ECE 220: Computer Systems & Programming

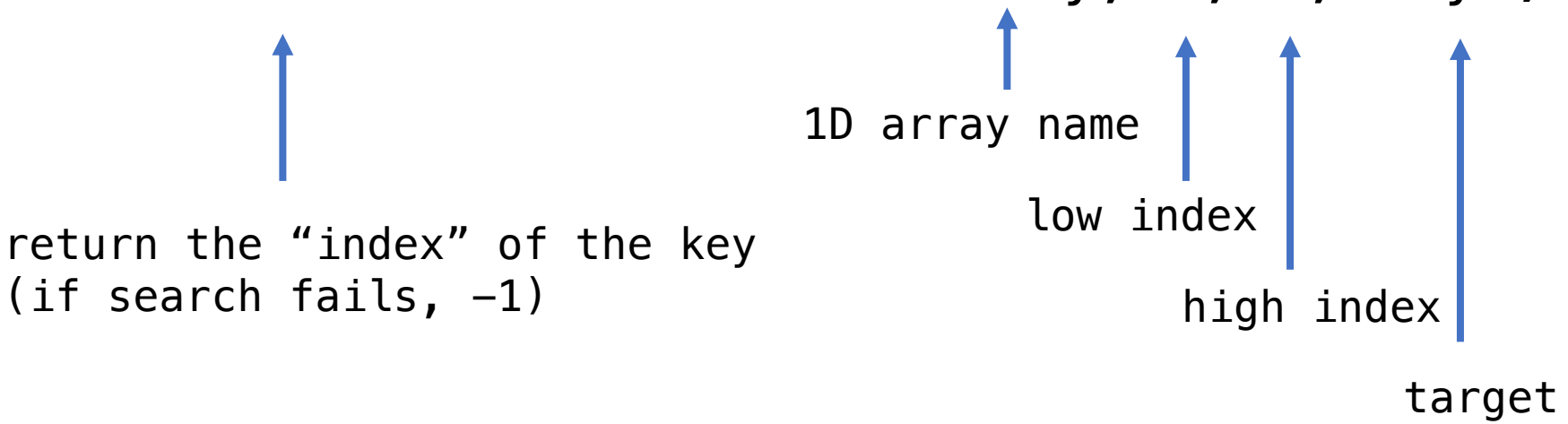
Lecture 11: Problem Solving with Pointers and Arrays

