# ECE 220 Computer Systems & Programming

## Lecture 11: Problem Solving with Pointers and Arrays

ECE ILLINOIS

ILLINOIS

# 2. Sort Algorithms

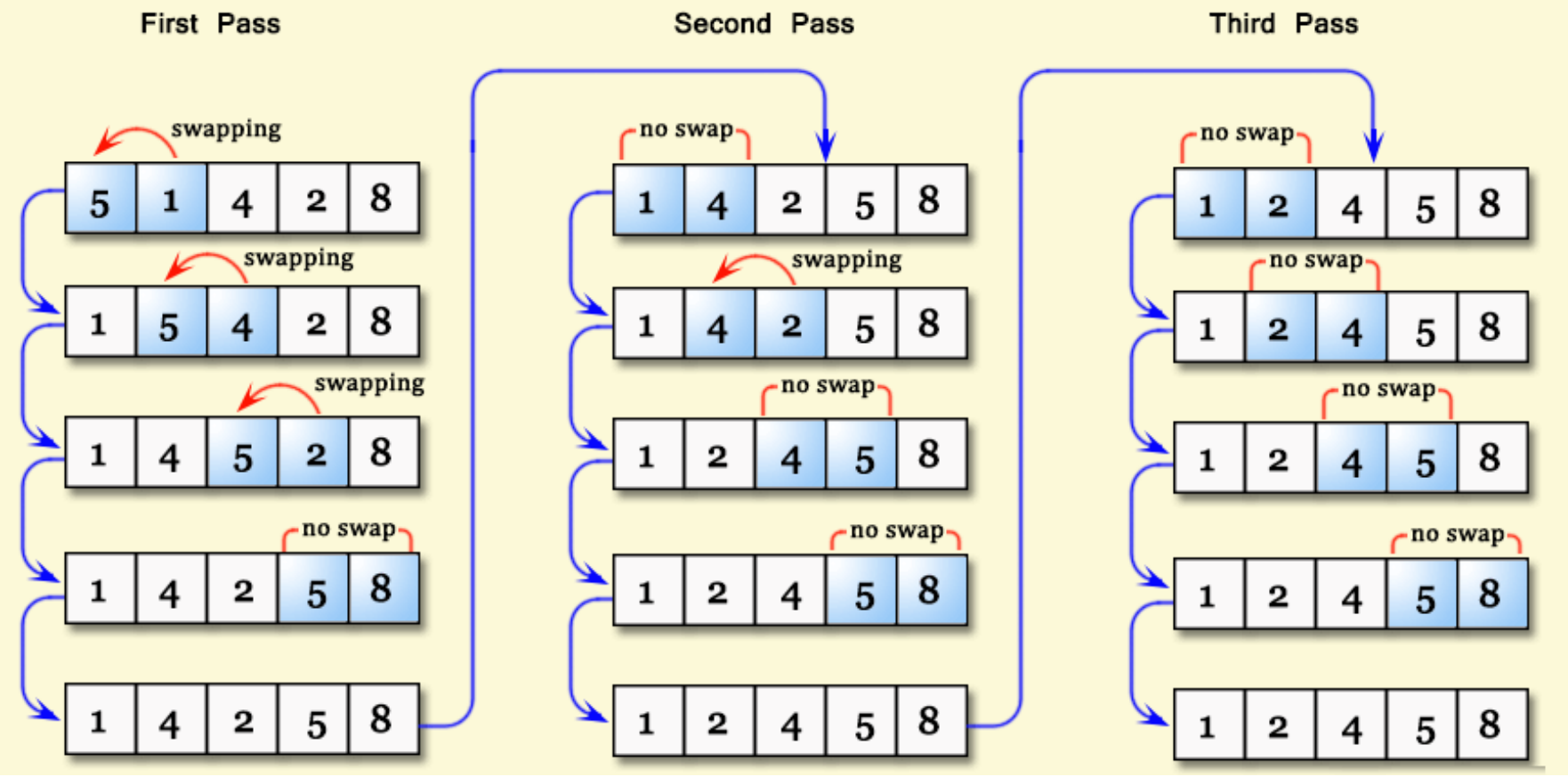
- Bubble sort
- Insertion sort
- Selection sort
- Merge sort
- Quick sort

….

