Irregular 2D Arrays

the
quick
brown
fox
jumps
over
Topic 12: Irregular Arrays

- Why Irregular Arrays?
- Checking the size
- Allocating an irregular array
- Algorithms on irregular arrays
Why 2D Arrays?

Perhaps you want to store a table of population changes over time...

<table>
<thead>
<tr>
<th></th>
<th>2000</th>
<th>2001</th>
<th>2002</th>
<th>2003</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arizona</td>
<td>5,160,586</td>
<td>5,273,477</td>
<td>5,396,255</td>
<td>5,510,364</td>
</tr>
<tr>
<td>California</td>
<td>33,987,977</td>
<td>34,479,458</td>
<td>34,871,843</td>
<td>35,253,159</td>
</tr>
<tr>
<td>Colorado</td>
<td>4,326,921</td>
<td>4,425,687</td>
<td>4,490,406</td>
<td>4,528,732</td>
</tr>
<tr>
<td>Nevada</td>
<td>2,018,741</td>
<td>2,098,399</td>
<td>2,173,791</td>
<td>2,248,850</td>
</tr>
<tr>
<td>New Mexico</td>
<td>1,821,204</td>
<td>1,831,690</td>
<td>1,855,309</td>
<td>1,877,574</td>
</tr>
<tr>
<td>Texas</td>
<td>20,944,499</td>
<td>21,319,622</td>
<td>21,690,325</td>
<td>22,030,931</td>
</tr>
<tr>
<td>Wyoming</td>
<td>494,300</td>
<td>494,657</td>
<td>500,017</td>
<td>503,453</td>
</tr>
</tbody>
</table>

Why 2D Arrays?

or the pixels of an image...
Why 2D Arrays?

or a map of a city...
Why 2D Arrays?

or a set of Strings.

```
the
quick
brown
fox
jumps
over
```

```
“the”
“quick”
“brown”
“fox”
“jumps”
“over”
```
Why Irregular Arrays?

A set of `String` s is (basically) a set of arrays.

Each element in the array is a different length.
Why Irregular Arrays?

String[]

words

[0]
[1]
[2]
[3]
[4]
[5]

"the"
"quick"
"brown"
"fox"
"jumps"
"over"
Irregular Arrays

- It's not just `String[]`

- There are lots of situations where you might have lots of arrays of different lengths.
Irregular Arrays

int [ ] [ ]

words

[0] → {1, 2, 3}
[1] → {0, 0}
[2] → {-11, 7}
[3] → null
[4] → {}
[5] → {1}
Irregular Arrays

The sub-arrays can be different lengths.

int [[[ ] ] ]

words

The sub-arrays can be different lengths.
The sub-arrays can be very short.
Irregular Arrays

int [[[] words

The sub-arrays can be empty.
The sub-arrays are actually references. They can be `null`.
Irregular Arrays

• In this declaration:
  – What is the type of arr?
  – What is the type of arr[0]?

```java
int[][] arr = new int[4][4];
```
Irregular Arrays

- In this declaration:
  - Is `arr` a reference variable?
  - Is `arr[0]` a reference variable?

```java
int[][] arr = new int[4][4];
```
Topic 12: Irregular Arrays

- Why Irregular Arrays?
- Checking the Size
- Allocating an Irregular Array
- Algorithms on Irregular Arrays
Checking the Size

- In an irregular array, there are many lengths that matter!
  - Length of the outer array
  - Length of each inner array (could vary)

- Also, think about whether or not null arrays are possible in your program
Finding the Size of a 2D Array

```java
char grid[][] = new char[4][2];

int firstSize = grid.length;
int secondSize = grid[0].length;
```

This code example is a rectangular array, but the principle still holds:

You can ask for the length of a sub-array
Finding the Size of a 2D Array

```java
int data[][] = ...;

for (int i=0; i<data.length; i++)
{
    S.o.p(i+" : len="+data[i].length);

    for (int j=0; j<data[i].length; i++)
        S.o.p(data[i][j]);
}
```
The outer `for()` loop loops over the array of arrays.

```java
int data[][] = ... ;

for (int i=0; i<data.length; i++)
{
    S.o.p(i+" : len="+data[i].length);

    for (int j=0; j<data[i].length; i++)
        S.o.p(data[i][j]);
}
```
We can read the length of any given sub-array.

```java
int data[][] = ...;

for (int i=0; i<data.length; i++)
{
    S.o.p(i+": len=“+data[i].length+"\n");
    for (int j=0; j<data[i].length; i++)
    {
        S.o.p(data[i][j]);
    }
}
```
We can loop over each of the sub-arrays in turn.

```java
int data[][] = ... ;

for (int i=0; i<data.length; i++)
{
    S.o.p(i+" : len="+data[i].length);
    for (int j=0; j<data[i].length; i++)
        S.o.p(data[i][j]);
}
```
We can access a given element of the 2D array.

```java
int data[][] = ...;

for (int i=0; i<data.length; i++)
{
    S.o.p(i + " : len = " + data[i].length);

    for (int j=0; j<data[i].length; i++)
        S.o.p(data[i][j]);
}
```
What if one of the elements was **null**? Where would we have a problem?

```java
int data[][] = ... ;

for (int i=0; i<data.length; i++)
{
    S.o.p(i+"": len=""+data[i].length);

    for (int j=0; j<data[i].length; i++)
    {
        S.o.p(data[i][j]);
    }
}
```
We'd hit a **NullPointerException** here.

