Tutorial 1 Solution Sketch

EECS3101 Z:
Design and Analysis of Algorithms
Winter 2026

CHEN-WEI WANG

2-D Arrays: Motivating Example (1)

Here is a solution based on what we've learnt so far:

- Fix the "positions" of cities in the table as constants:

```
final int CHICAGO = 0;
final int BOSTON = 1;
final int MIAMI = 4;
```

- Represent each (horizontal) row using a one-dimensional array:

```
int[] fromChicago = {0, 983, 787, 714, 1375, 967, 1087}
int[] fromBoston = {983, 0, 214, 1102, 1763, 1723, 1842}
int[] fromMiami = {1375, 1763, 1549, 661, 0, 1426, 1187}
```

- Given an itinerary {Boston, Chicago, Miami, Houston}, choose the corresponding arrays in the right order:

```
int[] dist = fromBoston[CHICAGO]
           + fromChicago[MIAMI]
           + fromMiami[HUSTON];
```

2-D Arrays: Motivating Example (2)

What if cities of an itinerary are read from the user?

```
1  Scanner input = new Scanner(System.in);
2  System.out.println("How many cities?");
3  int howMany = input.nextInt(); input.nextLine();
4  String[] trip = new String[howMany];
5  /* Read cities in the trip from the user. */
6  for(int i = 0; i < howMany; i ++) {
7    System.out.println("Enter a city:");
8    trip[i] = input.nextLine();
9  }
10 /* Add up source-to-destination distances. */
11 int dist = 0;
12 for(int i = 0; i < howMany - 1; i ++) {
13   String src = trip[i];
14   String dst = trip[i + 1];
15   /* How to accumulate the distance between src and dst?  */
16 }
```

2-D Arrays: Motivating Example (3)

Given a source and a destination, we need to *explicitly* select:

- The corresponding *source row* [e.g., `fromBoston`]
- The corresponding *destination index* [e.g., `CHICAGO`]

```
13  String src = trip[i];
14  String dst = trip[i + 1];
15  if(src.equals("Chicago")) {
16    if(dst.equals("Boston")) {dist += fromChicago[BOSTON];}
17    else if(dst.equals("New York")) {dist += fromChicago[NY];}
18    ...
19  }
20  else if(src.equals("Boston")) {
21    if(dst.equals("Chicago")) {dist += fromBoston[CHICAGO];}
22    else if(dst.equals("NEW YORK")) {dist += fromBoston[NY];}
23    ...
24  }
25  ...
```

- Drawback? $7 \times (7 - 1)$ possibilities to program!

2-D Arrays: Initialization (1)

A *2D array* is really *an array of 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

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

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

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

2-D Arrays: Initialization (1)

A *2D array* may be initialized either at the time of declaration, or after declaration.

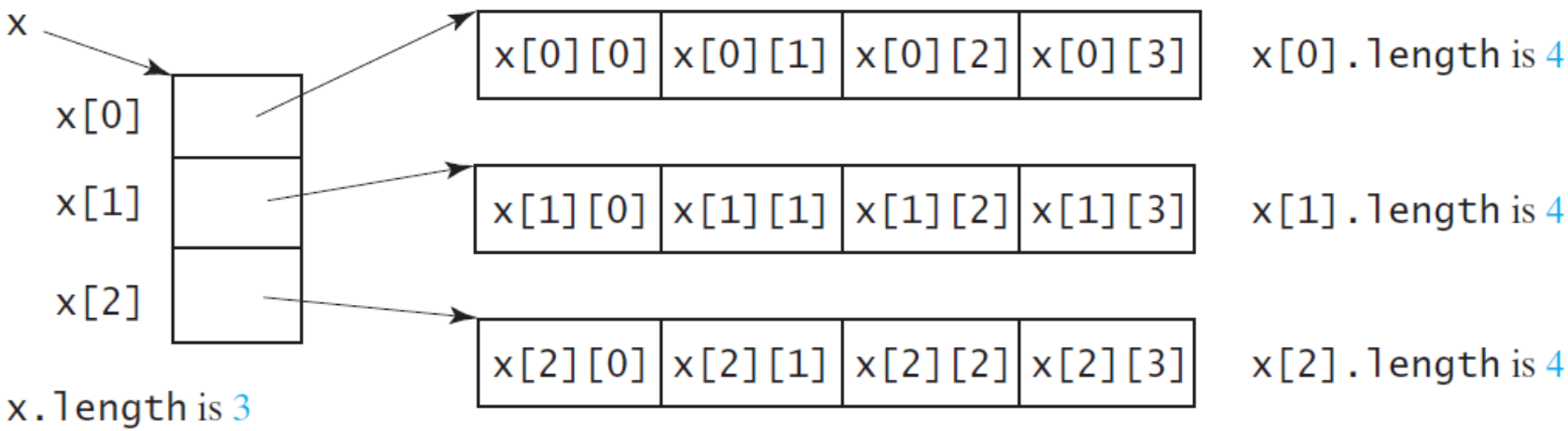
```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

Same as

