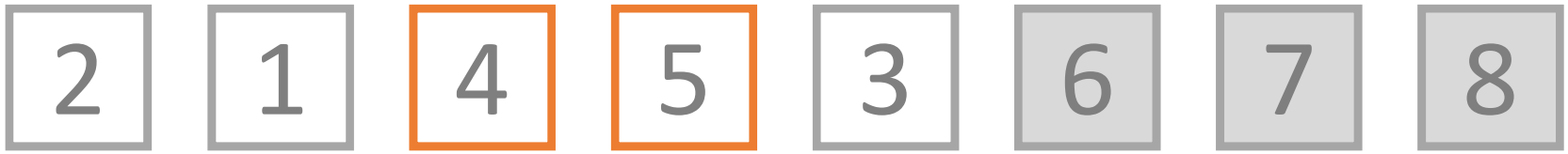
BUBBLE SORT



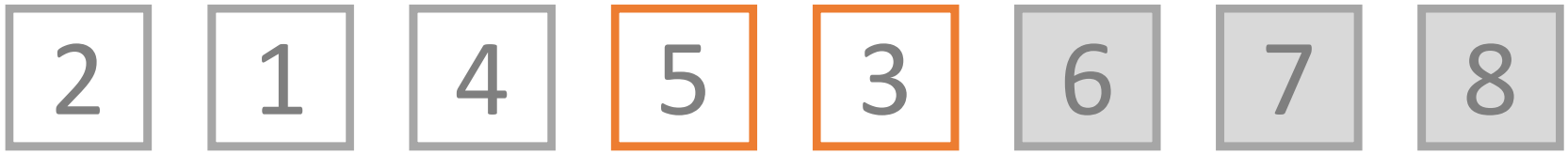
BUBBLE SORT



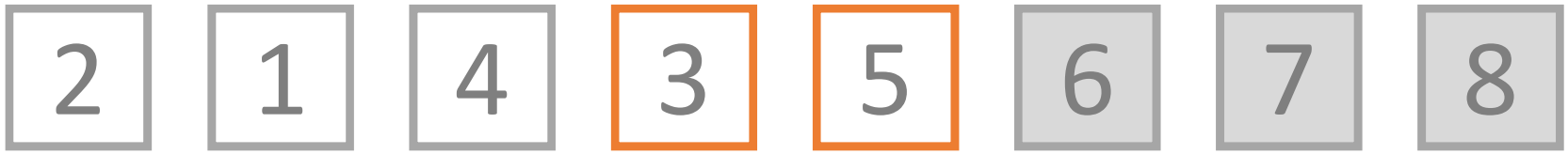
BUBBLE SORT



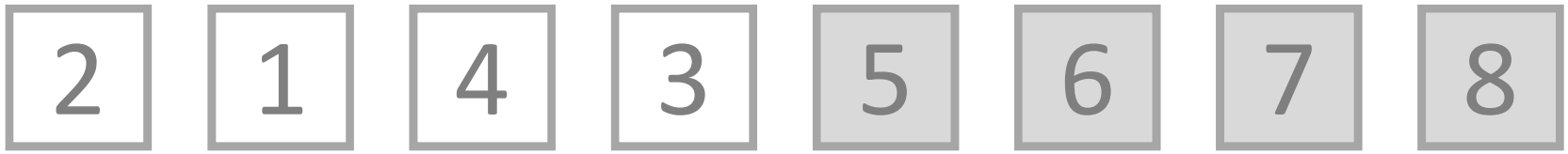
BUBBLE SORT



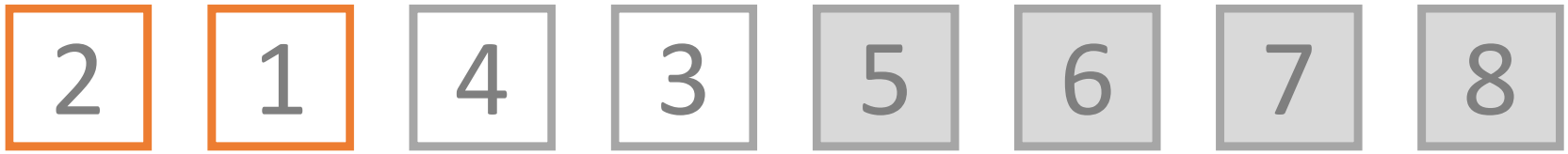
BUBBLE SORT



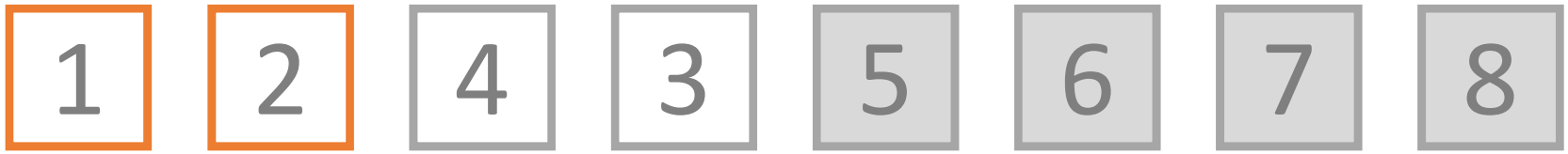
BUBBLE SORT



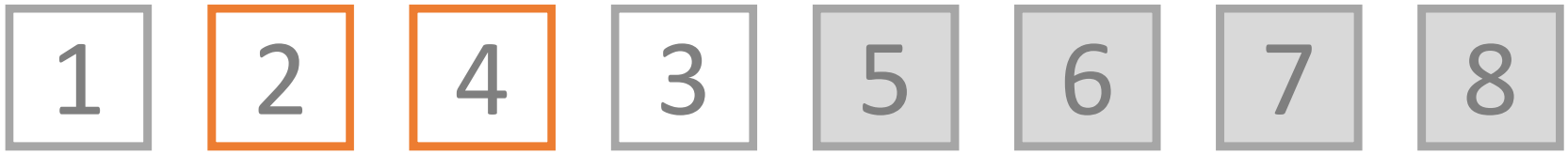
BUBBLE SORT



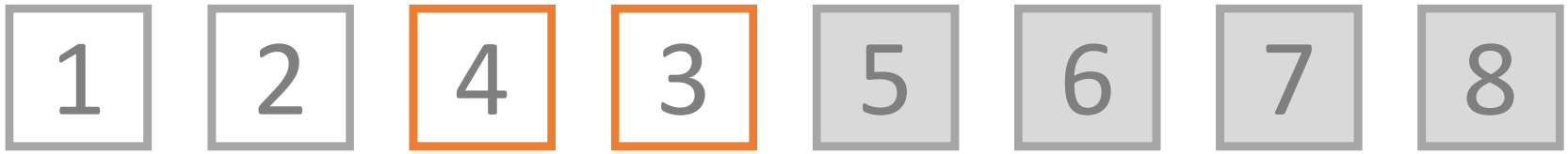
BUBBLE SORT



BUBBLE SORT

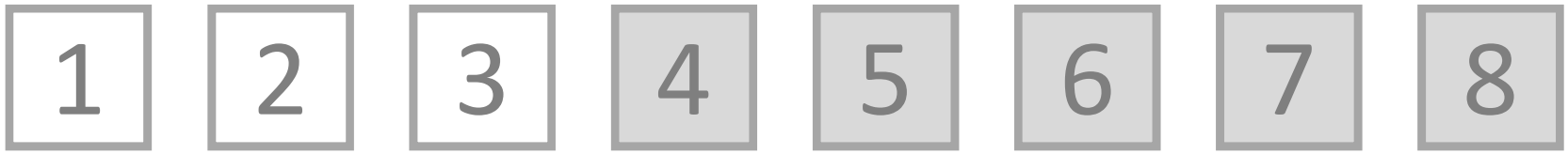


BUBBLE SORT

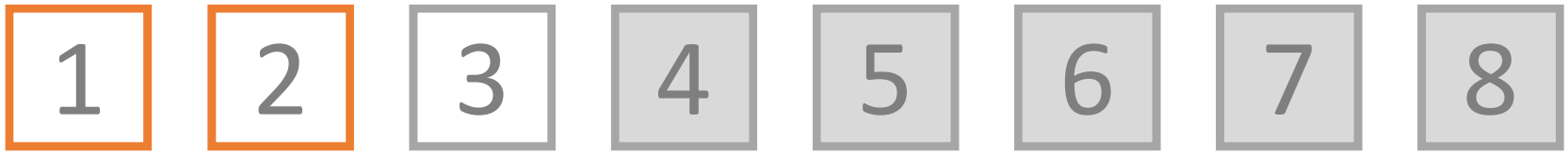


BUBBLE SORT

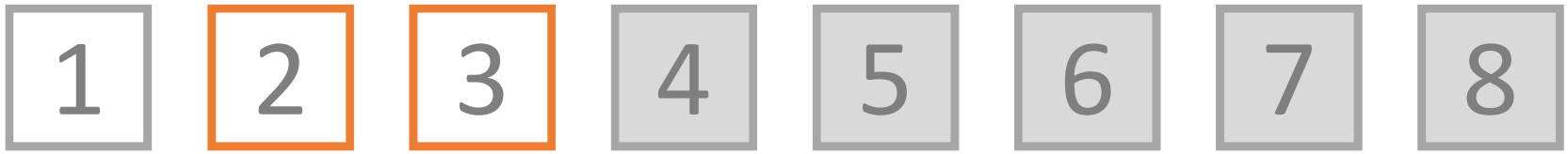
BUBBLE SORT



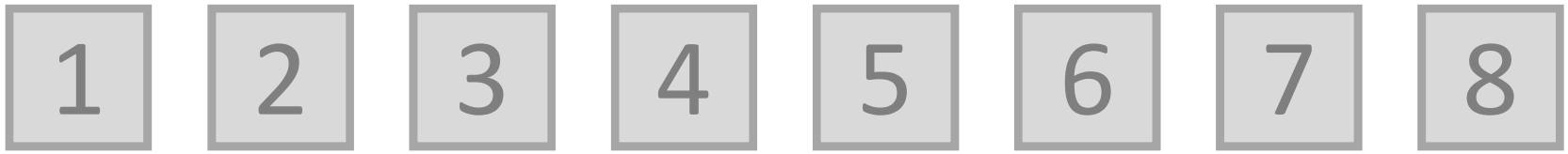
BUBBLE SORT



BUBBLE SORT



BUBBLE SORT



BUBBLE SORT

Optimización:

Nótese que si en la última pasada el algoritmo no hizo ningún intercambio, los elementos ya se encuentran ordenados y por lo tanto puede parar.

BUBBLE SORT

```
void bubblesort(int v[N], int n)
{
        int i, j, band;
        i = n - 1;
        band = 1;
        while(i > 0 && band)
        {
                band = 0;
                for(j = 0; j < i; j++)
                {
                        if(v[j] > v[j+1])
                        {
                                intercambio(&v[j], &v[j+1]);
                                band = 1;
                        }
                }
                i--;
        }
}
```

BUBBLE SORT

Peor Caso

$O(n^2)$

Cuando el arreglo está ordenado de manera inversa.

Mejor Caso

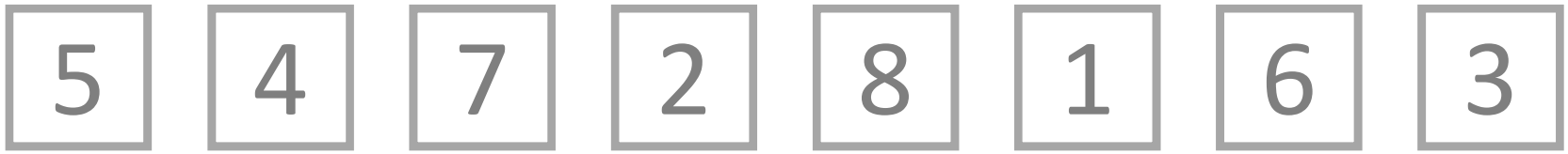
$\Omega(n)$

Cuando el arreglo ya se encuentra ordenado correctamente.

