CS 2050
Computer Science II

Thyago Mota

METROPOLITAN STATE UNIVERSITY OF DENVER
LIVES TRANSFORMED

Agenda

- Sorting Algorithms:
  - Insertion Sort

Insertion Sort

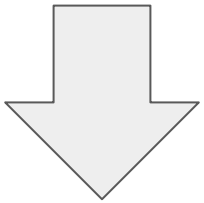
- Insertion sort works by taking one element at a time and finding its correct point of insertion in the sorted collection

Insertion Sort

```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```

Insertion Sort

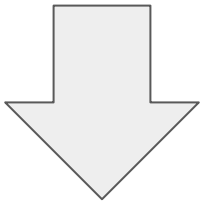
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[][13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```

Insertion Sort

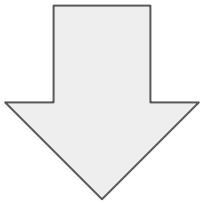
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]



[][13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]

Insertion Sort

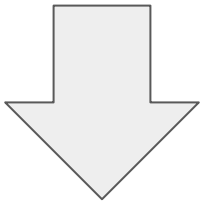
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]



[13][12, 84, 79, 10, 77, 56, 1, 34, 27, 3]

Insertion Sort

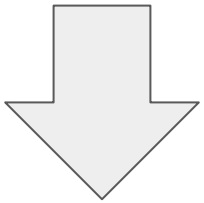
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]



[12, 13][84, 79, 10, 77, 56, 1, 34, 27, 3]

Insertion Sort

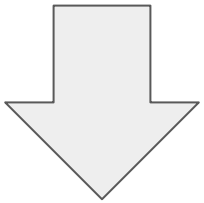
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]



[12, 13, 84][79, 10, 77, 56, 1, 34, 27, 3]

Insertion Sort

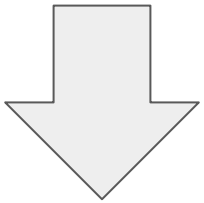
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[12, 13, 79, 84][10, 77, 56, 1, 34, 27, 3]
```

Insertion Sort

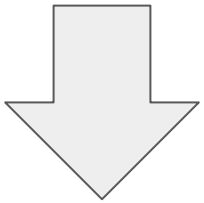
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[12, 13, 79, 10, 84][77, 56, 1, 34, 27, 3]
```

Insertion Sort

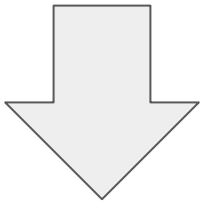
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[12, 13, 10, 79 84][77, 56, 1, 34, 27, 3]
```

Insertion Sort

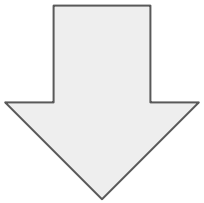
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[12, 10, 13, 79 84][77, 56, 1, 34, 27, 3]
```

Insertion Sort

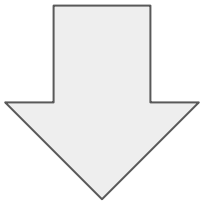
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[10, 12, 13, 79 84][77, 56, 1, 34, 27, 3]
```

Insertion Sort

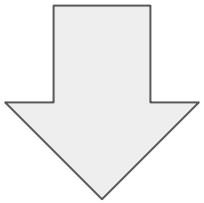
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[10, 12, 13, 79, 84][77, 56, 1, 34, 27, 3]
```

Insertion Sort

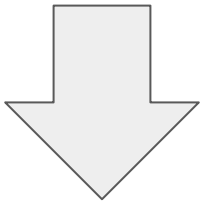
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[10, 12, 13, 79, 77, 84][56, 1, 34, 27, 3]
```