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.



```c
void swap_sort(                 )
{
    int temp;



}
```

```c
#include "sort_header.h"
void bubble_sort(int *a, int size)
{

    int  i;


        for(i=0;i<size-1;i++)
        {

            {
                swap_sort(&a[i],&a[i+1]);

            }
        }


}
```

```c
#include "sort_header.h"
void bubble_sort(int *a, int size)
{

    int flag, i;
    do{
        flag=0;
        for(i=0;i<size-1;i++)
        {
            if(a[i]>a[i+1])
            {
                swap_sort(&a[i],&a[i+1]);
                flag=1;
            }
        }
    }while(flag);

}
```

```c
void swap_sort(int *a, int *b)
{
    int temp;
    temp=*a;
    *a=*b;
    *b=temp;
}
```

# 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.

6 5 3 1 8 7 2 4

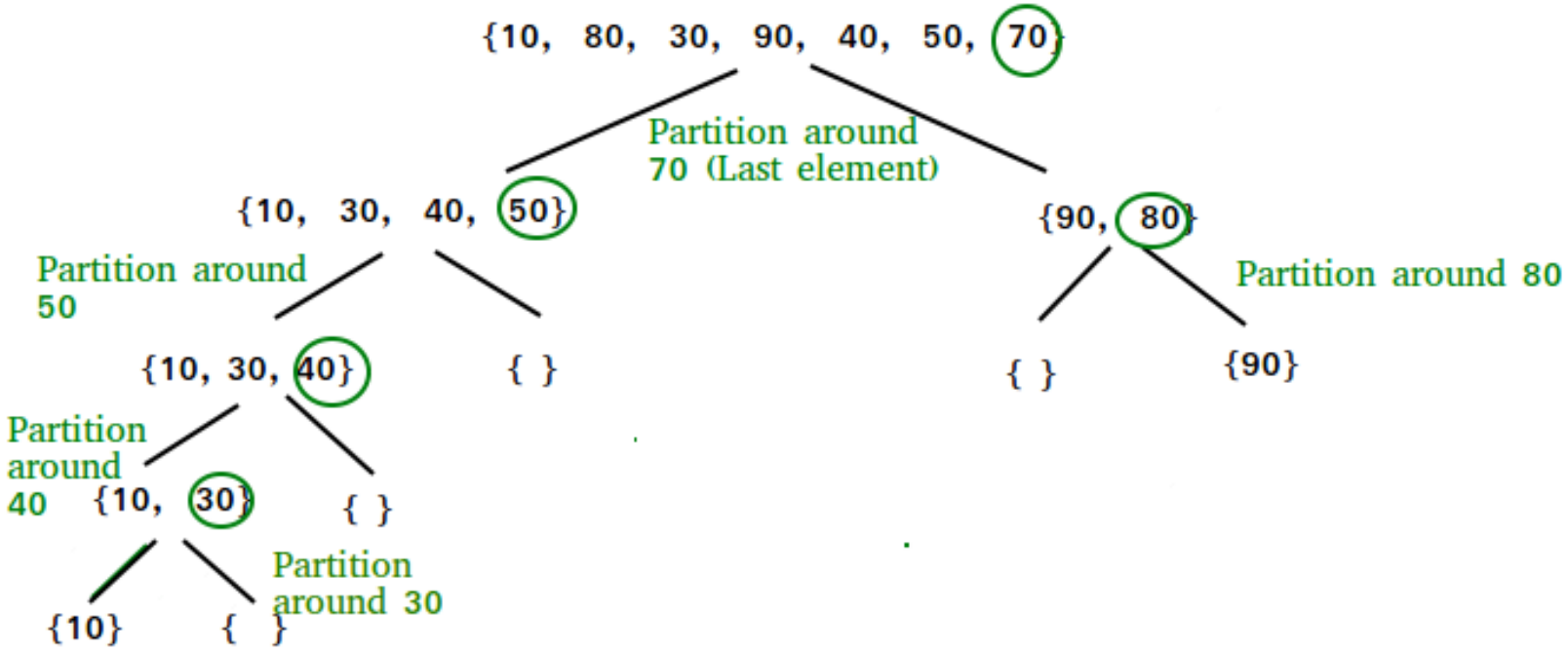
```
void insert_sort(int *a, int size)
{

    int sorted_ind,unsortedItem,unsorted_ind;

    for(unsorted_ind=1;unsorted_ind<size;unsorted_ind++)
    {
        unsortedItem=;
        sorted_ind=;
        while(  )
        {
            // slide the current book to the right


            sorted_ind--;
        }

        //insert the unsortemItem to i


    }

}
```

