CS1110 23 April 2009 Ragged arrays

Reading for today: sec. 9.3.
Reading for next time: sec. 2.5 pp. 83-86 and sec. 4.5 pp. 156-162.

We posted the skeleton for class BigInt from prelim III to the course website; you can use it to try re-doing P3 *on a computer* for study purposes. (Again: we recommend testing your sample-exam solutions at a computer; this *includes* the "draw folders" questions.)

- **The final exam is Friday May 8th, 9:00am-11:30am, Barton Hall (west side).** Please contact mwitlox@cs.cornell.edu ASAP regarding conflicts.
- Graded prelim IIIs can be retrieved from Upson 360, M-F 10am-noon and 2-4pm; bring ID.
- Assignment A7 is due Thursday the 30th.
- The labs next week are optional, and will simply serve as office hours (in the usual lab location).

Review of two-dimensional arrays

Type of d is **int**[][]

("**int** array array"/ "an array of **int** arrays")

To declare variable d:

int d[][];

To create a new array and assign it to d:

d= **new int**[5][4];

or, using an array initializer,

d= **new int**[][]{ {5,4,7,3}, {4,8,9,7}, {5,1,2,3}, {4,1,2,9}, {6,7,8,0} };

d	0	1	2	3
0	5	4	7	3
1	4	8	9	7
2	5	1	2	3
3	4	1	2	9
4	6	7	8	0

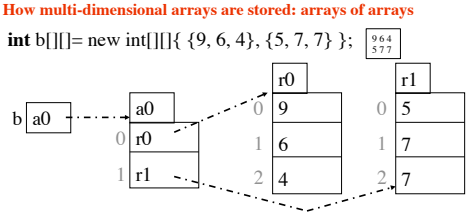
Some mysteries: an odd asymmetry, and strange toString output (see demo).

Number of rows of d: d.length

Number of columns in row r of d: d[r].length

How multi-dimensional arrays are stored: arrays of arrays

int b[][]= new int[][]{ {9, 6, 4}, {5, 7, 7} };



b holds the name of a one-dimensional array object of b.length elements; its elements are 1D arrays.

b[i] holds the name of a 1D array of **int**s of length b[i].length.

java.util.Arrays.deepToString recursively creates an appropriate String.

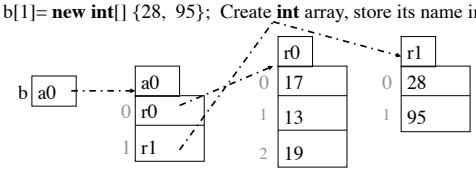
Ragged arrays: rows have different lengths

int[][] b; Declare variable b of type **int**[][]

b= **new int**[2][] Create a 1-D array of length 2 and store its name in b. Its elements have type **int**[] (and start as **null**).

b[0]= **new int**[] {17, 13, 19}; Create **int** array, store its name in b[0].

b[1]= **new int**[] {28, 95}; Create **int** array, store its name in b[1].



Application: "triangular" data

One example: array dist in which dist[i][j] would be the same as dist[j][i].

Another: *Pascal's triangle* (represents a function with interesting symmetries)

			1				0
		1		1			1
	1		2		1		2
1		3		3		1	3
	1		4		1		4
1		5		1			5
							…

The first and last entries of each row are 1.

Each other entry is the sum of the two entries above it.

Row r has r+1 values.

(Coloring the odd numbers starts to look like Sierpinski's triangle…)

5

Pascal's Triangle

			1				0
		1		1			1
	1		2		1		2
1		3		3		1	3
	1		4		1		4
1		5		1			5

Entry p[i][j], entry j of row i, is the number of ways j elements can be chosen from a set of size i !

p[i][j] = "i choose j" = $\binom{i}{j}$

recursive formula (computed via *dynamic programming*):
for $0 < i < j$, p[i][j] = p[i–1][j–1] + p[i–1][j]

6

Pascal's Triangle

			1				0
		1		1			1
	1		2		1		2
1		3		3		1	3
	1		4		1		4
1		5		1			5

Binomial theorem: Row r gives the coefficients of $(x + y)^r$

$(x + y)^2 = 1x^2 + 2xy + 1y^2$

$(x + y)^3 = 1x^3 + 3x^2y + 3xy^2 + 1y^3$

$(x + y)^r = \sum_{0 \le k \le r} (k \text{ choose } r)\, x^k y^{r-k}$

7

Application: representation of (irregular) sparse data

Large collections of *association data* abound, but often, many possible associations have the default value.

Netflix data: (movie, rater, score): 480K × 18K = 8.6B possible scores to track, but there are only (!) 100M actual scores.

GroupLens data (freely distributed by U. Minn): the small set has 943×1682= 1.5M possibilities, but only 100K actual scores.

Main idea:

DON'T store an **int** array of length 1682 for each movie; store a rater-sorted array of score objects corresponding to just the raters who scored that movie (avg. length: 59!).

Another very useful technique (among many more substantive ones; take more CS courses!): map the movie/rater names to ints, b/c they can be meaningful array indices.

8