Insertion Sort

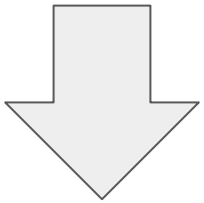
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[10, 12, 13, 77, 79, 84][56, 1, 34, 27, 3]
```

Insertion Sort

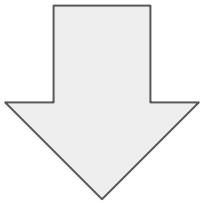
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]



[10, 12, 13, 77, 79, 84][56, 1, 34, 27, 3]

Insertion Sort

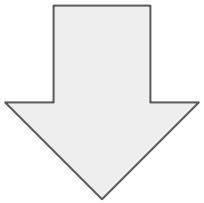
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]



[10, 12, 13, 77, 79, 56, 84][1, 34, 27, 3]

Insertion Sort

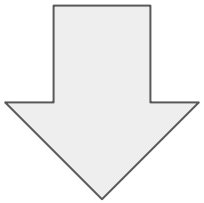
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[10, 12, 13, 77, 56, 79, 84][1, 34, 27, 3]
```

Insertion Sort

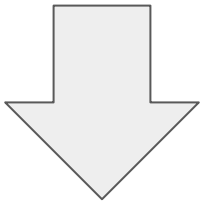
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[10, 12, 13, 56, 77, 79, 84][1, 34, 27, 3]
```

Insertion Sort

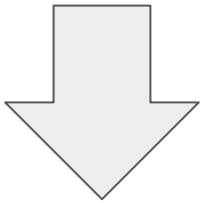
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[10, 12, 13, 56, 77, 79, 84][1, 34, 27, 3]
```

Insertion Sort

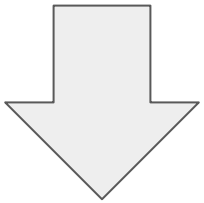
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[10, 12, 13, 56, 77, 79, 1, 84][34, 27, 3]
```

Insertion Sort

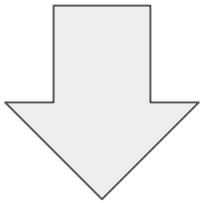
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[10, 12, 13, 56, 77, 1, 79, 84][34, 27, 3]
```

Insertion Sort

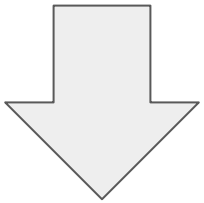
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]



[10, 12, 13, 56, 1, 77 79, 84][34, 27, 3]

Insertion Sort

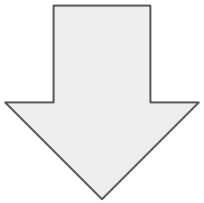
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[10, 12, 13, 1, 56, 77 79, 84][34, 27, 3]
```

Insertion Sort

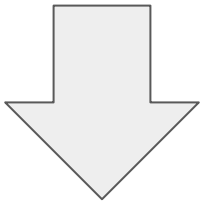
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[10, 12, 1, 13, 56, 77 79, 84][34, 27, 3]
```

Insertion Sort

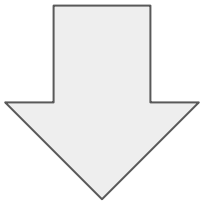
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[10, 1, 12, 13, 56, 77 79, 84][34, 27, 3]
```

Insertion Sort

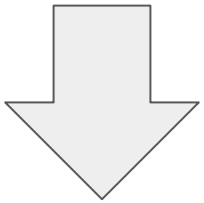
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[1, 10, 12, 13, 56, 77 79, 84][34, 27, 3]
```

Insertion Sort

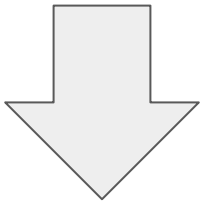
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[1, 10, 12, 13, 56, 77, 79, 84][34, 27, 3]
```

Insertion Sort

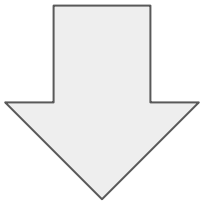
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[1, 10, 12, 13, 56, 77, 79, 34, 84][27, 3]
```

Insertion Sort

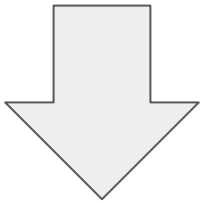
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[1, 10, 12, 13, 56, 77, 34, 79, 84][27, 3]
```

Insertion Sort

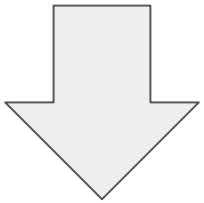
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[1, 10, 12, 13, 56, 34, 77, 79, 84][27, 3]
```

