

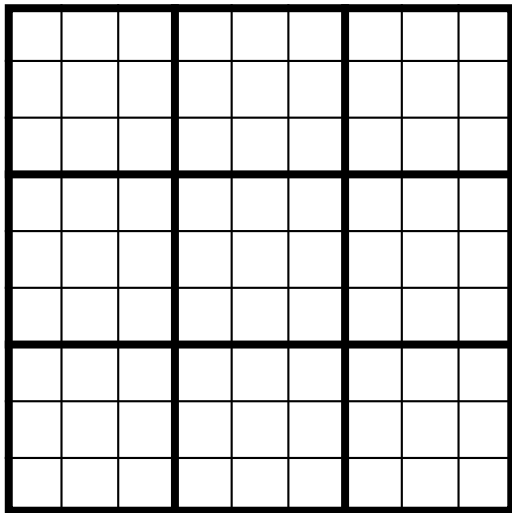
A Systematic Approach to Filling m -by- n Numerical Arrays

Gennady Stolyarov II

Department of Mathematics and Computer Science
Hillsdale College
Hillsdale, MI 49242

November 11, 2008

Sudoku Puzzles



Sudoku Puzzles

Rules of Sudoku

- ▶ The numbers used must be integers 1 through 9.
- ▶ The numbers used must not repeat within a particular row, column, or smaller 3-by-3 bold-outlined square within the larger grid.
- ▶ Each of the integers 1 through 9 must be used once within every particular row, column, or smaller 3-by-3 bold-outlined square.

Sudoku Puzzles

Rules of Sudoku

- ▶ The numbers used must be integers 1 through 9.
- ▶ The numbers used must not repeat within a particular row, column, or smaller 3-by-3 bold-outlined square within the larger grid.
- ▶ Each of the integers 1 through 9 must be used once within every particular row, column, or smaller 3-by-3 bold-outlined square.
- ▶ Every particular puzzle begins with some numbers already filled in, and the solver must complete the puzzle in accord with this given information. This enables a particular *Sudoku* puzzle to have a unique solution.

Sudoku Puzzles

Rules of Sudoku

- ▶ The numbers used must be integers 1 through 9.
- ▶ The numbers used must not repeat within a particular row, column, or smaller 3-by-3 bold-outlined square within the larger grid.
- ▶ Each of the integers 1 through 9 must be used once within every particular row, column, or smaller 3-by-3 bold-outlined square.
- ▶ Every particular puzzle begins with some numbers already filled in, and the solver must complete the puzzle in accord with this given information. This enables a particular *Sudoku* puzzle to have a unique solution.

In 2006, Felgenhauer and Jarvis showed that there are exactly

6,670,903,752,021,072,936,960

valid possibilities for 9-by-9 *Sudoku* grids.

Rules of Kakuro

- ▶ An empty grid is given, with sums specified for each row, column, or part of a row or column.

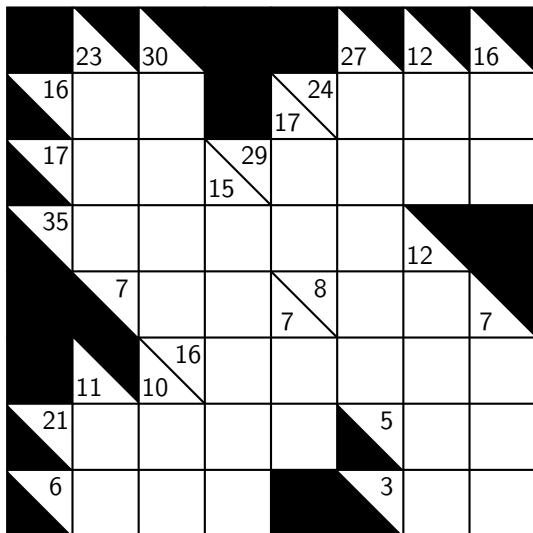
Rules of Kakuro

- ▶ An empty grid is given, with sums specified for each row, column, or part of a row or column.
- ▶ The numbers used must be integers 1 through 9.

Rules of Kakuro

- ▶ An empty grid is given, with sums specified for each row, column, or part of a row or column.
- ▶ The numbers used must be integers 1 through 9.
- ▶ No integer can appear twice within any series of cells for which a sum is specified.

