

Searching and Sorting

CIS 15

Functionalia

- Next Class in 2 Wednesdays
- Extra Office Hours Next Week (Tuesday evening)
- No more Office Hours on Thursdays
- HW 1 is Out (Review C++ / Arrays / Buffet)
 - DUE FRIDAY 23rd, 11:59 PM
 - Submission Details TBA!
- 14 out of 19 students e-mailed me their history:

```
$ history > my_history.txt
```

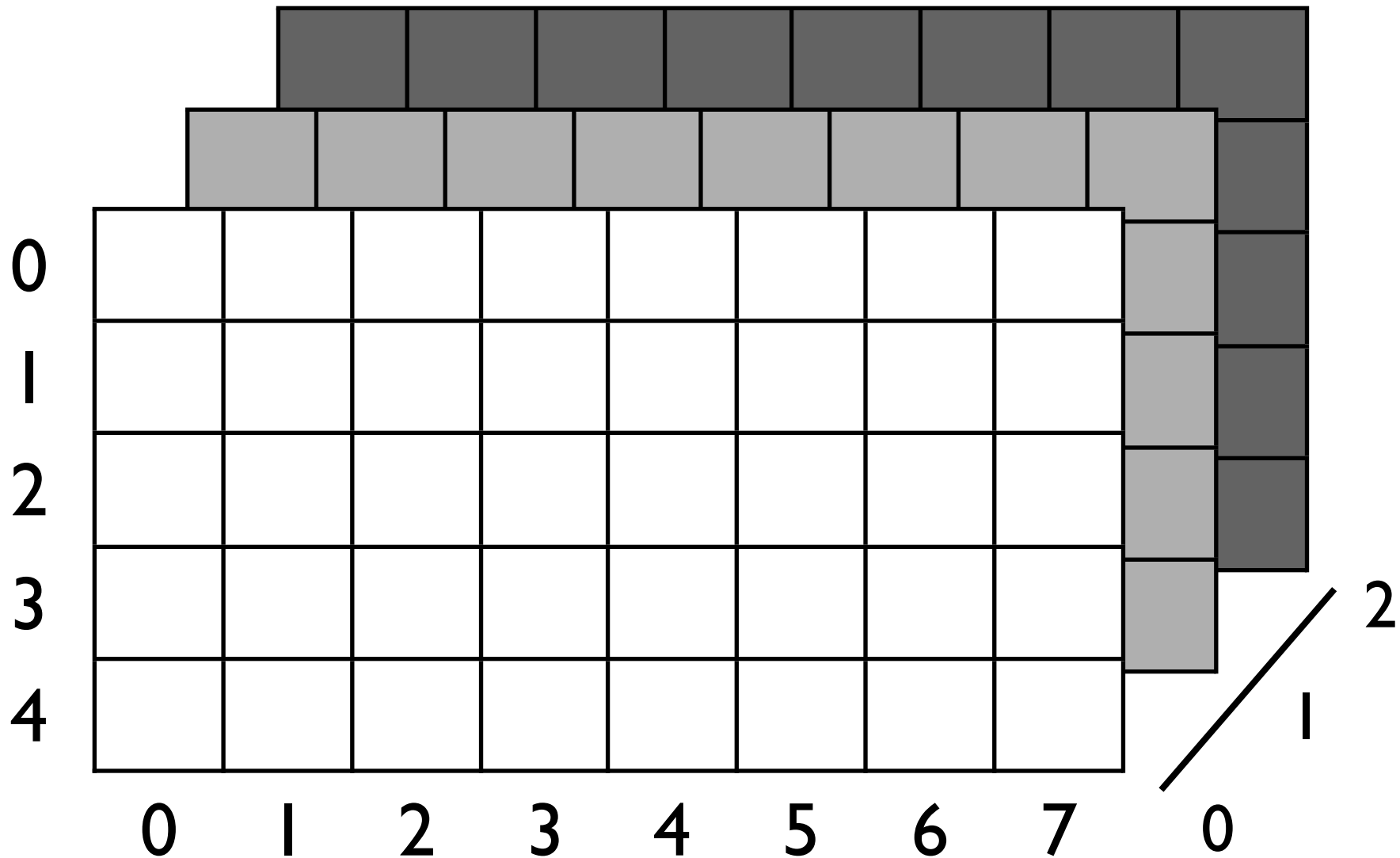
```
$ mail chipp@sci.brooklyn.cuny.edu <
```

