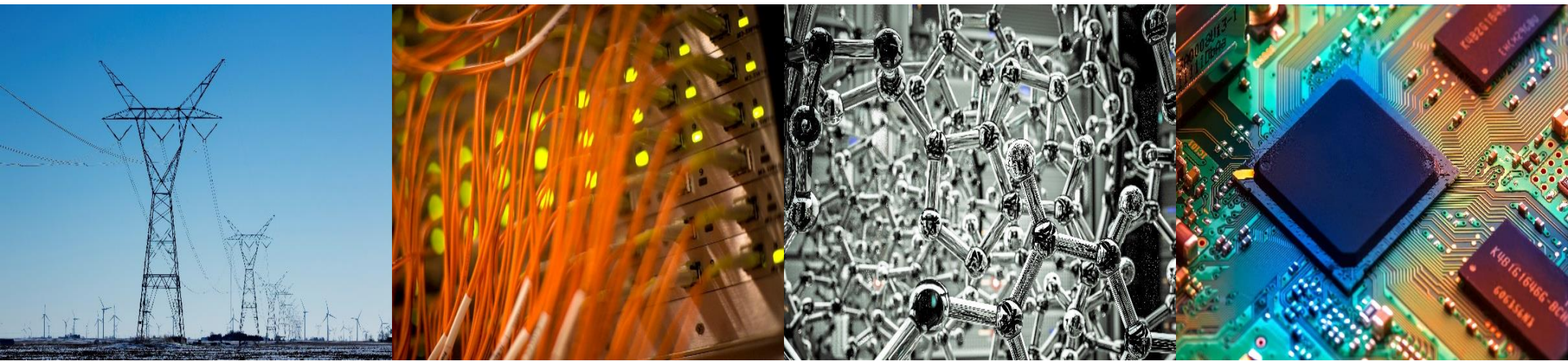


ECE 220 Computer Systems & Programming

Lecture 11 – Problem Solving with Pointers and Arrays

October 2, 2025



- Quiz3 is next week

I ILLINOIS

Electrical & Computer Engineering

GRAINGER COLLEGE OF ENGINEERING

Exercise: implement a function to exchange two rows of an $M \times N$ matrix. This function takes three arguments: pointer to the matrix, row index x and row index y .

```
#define M 3 /* row size */
#define N 4 /* col size */
void row_exchange(int matrix[M][N], int row_x, int row_y){

}
```

Exercise: implement a function to transpose an $M \times N$ input matrix into an $N \times M$ output matrix. Both input and output matrices have been allocated and stored as 1D in memory.

```
#define M 2 /* row size of input matrix */
#define N 3 /* col size of input matrix */
void transpose(int *in_matrix, int *out_matrix){
```

```
}
```

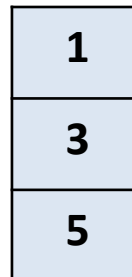
Pointer Array vs. Pointer to an Array

```
/* declare two integer arrays */  
int a[3] = {1,3,5};  
int b[4] = {2,4,6,8};  
  
int *ptr_array[2];  
ptr_array[0] = a;  
ptr_array[1] = b;
```

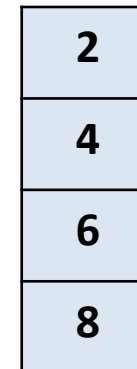
ptr_array



a



b



Search Algorithms

Linear Search: search from the beginning of the array until item is found

Binary Search: (for a sorted array in ascending order)

1. find the **middle** of the array and check if it's the search item;
2. search first half of the array if the search item is smaller than middle, else search the second half;
3. repeat steps 1 & 2 until search item is found.

