

Exercise walk-through

4.12

2D Array Traversals

AP Computer Science A

The five standard array algorithms

sum or average

an accumulator declared outside the loop; divide by `length` at the end

max or min

start from `arr[0]`, not from zero – an array of negatives has no element above zero

count matching

a counter, incremented only when the element passes the test

check for a duplicate

a **nested** loop, with the inner one starting at `i+1` so no pair is checked twice

reverse or copy

build into a **new** array, or swap ends inward for an in-place reverse

Every one of these is a traversal plus a small body. Recognising which one a question wants is most of the work.

Why a data structure at all

the problem

a single variable holds **one** value; real problems need many related values – a class roster, pixels, sensor readings

a data structure

holds many values under one name, reached by **index**

an array

fixed length, set when it is created, and `arr.length` is a field, not a method

the bounds

valid indices run 0 to `length-1`; anything outside throws an `ArrayIndexOutOfBoundsException`

the run-time cost

reaching any element takes the same time whatever the index – that is the point of an array

Off-by-one at the top end is the classic array bug: `i <= arr.length` throws, `i < arr.length` does not.

Removing from a list while looping over it

the problem

`remove(i)` shifts every later element down, so the element after the removed one is skipped

fix one

loop backwards, from `size()-1` down to 0 – later shifts cannot affect indices you have passed

fix two

do not increment `i` after a removal, so the next pass re-examines the same index

never

remove inside an enhanced for loop – it throws a `ConcurrentModificationException`

the rest

the array algorithms carry over unchanged, plus insertion and deletion, which arrays cannot do easily

This is the single most-tested ArrayList point. If a question removes items in a loop, it is testing exactly this.

Exercises 2.1-2.2

2.1 State the output of the **column-major** loop for **g**.

[1]

2.2 Write nested enhanced **for** loops that set **total** to the sum of every element of **g**.
Include the declaration of **total**.

[3]

Exercises 2.1-2.2

2.1 State the output of the **column-major** loop for `g`.

[1]

2.1

- down each column: **2 1 9 4 3 8 6 5 7**

2.2 Write nested enhanced `for` loops that set `total` to the sum of every element of `g`. Include the declaration of `total`. [3]

2.2

```
int total = 0;
for (int[] row : g)
    for (int v : row)
        total += v;
```

- outer loop over rows with type `int[]`
- inner loop over each value with type `int`
- adds every value to `total`, which starts at 0

Exercises 2.3-2.4

2.3 Write a loop that prints the **last** column of any 2D array **a**.

[2]

2.4 State the values visited by `for (int i = 0; i < g.length; i++) g[i][g.length - 1 - i]`.

[1]

Exercises 2.3-2.4

2.3 Write a loop that prints the **last** column of any 2D array `a`.

[2]

2.3

```
for (int r = 0; r < a.length; r++)  
    System.out.println(a[r][a[0].length - 1]);
```

- loop over every row
- fixed column `a[0].length - 1`

2.4 State the values visited by `for (int i = 0; i < g.length; i++) g[i][g.length - 1 - i]`.

[1]

2.4

- `g[0][2]`, `g[1][1]`, `g[2][0]`: **6, 3, 9**

Exercise 3.1

3.1 What is printed?

[1]

```
int[][] a = { {1, 2, 3}, {4, 5, 6} };  
for (int c = 0; c < a[0].length; c++)  
    System.out.print(a[1][c] - a[0][c]);
```

☐

A 333

B 123

C 456

D -3-3-3

Exercise 3.1

3.1 What is printed?

[1]

```
int[][] a = { {1, 2, 3}, {4, 5, 6} };  
for (int c = 0; c < a[0].length; c++)  
    System.out.print(a[1][c] - a[0][c]);
```

☐

A 333

B 123

C 456

D -3-3-3

3.1 A

- $4 - 1$, $5 - 2$, $6 - 3$: each 3

Exercise 3.2

3.2 Which loop header pair visits every element of `int[] [] m` in **column-major** order? [1]

☐

A `for (r = 0; r < m.length; r++) then for (c = 0; c < m[0].length; c++)`

B `for (c = 0; c < m[0].length; c++) then for (r = 0; r < m.length; r++)`

C `for (c = 0; c < m.length; c++) then for (r = 0; r < m[0].length; r++)`

D `for (int[] row : m) then for (int v : row)`

Exercise 3.2

3.2 Which loop header pair visits every element of `int[] [] m` in **column-major** order? [1]

☐

A `for (r = 0; r < m.length; r++) then for (c = 0; c < m[0].length; c++)`

B `for (c = 0; c < m[0].length; c++) then for (r = 0; r < m.length; r++)`

C `for (c = 0; c < m.length; c++) then for (r = 0; r < m[0].length; r++)`

D `for (int[] row : m) then for (int v : row)`

3.2 B

- columns outside, rows inside, each with its own bound

Exercise 3.3

3.3 How many times is `count++` executed for a grid with 4 rows and 5 columns? [1]

```
for (int r = 1; r < grid.length; r++)  
    for (int c = 0; c < grid[0].length - 1; c++)  
        count++;
```

☐

A 20

B 16

C 12

D 9

Exercise 3.3

3.3 How many times is `count++` executed for a grid with 4 rows and 5 columns? [1]

```
for (int r = 1; r < grid.length; r++)  
    for (int c = 0; c < grid[0].length - 1; c++)  
        count++;
```

☐

A 20

B 16

C 12

D 9

3.3 C

- rows 1 to 3 (3 rows) times columns 0 to 3 (4 columns): 12

Exercise 3.4 (a)

3.4 A teacher stores test marks in `int[][] marks`: each **row** is a student and each **column** is a test.

(a) Write a method `studentTotal` that returns the total of all marks for the student in row `s`. [3]

```
public static int studentTotal(int[][] marks, int s)
```

Exercise 3.4 (a)

3.4 A teacher stores test marks in `int[][] marks`: each **row** is a student and each **column** is a test.

(a) Write a method `studentTotal` that returns the total of all marks for the student in row `s`. [3]

```
public static int studentTotal(int[][] marks, int s)
```

(a)

```
public static int studentTotal(int[][] marks, int s)
{
    int total = 0;
    for (int c = 0; c < marks[s].length; c++)
    {
        total += marks[s][c];
    }
    return total;
}
```

- loops over every column of row `s`
- adds `marks[s][c]` to a total that starts at 0
- returns the total

Exercise 3.4 (b)

3.4 A teacher stores test marks in `int[] [] marks`: each **row** is a student and each **column** is a test.

(b) Write a method `testAverage` that returns the **average** mark, as a **double**, for the test in column `t`. [4]

```
public static double testAverage(int[] [] marks, int t)
```

Exercise 3.4 (b)

3.4 A teacher stores test marks in `int[] [] marks`: each **row** is a student and each **column** is a test.

(b) Write a method `testAverage` that returns the **average** mark, as a **double**, for the test in column `t`. [4]

```
public static double testAverage(int[] [] marks, int t)
```

(b)

```
public static double testAverage(int[] [] marks, int t)
{
    int total = 0;
    for (int r = 0; r < marks.length; r++)
    {
        total += marks[r][t];
    }
}
```

Full solution 3.4 (b)

(b)

```
public static double testAverage(int[] [] marks, int t)
{
    int total = 0;
    for (int r = 0; r < marks.length; r++)
    {
        total += marks[r][t];
    }
    return (double) total / marks.length;
}
```

- loops over every **row**
- adds `marks[r][t]` with the column fixed
- divides by the number of rows, `marks.length`
- casts to `double` to avoid integer division, and returns it