

# Sorting Data



## Selection sort Part 2



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/)  
by Christine Alvarado, Mia Minnes, and Leo Porter, 2015.

## By the end of this video you will be able to...

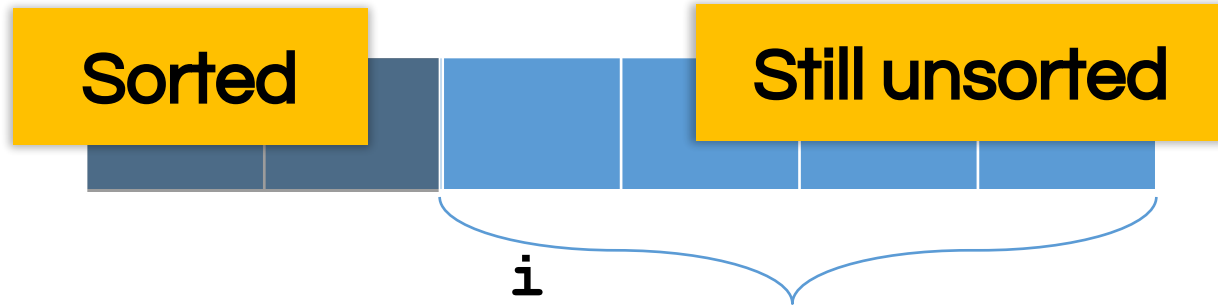
- Explain the selection sort algorithm
- Write code to perform selection sort

- Find smallest element, swap it with element in location 0
- Find next smallest element swap it with element in location 1
- etc.

## Selection Sort

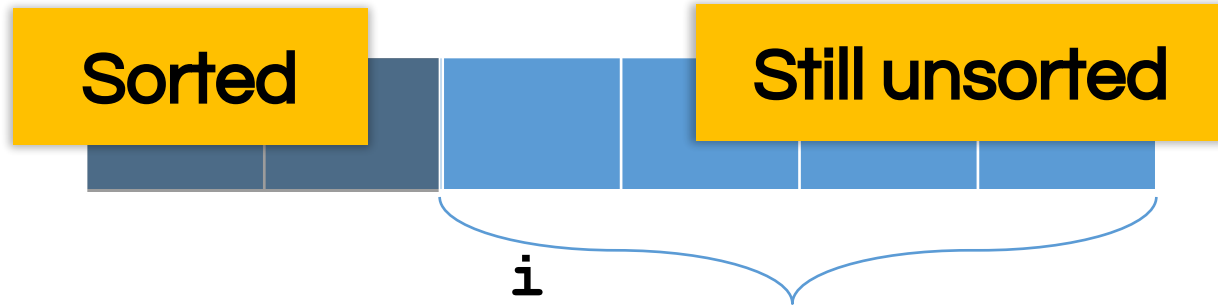
## Selection Sort: Basic Algorithm

For each **position**  $i$  from 0 to  $\text{length}-2$



## Selection Sort: Basic Algorithm

For each **position  $i$**  from 0 to **length-2**

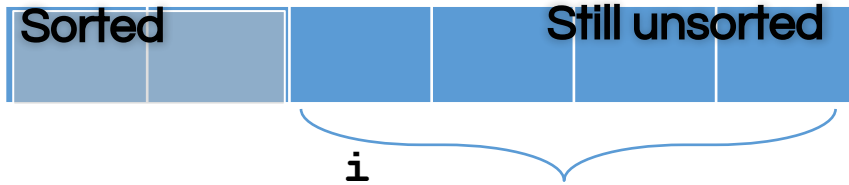


## Selection Sort: Basic Algorithm

For each **position  $i$**  from 0 to  **$\text{length}-2$**

Find smallest element in "still unsorted"

