CS 171: Introduction to Computer Science II

Arrays

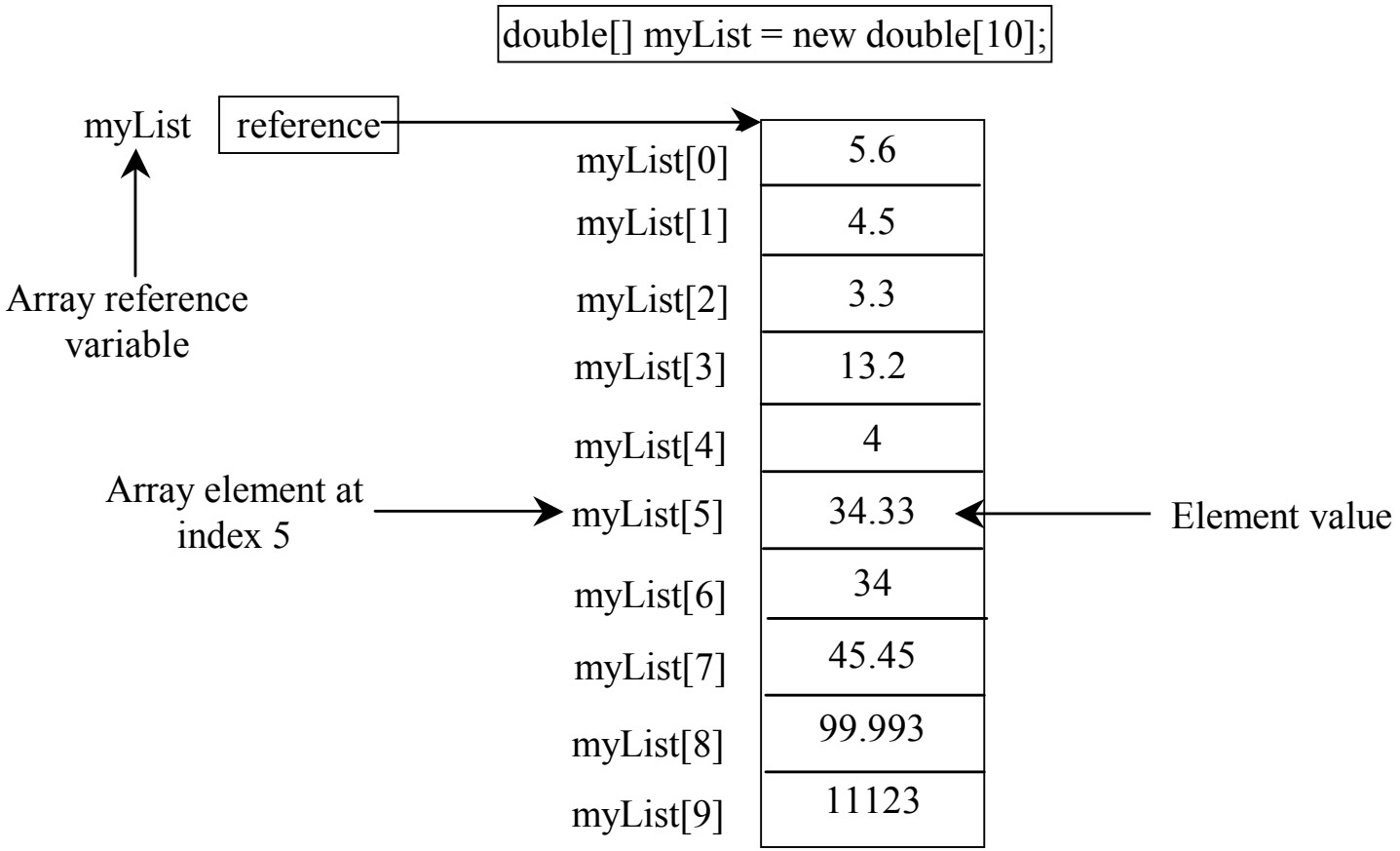
Li Xiong

Today

- Basics of Array (review)
- Searching
- Ordered Array and Binary Search
- Arraylist

Arrays

- Array is a data structure that represents a collection of the same types of data.



Java Array

- **Initialization**
  1. For primitive data types, all elements are initialized to be zero by default.
  - But you can explicitly initialize the array by providing the list of elements:

```
int[] myarray = {2,3,5,7,9,11,13,17};
```

  - This will both create the array and initialize the elements.

Java Array

- **Initialization**
  2. For class types, all elements are initialized to be **null**

```
Apple[] appArray = new Apple[100];
```

  – At this point, all elements in `appArray` are null.