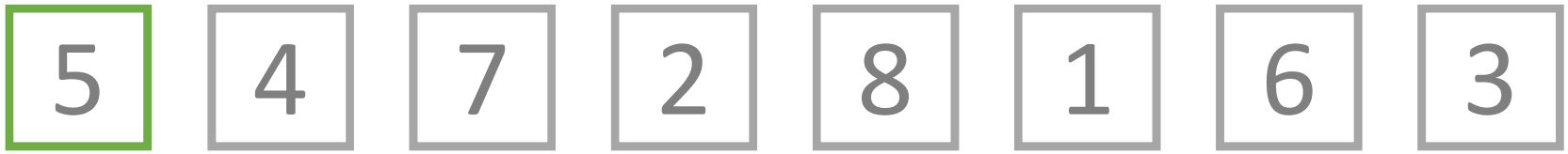
SELECTION SORT

- Este algoritmo encuentra el elemento mas pequeño del arreglo y lo intercambia con el elemento en la primera posición.
- Luego encuentra el segundo elemento mas pequeño y lo intercambia con el elemento en la segunda posición.
- Realiza esto hasta que el arreglo esté ordenado.

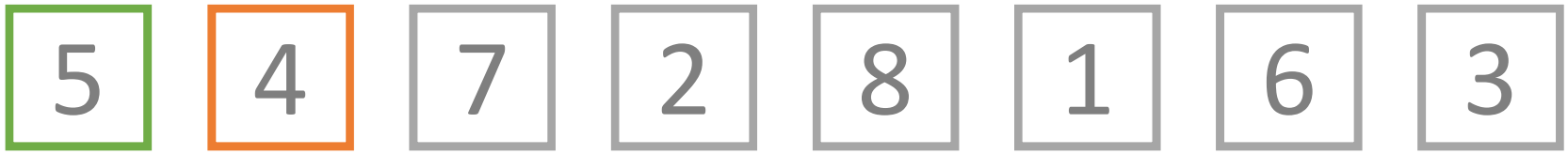
SELECTION SORT



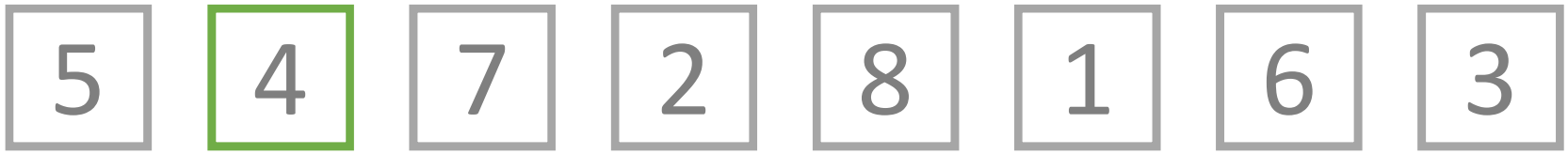
SELECTION SORT



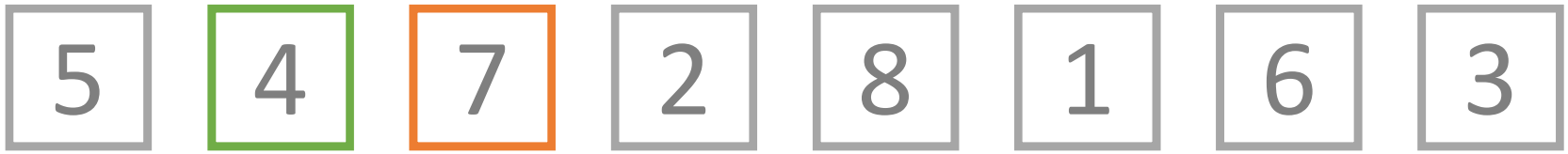
SELECTION SORT



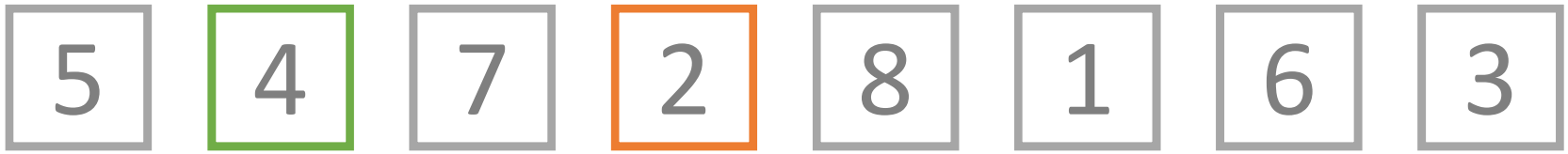
SELECTION SORT



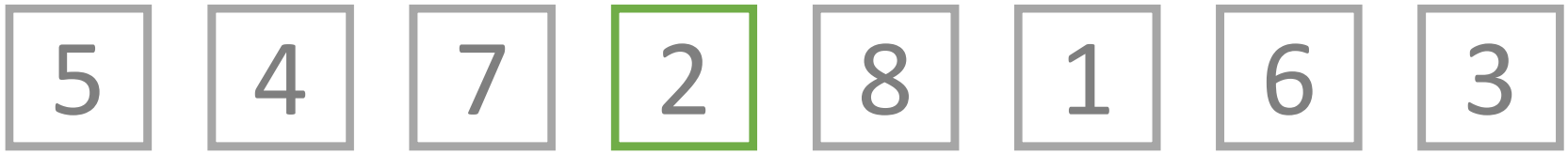
SELECTION SORT



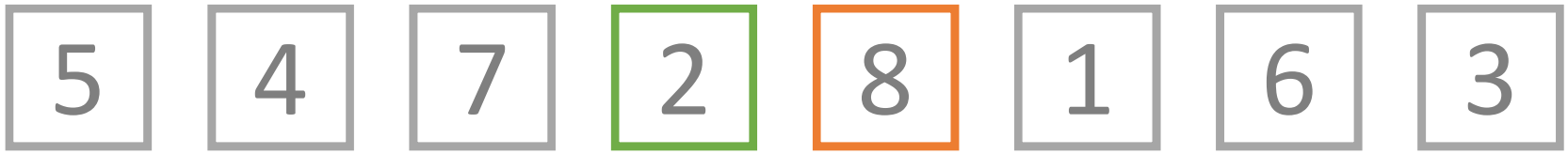
SELECTION SORT



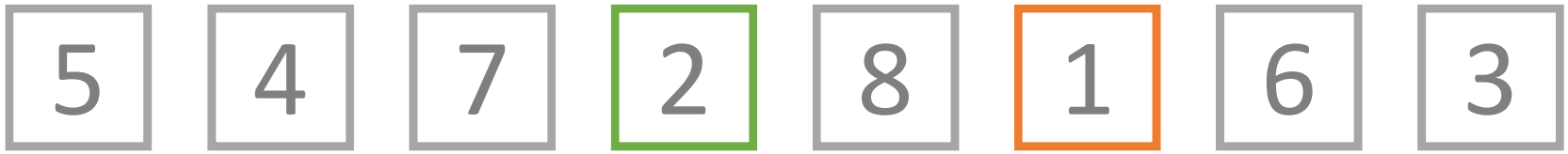
SELECTION SORT



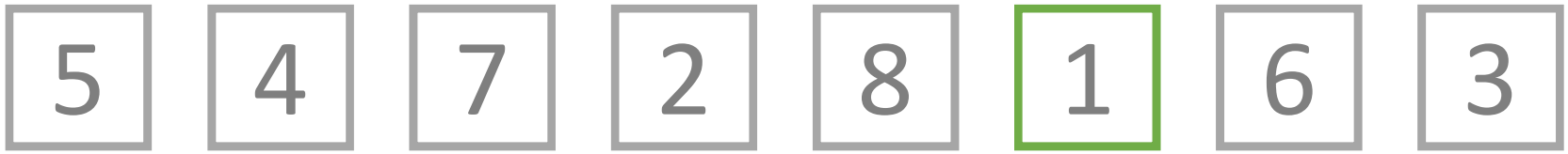
SELECTION SORT



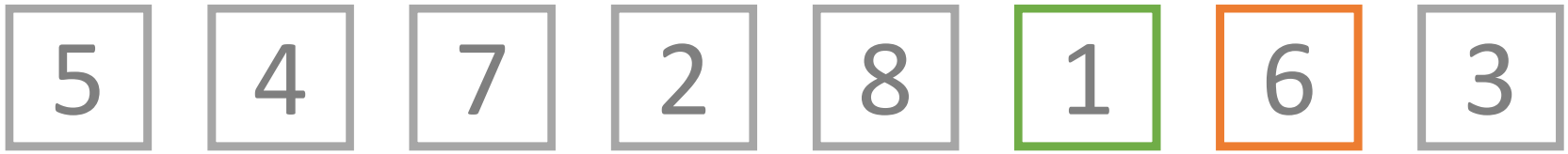
SELECTION SORT



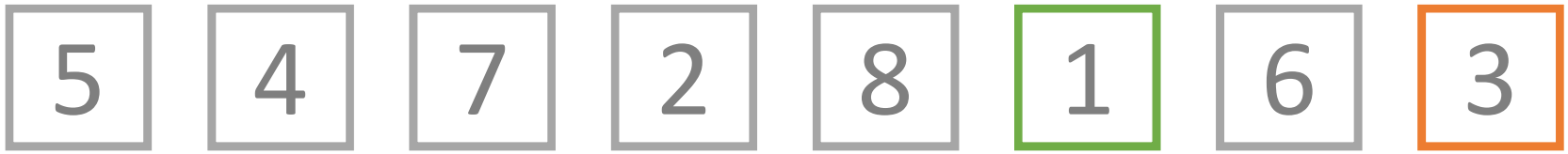
SELECTION SORT



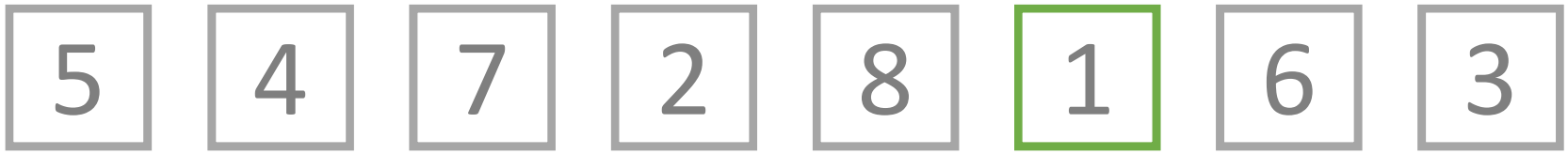
SELECTION SORT



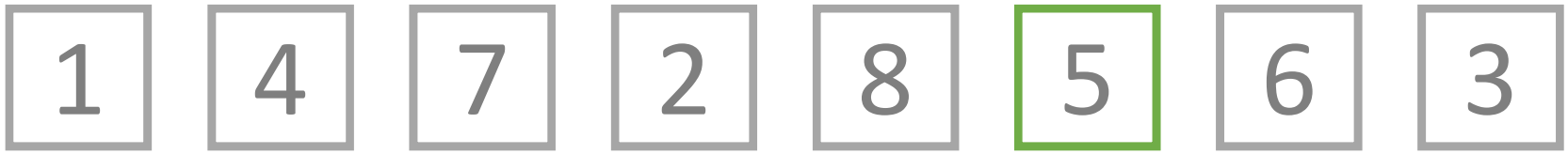
SELECTION SORT



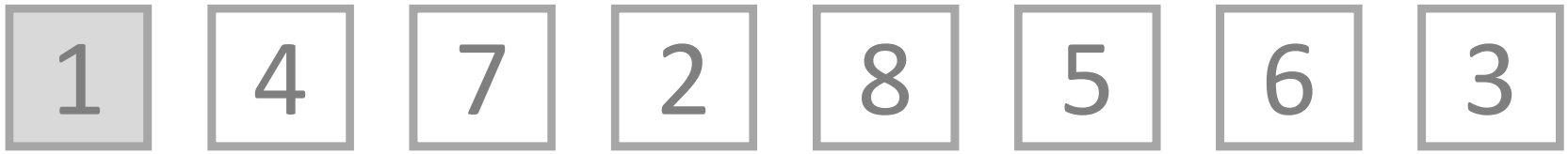
SELECTION SORT



