Searching

- *Searching:* Looking for an item in a list of items.
  - Linear (Sequential) Search
  - Binary Search

Linear (Sequential) Search

```
// Sequential search for an unsorted array
int seqSearch(int ar[], int size, int num)
{
    int k = 0;
    /* Repeat until the end of the array is reached or the
       number is found */
    while (k < size && ar[k] != num)
       k++;
    /* If the end of the array is reached, it is not found*/
    if (k == size)
       return (-1);
    else
       return (k);
}
```

<u>Home Exercise:</u>

2	3	6	7	9	11	16

Trace the following calls using the list given above:

```
seqSearch(list, 7, 9);

seqSearch(list, 7, 20);

seqSearch(list, 7, 5);
```

```
// Sequential search for a sorted array
int seqSearchSorted(int ar[], int size, int num)
{
    int k = 0;
    /* Repeat until the end of the array or a larger value is
       reached or the number is found */
    while (k < size && ar[k] < num)
       k++;
    /* If the end of the array or a larger value is reached,
       it is not found */
    if (k == size || ar[k] > num)
       return (-1);
    else
       return (k);
}
```

Home Exercise:

2	3	6	7	9	11	16

Trace the following calls using the list given above:

```
seqSearchSorted(list, 7, 9);

seqSearchSorted(list, 7, 20);

seqSearchsorted(list, 7, 5);
```

In which call(s) is the seqSearchSorted function more efficient than the seqSearch function?

Searching – Binary Search

- *Searching* means looking for an item in a list of items. There are many searching algorithms in the literature. You already know one of them named *Linear (Sequential) Search*. This is the easiest but slowest search algorithm. Today, you will learn one other algorithm, named *Binary Search*.

- Binary Search algorithm assumes that the list to be searched is sorted and placed into an array.

- This algorithm works similar to what you usually do while finding a name in a telephone book:

  1. Open the book in the middle, and look at the middle name on the page.

  2. If the middle name is not the one you are looking for, decide whether it comes before or after the name you want.

  3. Take the appropriate half of the book you are looking in, and repeat these steps until you find the name.

Binary Search Algorithm in narrative description:

```
Let top be the subscript of the initial array element.
Let bottom be the subscript of the last array element.
Repeat until top exceeds bottom, thus there are no more elements to check
    Let middle be the subscript of the element halfway from top to bottom.
    If the element at middle is the target.
        Return middle.
    else if the element at middle is larger than the target
        Let bottom be  middle-1, thus continue the search in the first half.
    else
        Let top be middle+1, thus continue the search in the second half.
Return -1 since the loop terminated, but the number is not found
```

The C function:

```
int binarySearch(int ar[], int top, int bottom, int num) {
      int middle;
      while (top <= bottom) {
           middle = (top + bottom) / 2;
           if (num == ar[middle])
                return (middle);
           else
                if (ar[middle] > num)
                     bottom = middle-1;
                else
                     top = middle+1;
      }
      return (-1);
}
```

- Notice that, since we send both the top and the bottom indices as parameters to the function, it also allows us to search only some part of the array. For example, if list is an array with 10 elements,

  ```
  binarySearch(list, 2, 6, 9)
  ```

  searches 9 between list[2] and list[6], inclusively.

  ```
  binarySearch(list, 0, 9, 9)
  ```

  searches 9 in the entire array.

<u>Home Exercise:</u>

2	3	6	7	9	11	16

Trace the following calls using the list given above:

```
binarySearch (list, 2, 5, 9)

binarySearch (list, 0, 6, 16)

binarySearch (list, 0, 6, 1)

binarySearch (list, 1, 4, 11)
```