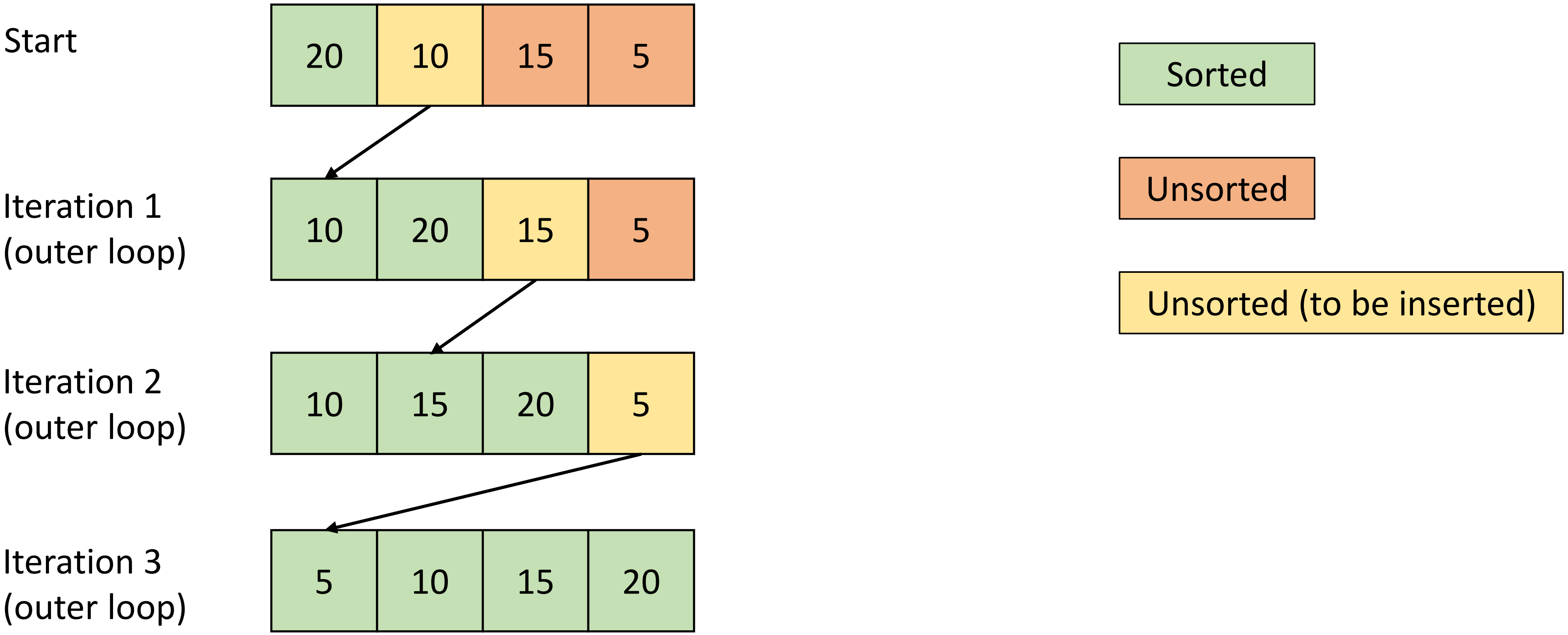
Sorting Algorithms

Insertion and Selection

Alan Johnson 2023

What is Insertion Sort?

- Split array into **sorted** and **unsorted** section
- Select item in index 1 and insert into propper sorted order
- Repeat previous step for all items in the array



Start	20	10	15	5
Iteration 1 (outer loop)	10	20	15	5
Iteration 2 (outer loop)	10	15	20	5
Iteration 3 (outer loop)	5	10	15	20

🟩 Sorted

🟥 Unsorted

🟨 Unsorted (to be inserted)