Kakuro Puzzles: An Example



Kakuro Puzzles: Solution to Example

	23	30			27	12	16	
16	9	7		24	8	7	9	
17	8	9	29	17	8	9	5	7
35	6	8	15	5	9	7		
	7	6	1	8	2	6	7	
	11	10	4	7	6	1	3	2
21	8	9	3	1	5	1	4	
6	3	1	2		3	2	1	

Puzzles Under Consideration: m -by- n Arrays

$a_{1,1}$	$a_{1,2}$	$a_{1,n}$	μ_1
$a_{2,1}$	$a_{2,2}$	$a_{2,n}$	μ_2
			
			
			
$a_{m,1}$	$a_{m,2}$	$a_{m,n}$	μ_m

ν_1 ν_2 ν_n | $d = \sum_{i=1}^m \mu_i = \sum_{j=1}^n \nu_j$

Puzzles Under Consideration: m -by- n Arrays

Let d be a positive integer.

Puzzles Under Consideration: m -by- n Arrays

Let d be a positive integer.

Let $\mu = (\mu_1, \mu_2, \dots, \mu_m)$ and $\nu = (\nu_1, \nu_2, \dots, \nu_n)$ be *compositions* of d ,
i.e., $d = \sum \mu_i = \sum \nu_j$.

Puzzles Under Consideration: m -by- n Arrays

Let d be a positive integer.

Let $\mu = (\mu_1, \mu_2, \dots, \mu_m)$ and $\nu = (\nu_1, \nu_2, \dots, \nu_n)$ be *compositions* of d , i.e., $d = \sum \mu_i = \sum \nu_j$.

Question: Is there an array $A = [a_{ij}]_{m \times n}$, whose entries are nonnegative integers, satisfying

$$\sum_{k=1}^n a_{ik} = \mu_i \quad \text{and} \quad \sum_{\ell=1}^m a_{\ell j} = \nu_j$$

for $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$?

Filling Rules

- ▶ Each cell $a_{i,j}$ must be filled with a nonnegative integer. The number 0 and repeats are allowed.

Puzzles Under Consideration: m -by- n Arrays

Filling Rules

- ▶ Each cell $a_{i,j}$ must be filled with a nonnegative integer. The number 0 and repeats are allowed.
- ▶ The sum of the entries in a particular row i must be equal to μ_i .

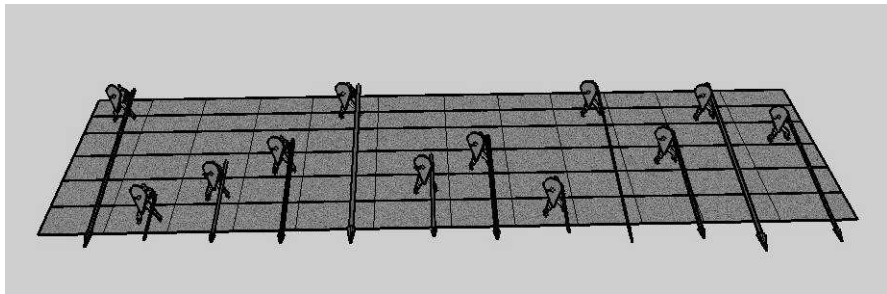
Filling Rules

- ▶ Each cell $a_{i,j}$ must be filled with a nonnegative integer. The number 0 and repeats are allowed.
- ▶ The sum of the entries in a particular row i must be equal to μ_i .
- ▶ The sum of the entries in a particular column j must be equal to ν_j .

Filling Rules

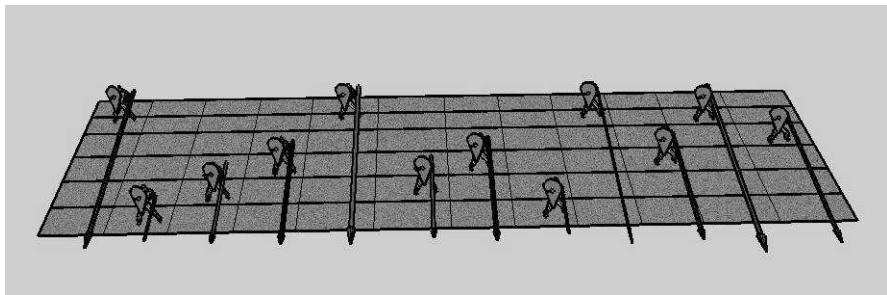
- ▶ Each cell $a_{i,j}$ must be filled with a nonnegative integer. The number 0 and repeats are allowed.
- ▶ The sum of the entries in a particular row i must be equal to μ_i .
- ▶ The sum of the entries in a particular column j must be equal to ν_j .
- ▶ Recall that in each array, the sum of the row constraints is equal to the sum of the column constraints. We can call this sum d , where
$$d = \sum_{i=1}^m \mu_i = \sum_{j=1}^n \nu_j.$$