```
int[][] array = new int[4][3];
array[0][0] = 1; array[0][1] = 2; array[0][2] = 3;
array[1][0] = 4; array[1][1] = 5; array[1][2] = 6;
array[2][0] = 7; array[2][1] = 8; array[2][2] = 9;
array[3][0] = 10; array[3][1] = 11; array[3][2] = 12;
```

2-D Arrays: Lengths (1)

For a *2D array*, you may query about its *size*, or *sizes* of its component arrays.

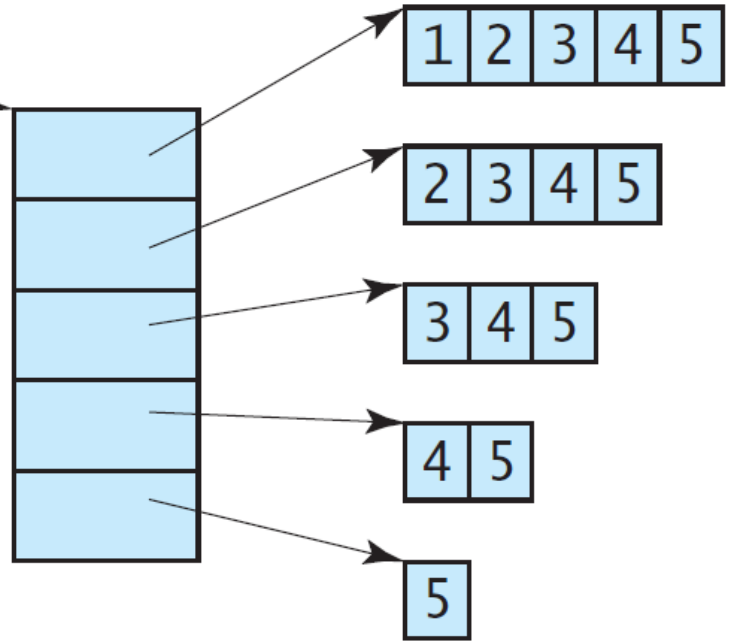


2-D Arrays: Lengths (2)

For a *2D array*, its components may have different *sizes*.

e.g.,

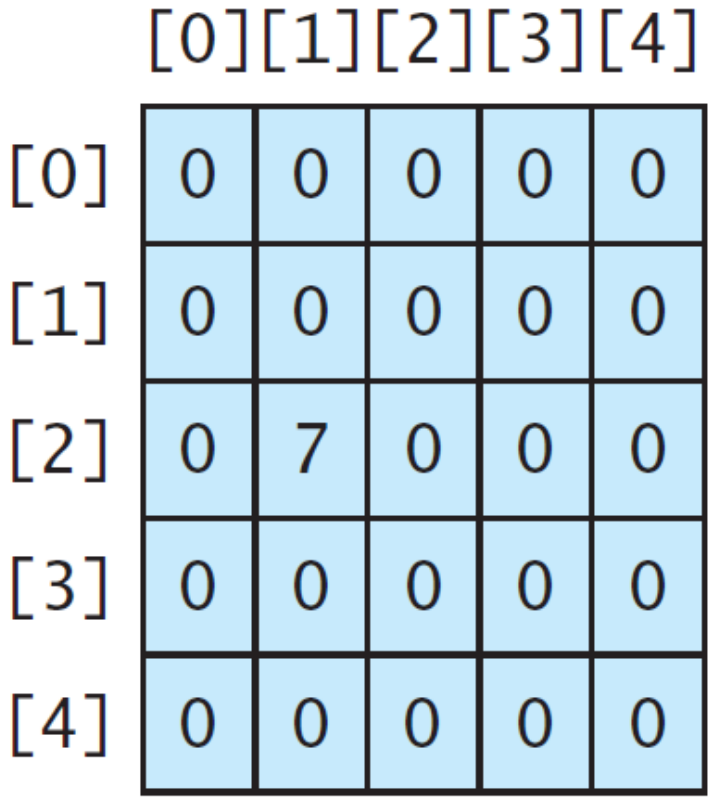
```
int[][] triangleArray = {
   {1, 2, 3, 4, 5},
   {2, 3, 4, 5},
   {3, 4, 5},
   {4, 5},
   {5}
};
```



2-D Arrays: Assignments

For a *2D array*, access a slot via its *row* and *column*.

e.g.,




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

Revisiting the Motivating Example

```
double[][] distances = {
    {0, 983, 787, 714, 1375, 967, 1087},
    {983, 0, 214, 1102, 1763, 1723, 1842},
    {787, 214, 0, 888, 1549, 1548, 1627},
    {714, 1102, 888, 0, 661, 781, 810},
    {1375, 1763, 1549, 661, 0, 1426, 1187},
    {967, 1723, 1548, 781, 1426, 0, 239},
    {1087, 1842, 1627, 810, 1187, 239, 0},
};
```

```
final int CHICAGO = 0;
final int BOSTON = 1;
...
final int HOUSTON = 6;

int MiamiToBoston = distances[MIAMI][BOSTON];
int BostonToNewYork = distances[BOSTON][NEWYORK];
int MiamiToNewYork = MiamiToBoston + BostonToNewYork;
```

Index (1)