Java Programming

Zheng-Liang Lu

Department of Computer Science & Information Engineering
National Taiwan University

Java 415
Summer 2024

```java
class Lecture4 {

                    "Arrays and More Data Structures"

}
```

Arrays

An array is an object which stores multiple values of the same type.

```
1 ...
2         // Assume that T is any type and the size is known.
3         T[] A = new T[size];
4 ...
```

- We now proceed to explain Line 3 in two stages.

Stage 1: Array Creation

- We first focus on the RHS of Line 3.
- One array is allocated in the heap by invoking the new operator followed by **T** and [ ] surrounding its size,
- Then its starting address is returned and should be cached.
- Note that the size cannot be changed after allocation.[1]

[1] What if the array is full?! Stay tuned.

Stage 2: Reference

- We then declare one variable, say A in this case, to store the starting address of the array.
- I strongly emphasize that A is not the array, but the reference to the array!
- To understand the type correctly, one should read the type from right to left.
- For example, A is the reference to an array whose elements are of the **T** type.
- Note that the array type is declared like **T**[ ] but without the size.

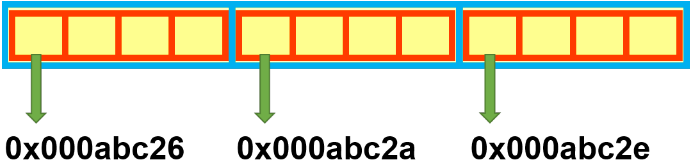
Zero-Based Array Indexing

- We access any array element by using its index, which starts from 0 but not 1.
- Explicitly, the first element is $A[0]$, followed by $A[1]$, $A[2]$, and so on. (Why?)
- So the last index of one array is $size - 1$.
- When the index is out of range, the program will fail due to the runtime exception **ArrayIndexOutOfBoundsException**.

Memory Allocation for Arrays

- An array is allocated contiguously in the memory.
- To fetch the second element, jump to the address stored by A and shift by 1 unit size of **T**, denoted by A[1].
- For example,

```java
...
        int[] A = new int[3];
...
```



Zero-Based Array Indexing (Concluded)

- You now could explain why A[0] denotes the first array element.
- Array index clearly acts as an offset from the starting address of the array!
- It is worth to mention that we can treat the whole memory as an array, indeed.
- This convention is applicable commonly among the mainstream languages![2] (Why?)

[2]For example, C, C++, Java, JavaScript, and even Python. However, to the best of my knowledge, R and MATLAB manipulate arrays with the first index starting from 1. So it is just an option between choosing 0 and 1.

Array Initialization

- Every array is initialized implicitly once the array is created.
- Default values for different types are listed below:
  - 0 for all numeric types;
  - \u0000 for char type;
  - false for boolean type;
  - null for all reference types.[3]
- An array can also be created by enumerating all elements without using the new operator, for example,

```
...
        int[] A = { 10, 20, 30 }; // Syntax sugar.
...
```

[3]We will visit the keyword null in the chapter of OOP.

Arrays & Loops

> We often use for loops to process array elements.

- Arrays have the attribute called length, which indicates the array capacity.
  - For example, *A*.length.
- So it is natural to use a for loop to manipulate arrays.

Examples

```java
...
        // Create an integer array of size 5.
        int[] A = new int[5];

        // Generate 5 random integers ranging from 0 to 99.
        for (int i = 0; i < A.length; ++i) {
            A[i] = (int) (Math.random() * 100);
        }

        // Display all elements of A: O(n).
        for (int i = 0; i < A.length; ++i) {
            System.out.printf("%d ", A[i]);
        }
        System.out.println();
...
```

- To show all elements, you need to iterate over the array by loops instead of simply printing *A*. (Why?)

```java
...
        // Find maximum and minimum of A: O(n).
        int max = A[0];
        int min = A[0];
        for (int i = 1; i < A.length; ++i) {
            if (max < A[i]) max = A[i];
            if (min > A[i]) min = A[i];
        }
...
```

- How to find the locations of extreme values?[4]
- Can you find the 2nd maximum value in A?
- Can you track and maintain a record of the first k maximum values in A?

[4]See also Arguments of Maxima (argmax) and Arguments of Minima (argmin).

```java
...
        // Sum of A: O(n).
        int sum = 0;
        for (int i = 0; i < A.length; ++i) {
            sum += A[i];
        }
...
```

- Calculate the following descriptive statistics:
  - the mean of A;
  - the median[5] of A;
  - the mode[6] of A.