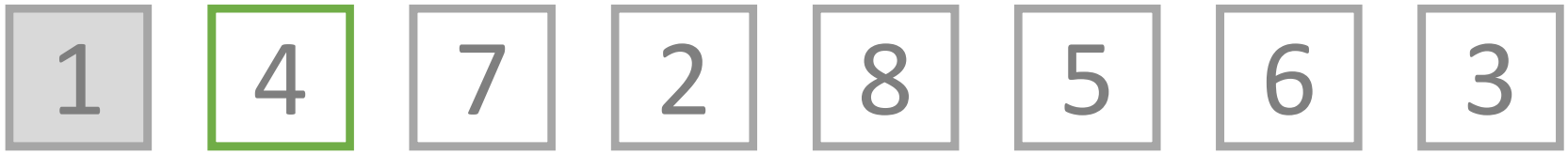
SELECTION SORT



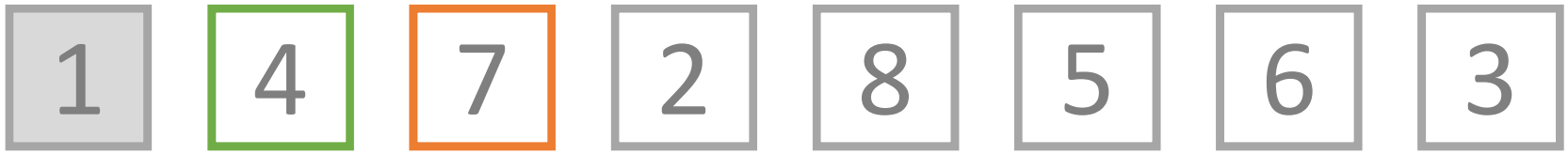
SELECTION SORT



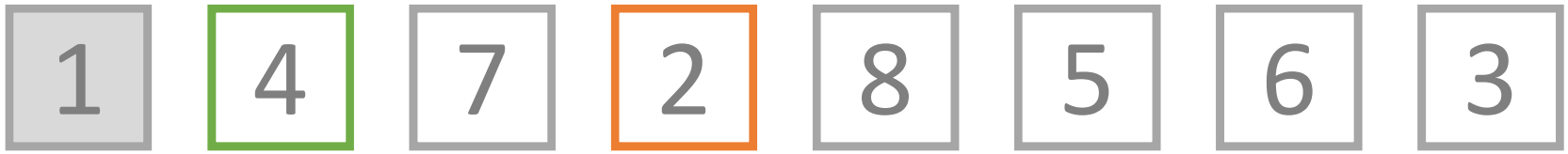
SELECTION SORT



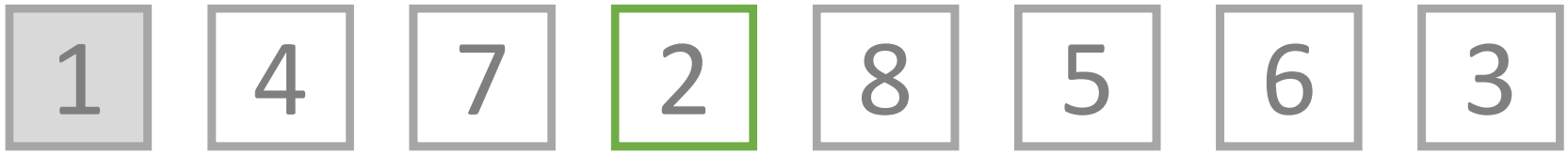
SELECTION SORT



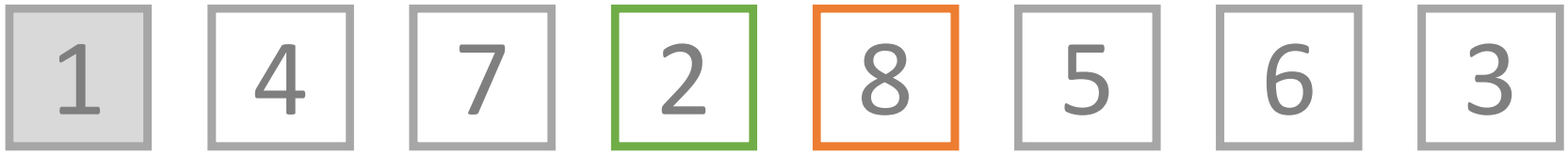
SELECTION SORT



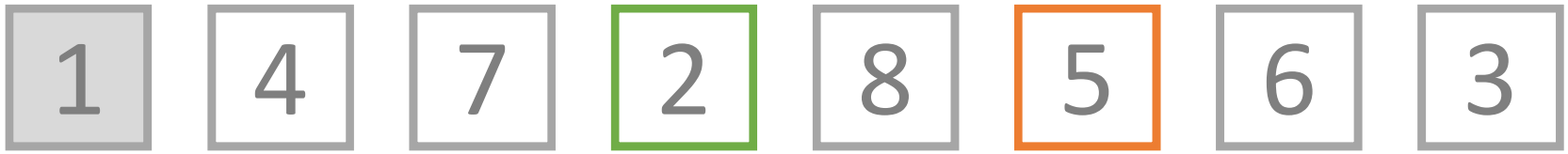
SELECTION SORT



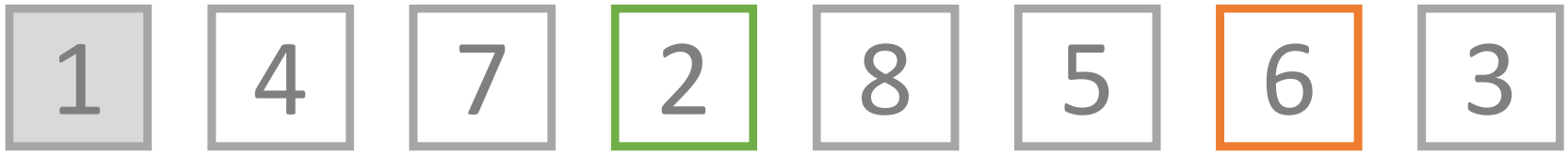
SELECTION SORT



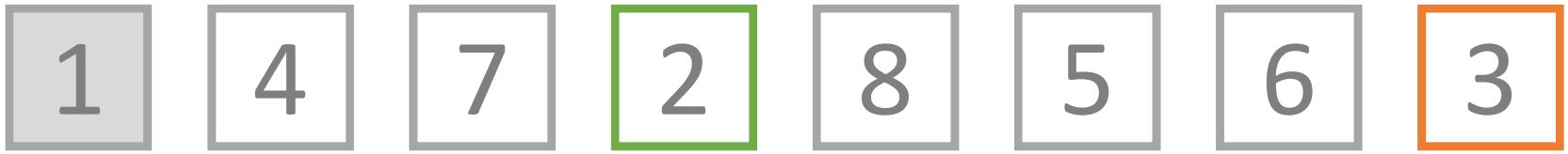
SELECTION SORT



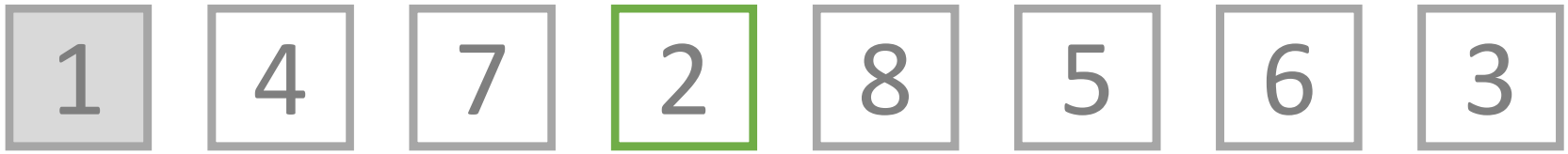
SELECTION SORT



SELECTION SORT

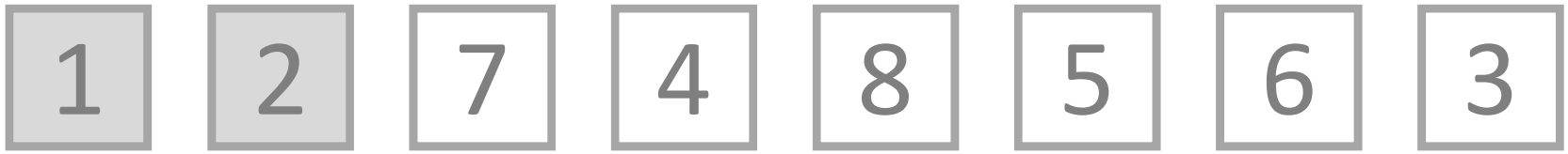


SELECTION SORT

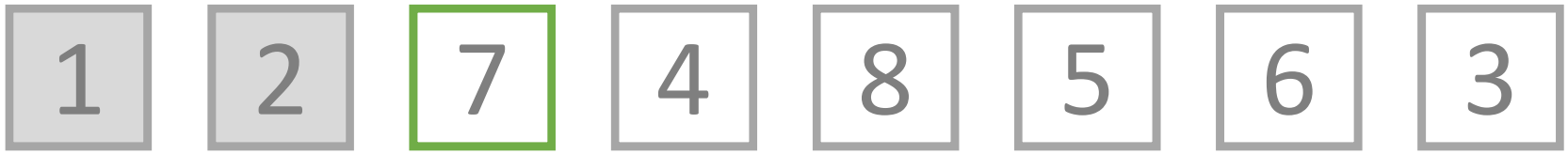


SELECTION SORT

SELECTION SORT



SELECTION SORT



SELECTION SORT

SELECTION SORT

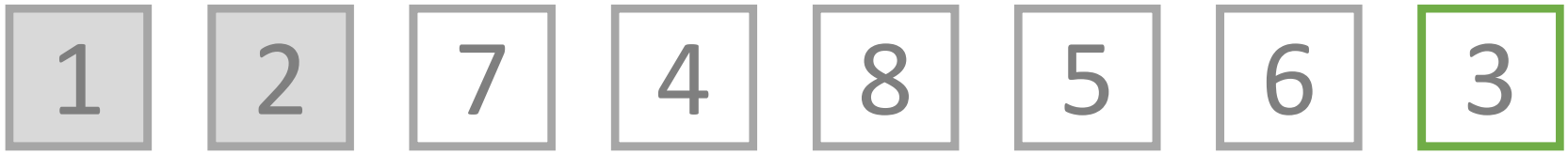
SELECTION SORT

SELECTION SORT

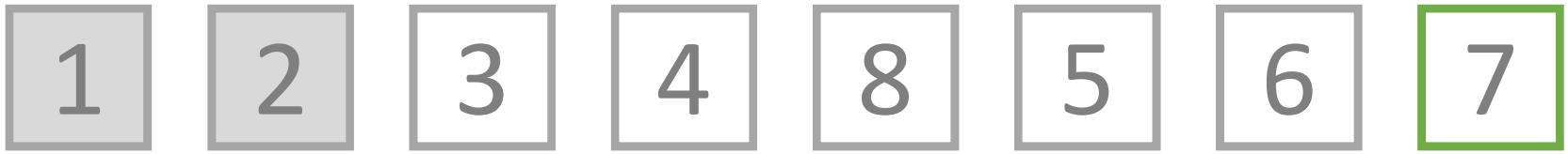
SELECTION SORT

SELECTION SORT

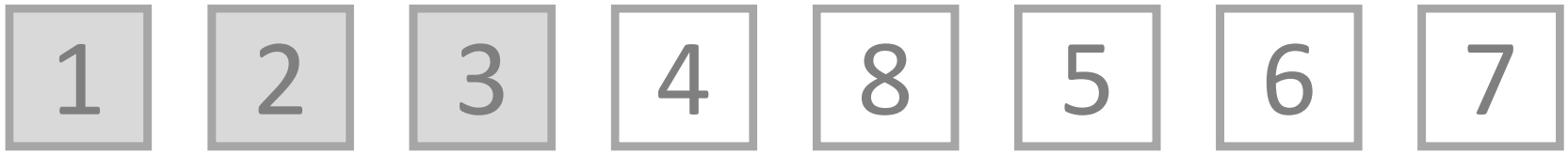
SELECTION SORT



SELECTION SORT



SELECTION SORT



SELECTION SORT

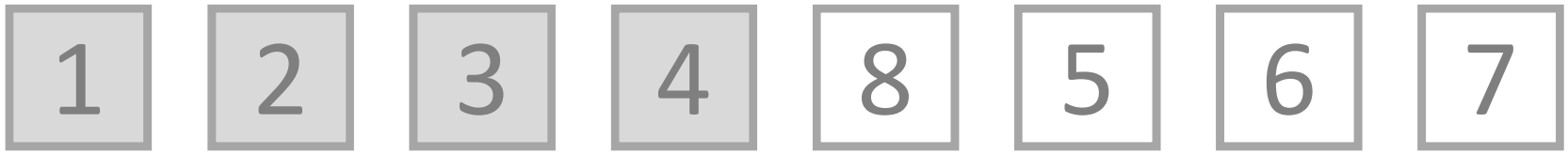
SELECTION SORT

SELECTION SORT

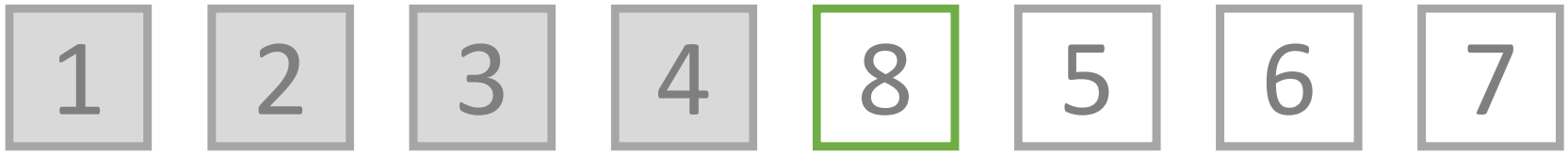
SELECTION SORT

SELECTION SORT