Insertion Sort – Tracing

Array = {4, 20, 9, 16, 1, 10, 6, 5}

4, 20, 9, 16, 1, 10, 6, 5

4, 20, 9, 16, 1, 10, 6, 5

4, 9, 20, 16, 1, 10, 6, 5

4, 9, 16, 20, 1, 10, 6, 5

1, 4, 9, 16, 20, 10, 6, 5

1, 4, 9, 10, 16, 20, 6, 5

1, 4, 6, 9, 10, 16, 20, 5

1, 4, 5, 6, 9, 10, 16, 20

Insertion Sort – Code

```
void insertionSort(int arr[], int size){
    for(int i = 1; i < size; i++){
        int j = i;
        while(j > 0 && arr[j] < arr[j-1]){
            swap(&arr[j-1], &arr[j]);
            j--;
        }
    }
}
```

Swap adjacent nodes

```
void insertionSort(int arr[], int size){
    for(int i = 1; i < size; i++){
        int next = arr[i];
        int j = i -1;
        while(j >= 0 && arr[j] > next){
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j+1] = next;
    }
}
```

Saving next item to be inserted

Shift sorted items to make place for next

Insert next to correct location

Insertion Sort – Code

```
void insertionSort(int arr[], int size){
      for(int i = 1; i < size; i++){
            int j = i;
            while(j > 0 && arr[j] < arr[j-1]){
                  swap(&arr[j-1], &arr[j]);
                  j--;
            }
      }
 }
```

```
void insertionSort(int arr[], int size){
      for(int i = 1; i < size; i++){
            int next = arr[i];
            int j = i -1;
            while(j >= 0 && arr[j] > next){
                  arr[j + 1] = arr[j];
                  j--;
            }
            arr[j+1] = next;
      }
 }
```