[5] See https://en.wikipedia.org/wiki/Median.

[6] See https://en.wikipedia.org/wiki/Mode_(statistics).

Alternative Way: for-each Loops

- A for-each loop is designed to iterate over a collection of objects, such as arrays and other data structures, in strictly sequential fashion, from start to finish.

```
...
        T[] A = { ... };
        for (T element : A) {
            // Loop body.
        }
...
```

Example

```java
...
        int s = 0;
        for (int i = 0; i < A.length; ++i) {
            s += A[i];
        }
...
```

```java
...
        int s = 0;
        for (int item : A) {
            s += item;
        }
...
```

- Short and sweet!
- You may consider using the for-each loop if you iterate over all elements and the order of iteration is irrelevant.

Exercise

```java
...
        String[] letters = { "A", "B", "C", "D", "E" };

        for (String letter: letters) {
            System.out.printf("%s ", letter);
        }
        System.out.println();
...
```

More Examples (1/4): Cloning Arrays

- One might duplicate an array for some purpose, say a backup.
- For example,

```
...
        int x = 1;
        int y = x; // You can say that y copies the value of x.
        x = 2;
        System.out.println(y); // Output 1.

        int[] A = { 10, ... }; // Ignore the rest of elements.
        int[] B = A;
        A[0] = 100;
        System.out.println(B[0]); // Output?
...
```

- The result differs from our expectation. (Why?)
- This is called the shallow copy.

- To clone an array, you should create a new array and use loops to copy every element, one by one.

```
...
        // Let A be an array to be copied.
        int[] B = new int[A.length];
        for (int i = 0; i < A.length; ++i) {
            B[i] = A[i];
        }
...
```

- This is called the deep copy.
- Alternatively, you may use the method **System**.arraycopy() for the same purpose.

```
...
        // Assume that B is ready.
        System.arraycopy(A, 0, B, 0, A.length);
...
```

More Examples (2/4): Shuffle Algorithm

```java
...
        for (int i = 0; i < A.length; ++i) {

            // Choose a randon integer j.
            int j = (int) (Math.random() * A.length);

            // Swap A[i] and A[j].
            int tmp = A[i];
            A[i] = A[j];
            A[j] = tmp;

        }
...
```

- However, this naive algorithm is fundamentally broken![7]
- How to swap by using XOR (that is, $\wedge$)?

[7] See https://blog.codinghorror.com/the-danger-of-naivete/.

Exercise

Write a program to deal the first 5 cards from a deck of 52 shuffled cards.

- As you can see, RNG produces only random numbers.
- How to shuffle nonnumerical objects?
- Simply label 52 cards by $0, 1, \ldots, 51$.
- Shuffle these numbers!

```java
...
        String[] suits = { "Club", "Diamond", "Heart", "Spade" };
        String[] ranks = { "3", "4", "5", "6", "7", "8", "9",
                           "10", "J", "Q", "K", "A", "2" };

        int size = 52;
        int[] deck = new int[size];
        for (int i = 0; i < deck.length; i++)
            deck[i] = i;

        // Shuffle algorithm: correct version.
        for (int i = 0; i < size - 1; i++) {
            int j = (int) (Math.random() * (size - i)) + i;
            int z = deck[i];
            deck[i] = deck[j];
            deck[j] = z;
        }

        for (int i = 0; i < 5; i++) {
            String suit = suits[deck[i] / 13];
            String rank = ranks[deck[i] % 13];
            System.out.printf("%2s %-7s\n", rank, suit);
        }
...
```

More Examples (3/4): Sorting Problem

- In computer science, a sorting algorithm is an algorithm that puts elements of a list in a certain order.
- You may call **Arrays**.sort(A) to rearrange A in ascending order, for example,

```
import java.util.Arrays;

...
        int[] A = { 5, 2, 8 };
        Arrays.sort(A); // Becomes { 2, 5, 8 }.

        String[] B = { "www", "csie", "ntu", "edu", "tw" };
        Arrays.sort(B); // Result?
...
```

- Note that we sort strings in lexicographical (dictionary) order for most cases.

Exercise: Bubble Sort

```
...
        // Bubble sort: O(n ^ 2).
        boolean swapped;
        do {
            swapped = false;
            for (int i = 0; i < A.length - 1; i++) {
                if (A[i] > A[i + 1]) {
                    int tmp = A[i];
                    A[i] = A[i + 1];
                    A[i + 1] = tmp;
                    swapped = true;
                }
            }
        } while (swapped);
...
```

- Try to implement selection sort and insertion sort.[8]

[8] See https://visualgo.net/en/sorting.