Insertion Sort

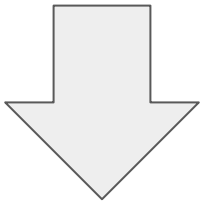
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[1, 10, 12, 13, 34, 56, 77, 79, 84][27, 3]
```

Insertion Sort

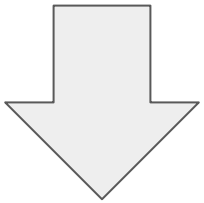
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]



[1, 10, 12, 13, 34, 56, 77, 79, 84][27, 3]

Insertion Sort

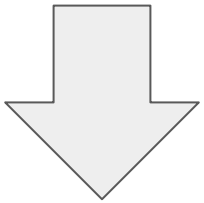
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[1, 10, 12, 13, 34, 56, 77, 79, 27, 84][3]
```

Insertion Sort

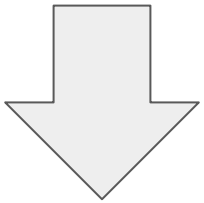
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[1, 10, 12, 13, 34, 56, 77, 27, 79, 84][3]
```

Insertion Sort

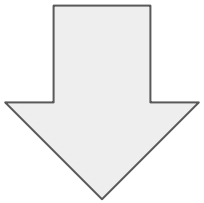
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[1, 10, 12, 13, 34, 56, 27, 77, 79, 84][3]
```

Insertion Sort

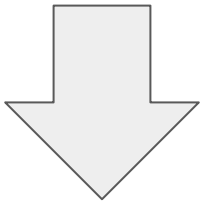
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[1, 10, 12, 13, 34, 27, 56, 77, 79, 84][3]
```

Insertion Sort

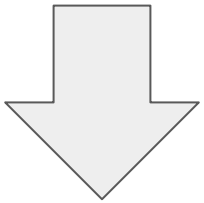
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[1, 10, 12, 13, 27, 34, 56, 77, 79, 84][3]
```

Insertion Sort

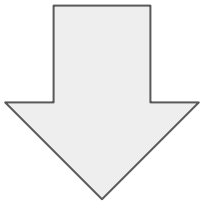
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[1, 10, 12, 13, 27, 34, 56, 77, 79, 84][3]
```

Insertion Sort

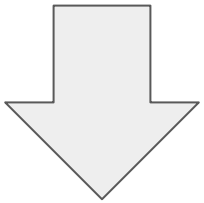
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[1, 10, 12, 13, 27, 34, 56, 77, 79, 3, 84][]
```

Insertion Sort

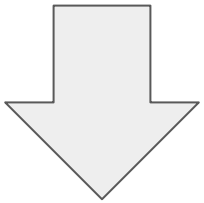
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[1, 10, 12, 13, 27, 34, 56, 77, 3, 79, 84][]
```

Insertion Sort

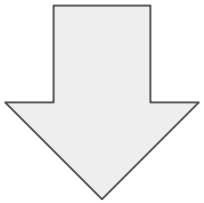
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[1, 10, 12, 13, 27, 34, 56, 3, 77, 79, 84][]
```

Insertion Sort

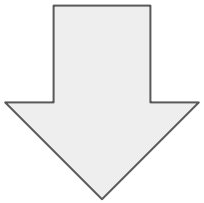
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[1, 10, 12, 13, 27, 34, 3, 56, 77, 79, 84][]
```

Insertion Sort

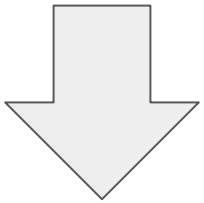
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[1, 10, 12, 13, 27, 3, 34, 56, 77, 79, 84][]
```

Insertion Sort

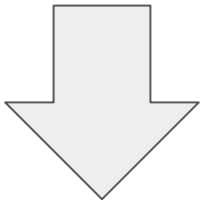
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[1, 10, 12, 13, 3, 27, 34, 56, 77, 79, 84][]
```

Insertion Sort

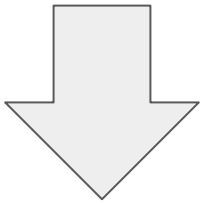
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[1, 10, 12, 3, 13, 27, 34, 56, 77, 79, 84][]
```