Search for 23 in a sorted array

2	5	8	12	16	23	38	56	72	91
---	---	---	----	----	----	----	----	----	----

23 > 16 (middle), search second half

L								H	
2	5	8	12	16	23	38	56	72	91

23 < 56 (middle), search first half

					L				H
2	5	8	12	16	23	38	56	72	91

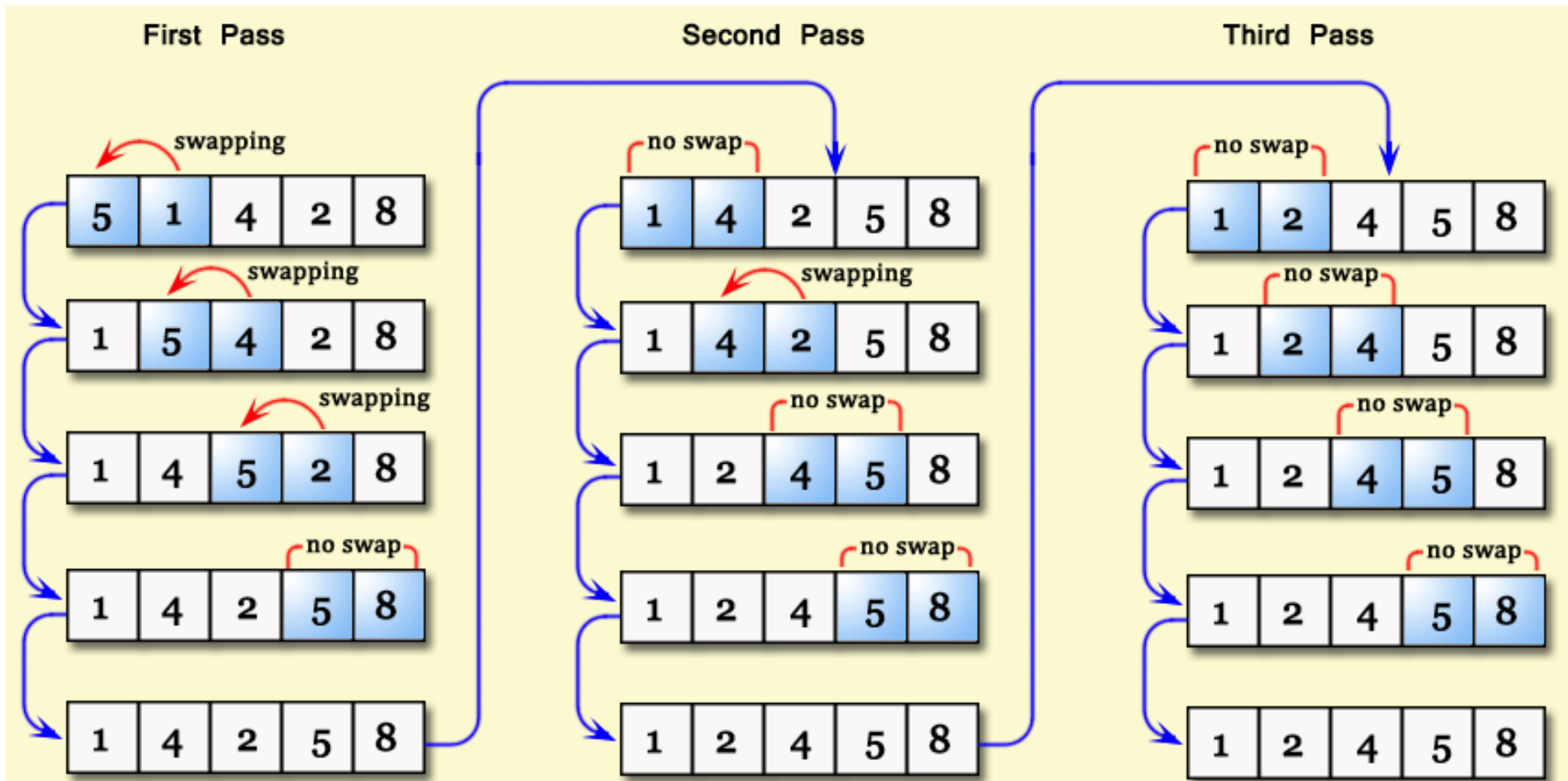
23 == 23 (middle), return 5 (index of 23)

					L	H			
2	5	8	12	16	23	38	56	72	91

5

Sorting Algorithms (<http://visualgo.net/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.