## 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.

```c
void insert_sort(int *a, int size)
{

    int sorted_ind,unsortedItem,unsorted_ind;

    for(unsorted_ind=1;unsorted_ind<size;unsorted_ind++)
    {
        unsortedItem=a[unsorted_ind];
        sorted_ind=unsorted_ind-1;
        while((sorted_ind>=0) && (unsortedItem<a[sorted_ind]))
        {
            a[sorted_ind+1]=a[sorted_ind];
            sorted_ind--;
        }

        a[sorted_ind+1]=unsortedItem;
    }

}
```

# Quick Sort



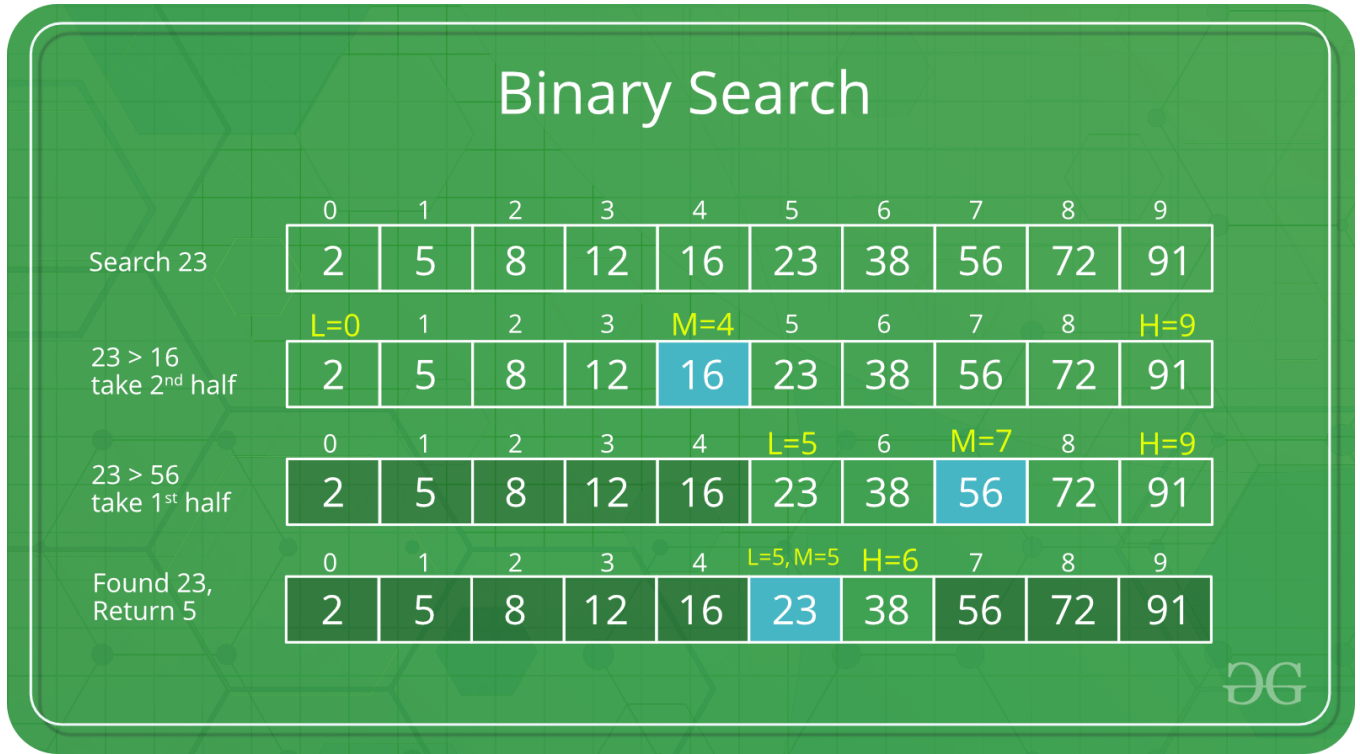

Ref: https://www.geeksforgeeks.org/quick-sort/

```
int partition(int array[], int l, int h)
```

```
void quickSort(int array[], int l, int h)
```