More Examples (4/4): Searching Problem

- It is often to query one key for its corresponding value.
- For example, the program plans to find one client's credit card number.
- In this case, the client name is the query key and his/her credit card number is the value associated.

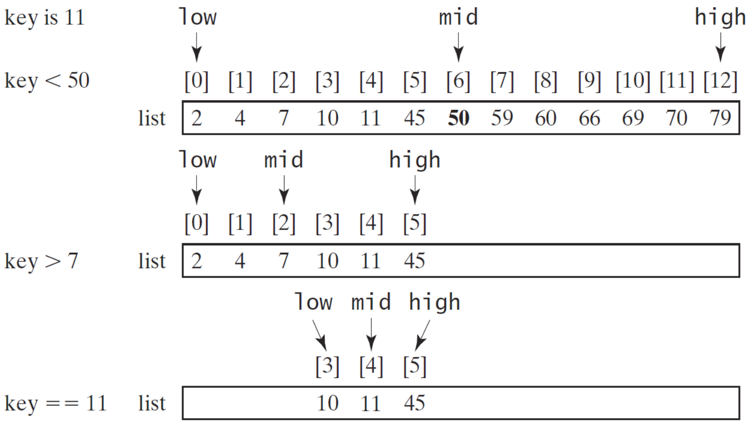
Solution 1: Linear Search

- Linear search compares the query key with all elements in sequential order.

```
...
        // Linear search: O(n).
        int[] A = { ... };
        int founds = 0;
        for (int i = 0; i < A.length; i++) {
            if (A[i] == key) {
                System.out.printf("%d ", i);
                founds++;
            }
        }
        System.out.println("\nFounds: " + founds);
...
```

- Could we do better?

Solution 2: Binary Search (Revisited)



```java
...
        int idx = -1; // Why?
        int high = A.length - 1, low = 0, mid;
        while (high > low && idx < 0) {
            mid = low + (high - low) / 2; // Why?
            if (A[mid] < key)
                low = mid + 1;
            else if (A[mid] > key)
                high = mid - 1;
            else
                idx = mid;
        }

        if (idx > -1)
            System.out.printf("%d: %d\n", key, idx);
        else
            System.out.printf("%d: not found\n", key);
...
```

- It can be shown that binary search runs in $O(\log n)$ time.
- However, binary search works only for ordered data!

Discussions

Scenario / Operation	Insert	Search
Immutable unsorted array	N/A	$O(n)$
Immutable sorted array	N/A	$O(\log n)$
Mutable unsorted array	$O(1)^*$	$O(n)$
Mutable sorted array	$O(n)$	$O(\log n)$

*: insert by attaching behind the array.

- Assume that the data is immutable (unchangeable).
- We sort the data once for all and the binary search works well.
- What if the data may be changed all the time?
- Is it possible to make it run in $O(1)$ time for both operations?[9]

[9] See https://en.wikipedia.org/wiki/Hash_table.

Short Introduction to Data Structures

- A data structure is a particular way of organizing data in a program so that it can perform efficiently.[10]
- The choice for data structures depends on applications.
- As an alternative to arrays, linked lists[11] are used to store data in the way different from arrays.
- You will see plenty of data structures in the future.[12]
  - For example, trees, graphs, tables, and more.
- You could also find a huge number of questions about data structures on LeetCode.

[10] See http://bigocheatsheet.com/.
[11] See https://en.wikipedia.org/wiki/Linked_list.
[12] See Introduction to Collections by Oracle and Java Collections Framework from Wikipedia.

Beyond 1D Arrays

- 2D or higher dimensional arrays are widely used in various applications.
  - For example, RGB images are stored as 3D arrays.
- We can create 2D **T**-type arrays simply by adding one more [ ] with its size.
- For example,

```
...
        int rows = 4; // Row size.
        int cols = 3; // Column size.
        T[][] M = new T[rows][cols];
...
```

- It is similar to create 3D or higher-dimensional arrays.