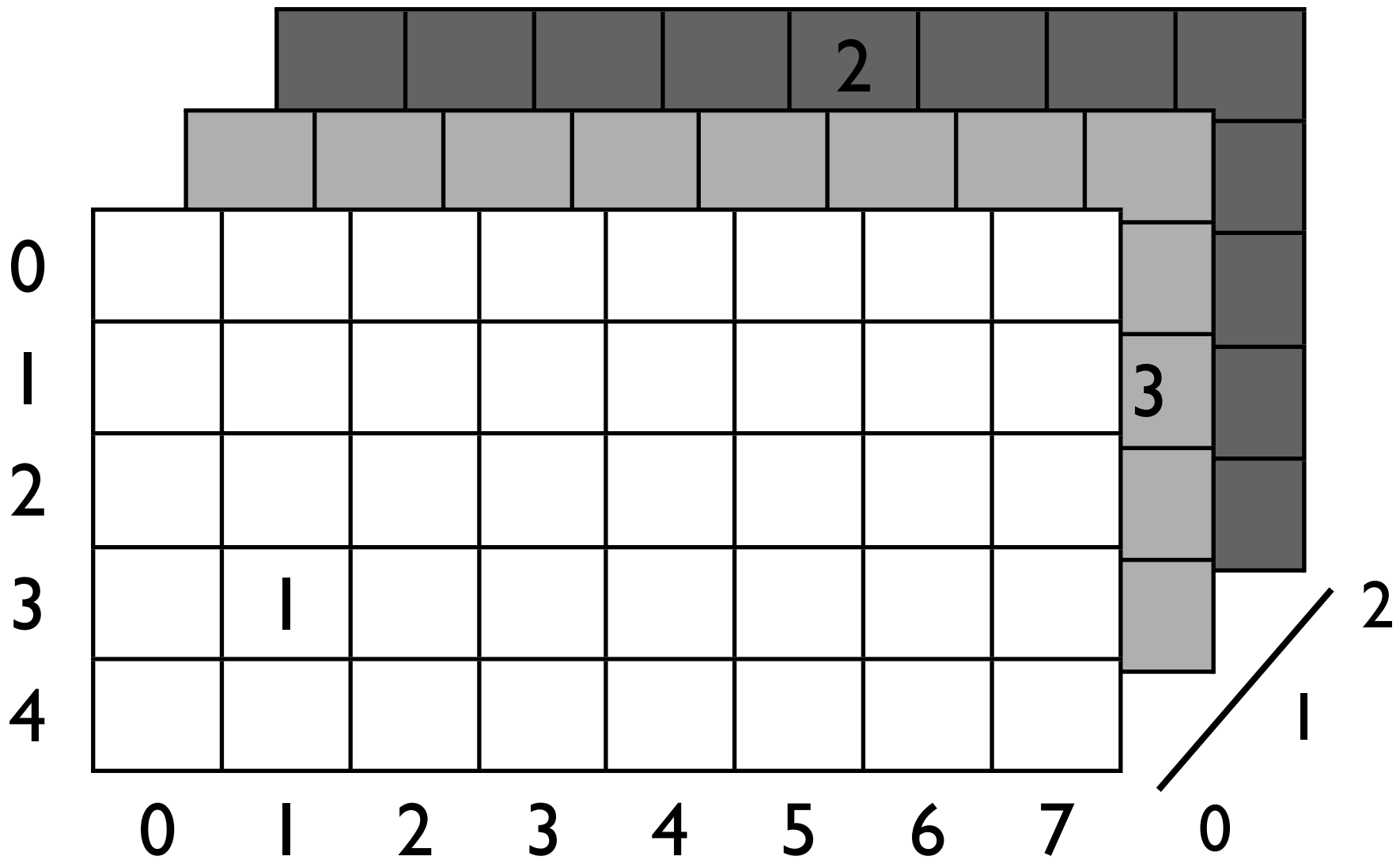
Summary

- Finish Multi-Dimensional Arrays
- Array Sorting
 - Linear Search
 - Binary Search