How can we avoid it?

```java
int data[][] = ... ;

for (int i=0; i<data.length; i++)
{
    S.o.p(i:" len="+data[i].length);

    for (int j=0; j<data[i].length; i++)
        S.o.p(data[i][j]);
}
```
We'd hit a **NullPointerException** here.

How can we avoid it?

```java
int data[][] = ...;

for (int i=0; i<data.length; i++)
{
    if (data[i] == null)
        continue;

    S.o.p(i+" : len="+data[i].length);
    for (int j=0; j<data[i].length; i++)
        S.o.p(data[i][j]);
}
```
Topic 12: Irregular Arrays

- Why Irregular Arrays?
- Checking the size
- Allocating an irregular array
- Algorithms on irregular arrays
Allocating an Irregular Array

```java
int[][] foo = new int[4][];
```

This allocates an array-of-arrays.
Allocating an Irregular Array

```java
int[][] foo = new int[4][];
```

The length of the array is 4.
Allocating an Irregular Array

```
int[][] foo = new int[4][];
```

The type of each element is `int[]`.

Yes, the syntax is weird.
Allocating an Irregular Array

```java
int[][] foo = new int[4][];
```

Since the elements are `int[]` ...

...the array is `int[][]`.
Allocating an Irregular Array

```java
int[][] foo = new int[4][];
```

What are the default values of these four elements?

Hint: array variables are references!
Allocating a Rectangular Array

● Therefore, the old syntax:
  ```java
  int[][] grid = new int[4][8];
  ```

● Is just shorthand for this longer code:
  ```java
  int[][] grid = new int[4][];
  for (int i=0; i<4; i++)
      grid[i] = new int[8];
  ```
Allocating a Rectangular Array

• Therefore, the old syntax:
  ```java
  int[][][] grid = new int[4][8];
  ```

• Is just shorthand for this longer code:
  ```java
  int[][][] grid = new int[4][];
  for (int i=0; i<4; i++)
    grid[i] = new int[8];
  Allocate the outer array.
  ```
Allocating a Rectangular Array

- Therefore, the old syntax:
  ```java
  int[][] grid = new int[4][8];
  ```

- Is just shorthand for this longer code:
  ```java
  int[][] grid = new int[4][];
  for (int i=0; i<4; i++)
    grid[i] = new int[8];
  ```

  For each element in the outer array...
Allocating a Rectangular Array

- Therefore, the old syntax:
  ```java
  int[][] grid = new int[4][8];
  ```

- Is just shorthand for this longer code:
  ```java
  int[][] grid = new int[4][];
  for (int i=0; i<4; i++)
    grid[i] = new int[8];
  ```

For each element in the outer array...
...allocate an inner array.
An Array of Arrays

main():
grid = Stack

Array type = int[] length = 4

Array type = int length = 8
Array type = int length = 8
Array type = int length = 8
Array type = int length = 8

This is how the data **really** looks.
An Array of Arrays

```plaintext
main():
grid =

But it's often OK to think of it as a grid.
```
Even More Dimensions...

- A 3D array is an array of 2D arrays...
- A 4D array is an array of 3D arrays...
Even More Dimensions

Stack

main():
grid =

3D Array
type = int size = 4x4x4
Topic 12: Irregular Arrays

- Why Irregular Arrays?
- Checking the size
- Allocating an irregular array
- Algorithms on irregular arrays
Adding all of the Elements

int sum(int[][] arr2d)
{
    int retval = 0;

    for (int i=0; i<arr2d.length; i++)
        for (int j=0; j<arr2d[i].length; j++)
            retval += arr2d[i][j];

    return retval;
}
Total Length

```c
int totalLen(int[][][] arr2d)
{
    int retval = 0;

    for (int i=0; i<arr2d.length; i++)
        retval += arr2d[i].length;

    return retval;
}
```
Max Length

```java
int maxLen(int[][][] arr2d)
{
    int retval = arr2d[0].length;

    for (int i=1; i<arr2d.length; i++)
        if (arr2d[i].length > retval)
            retval = arr2d[i].length;

    return retval;
}
```
Has null

```java
boolean hasNull(int[][][] arr2d)
{
    for (int i=0; i<arr2d.length; i++)
        if (arr2d[i] == null)
            return true;

    return false;
}
```
int[][] dup(int[][] arr2d)
{
    int[][] retval = new int[arr2d.length][];

    for (int i=0; i<arr2d.length; i++)
    {
        if (arr2d[i] == null)
            continue;

        retval[i] = new int[arr2d[i].length];
        for (int j=0; j<arr2d[i].length; j++)
            retval[i][j] = arr2d[i][j];
    }

    return retval;
}
Topic 12: Irregular Arrays

- Why Irregular Arrays?
- Checking the size
- Allocating an irregular array
- Algorithms on irregular arrays

Summary