Applications to Military Formations: Spearmen



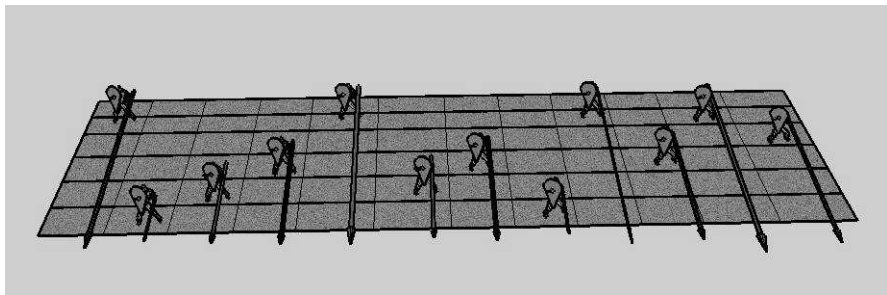
- ▶ Spearmen carry spears of varying lengths.

Applications to Military Formations: Spearmen



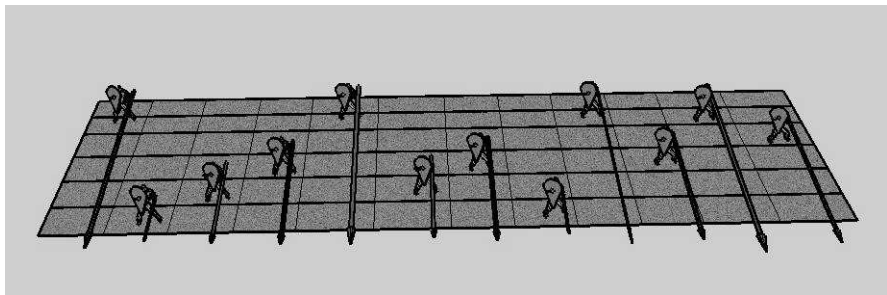
- ▶ Spearmen carry spears of varying lengths.
- ▶ Every single spear must have access to the enemy.

Applications to Military Formations: Spearmen



- ▶ Spearmen carry spears of varying lengths.
- ▶ Every single spear must have access to the enemy.
- ▶ Number of spears of each type determines row constraints.

Applications to Military Formations: Spearmen



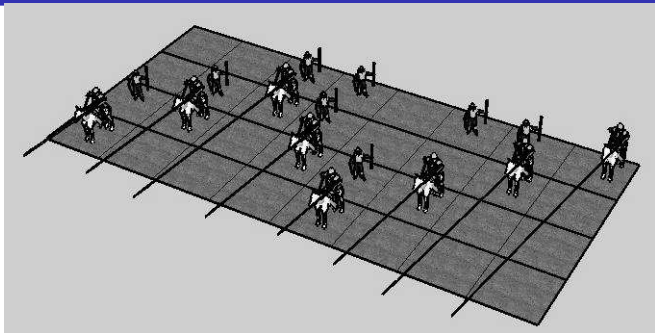
- ▶ Spearmen carry spears of varying lengths.
- ▶ Every single spear must have access to the enemy.
- ▶ Number of spears of each type determines row constraints.
- ▶ Number of spearmen determines column constraints.

Applications to Military Formations: Spearmen

$a_{1,1}$	$a_{1,2}$	$a_{1,d}$	μ_1
$a_{2,1}$	$a_{2,2}$	$a_{2,d}$	μ_2
			
			
			
			
$a_{m,1}$	$a_{m,2}$	$a_{m,d}$	μ_m

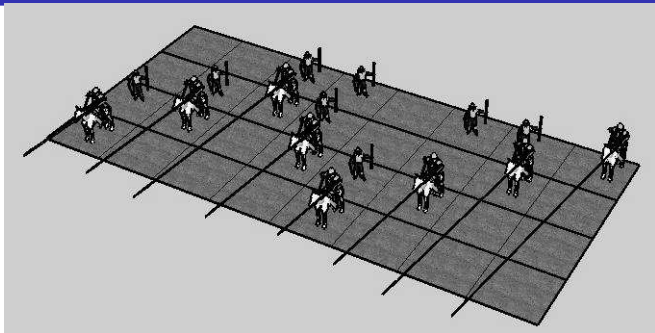
$1 \quad 1 \quad \quad \quad 1$ | $d = \sum_{i=1}^m \mu_i = \sum_{j=1}^d 1$

Applications to Military Formations: Knights and Squires



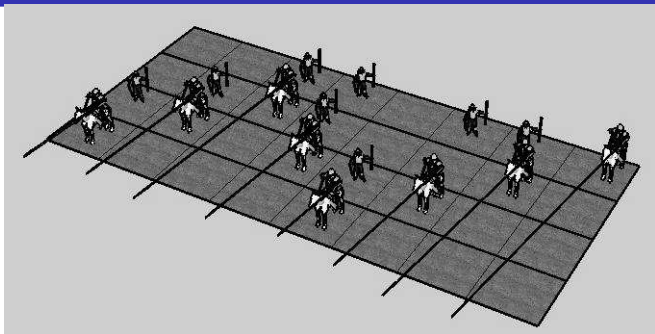
- ▶ Each knight has some number of squires who always stay behind their knights.