More than 2 Dimensions

```
int cubes[3][5][8];
```



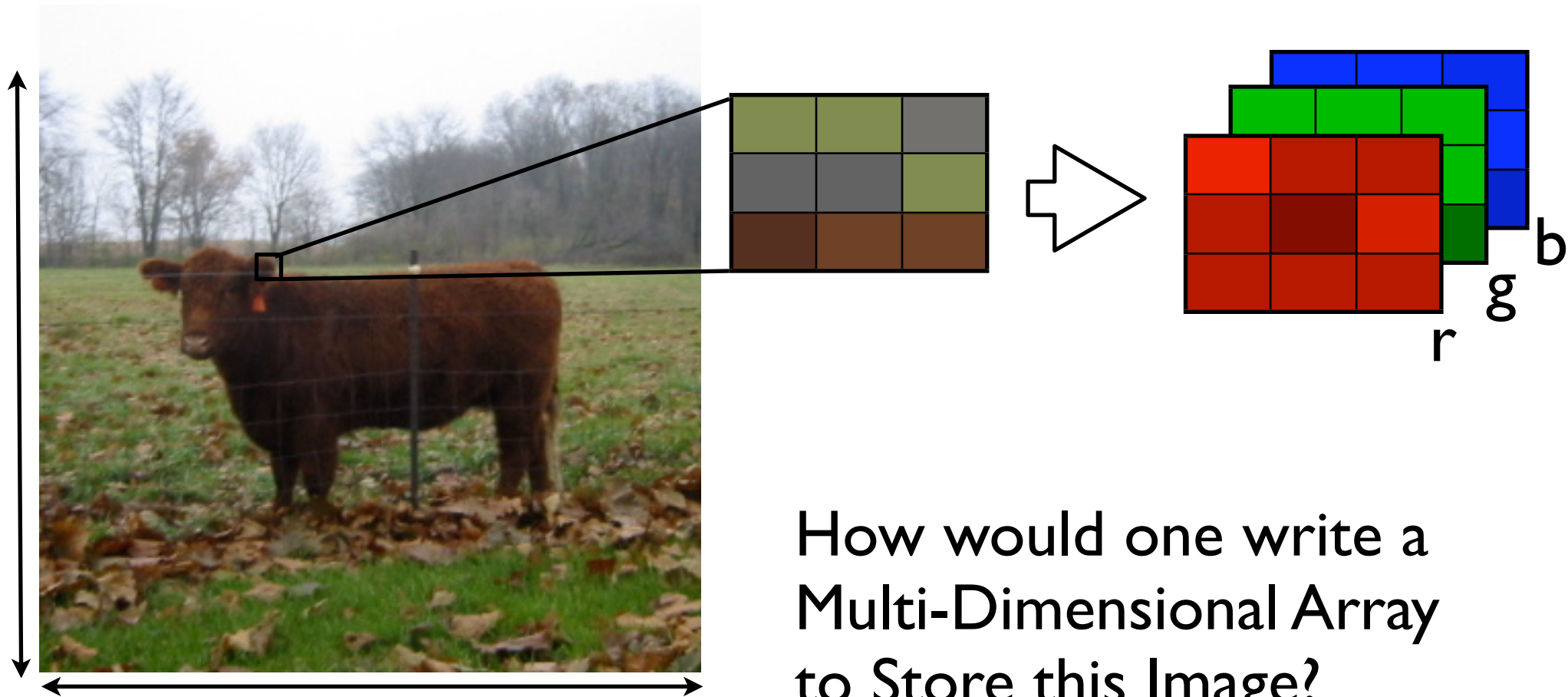


`cubes[0][3][1] = 1;`

`cubes[2][0][4] = 2;`

`cubes[1][2][7] = 3;`

Example: Images Stored As Multi-Dimensional Arrays



How would one write a
Multi-Dimensional Array
to Store this Image?

What about a Photo Album?

Multi-Arrays in Functions

First Dimension is Unspecified (Number of Photos).

```
#define SIZE_X 320
```

```
#define SIZE_Y 240
```

```
#define RGB_DEPTH 3
```

```
void process(int album[][SIZE_Y][SIZE_X][RGB_DEPTH]) ;
```

Search

Arrays are useful in performing search functions.

What is the access time to retrieve an item in an array?
(In other words, how many steps to read and write a member of an array?)