# Search Algorithms

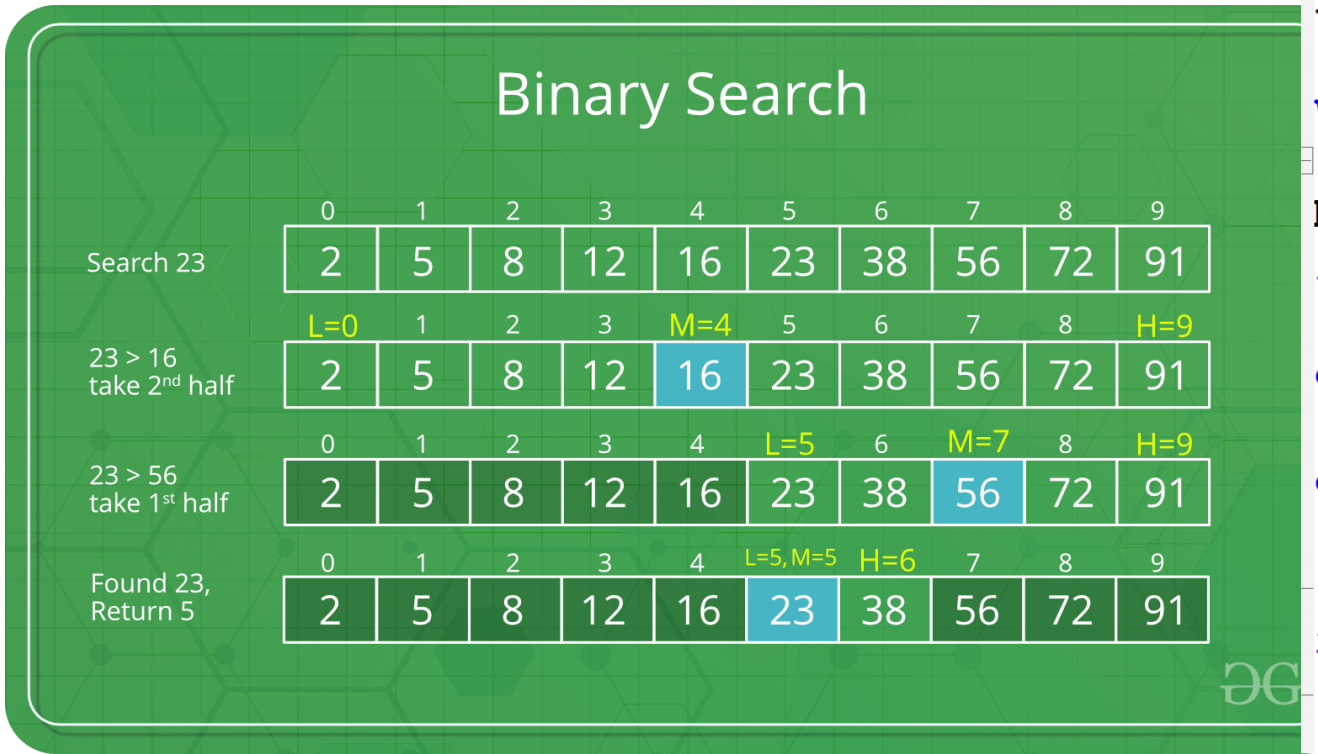
- Binary search (for sorted array)
  1. Find the middle of array and check if it's the search key
  2. If key < middle → search the first half
     If key > middle → search the second half
     If key == middle → return the key
  3. Repeat 1 and 2 until search space contains no items



```c
int binary_search(int *a, int size, int key)
{
int l, h, M;
l=0;
h=size-1;

while(h>=l)
{
M=;
if(key==a[M])
    return M;



}
return -1;
}
```

# 1. Search Algorithms

- Binary search (for sorted array)
  1. Find the middle of array and check if it's the search key
  2. If key < middle → search the first half
     If key > middle → search the second half
     If key == middle → return the key
  3. Repeat 1 and 2 until search space c




```c
int binary_search(int *a, int size, int key)
{
int l, h, M;
l=0;
h=size-1;

while(h>=l)
{
M=(l+h)/2;
if(key==a[M])
     return M;
else if(key<a[M])
     h=M-1;
else
     l=M+1;
}
return -1;
}
```