  – To create object for each element, you can do:

```
for (int i=0; i<appArray.length; i++) {
  appArray = new Apple();
}
```

The Length of an Array

Once an array is created, its size is fixed. It cannot be changed. You can find its size using

```
arrayRefVar.length
```

For example:

```
int[] numbers = new int[10];

int len = numbers.length; // 10
```

Accessing Array Elements

- The array elements are accessed through the index.
- Each element in the array is represented using:

```
arrayRefVar[index]
```

- The array indices starts from 0 to array.length-1
- Example:

```
double[] myList = new double[10];
myList[0] = 5.6;
myList[1] = 4.5;
myList[10] = 3.33; //ArrayIndexOutOfBoundsException
```

Using for loops to process arrays

- Using for loops to traverse or process array elements
  - Elements are of the same type and processed repeatedly
  - The size of the array is known

```
for (int i=0; i<myList.length; i++) {
   // process ith element myList[i]
   System.out.println(myList[i]);
}
```

Typical Array Processing Code

- Finding maximum/minimum of array values
- Computing the average/sum of array values
- Copy to another array
- Reverse the elements within an array

Typical Array Processing Code

- Finding maximum of array values

```
double max = a[0];
for (int i=1; i< a.length; i++)
   if (a[i] > max)
        max = a[i];
```

Typical Array Processing Code

- Compute the average of array values

```
double sum = 0;
for (int i=1; i< a.length; i++)
   sum += a[i];
```

Typical Array Processing Code

- Copy values to another array

```
double[]  b = a; //?
```

Typical Array Processing Code

- Copy values to another array

```
int N = a.length;
double[]  b = new double[N];
for (int i=0; i<N; i++)
   b[i] = a[i]
```

Problem

- Given an array of elements, find the count of elements that are smaller than the element immediately following it. For example, if the array consists of 3, 7, 8, 5, 4, 9, the method should return 3, because 3<7, 7<8, and 4<9.
- Algorithm

  1. Initialize counter to 0

  2. for each element, compare it with the one following it, if smaller, increase counter by 1

Is this solution correct?

```
public int inorder(int[] a)
  {
    int counts = 0;
    for (int i=0; i<a.length; i++)
        if (a[i] < a[i+1])
                counts ++;
    return counts;
  }
```

Operations on Array

- Provided operations by Java Array
  - Accessing or updating a particular item by index
- Desired operations
  - insert an item
  - Delete an item
  - search an item with a particular value (key)
    - What's the cost in number of steps?

Search in an Array

- **Searching cost**
  - Best-case: 1 step
  - Worst-case: N steps (N: number of elements)
  - Average case: N/2 steps

Array

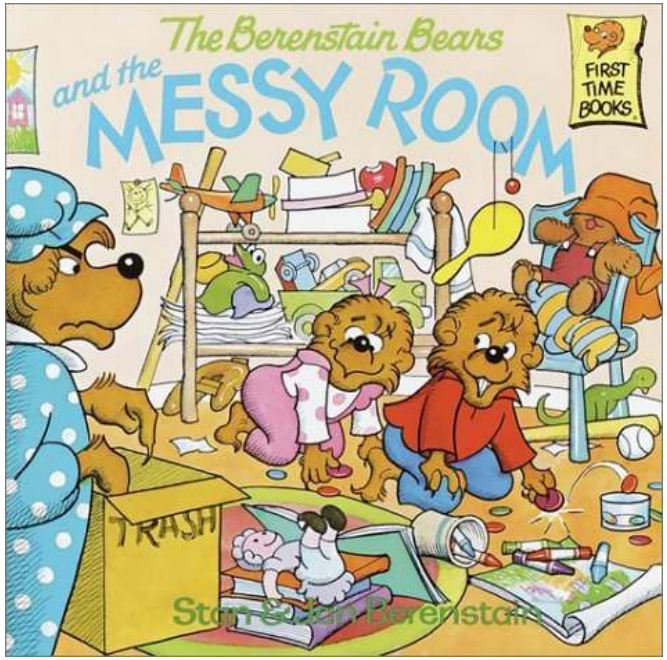
- **Duplication issue**
  - In many cases duplicates are not allowed, such as an employee's SSN
  - In many other cases, duplication is common (such as one's last name)
  - If duplicates are allowed, how does the search cost change?

Array with Duplication Allowed

- **Searching**
  - Always takes N steps

Ordered Array

- An array in which data elements are **sorted** in ascending values.
  - The smallest value at index 0
  - Each element is larger that its previous element
- Search
  - The main advantage of an ordered array is that it allows faster search using **binary search**.



Guess-a-Number Game

- I am asking you to guess a number that I am thinking of between 1 and 100
- When you make a guess, I will tell you one of three things: too large, too small, your guess is correct.