Insertion Sort

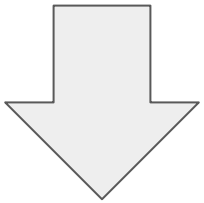
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[1, 10, 3, 12, 13, 27, 34, 56, 77, 79, 84][]
```

Insertion Sort

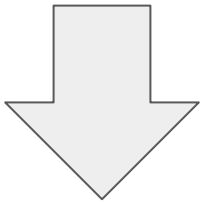
```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[1, 3, 10, 12, 13, 27, 34, 56, 77, 79, 84][]
```

Insertion Sort

```
[13, 12, 84, 79, 10, 77, 56, 1, 34, 27, 3]
```



```
[1, 3, 10, 12, 13, 27, 34, 56, 77, 79, 84][]
```

Insertion Sort

Pause the video now and try to implement the insertion sort algorithm!

Insertion Sort

```
public static void insertionSort(int data[]) {
    for (int i = 1; i < data.length; i++) {
        int j = i;
        int k = i - 1;
        while (k >= 0) {
            if (data[j] < data[k]) {
                int temp = data[j];
                data[j] = data[k];
                data[k] = temp;
                j = k;
                k--;
            }
            else
                break;
        }
    }
}
```

Insertion Sort

```
public static void insertionSort(int data[]) {
    for (int i = 1; i < data.length; i++) {
        int j = i;
        int k = i - 1;
        while (k >= 0) {
            if (data[j] < data[k]) {
                int temp = data[j];
                data[j] = data[k];
                data[k] = temp;
                j = k;
                k--;
            }
            else
                break;
        }
    }
}
```

$O(n^2)$

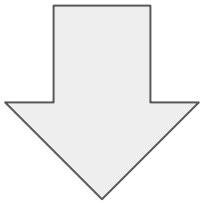
Sorting Algorithms

- Which sorting algorithm do you think is best?
  - selection sort OR
  - insertion sort ?

Sorting Algorithms

Selection Sort

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

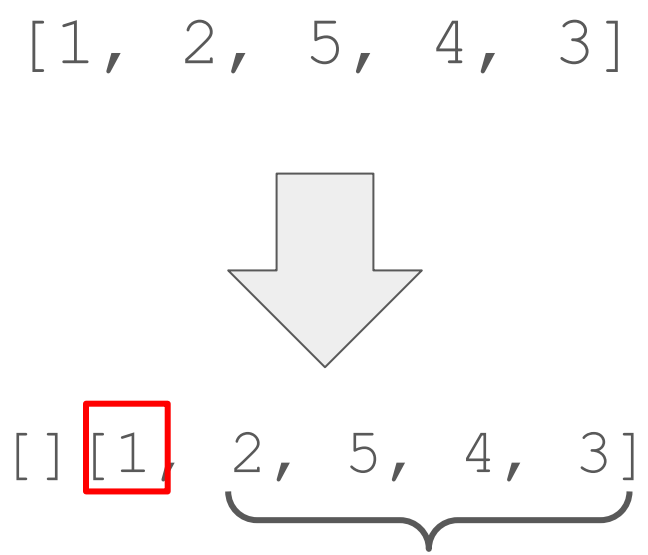