Thomas Moon

February 22, 2024

Search an item – Linear search

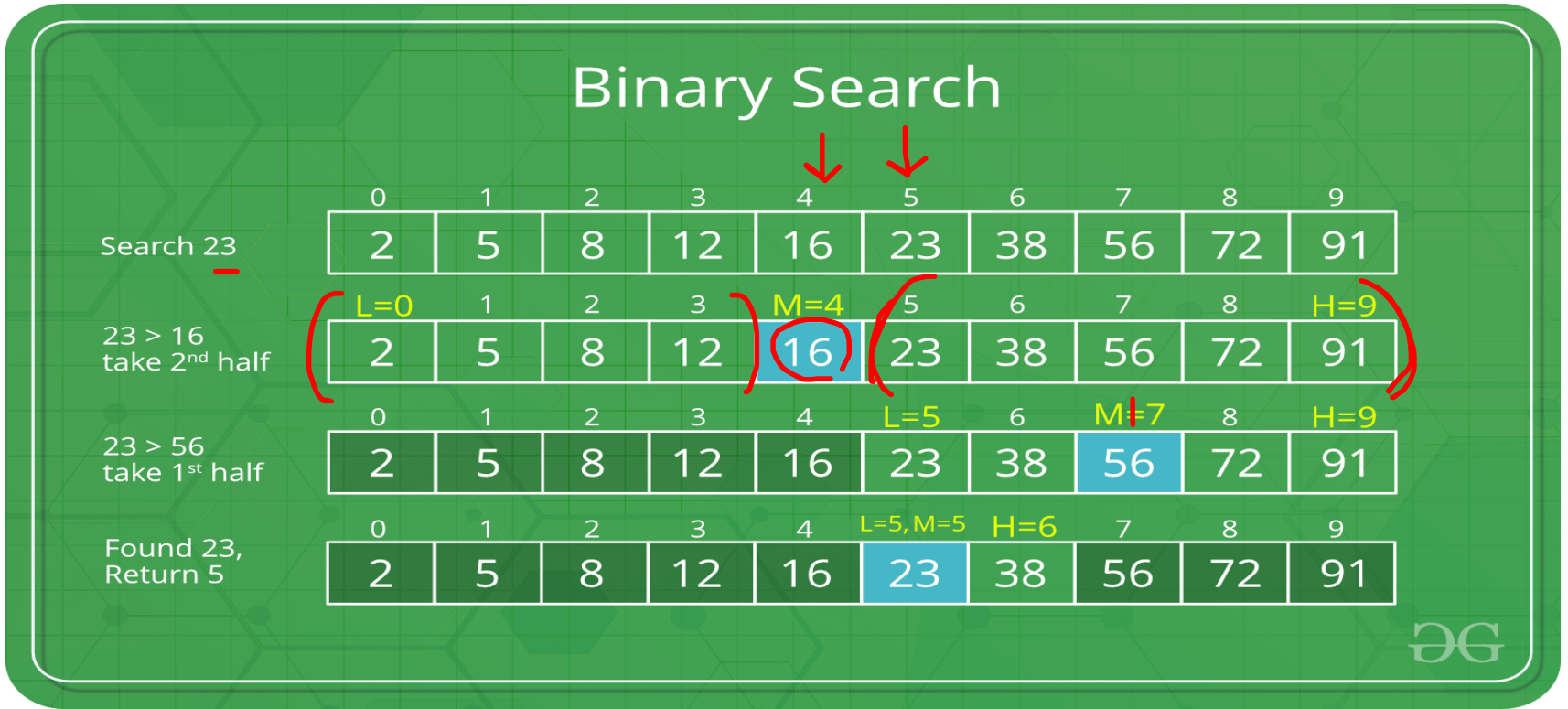
```
int array[] = {11, 12, 22, 25, 34, 64};
int key=22, found;

found = linearSearch(array, 0, 5, key);
```



```
int linearSearch(int array[], int l, int h, int key)
{
    int i;
    for(i = l; i <= h;i++_____){
        if(array[i] == key_____________________)
            return  i____;
    }
    return -1;
}
```

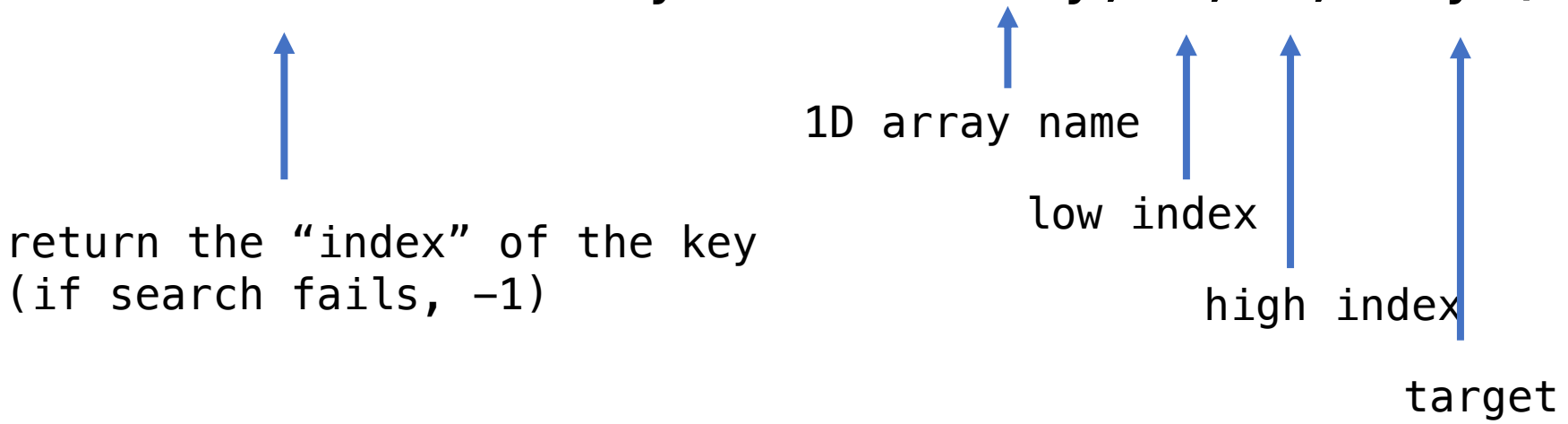
1. Search Algorithms

- Binary search (for sorted array)
  1. Repeat 2 and 3 until search space contains no items
  2. Find the middle of array and check if it's the search key
  3. If key < middle → search the first half
     If key > middle → search the second half
     If key == middle → return the key