Applications to Military Formations: Knights and Squires



- ▶ Each knight has some number of squires who always stay behind their knights.
- ▶ Knights have lances of varying lengths. Every lance must have access to the enemy. The length of a knight's lance will determine his row placement.

Applications to Military Formations: Knights and Squires



- ▶ Each knight has some number of squires who always stay behind their knights.
- ▶ Knights have lances of varying lengths. Every lance must have access to the enemy. The length of a knight's lance will determine his row placement.
- ▶ Number of people in each knight's retinue determines column constraints.

Applications to the Assignment of Tasks

- ▶ Decision-maker has d people under his supervision, n tasks needing to be accomplished, and m different groups of workers.

Applications to the Assignment of Tasks

- ▶ Decision-maker has d people under his supervision, n tasks needing to be accomplished, and m different groups of workers.
- ▶ Military groups can be contingents from different countries in a multinational coalition.

Applications to the Assignment of Tasks

- ▶ Decision-maker has d people under his supervision, n tasks needing to be accomplished, and m different groups of workers.
- ▶ Military groups can be contingents from different countries in a multinational coalition.
- ▶ Civilian groups can be different crews of workers who are not highly specialized.

Applications to the Assignment of Tasks

- ▶ Decision-maker has d people under his supervision, n tasks needing to be accomplished, and m different groups of workers.
- ▶ Military groups can be contingents from different countries in a multinational coalition.
- ▶ Civilian groups can be different crews of workers who are not highly specialized.
- ▶ The column constraints are the numbers of workers required for each task.

Applications to the Assignment of Tasks

- ▶ Decision-maker has d people under his supervision, n tasks needing to be accomplished, and m different groups of workers.
- ▶ Military groups can be contingents from different countries in a multinational coalition.
- ▶ Civilian groups can be different crews of workers who are not highly specialized.
- ▶ The column constraints are the numbers of workers required for each task.
- ▶ The row constraints are the numbers of workers in each group.

Questions Considered

- ▶ Is it possible to always fill such m -by- n arrays?

Questions Considered

- ▶ Is it possible to always fill such m -by- n arrays?
- ▶ If a filling is always possible, is some particular filling of an array unique?

Questions Considered

- ▶ Is it possible to always fill such m -by- n arrays?
- ▶ If a filling is always possible, is some particular filling of an array unique?
- ▶ If a unique filling does not exist for an array, how many distinct fillings exist?

Questions Considered

- ▶ Is it possible to always fill such m -by- n arrays?
- ▶ If a filling is always possible, is some particular filling of an array unique?
- ▶ If a unique filling does not exist for an array, how many distinct fillings exist?
- ▶ Can we arrive at a formula for counting the number of fillings for a general m -by- n array?

Row-by-Row Filling Algorithm

- ▶ We move through each row from left to right, filling each cell with the minimum of the revised row constraint (the μ of the row minus the sum of the values in cells already filled in the row) and the revised column constraint.

Row-by-Row Filling Algorithm

- ▶ We move through each row from left to right, filling each cell with the minimum of the revised row constraint (the μ of the row minus the sum of the values in cells already filled in the row) and the revised column constraint.
- ▶ This row-by-row algorithm fills every cell $a_{i,j}$ with

$$\begin{aligned} a_{i,j} &= \min(\mu_i - a_{i,1} - a_{i,2} - \cdots - a_{i,j-1}, \nu_j - a_{1,j} - a_{2,j} - \cdots - a_{i-1,j}) \\ &= \min\left(\left[\mu_i - \sum_{t=1}^{j-1} a_{i,t} \right], \left[\nu_j - \sum_{t=1}^{i-1} a_{t,j} \right] \right). \end{aligned}$$

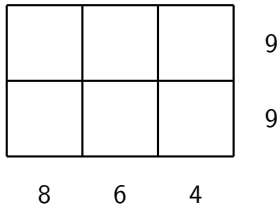
Row-by-Row Filling Algorithm

- ▶ We move through each row from left to right, filling each cell with the minimum of the revised row constraint (the μ of the row minus the sum of the values in cells already filled in the row) and the revised column constraint.
- ▶ This row-by-row algorithm fills every cell $a_{i,j}$ with

$$\begin{aligned} a_{i,j} &= \min(\mu_i - a_{i,1} - a_{i,2} - \cdots - a_{i,j-1}, \nu_j - a_{1,j} - a_{2,j} - \cdots - a_{i-1,j}) \\ &= \min\left(\left[\mu_i - \sum_{t=1}^{j-1} a_{i,t} \right], \left[\nu_j - \sum_{t=1}^{i-1} a_{t,j} \right] \right). \end{aligned}$$

- ▶ This algorithm was proved by induction on the number of rows.