```
[][1, 2, 5, 4, 3]
```

iteration
count = 0

Sorting Algorithms

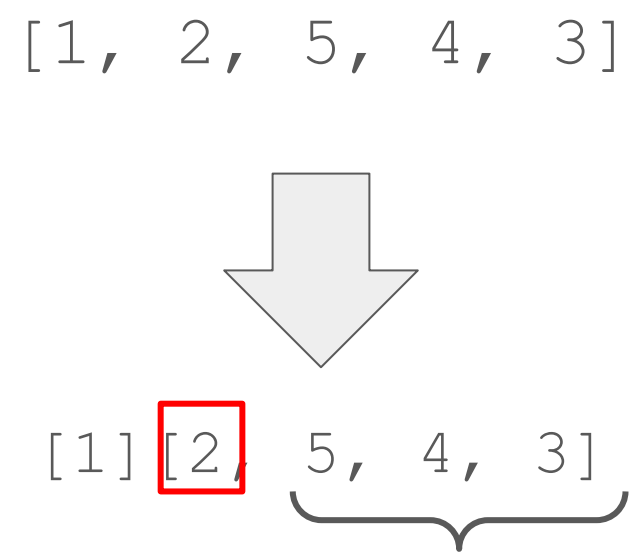
Selection Sort



iteration
count = 4

Sorting Algorithms

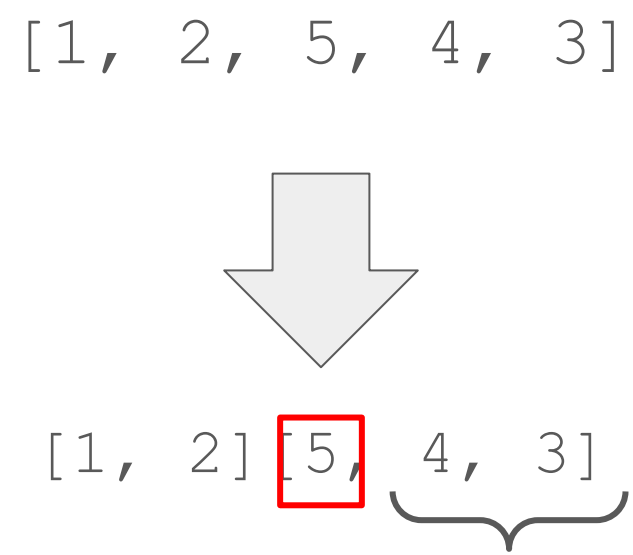
Selection Sort



iteration
count = 4+3

Sorting Algorithms

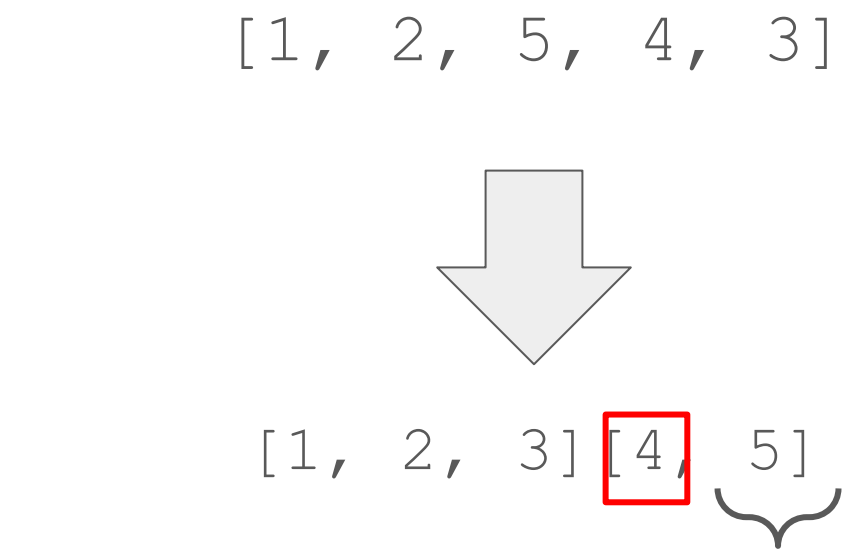
Selection Sort



iteration
count = 4+3+2

Sorting Algorithms

Selection Sort

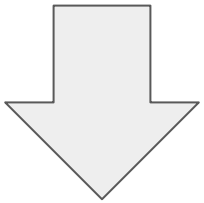


iteration
count = 4+3+2+1

Sorting Algorithms

Selection Sort

[1, 2, 5, 4, 3]



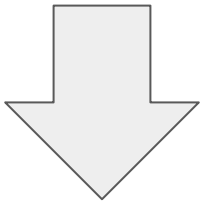
[1, 2, 3, 4][5]

iteration
count = 4+3+2+1+0

Sorting Algorithms

Selection Sort

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



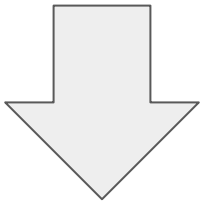
```
[1, 2, 3, 4, 5][]
```

iteration
count = 10

Sorting Algorithms

Insertion Sort

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

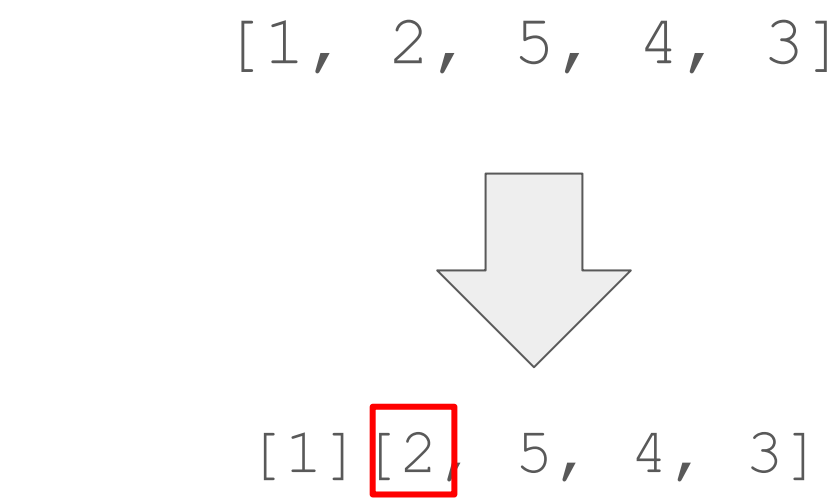