Implement linear search and binary search

```
int array[] = {11, 12, 22, 25, 34, 64};
int key=22, found;

//found = linearSearch(array, 0, 5, key);
found = binarySearch(array, 0, 5, key);
```



```
int binarySearch(int array[], int l, int h, int key)
{
    int mid;
    while( l <= h_____){
        mid = (l+h)/2;<-
        if(key < array[mid])
            h = mid -1____________________;
        else if(key > array[mid])
            l = mid +1____________________;
        else
            return mid;
    }
    return -1;
}
```

2. Sort Algorithms

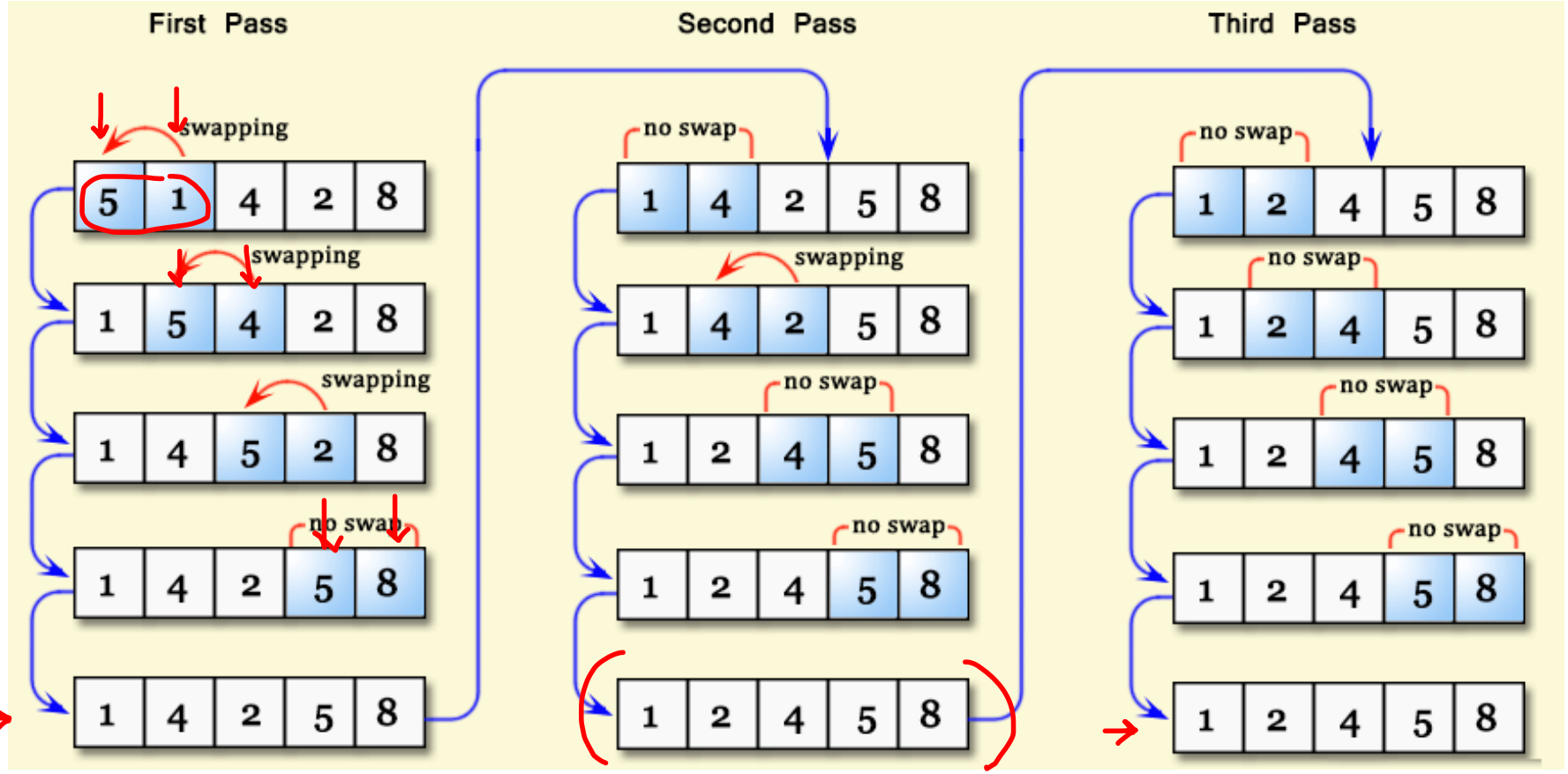
- Bubble sort
- Insertion sort
- Selection sort
- Merge sort
- Quick sort