Exercise: implement a function that performs bubble sort

This function takes one argument: a pointer to the array. Note: you can use the swap function:

```
void swap(int *x, int *y);
```

```
#define SIZE 5
```

```
void bubble_sort(int array[]){
```

```
    /* declare necessary variables */
```

```
    do{
```

```
        /* initialize the "swap" indicator */
```

```
        /* go through the entire array to compare and swap adjacent  
           elements */
```

```
    }while(                );
```

```
    /* go through the entire array again if there's a swap */
```

```
}
```

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 reach.

Step 1: assume first item is "sorted"

5	2	6	1	3	9
---	---	---	---	---	---

Step 2: remove the next item from array

5		6	1	3	9
	2				

Step 3: since $5 > 2$, shift 5 to create space

	5	6	1	3	9
	2				

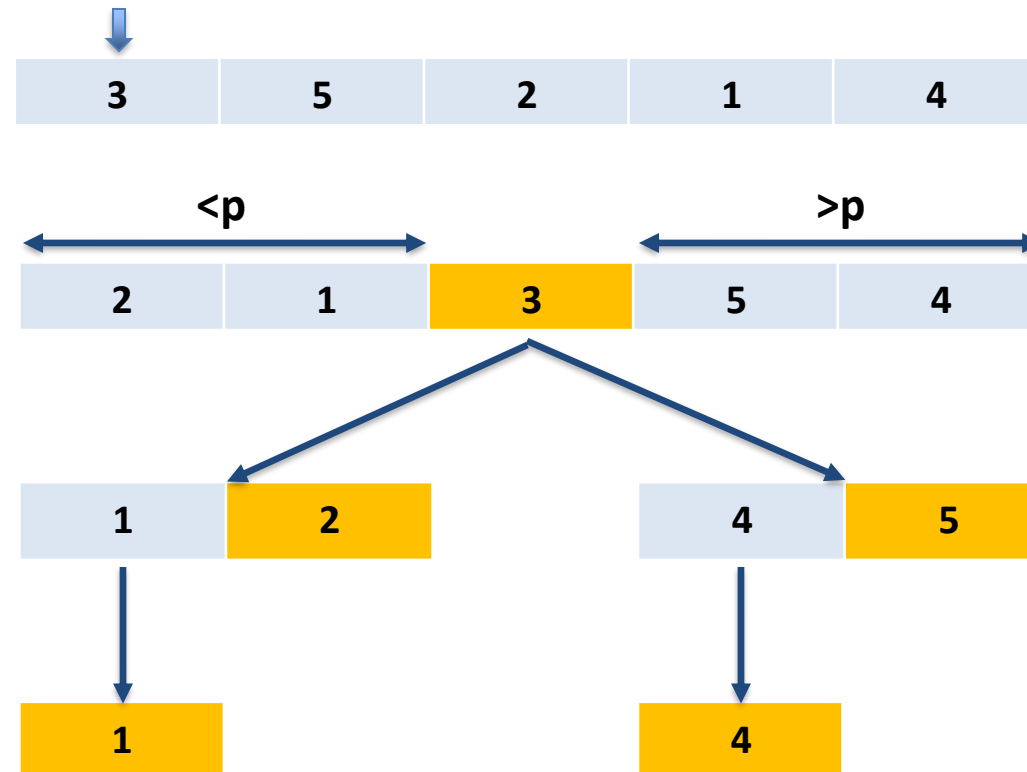
Step 4: insert 2 into the empty space

2	5	6	1	3	9
---	---	---	---	---	---

Quick Sort: also called divide-and-conquer

1. pick a pivot and partition array into 2 subarrays;
2. then sort subarrays using the same method.

Pivot Element p



Sorted Array