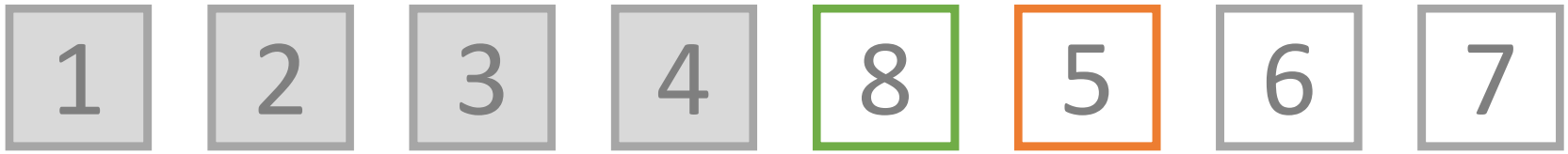
SELECTION SORT



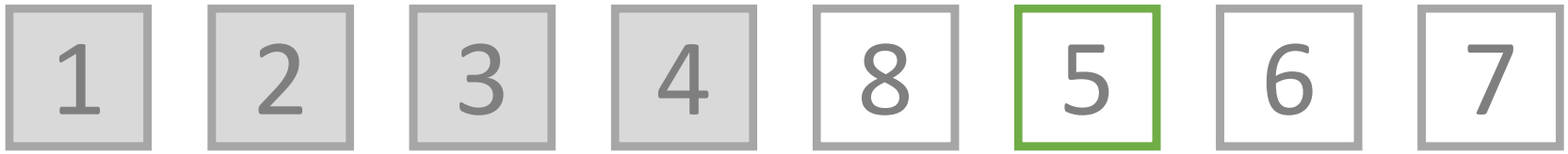
SELECTION SORT



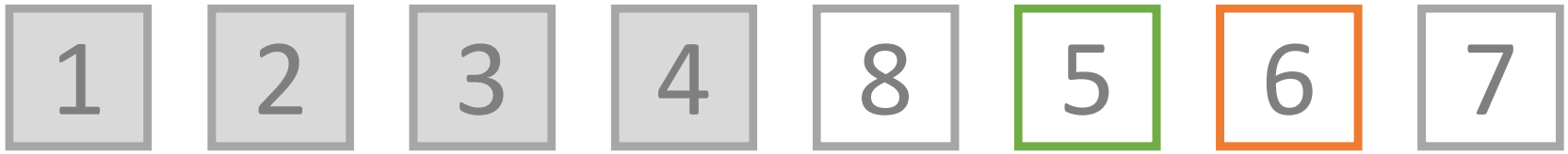
SELECTION SORT



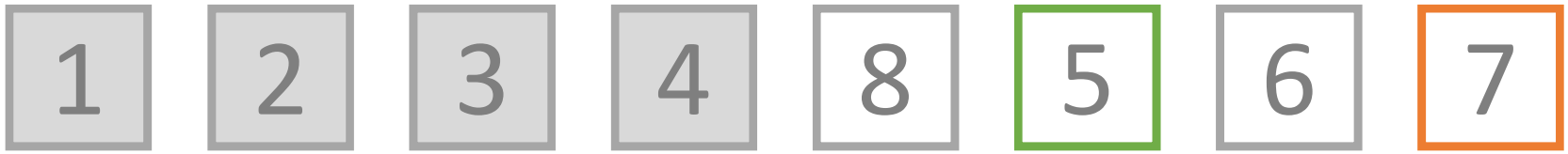
SELECTION SORT



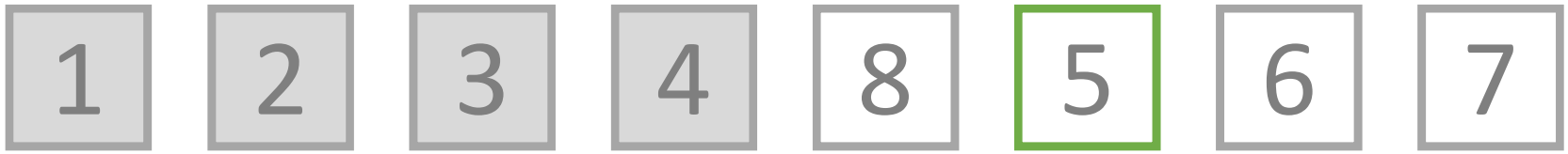
SELECTION SORT



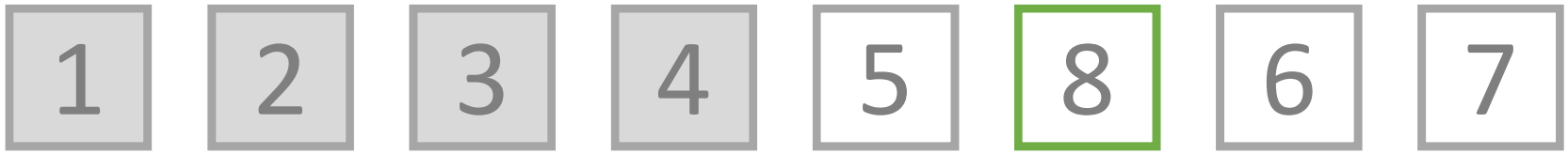
SELECTION SORT



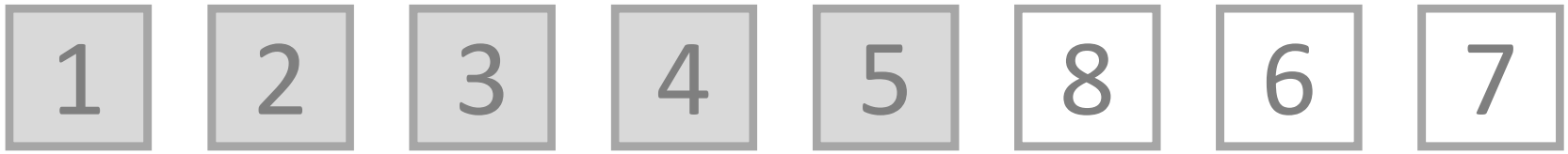
SELECTION SORT



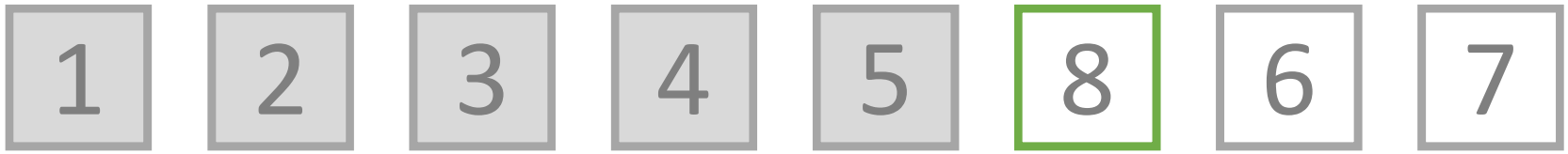
SELECTION SORT



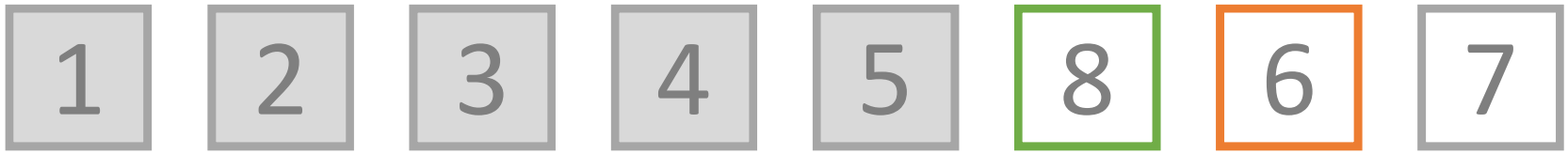
SELECTION SORT



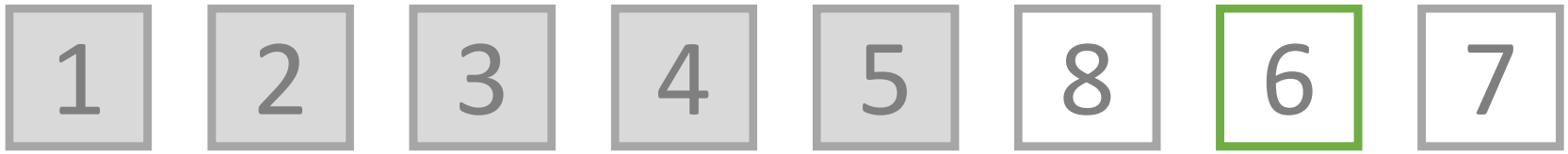
SELECTION SORT



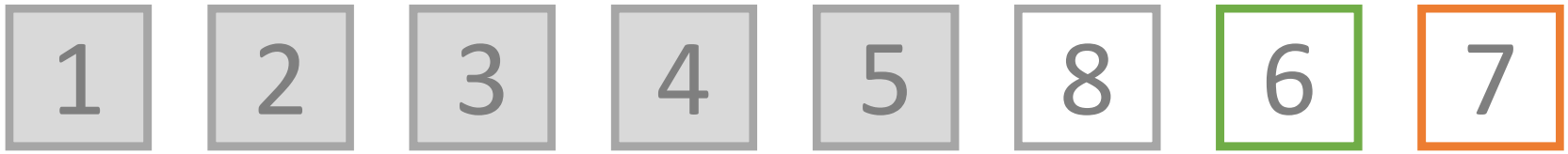
SELECTION SORT



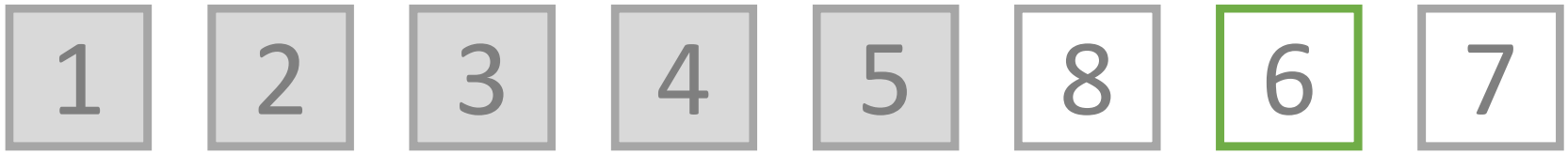
SELECTION SORT



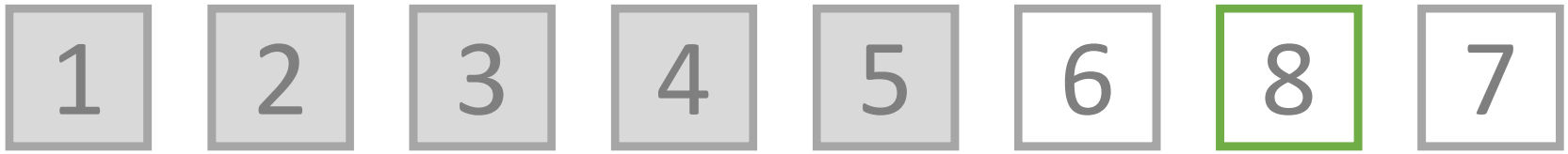
SELECTION SORT



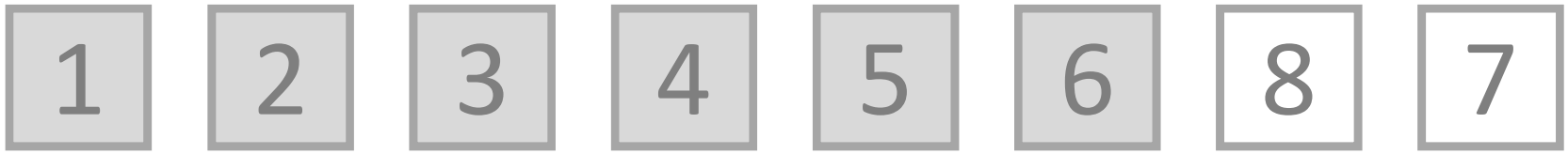
SELECTION SORT



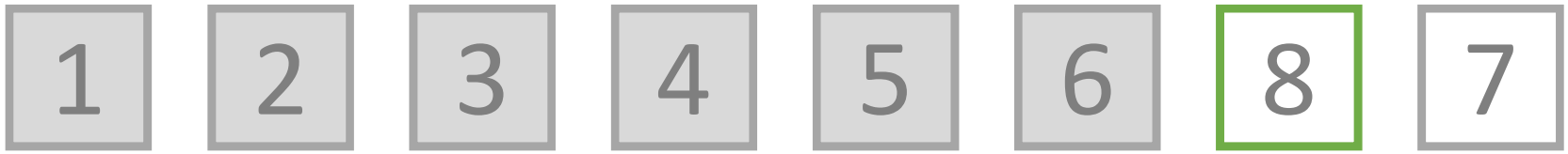
SELECTION SORT



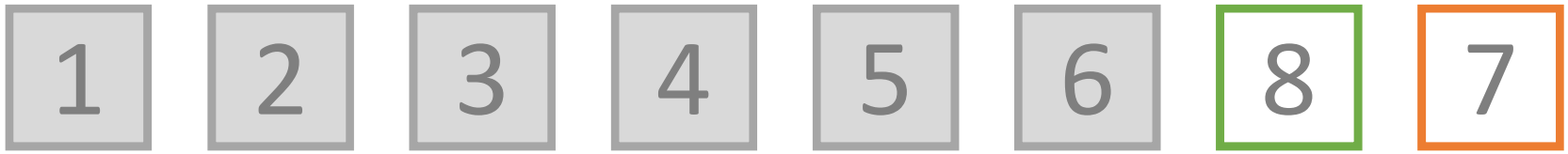
SELECTION SORT



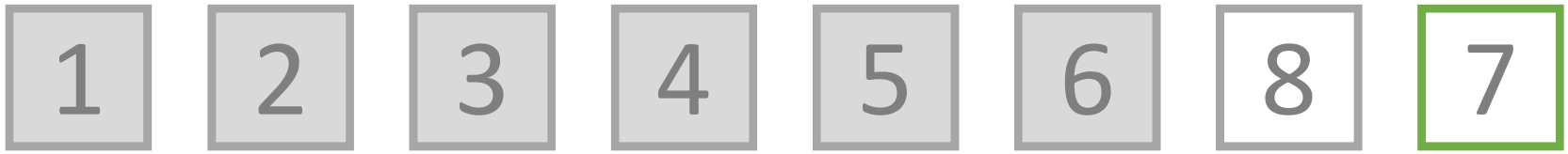
SELECTION SORT



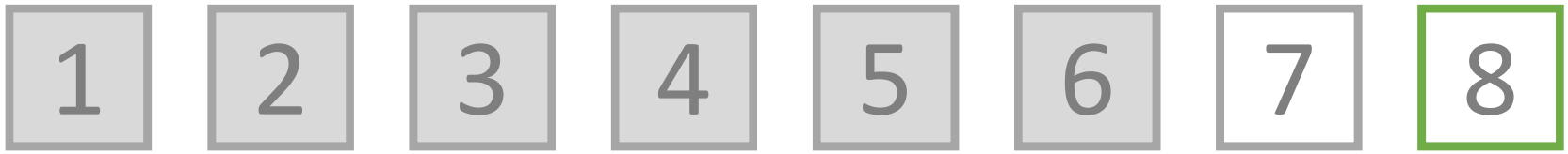
SELECTION SORT



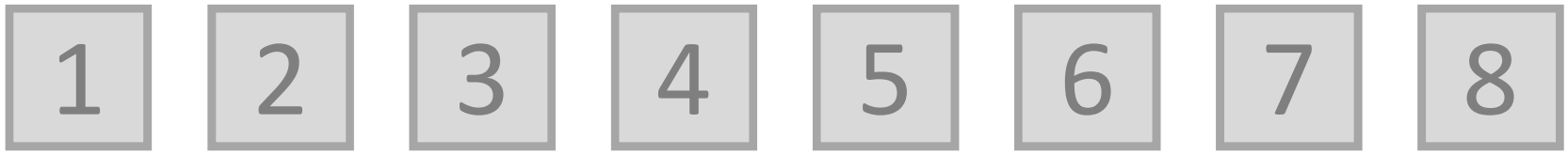
SELECTION SORT