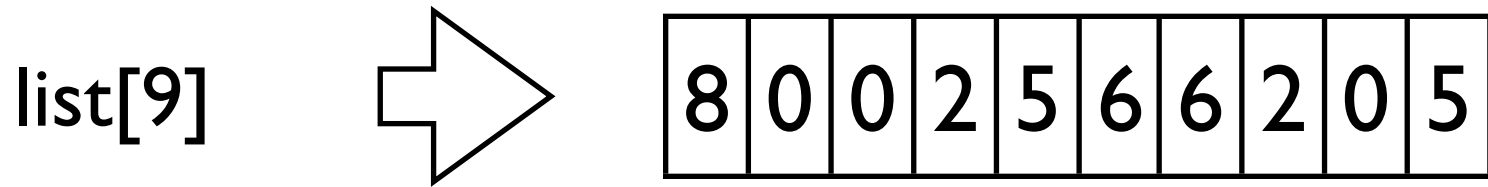


Two Kinds:

Linear Search (Sequential Search)

Binary Search

Linear Search

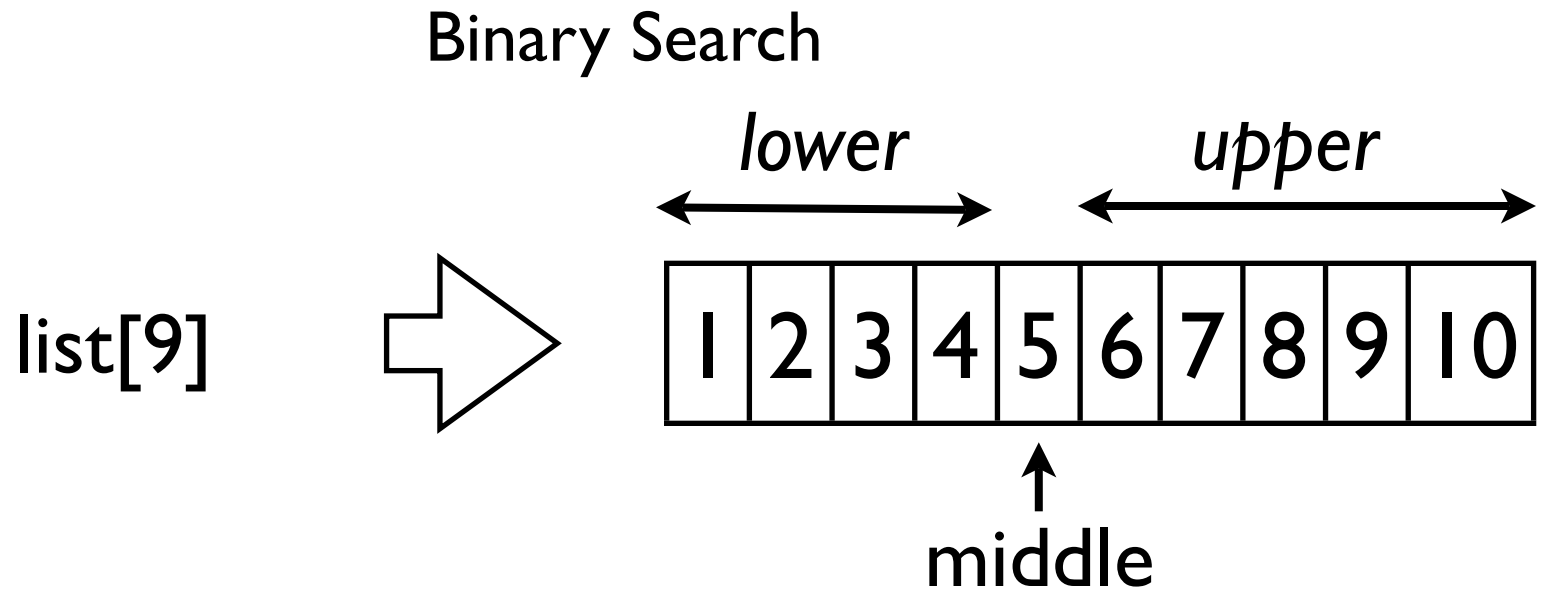


- Goal: Search for the number '6'
- Start at the beginning (i.e. list[0])
- Iterate through the array (step through) till one gets the number 6. What is its array index?
- Will you always find what your looking for?
- What is the MAXIMUM number of steps to find a number?
- Simple to Implement.
- What about looking up "H. Zzpitz" in the Phone Book?