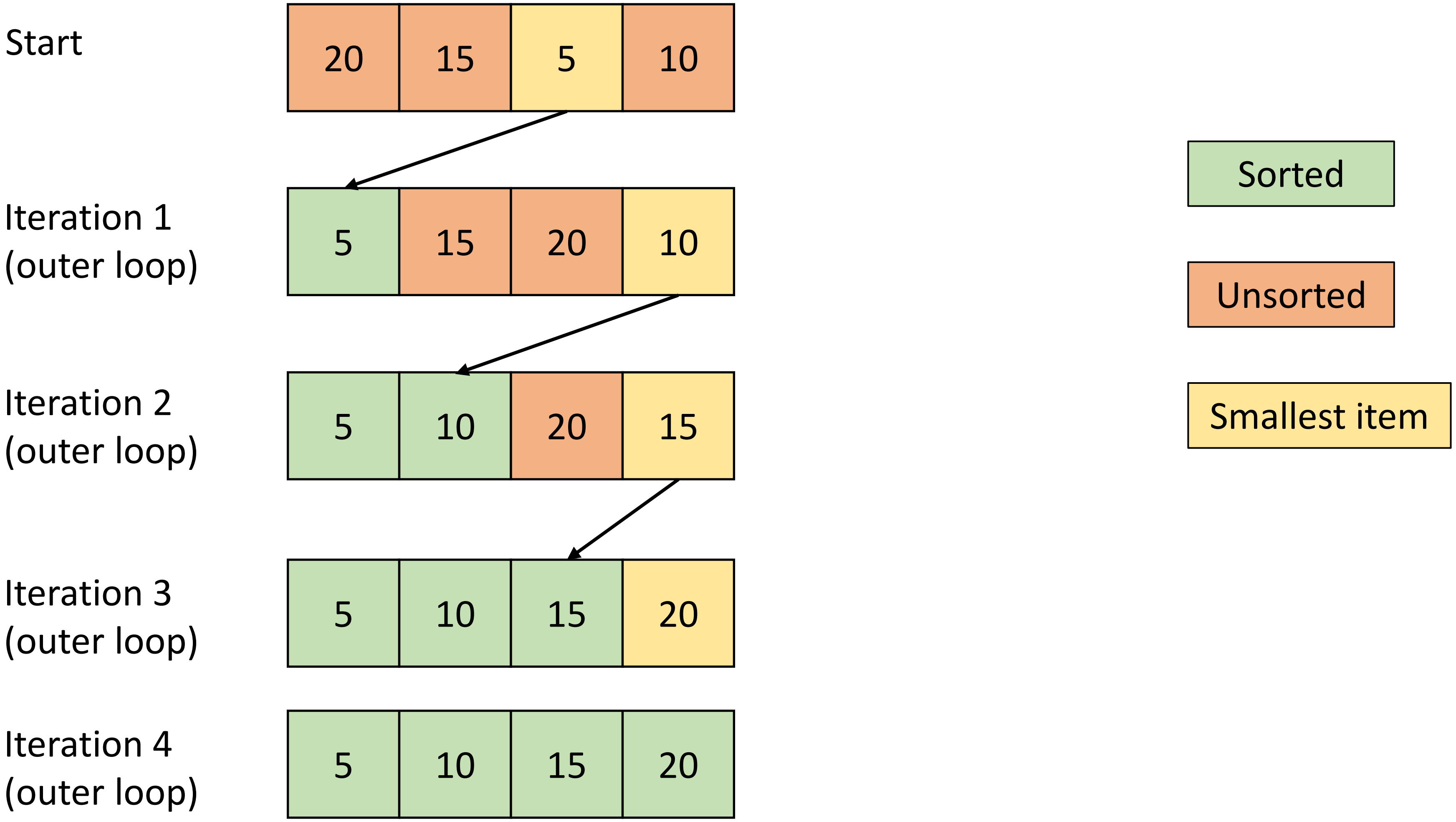
Insertion Sort – Time Complexity

- Outer loop executes (n-1) times
- Number of times inner loop executes depends on input
    - Best Case: array is already sorted so (arr[j] < arr[j-1]) is **always false**
        - No insertion/swapping occurs
    - Worst Case: array is sorted in reverse order so (arr[j] < arr[j-1]) is **always true**
        - Insertion to the front of the array

Worst Case: O(N^2)
Best Case: O(N)

What is Selection Sort?

- Find the smallest (or largest) item *x* in range of [0, n-1]
- Swap *x* with *ith* item
- Increment *i* by 1 and repeat previous steps



Start	20	15	5	10
Iteration 1 (outer loop)	5	15	20	10
Iteration 2 (outer loop)	5	10	20	15
Iteration 3 (outer loop)	5	10	15	20
Iteration 4 (outer loop)	5	10	15	20

Selection Sort – Tracing

Array = {4, 20, 9, 16, 1, 10, 6, 5}

4, 20, 9, 16, 1, 10, 6, 5
1, 20, 9, 16, 4, 10, 6, 5
1, 4, 9, 16, 20, 10, 6, 5
1, 4, 5, 16, 20, 10, 6, 9
1, 4, 5, 6, 20, 10, 16, 9
1, 4, 5, 6, 9, 10, 16, 20
1, 4, 5, 6, 9, 10, 16, 20
1, 4, 5, 6, 9, 10, 16, 20
1, 4, 5, 6, 9, 10, 16, 20

Selection Sort - Code

```
void selectionSort(int arr[], int size){
    int min_index;
    for(int i = 0; i < size-1; i++){
        min_index = i;
        for(int j = i+1; j < size; j++){
            if(arr[j] < arr[min_index]){
                min_index = j;
            }
            swap(&arr[min_index], &arr[i]);
        }
    }
}
```

Search for min element

```
void selectionSort(int arr[], int size){
    int max_index;
    for(int i = size-1; i >= 1; i--){
        max_index = i;
        for(int j = 0; j < i; j++){
            if(arr[j] > arr[max_index]){
                max_index = j;
            }
            swap(&arr[max_index], &arr[i]);
        }
    }
}
```

Search for max element

Selection Sort - Code

```
void selectionSort(int arr[], int size){
    int min_index;
    for(int i = 0; i < size-1; i++){
        min_index = i;
        for(int j = i+1; j < size; j++){
            if(arr[j] < arr[min_index]){
                min_index = j;
            }
            swap(&arr[min_index], &arr[i]);
        }
    }
 }
```

```
void selectionSort(int arr[], int size){
    int max_index;
    for(int i = size-1; i >= 1; i--){
        max_index = i;
        for(int j = 0; j < i; j++){
            if(arr[j] > arr[max_index]){
                max_index = j;
            }
            swap(&arr[max_index], &arr[i]);
        }
    }
 }
```

Selection Sort – Time Complexity

- Outer loop executes (n-1) times
- Inner loop executes size of unsorted section – 1
    - Best case, worst case, and average case of selection sort is same

Worst Case: O(N^2)
Best Case: O(N^2)