SELECTION SORT



SELECTION SORT



SELECTION SORT

```
void selectionsort(int v[N], int n)
{
        int i, j, posMenor;
        for(i = 0; i < n-1; i++)
        {
                posMenor = i;
                for(j = i+1; j < n; j++)
                {
                        if(v[j] < v[posMenor])
                        {
                                posMenor = j;
                        }
                }
                if(posMenor != i)
                {
                        intercambio(&v[i], &v[posMenor]);
                }
        }
}
```

SELECTION SORT

Peor Caso	Mejor Caso
$O(n^2)$	$\Omega(n^2)$

Todas las iteraciones de este algoritmo se harán independientemente de la distribución de los datos.

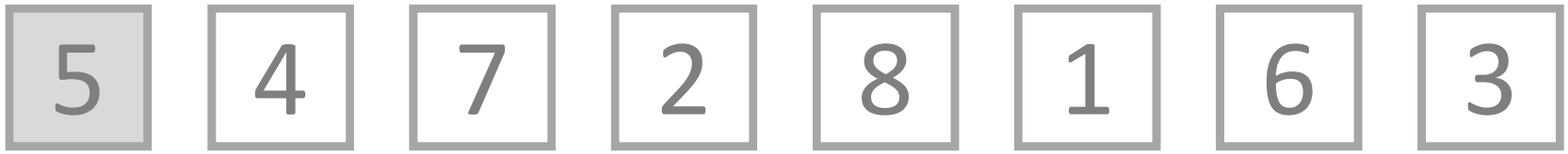
INSERTION SORT

- Este algoritmo divide el arreglo en dos partes: el lado izquierdo que se encuentra ordenado, y el lado derecho que se encuentra desordenado.

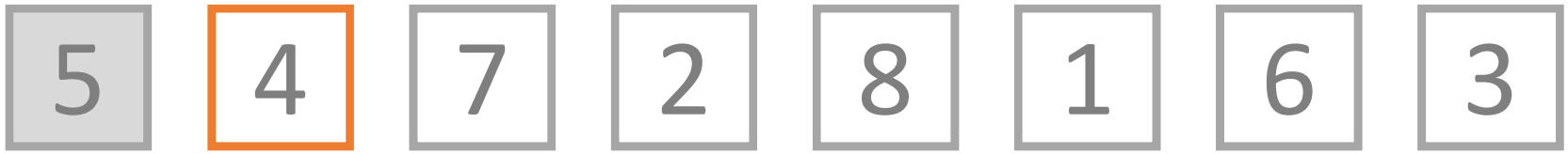
- Luego va insertando de manera ordenada cada elemento del arreglo derecho en el arreglo izquierdo.

- Realiza esto hasta que el arreglo esté ordenado.