- Think about a strategy that leads to the fewest number of guesses on average.

Guess-a-Number Game

- Start by guessing 50
- If I say it's too small, you know the correct number must be within the range 51 to 100, so take 75 (halfway) as your next guess.
- Similarly if I say it's too large, you know the correct number must be within the range 1 to 49, so your next guess is 25.
- Each guess allows you to divide the range of possible values in half.
- Finally when the range is only one number long, you have the correct answer.

Guess-a-Number Game

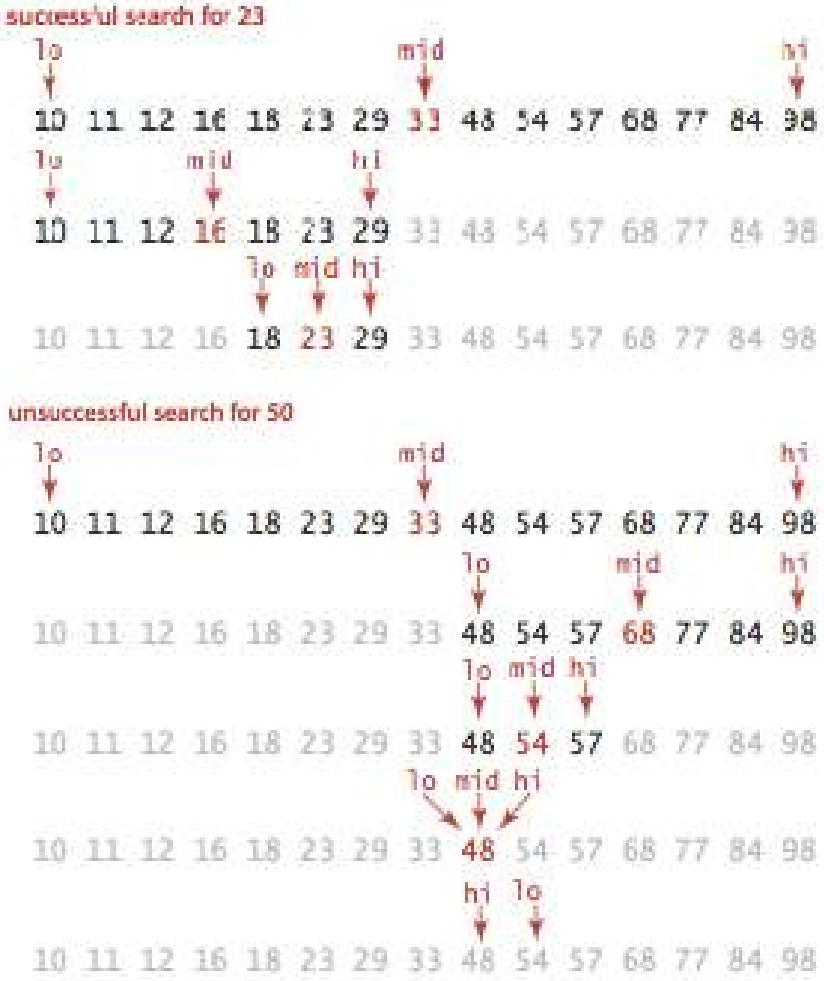
- Assuming the correct answer is 33:

TABLE 2.2 Guessing a Number

Step Number	Number Guessed	Result	Range of Possible Values
0			1–100
1	50	Too high	1–49
2	25	Too low	26–49
3	37	Too high	26–36
4	31	Too low	32–36
5	34	Too high	32–33
6	32	Too low	33–33
7	33	Correct	

Binary Search

- **Goal**: given a specific key, find it's location in the order array if it exists.



Binary Search - BinarySearch.java

```
// precondition: array a[] is sorted
public static int rank(int key, int[] a)
{
        int lo = 0;
        int hi = a.length - 1;
        while (lo <= hi) { // Key is in a[lo..hi] or not present.
                int mid = lo + (hi - lo) / 2;
                if (key < a[mid]) hi = mid - 1;
                else if (key > a[mid]) lo = mid + 1;
                else return mid;
        }
        return -1;
}
```

Binary Search Cost

- Given a sorted array of N elements
  - How many steps?

Operations on Array

- Provided operations by Java Array
  - Accessing or updating a particular item by index
- Desired operations
  - insert an item
  - Delete an item
  - search an item with a particular value (key)

Java ArrayList class

- **ArrayList<DataType>**
  - A useful Java class that implements the basic array operations we discussed
  - Use generics to allow different data types
  - Additionally allows for dynamic expanding of capacity.

Coming up

- Hw1: number guessing game
- Next lecture: algorithm cost analysis