Swap it with element in **position  $i$**

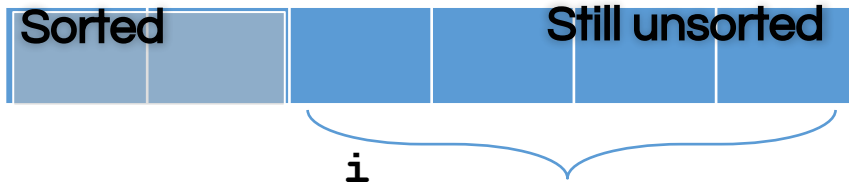


## Selection Sort: Basic Algorithm

For each **position  $i$**  from 0 to  **$\text{length}-2$**

Find smallest element in **positions  $i$  to  $\text{length}-1$**

Swap it with element in **position  $i$**



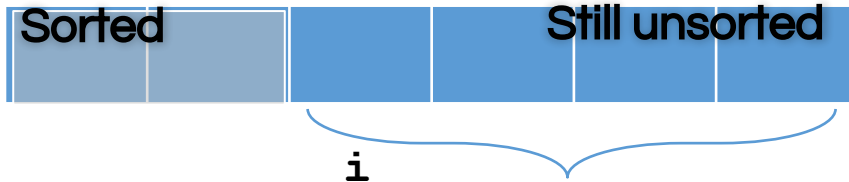
## Selection Sort: Basic Algorithm



For each **position  $i$**  from 0 to **length-2**

Find smallest element in **positions  $i$  to length-1**

Swap it with element in **position  $i$**





```
public static void selectionSort( int[] vals )    {
```

```
public static void selectionSort( int[] vals )    {
```

```
for ( int i=0; i < vals.length-1 ; i++ ) {
```

}

}

```
public static void selectionSort( int[] vals )    {

    int indexMin;

    for ( int i=0; i < vals.length-1 ; i++ ) {

        indexMin = i ;
        for ( int j=i+1; j < vals.length; j++ ) {
            if ( vals[j] < vals[indexMin] ) {
                indexMin = j ;
            }
        }

    }

}
```

```
public static void selectionSort( int[] vals )    {

    int indexMin;

    for ( int i=0; i < vals.length-1 ; i++ ) {

        indexMin = i ;
        for ( int j=i+1; j < vals.length; j++ ) {
            if ( vals[j] < vals[indexMin] ) {
                indexMin = j ;
            }
        }

    }

}
```

```
public static void selectionSort( int[] vals )    {  
  
    int indexMin;  
  
    for ( int i=0; i < vals.length-1 ; i++ ) {  
  
        indexMin = i ;  
        for ( int j=i+1; j < vals.length; j++ ) {  
            if ( vals[j] < vals[indexMin] ) {  
                indexMin = j ;  
            }  
        }  
  
    }  
}
```

```
public static void selectionSort( int[] vals )    {  
  
    int indexMin;  
  
    for ( int i=0; i < vals.length-1 ; i++ ) {  
  
        indexMin = i ;  
        for ( int j=i+1; j < vals.length; j++ ) {  
            if ( vals[j] < vals[indexMin] ) {  
                indexMin = j ;  
            }  
        }  
  
    }  
  
}
```

```
public static void selectionSort( int[] vals )    {

    int indexMin;

    for ( int i=0; i < vals.length-1 ; i++ ) {

        indexMin = i ;
        for ( int j=i+1; j < vals.length; j++ ) {
            if ( vals[j] < vals[indexMin] ) {
                indexMin = j ;
            }
        }

    }

}
```

```
public static void selectionSort( int[] vals )    {

    int indexMin;

    for ( int i=0; i < vals.length-1 ; i++ ) {

        indexMin = i ;
        for ( int j=i+1; j < vals.length; j++ ) {
            if ( vals[j] < vals[indexMin] ) {
                indexMin = j ;
            }
        }

        swap ( vals, indexMin , i );
    }
}
```



# Thought questions

- How do we know this algorithm **always** works?
- How well does it work?
- Can we do better?