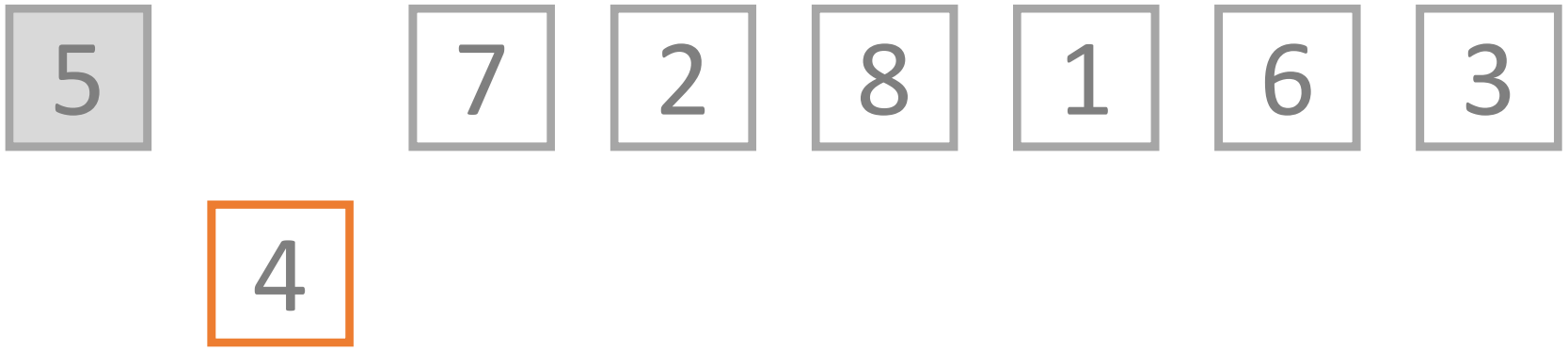
INSERTION SORT



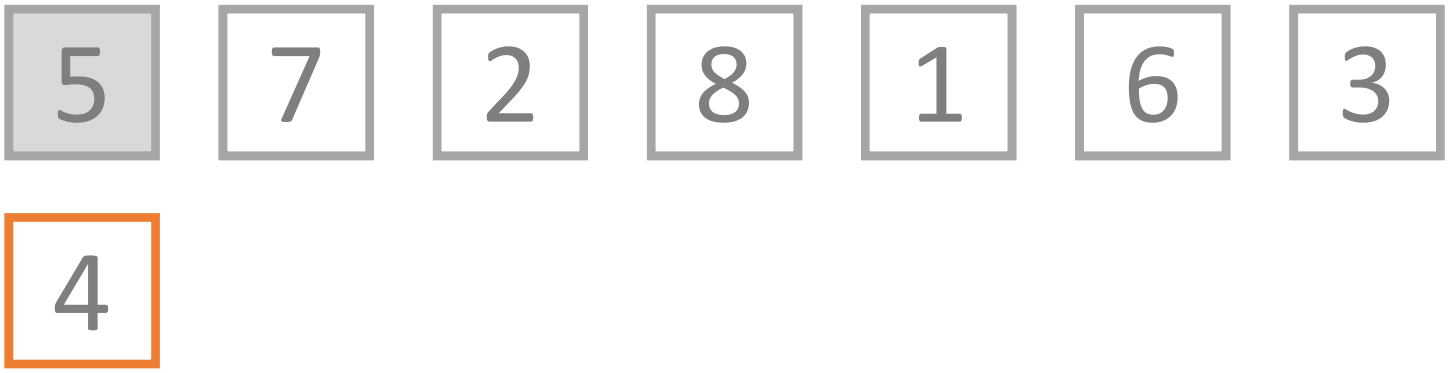
INSERTION SORT



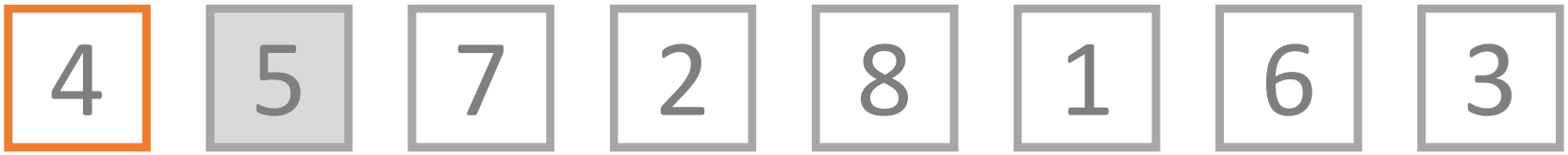
INSERTION SORT



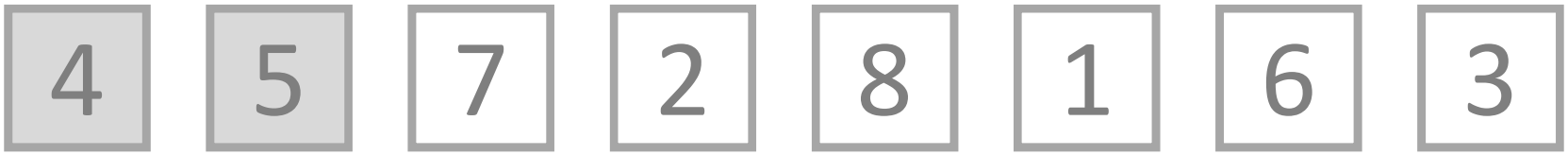
INSERTION SORT



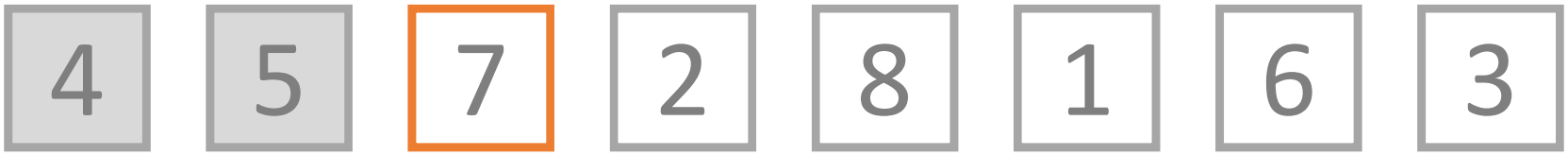
INSERTION SORT



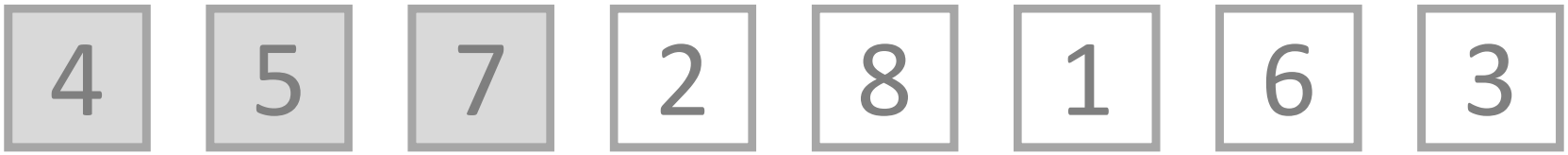
INSERTION SORT



INSERTION SORT

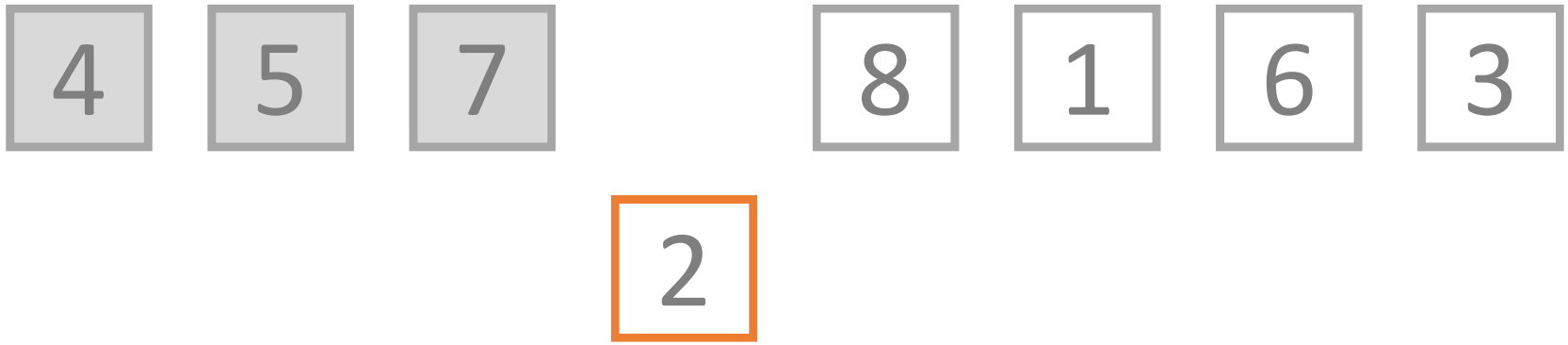


INSERTION SORT

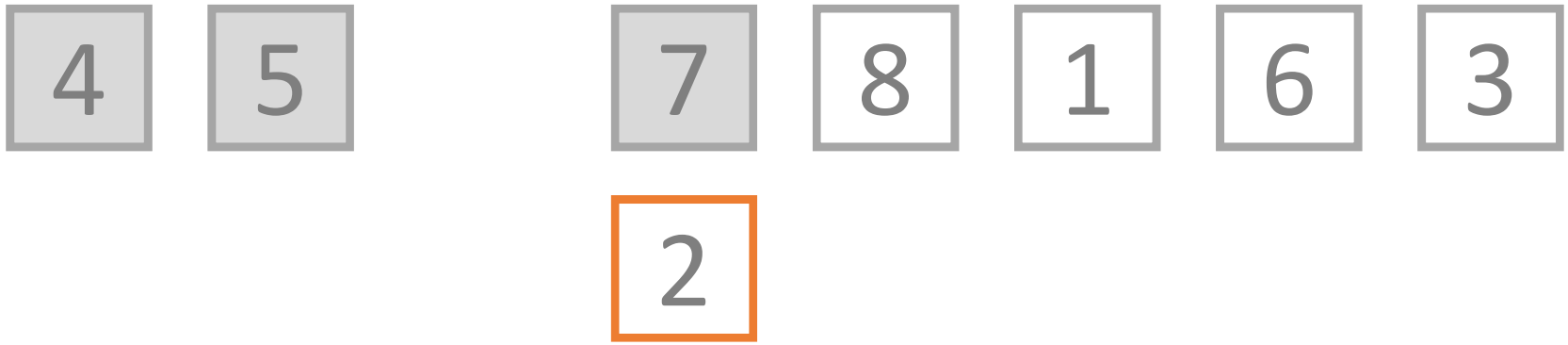


INSERTION SORT

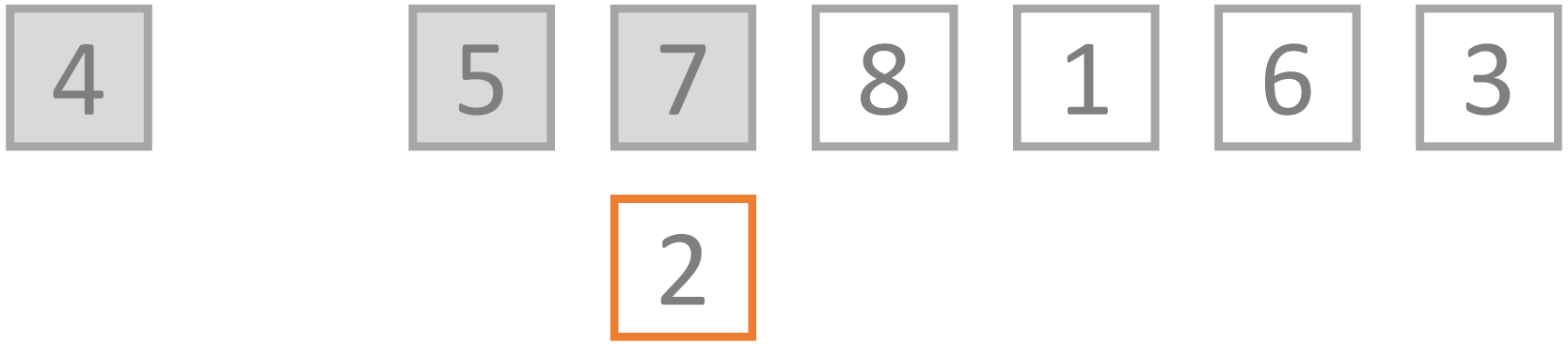
INSERTION SORT



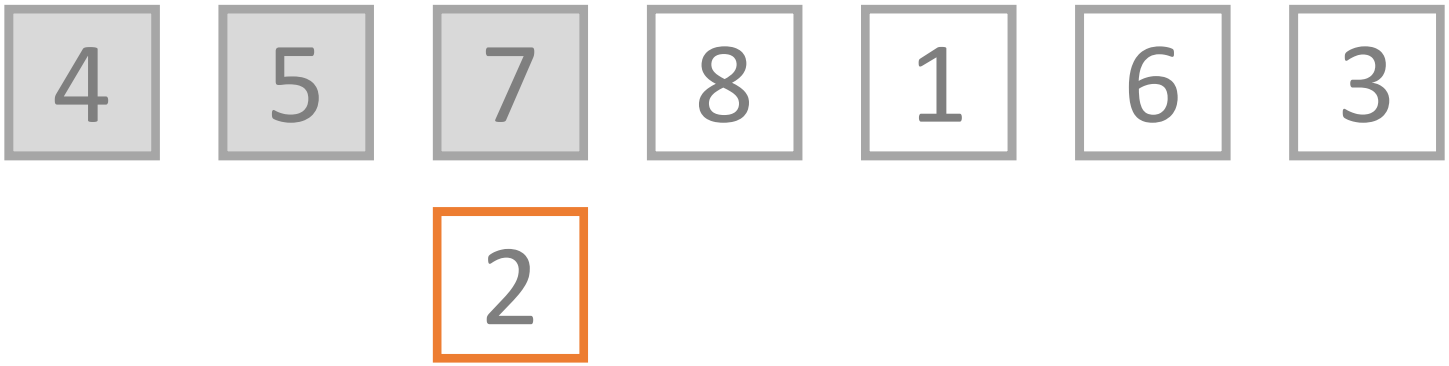
INSERTION SORT



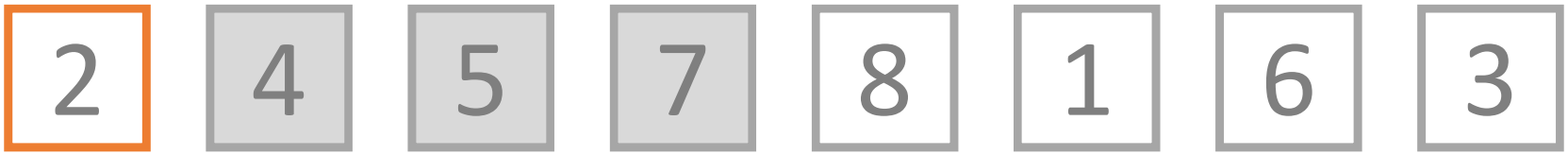
INSERTION SORT



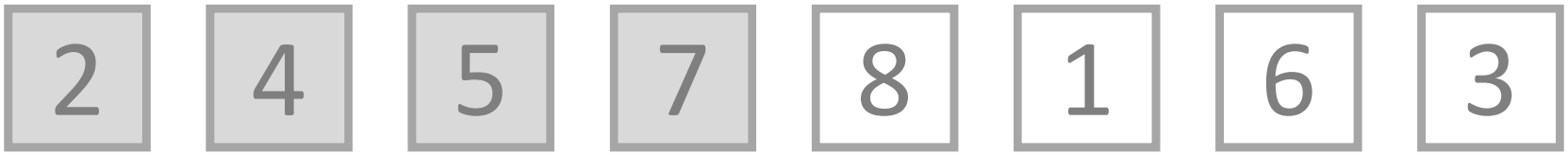
INSERTION SORT



INSERTION SORT

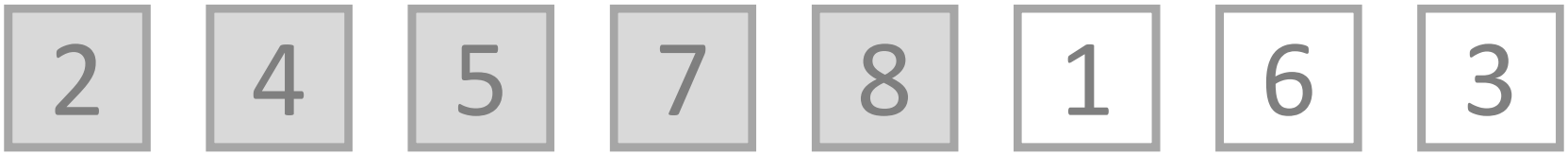


INSERTION SORT

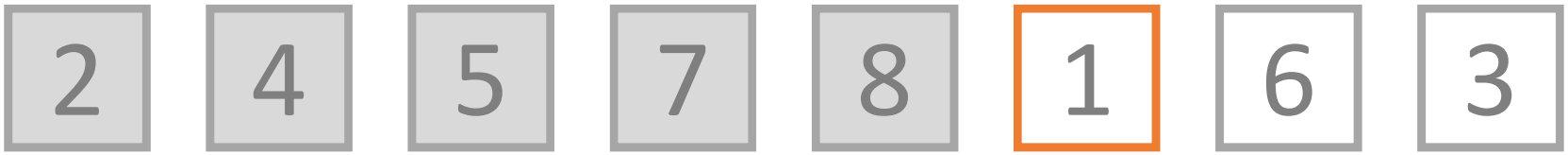