Linear Search

```
int searchList(int list[], int numElems, int value)
{
    int index = 0;           // Used as a subscript to search array
    int position = -1;       // To record position of search value
    bool found = false;      // Flag to indicate if the value was found

    while (index < numElems && !found)
    {
        if (list[index] == value) // If the value is found
        {
            found = true;          // Set the flag
            position = index;       // Record the value's subscript
        }
        index++;                  // Go to the next element
    }
    return position;              // Return the position, or -1
}
```



Goal: Search for the number ‘9’

Step 0 Assumption that the Array is sorted in **order**.

Step 1 Start in the MIDDLE of the array. Is the element you’re looking at the candidate?

Step 2 No? Then decide to consider the *upper* or *lower* part of the array based on the value of the candidate.

Step 3 Repeat with that part as a “new” array and go to **Step 1**

Binary Search for number '6'

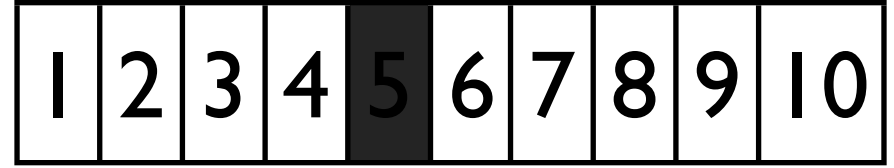
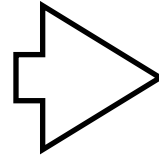
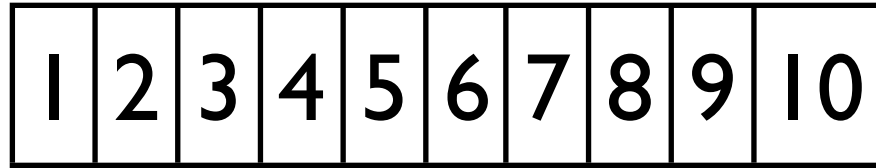
lower

upper

lower

upper

1.



↑
start

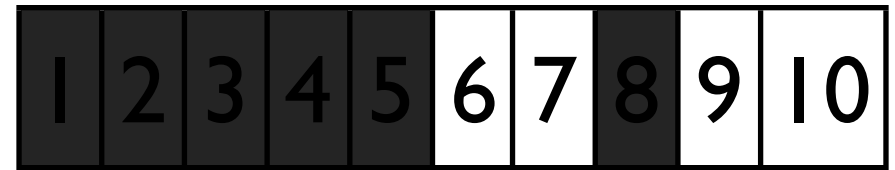
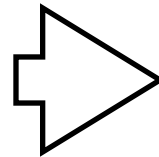
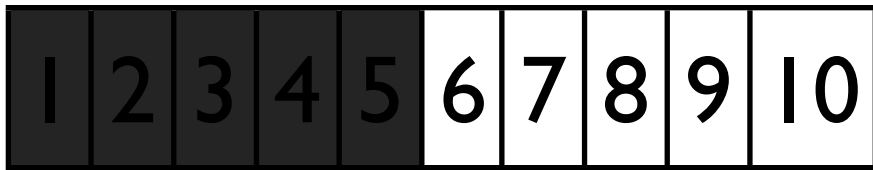
l

u

l

u

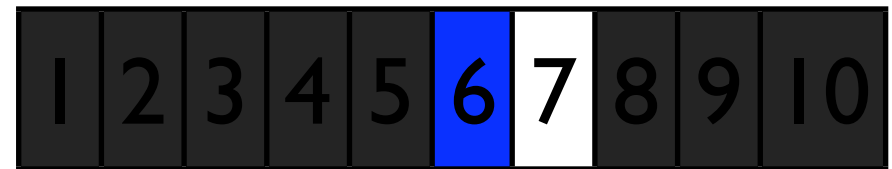
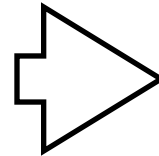
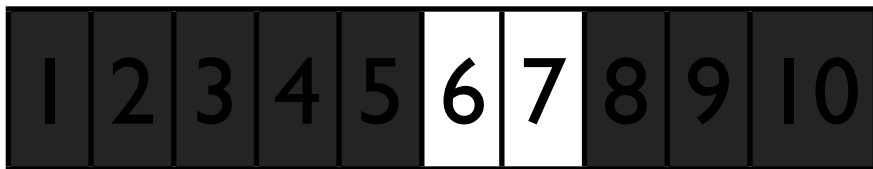
2.