....

https://visualgo.net/en/sorting

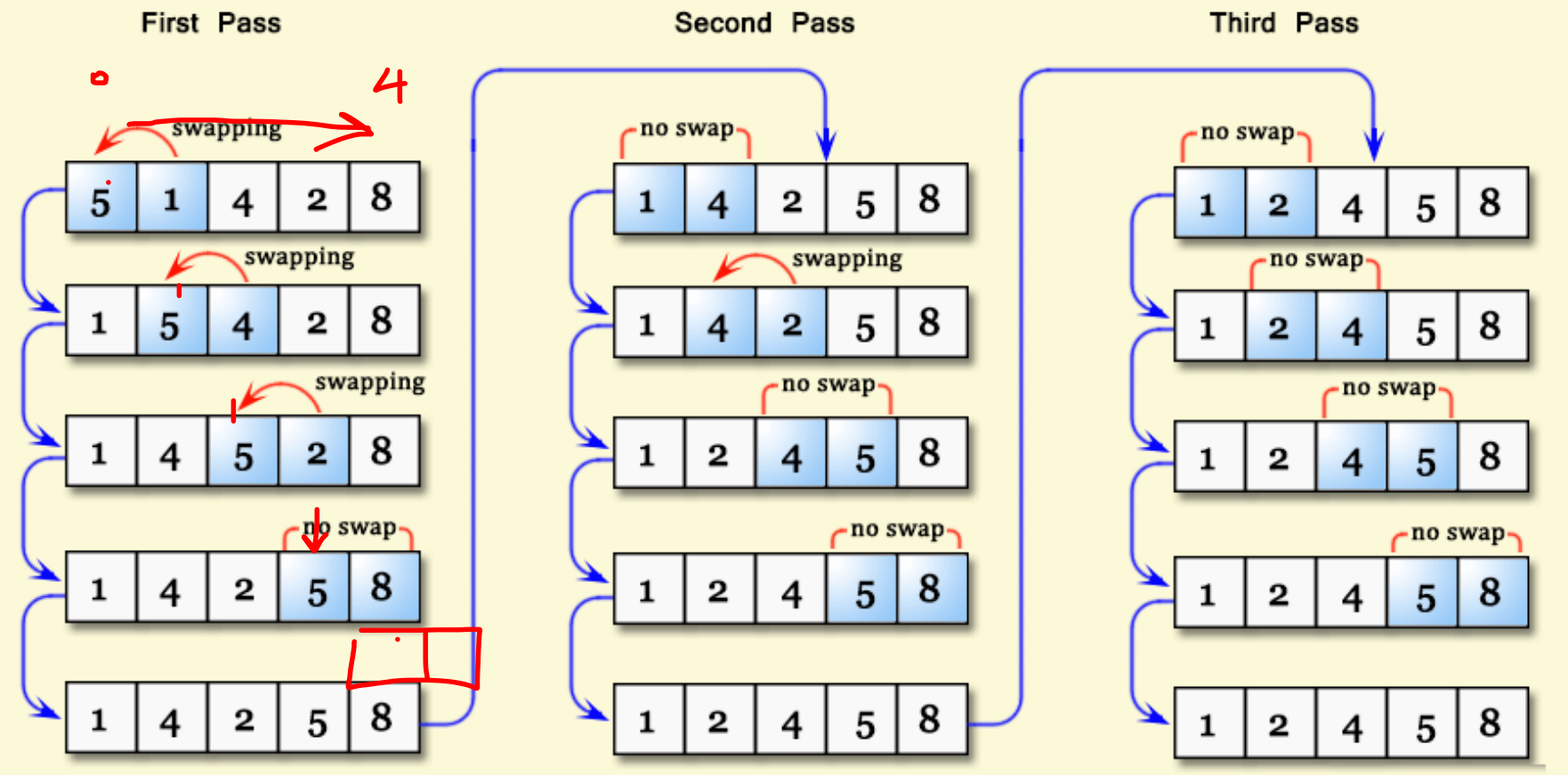
Bubble Sort

1. Compare items next to each other and swap them if needed.
2. Repeat this process until the entire array is sorted.



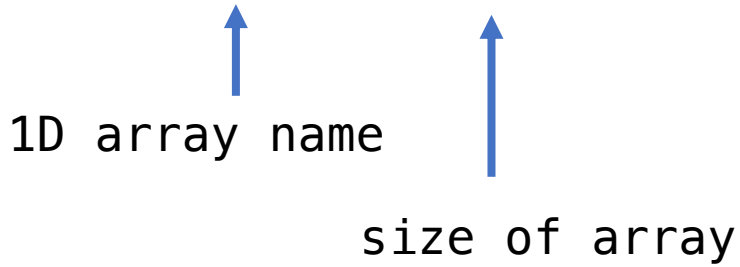
Bubble Sort

1. How many different loops? 2
2. If the array is already sorted, how many “Pass” we need? 1



Implement Bubble sort

```
int array[SIZE] = {64, 34, 25, 12, 22, 11, 1};

bubbleSort(array, SIZE);
```