INSERTION SORT

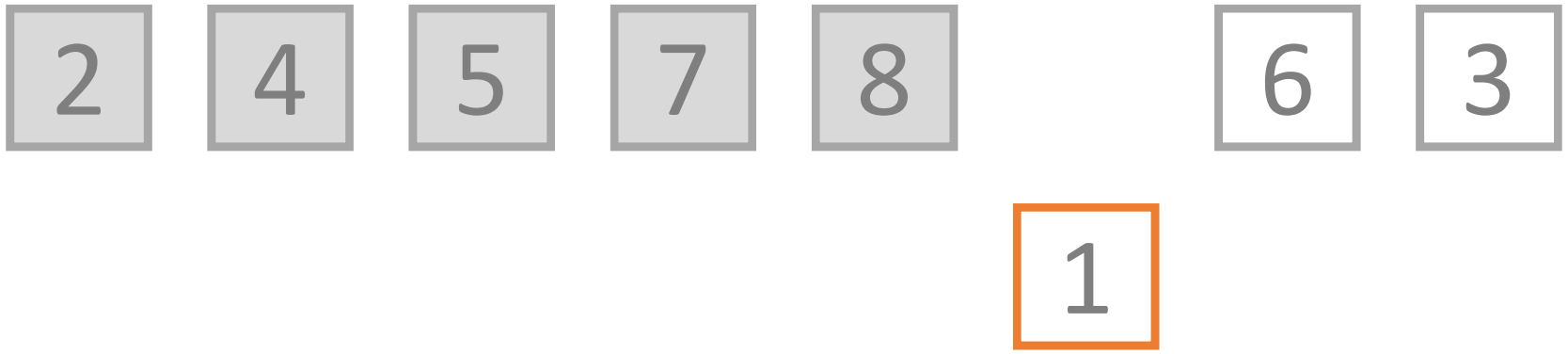
INSERTION SORT



INSERTION SORT



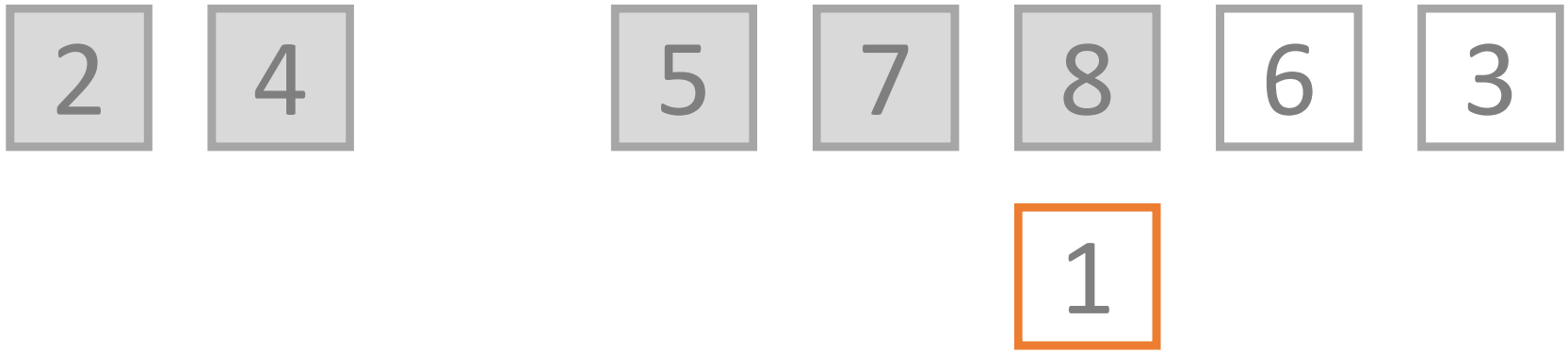
INSERTION SORT



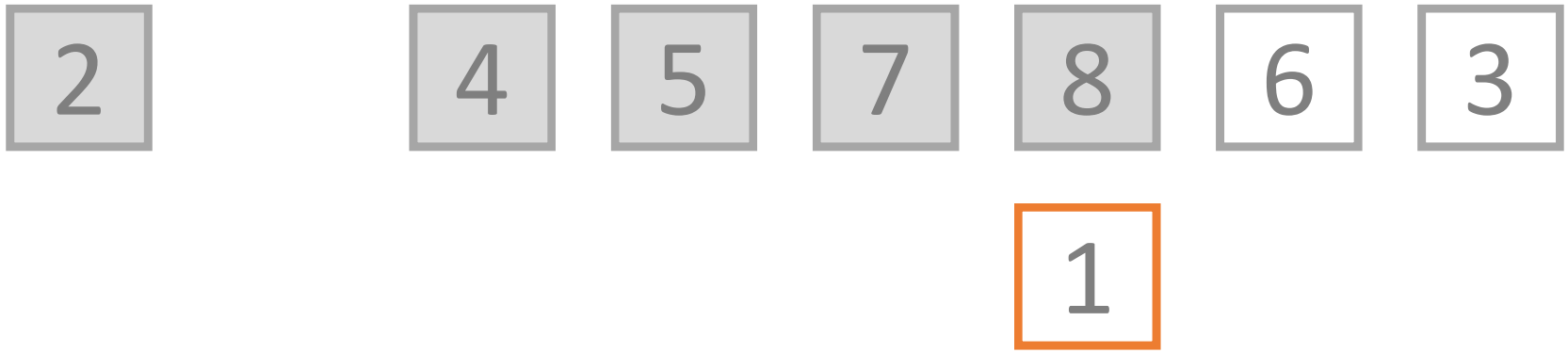
INSERTION SORT

INSERTION SORT

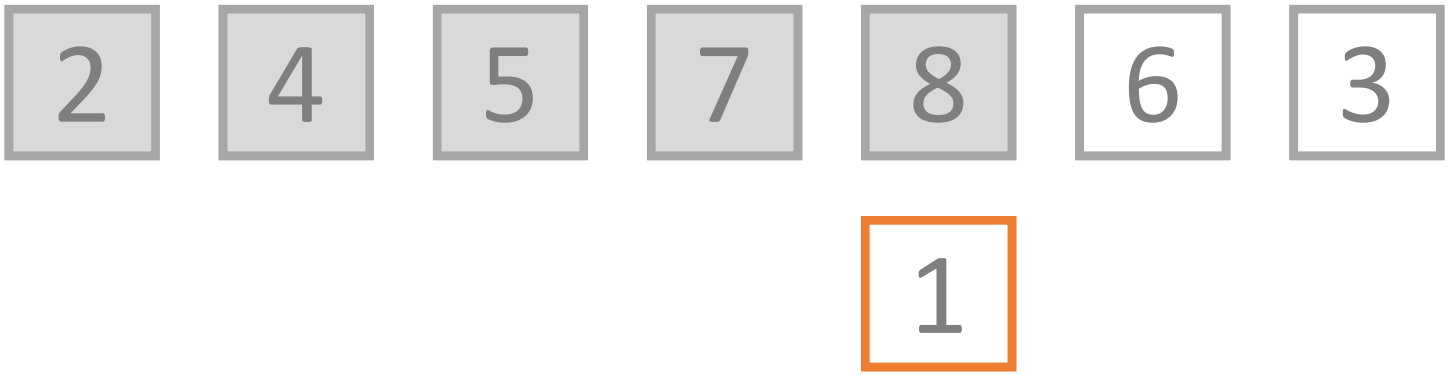
INSERTION SORT



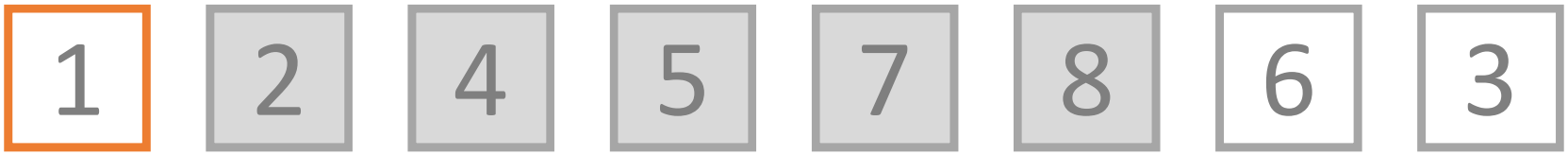
INSERTION SORT



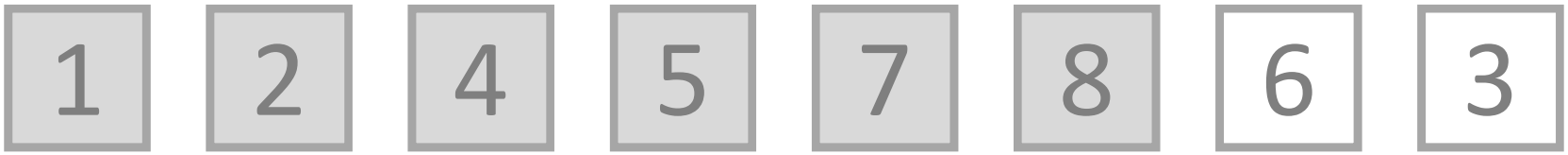
INSERTION SORT



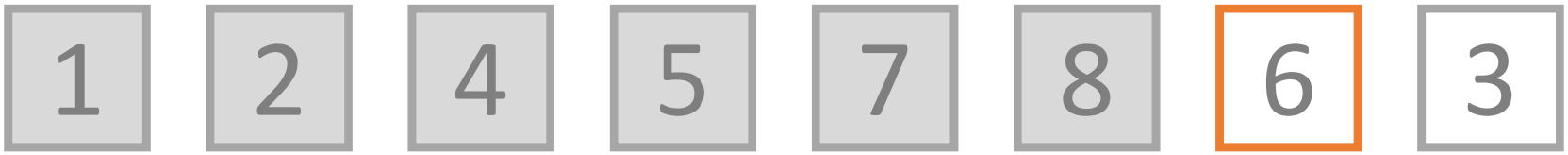
INSERTION SORT



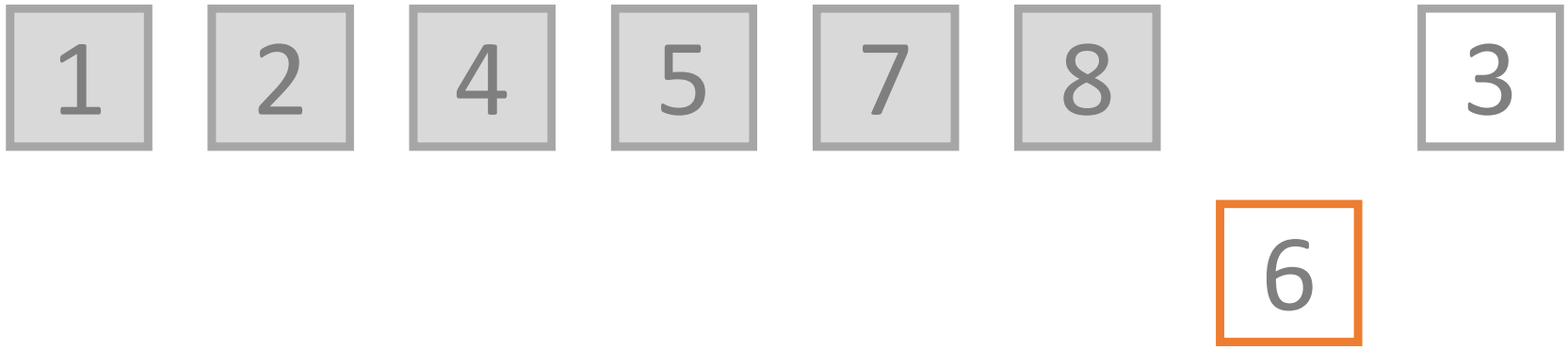
INSERTION SORT



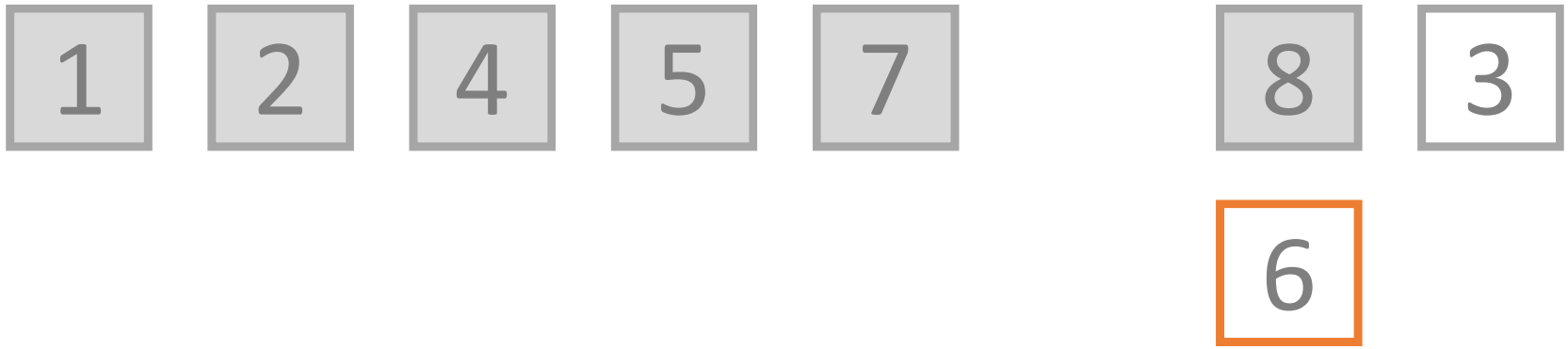
INSERTION SORT



INSERTION SORT



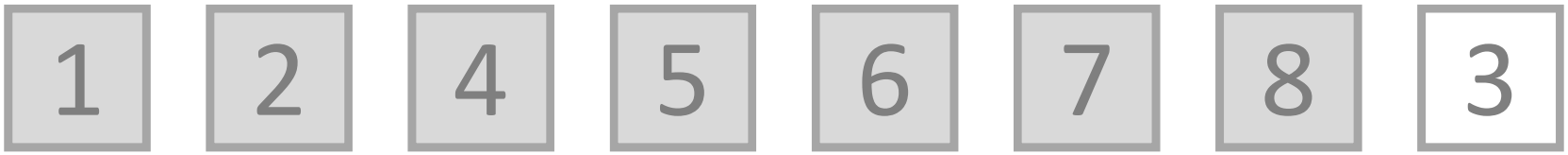
INSERTION SORT



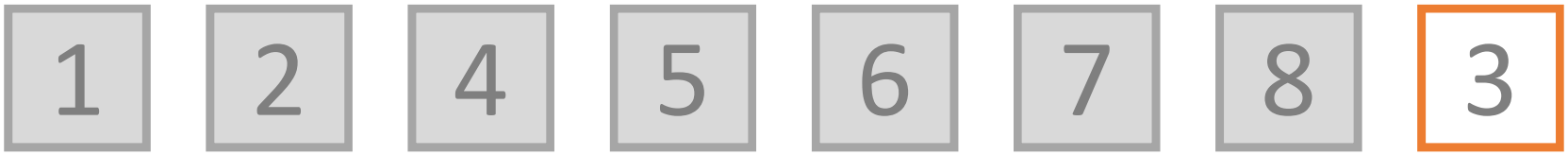
INSERTION SORT

INSERTION SORT

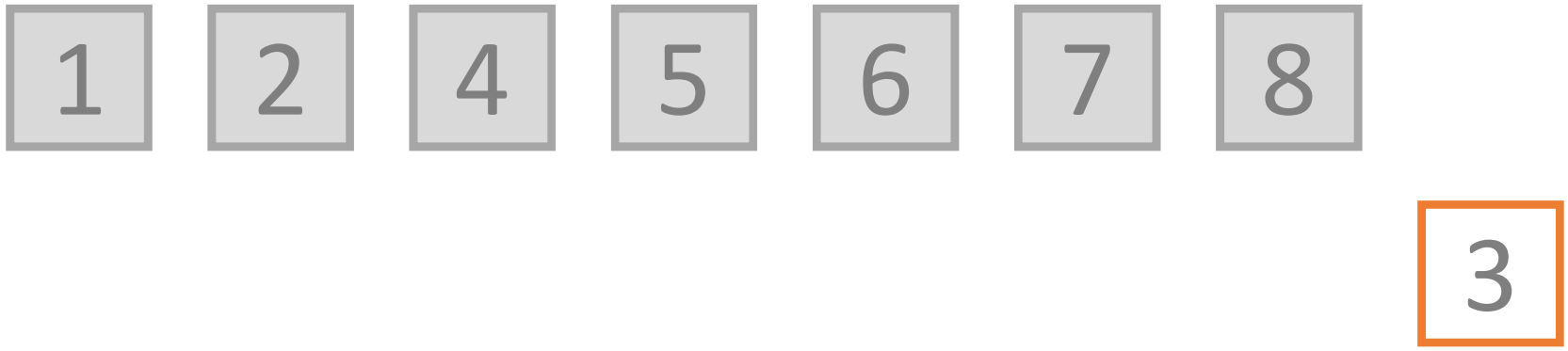
INSERTION SORT



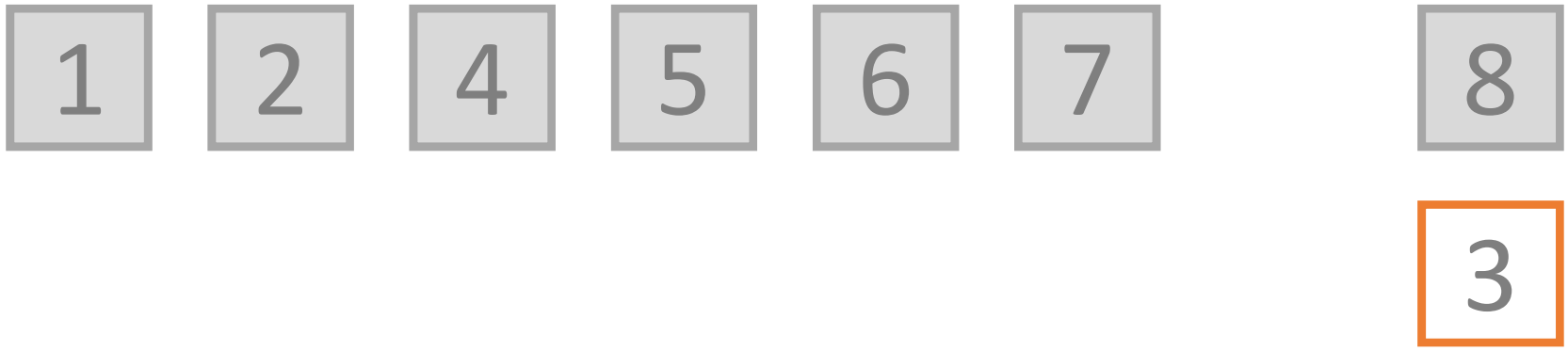
INSERTION SORT