↑
middle

u

3.



↑
middle

Success: Found "6"

Binary Search

```
int binarySearch(int list[], int numElems, int value)
{
```

Use Linear Search
as your template.

Without using your
book.

Write a Binary
Search
Algorithm.

```
}
```

Binary Search

```
int binarySearch(int array[], int numelems, int value)
{
    int first = 0,           // First array element
        last = numelems - 1, // Last array element
        middle,              // Mid point of search
        position = -1;       // Position of search value
    bool found = false;      // Flag

    while (!found && first <= last)
    {
        middle = (first + last) / 2;    // Calculate mid point
        if (array[middle] == value)     // If value is found at mid
        {
            found = true;
            position = middle;
        }
        else if (array[middle] > value) // If value is in lower half
            last = middle - 1;
        else
            first = middle + 1;          // If value is in upper half
    }
    return position;
}
```

Binary Search

Binary more efficient by a power of 2.

Array must be sorted!

N items	Linear Search	Binary Search
10	10	4
100	100	7
1000	1000	10
10,000	10,000	14
100,000	100,000	17
1,000,000	1,000,000	20

Sorting

Sorting aids in search.

Look at Two Algorithms:

Bubble Sort

Selection Sort



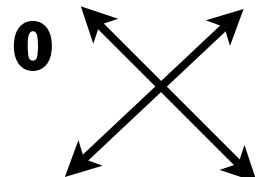
Bubble Sort

Choose *ascending* or *descending* order.

Example: (Sort in *ascending* order)

Compare

7	2	3	8	9	1
0	1	2	3	4	5



Swap

2	7	3	8	9	1
0	1	2	3	4	5

Compare

2	7	3	8	9	1
0	1	2	3	4	5

...

Bubble Sort

After First Pass

2	3	7	8	1	9
0	1	2	3	4	5

Are we there yet?

Bubble Sort

Compare

2	3	7	8	1	9
0	1	2	3	4	5

No Swap.

Bubble Sort

Compare

2	3	7	8	1	9
0	1	2	3	4	5

No Swap.

Bubble Sort

Compare

2	3	7	8	1	9
0	1	2	3	4	5

No Swap.

Bubble Sort

Compare

2	3	7	8	1	9
0	1	2	3	4	5

Swap.

2	3	7	1	8	9
0	1	2	3	4	5

Bubble Sort

Compare

2	3	7	1	8	9
0	1	2	3	4	5

No Swap.

Keep on Going.

Bubble Sort

Compare

2	3	7	1	8	9
0	1	2	3	4	5

No Swap.

Bubble Sort

Compare

2	3	7	1	8	9
0	1	2	3	4	5

No Swap.

Bubble Sort

Compare

2	3	7	1	8	9
0	1	2	3	4	5

Swap.

2	3	1	7	8	9
0	1	2	3	4	5

Bubble Sort

Compare

2	3	1	7	8	9
0	1	2	3	4	5

No Swap.

Bubble Sort

Compare

2	3	1	7	8	9
0	1	2	3	4	5

No Swap.

Keep on Going.

Bubble Sort

Compare

2	3	1	7	8	9
0	1	2	3	4	5

No Swap.

Bubble Sort

Compare

2	3	1	7	8	9
0	1	2	3	4	5

Swap.

2	1	3	7	8	9
0	1	2	3	4	5

Bubble Sort

Compare

2	1	3	7	8	9
0	1	2	3	4	5

No Swap.

Bubble Sort

Compare

2	1	3	7	8	9
0	1	2	3	4	5

No Swap.

Bubble Sort

Compare

2	1	3	7	8	9
0	1	2	3	4	5

No Swap.

Keep on Going.

Bubble Sort

Compare

2	1	3	7	8	9
0	1	2	3	4	5

Swap.

1	2	3	7	8	9
0	1	2	3	4	5

Bubble Sort

Compare

1	2	3	7	8	9
0	1	2	3	4	5

No Swap.

Bubble Sort

Compare

1	2	3	7	8	9
0	1	2	3	4	5

No Swap.

Bubble Sort

Compare

1	2	3	7	8	9
0	1	2	3	4	5

No Swap.

Bubble Sort

Compare

1	2	3	7	8	9
0	1	2	3	4	5

No Swap.