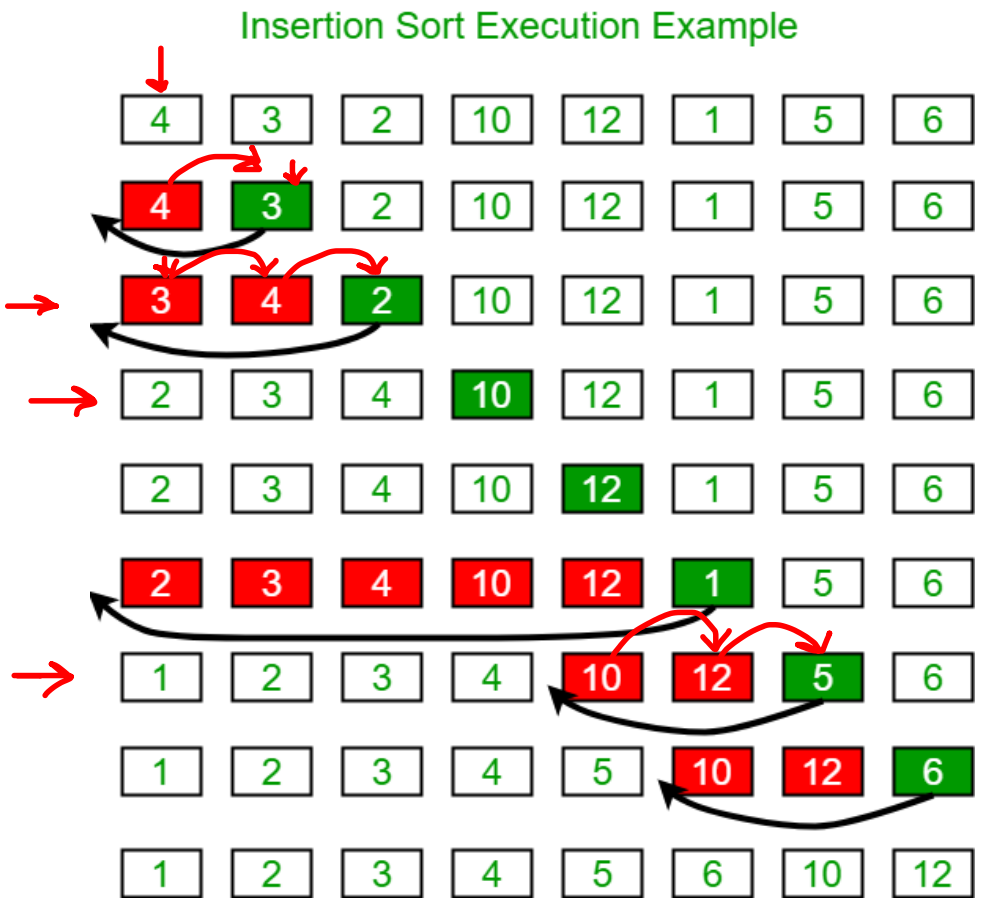


Implement Bubble sort

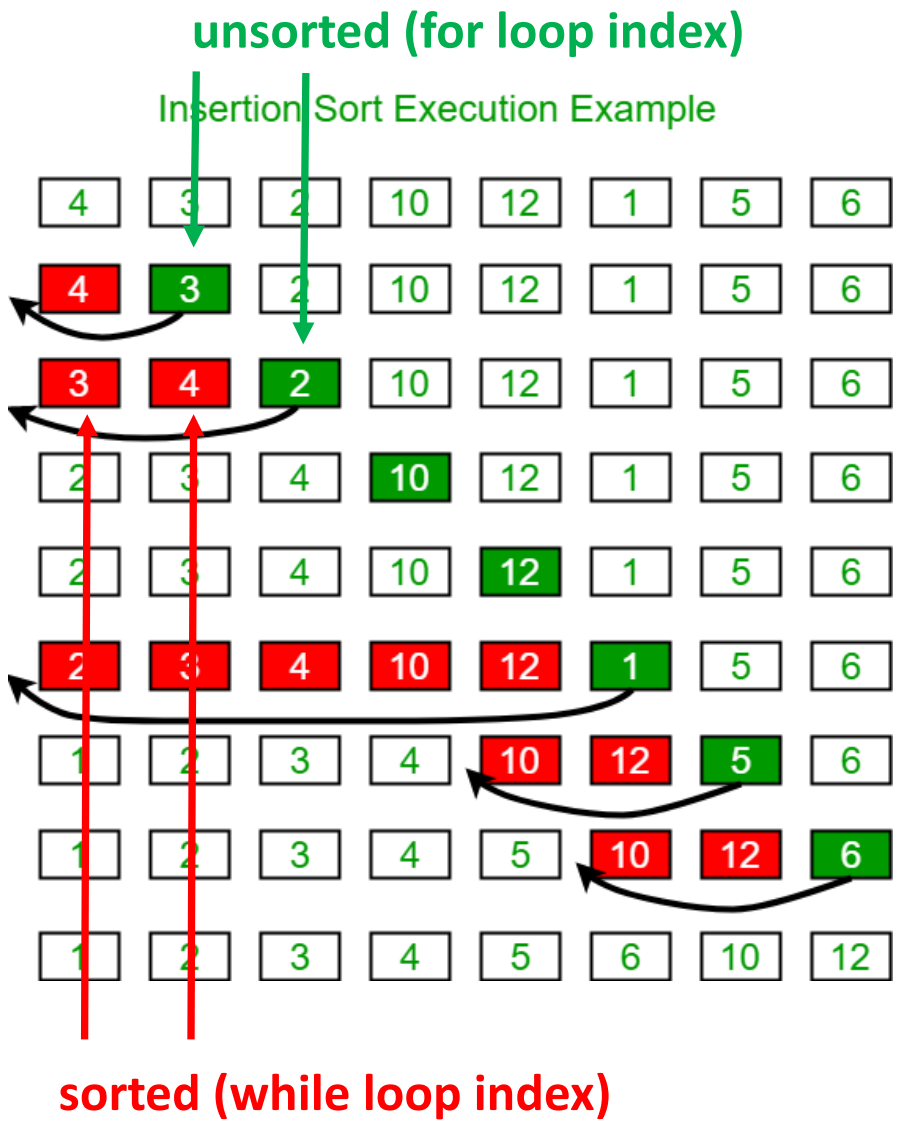
```
void swap(int *firstVal, int *secondVal);
void bubbleSort(int array[], int size)
{
    int i, is_swapped;
    do{
        is_swapped = 0;
        for(i=0; i< size-1_; i++){
            if(array[i] > array[i+1]_________){
                swap(  &array[i], & array[i+1] );
                is_swapped = 1;
            }
        }
    }while(is_swapped !=0 );
}
```

Insertion Sort

1. Remove item from array, insert it at the proper location in the sorted part by shifting other items.
2. Repeat this process until the end of array is reached.



unsortedItem = Green item value



```
void insertSort(int array[], int n)
{
    int unsorted;   // Index for unsorted list items
    int sorted;     // Index for sorted items
    int unsortedItem;   // Current item to be sorted
    // Loop from 1 thru n
    for(unsorted=1; unsorted<n; unsorted++){
        unsortedItem = array[unsorted];
        // Loop from unsorted-1 thru 0
        // if unsortedItem < current sorted item, shift the current item to
the right
        // stop if we hit a smaller element than unsortedItem
        sorted = unsorted-1;
        while(unsortedItem < arrary[sorted] && sorted>=0){
            array[sorted+1]= array[sorted]          ;
            sorted--;
        }
        array[sorted+1] = unsortedItem;
    }
}
```