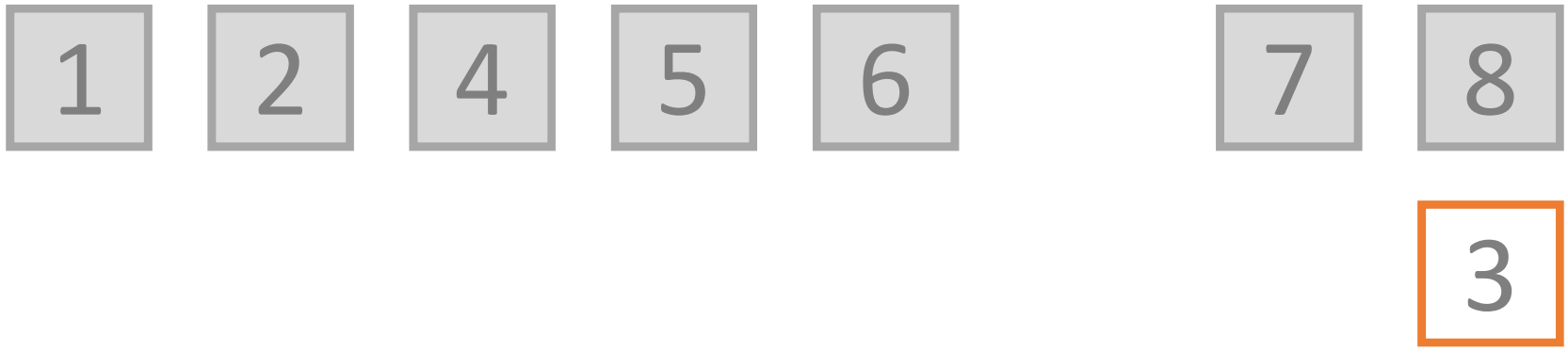
INSERTION SORT



INSERTION SORT



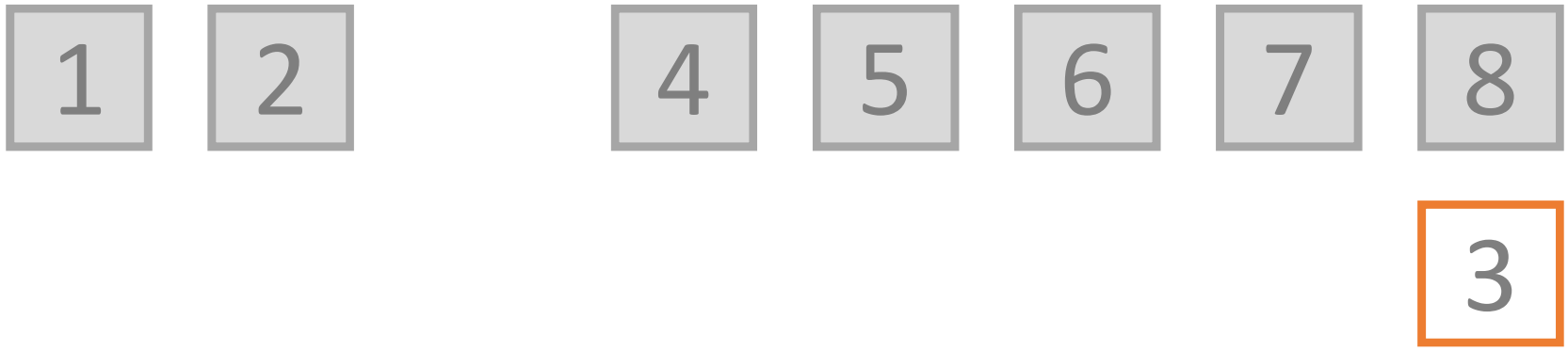
INSERTION SORT



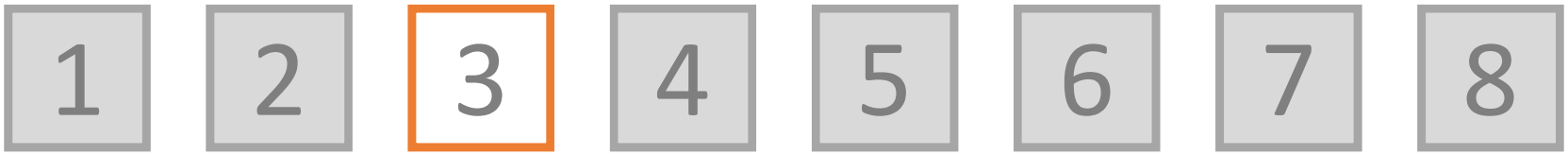
INSERTION SORT

INSERTION SORT

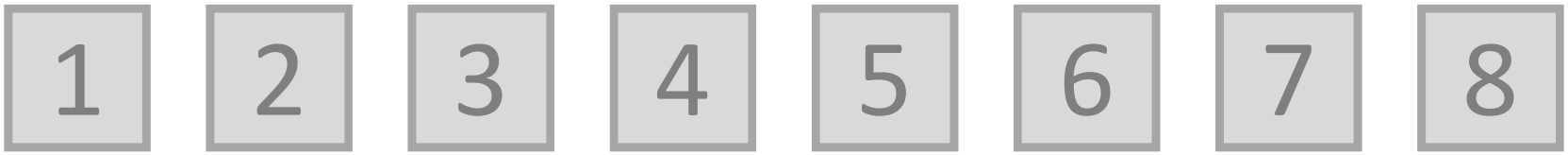
INSERTION SORT



INSERTION SORT



INSERTION SORT



INSERTION SORT

```
void insertionsort (int v[N], int n)
{
        int i, j, aux;
        for(i = 1; i < n; i++)
        {
                j = i;
                aux = v[j];
                while(j > 0 && aux < v[j-1])
                {
                        v[j] = v[j-1];
                        j--;
                }
                v[j] = aux;
        }
}
```

INSERTION SORT

Peor Caso

$O(n^2)$

Cuando el arreglo está ordenado de manera inversa.

Mejor Caso

$\Omega(n)$

Cuando el arreglo ya se encuentra ordenado correctamente.

¿PREGUNTAS?