```
[1][2, 5, 4, 3]
```

iteration
count = 0

Sorting Algorithms

Insertion Sort

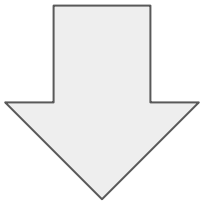


iteration
count = 1

Sorting Algorithms

Insertion Sort

[1, 2, 5, 4, 3]



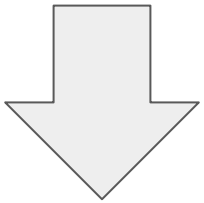
[1, 2][5, 4, 3]

iteration
count = 1+1

Sorting Algorithms

Insertion Sort

[1, 2, 5, 4, 3]



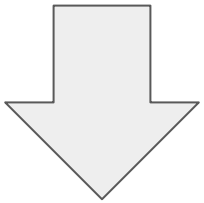
[1, 2, 5][4, 3]

iteration
count = 1+1+1

Sorting Algorithms

Insertion Sort

[1, 2, 5, 4, 3]



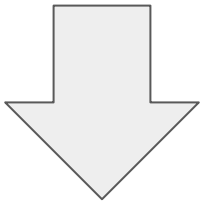
[1, 2, 4, 5][3]

iteration
count = 1+1+1+1

Sorting Algorithms

Insertion Sort

[1, 2, 5, 4, 3]



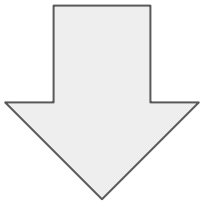
[1, 2, 4, 5][3]

iteration
count = 1+1+1+1+1

Sorting Algorithms

Insertion Sort

[1, 2, 5, 4, 3]



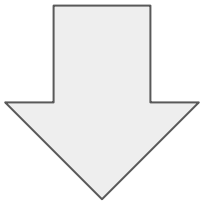
[1, 2, 4, 3, 5][]

iteration
count = 1+1+1+1+1+1

Sorting Algorithms

Insertion Sort

[1, 2, 5, 4, 3]



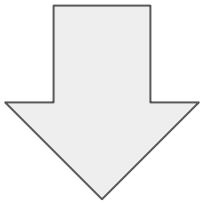
[1, 2, 3, 4, 5][]

iteration
count = 1+1+1+1+1+1+1

Sorting Algorithms

Insertion Sort

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



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

iteration
count = 1+1+1+1+1+1+1= 7

Sorting Algorithms

- Conclusions:
  - Selection sort performance is not affected if the input collection is (somehow) sorted
  - Insertion sort performance improves as the input collection is more sorted
  - In fact, insertion sort has linear time complexity if the input is completed sorted