We're Done. (But the computer isn't)

Need to go through for one last verification pass.

Bubble Sort

What's the worse case scenario?

How many passes?

Not so great on LARGE data set, only moves one element at a time.

Bubble Sort

```
void sortArray(int array[], int size)
{
    bool swap;
    int temp;

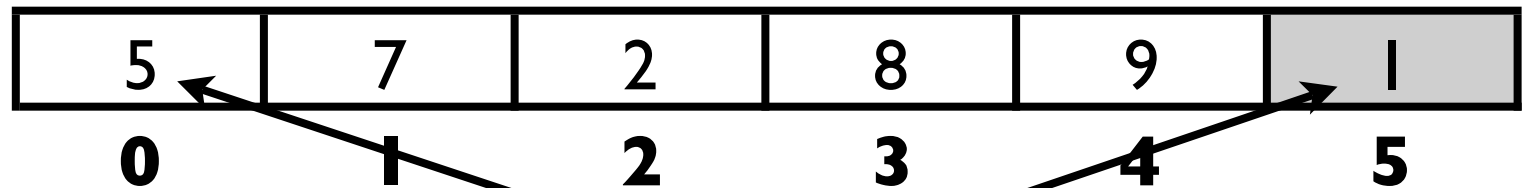
    do
    {
        swap = false;
        for (int count = 0; count < (size - 1); count++)
        {
            if (array[count] > array[count + 1])
            {
                temp = array[count];
                array[count] = array[count + 1];
                array[count + 1] = temp;
                swap = true;
            }
        }
    } while (swap);
}
```


Selection Sort

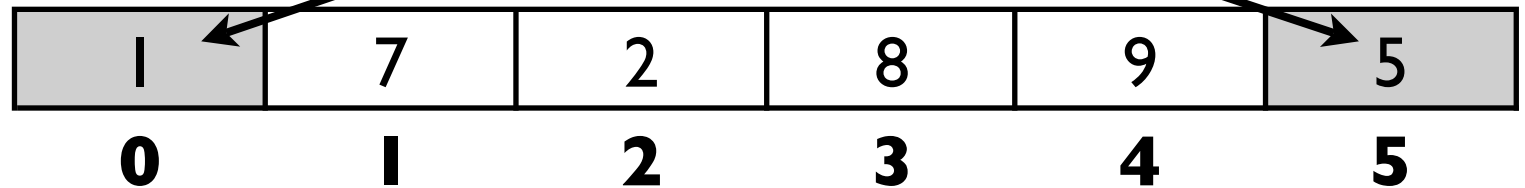
Finds smallest value.

Moves it to the first part of the array.

Search



Swap



Selection Sort

Array is sorted up to the first element.

Search



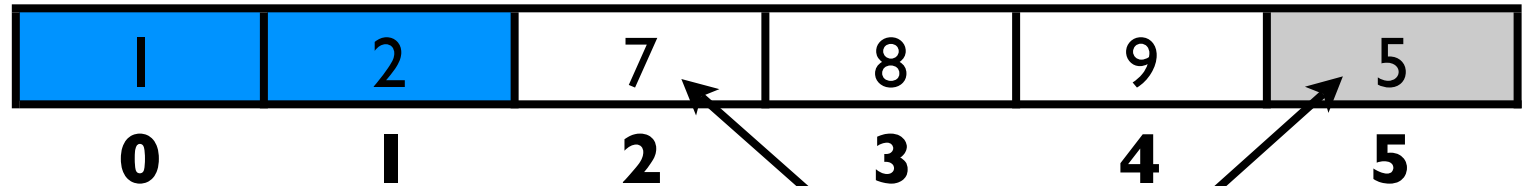
Swap



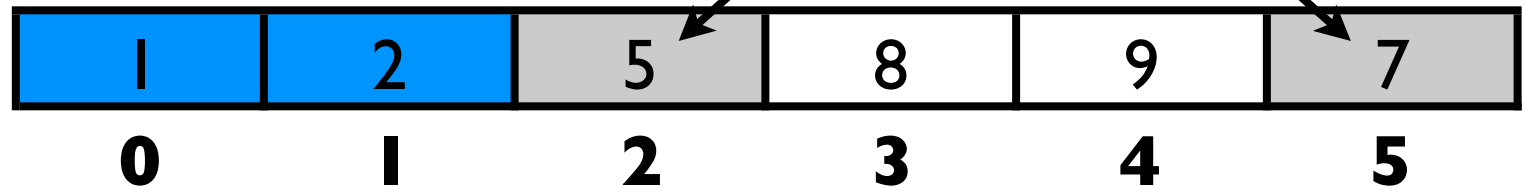
Selection Sort

Array is sorted up to element 1.