	[0]	[1]	[2]	[3]	[4]
[0]	0	0	0	0	0
[1]	0	0	0	0	0
[2]	0	0	0	0	0
[3]	0	0	0	0	0
[4]	0	0	0	0	0

```java
matrix = new int[5][5];
```

(a)

	[0]	[1]	[2]	[3]	[4]
[0]	0	0	0	0	0
[1]	0	0	0	0	0
[2]	0	7	0	0	0
[3]	0	0	0	0	0
[4]	0	0	0	0	0

```java
matrix[2][1] = 7;
```

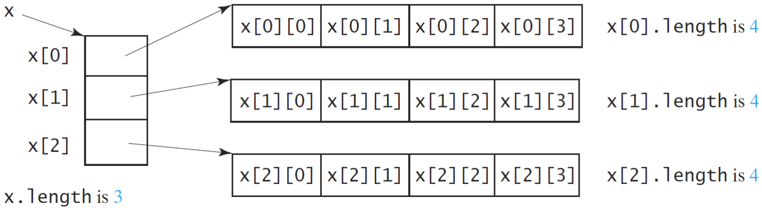
(b)

	[0]	[1]	[2]
[0]	1	2	3
[1]	4	5	6
[2]	7	8	9
[3]	10	11	12

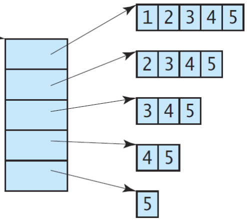
```java
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

(c)

Memory Allocation for 2D Arrays




```java
int[][] triangleArray = {
  {1, 2, 3, 4, 5},
  {2, 3, 4, 5},
  {3, 4, 5},
  {4, 5},
  {5}
};
```



Example: 2D Arrays & Loops[13]

```
...
        int[][] A = { { 10, 20, 30 }, { 40, 50 }, { 60 } };

        // Conventional for loop.
        for (int i = 0; i < A.length; i++) {
            for (int j = 0; j < A[i].length; j++)
                System.out.printf("%3d", A[i][j]);
            System.out.println();
        }

        // For-each loop.
        for (int[] row : A) {
            for (int item : row)
                System.out.printf("%3d", item);
            System.out.println();
        }
...
```

[13] Thanks to a lively discussion on January 31, 2016.

Exercise: Matrix Multiplication

Let $A_{m \times n}$ and $B_{n \times q}$ be two matrices for $m, n, q \in \mathbb{N}$. Write a program to calculate $C = AB$.

- Let a_{ik} and b_{kj} be elements of A and B, respectively.
- For $k = 1, 2, \ldots, n$, use the formula

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

  for $i = 1, 2, \ldots, m$ and for $j = 1, 2, \ldots, q$.
- Following the formula, it takes $O(n^3)$ time. (Why?)

Digression: **ArrayList**

```java
...
        int[] A = new int[3]; // The size should be known in advance.
        A[0] = 100;
        A[1] = 200;
        A[2] = 300;
        for (int item : A)
            System.out.printf("%d ", item);
        System.out.println();

        ArrayList<Integer> B = new ArrayList<>(); // Size?
        B.add(100);
        B.add(200);
        B.add(300);
        System.out.println(B); // Short and sweet!
...
```

- Arrays are the simplest form of data structures but not convenient to use.
- For example, resizing arrays can be costly when you frequently move data to a newly created, larger array. (Why?)
- So it is advisable to use **ArrayList**$<$E$>$, where E is the type parameter.
- Using angle brackets $< \cdot >$ in Java is called the generics starting from JDK5 in 2004.

Digression: Generics

- Generics are widely used in data structures, like **Stack**<T>, **Map**<K, V>, **Graph**<V, E>, etc.[14]
- To use **ArrayList**<E> correctly, we need to replace E with **Integer**, which is the wrapper class[15] for int values.
- Be aware that only reference types can substitute the type parameters.
- This technique is also utilized in C++ and C#.

[14] See also Generics by Orcale. ~~Stay tuned in Java Programming 2.~~
[15] See The Numbers Classes.

Case Study: Order Reversing

- How to rearrange an input array in reverse order?
- Let A be an integer array.
- The first attempt is to create another array with same size and copy each element from A to B.

```
...
        int[] A = { 1, 2, 3, 4, 5 };
        int[] B = new int[A.length];
        for (int i = 0; i < A.length; i++) {
            B[A.length - 1 - i] = A[i];
        }
        A = B; // Why?
...
```

Another Attempt

```java
...
        int[] A = { 1, 2, 3, 4, 5 };
        for (int i = 0; i < A.length / 2; i++) {
            int j = A.length - 1 - i;
            int tmp = A[i];
            A[i] = A[j];
            A[j] = tmp;
        }
...
```

Approach	Time Complexity	Space Complexity
1st attempt	$O(n)$	$O(n)$
2nd attempt	$O(n)$	$O(1)$

- The second is better in both time[16] and space.
- This is an in-place algorithm.

[16] It runs in only half time of the first attempt.