Search



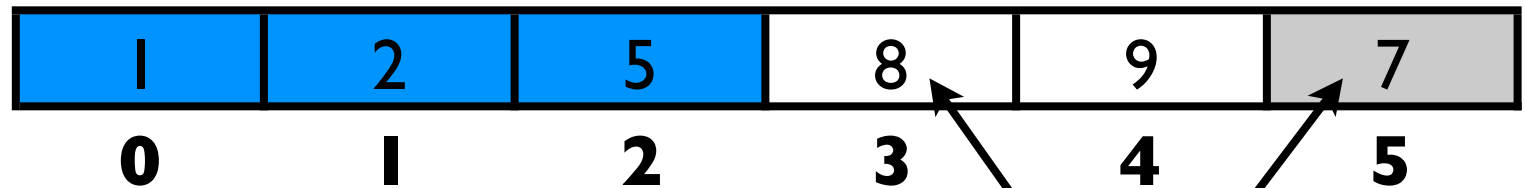
Swap



Selection Sort

Array is sorted up to element 2.

Search



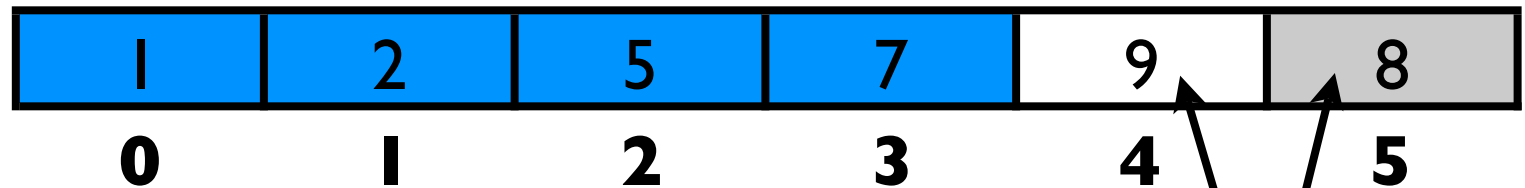
Swap



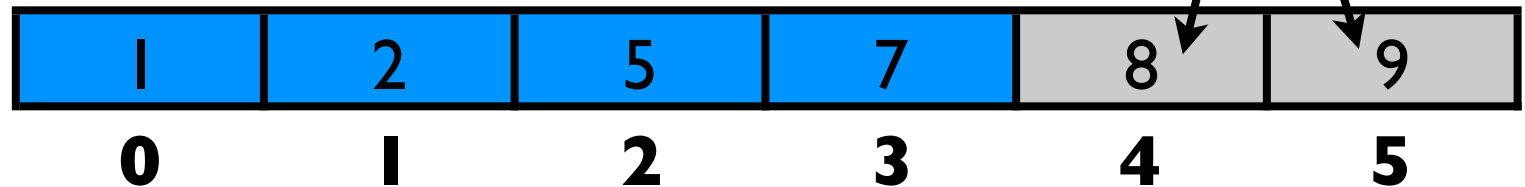
Selection Sort

Array is sorted up to element 3.

Search



Swap



Selection Sort

Array is sorted up to element 4.

Search

1	2	5	7	8	9
0	1	2	3	4	5

Done

1	2	5	7	8	9
0	1	2	3	4	5

Selection Sort

What's the worst case scenario?

How many passes through the data?

```
void selectionSort(int array[], int size)
```

```
{
```



?

```
}
```

Selection Sort

```
void selectionSort(int array[], int size)
{
    int startScan, minIndex, minValue;

    for (startScan = 0; startScan < (size - 1); startScan++)
    {
        minIndex = startScan;
        minValue = array[startScan];
        for(int index = startScan + 1; index < size; index++)
        {
            if (array[index] < minValue)
            {
                minValue = array[index];
                minIndex = index;
            }
        }
        array[minIndex] = array[startScan];
        array[startScan] = minValue;
    }
}
```


Readings Following Week

Chapter 8.1 - 8.4