Example of Row-Filling Algorithm's Application



*	*	*
*	*	*

→

8	1	0
*	*	*

→

8	1	0
0	5	4

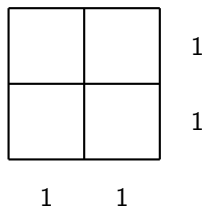
Another Example of Row-Filling Algorithm's Application

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



8	0	0	0	0	0	0	0	0	0
1	7	0	0	0	0	0	0	0	0
0	1	7	0	0	0	0	0	0	0
0	0	0	4	0	0	0	0	0	0
0	0	0	2	2	0	0	0	0	0
0	0	0	0	3	1	0	0	0	0
0	0	0	0	0	3	0	0	0	0
0	0	0	0	0	0	3	0	0	0
0	0	0	0	0	0	0	2	1	0

Example of Non-Unique Fillings



Two possible fillings:

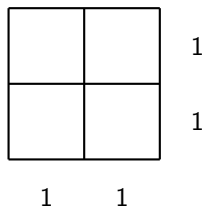
1	0
0	1

 and

0	1
1	0

.

Example of Non-Unique Fillings



Two possible fillings:

1	0
0	1

 and

0	1
1	0

.

For a d -by- d puzzle all of whose constraints are equal to 1, there are $d!$ possible fillings.

The Number of Fillings for an m -by- d Array

$a_{1,1}$	$a_{1,2}$	$a_{1,d}$	μ_1
$a_{2,1}$	$a_{2,2}$	$a_{2,d}$	μ_2
.....
$a_{m,1}$	$a_{m,2}$	$a_{m,d}$	μ_m

$1 \quad 1 \quad \dots \quad 1$ | $d = \sum_{i=1}^m \mu_i = \sum_{j=1}^d 1$

- Ways to fill first row: $C(d, \mu_1) = \frac{d!}{(\mu_1)!(d-\mu_1)!}$.

The Number of Fillings for an m -by- d Array

$a_{1,1}$	$a_{1,2}$	$a_{1,d}$	μ_1
$a_{2,1}$	$a_{2,2}$	$a_{2,d}$	μ_2
.....
$a_{m,1}$	$a_{m,2}$	$a_{m,d}$	μ_m

$1 \quad 1 \quad \dots \quad 1$ | $d = \sum_{i=1}^m \mu_i = \sum_{j=1}^d 1$

- ▶ Ways to fill first row: $C(d, \mu_1) = \frac{d!}{(\mu_1)!(d-\mu_1)!}$.
- ▶ Ways to fill second row: $C(d - \mu_1, \mu_2)$.

The Number of Fillings for an m -by- d Array

$a_{1,1}$	$a_{1,2}$	$a_{1,d}$	μ_1
$a_{2,1}$	$a_{2,2}$	$a_{2,d}$	μ_2
.....
$a_{m,1}$	$a_{m,2}$	$a_{m,d}$	μ_m

$1 \quad 1 \quad \dots \quad 1 \quad | \quad d = \sum_{i=1}^m \mu_i = \sum_{j=1}^d 1$

- ▶ Ways to fill first row: $C(d, \mu_1) = \frac{d!}{(\mu_1)!(d-\mu_1)!}$.
- ▶ Ways to fill second row: $C(d - \mu_1, \mu_2)$.
- ▶ Ways to fill m th row: $C(d - \mu_1 - \mu_2 - \dots - \mu_{m-1}, \mu_m)$.

Formula The Number of Fillings for an m -by- d Array

- ▶ There are

$$C(d, \mu_1)C(d - \mu_1, \mu_2)C(d - \mu_1 - \mu_2, \mu_3) \dots C(d - \mu_1 - \mu_2 - \dots - \mu_{m-1}, \mu_m) \\ = \frac{d!}{(\mu_1)! (\mu_2)! \dots (\mu_m)!}$$

ways to fill a m -by- d array, where d is the sum of the row constraints and the sum of the column constraints.

Formula The Number of Fillings for an m -by- d Array

- ▶ There are

$$C(d, \mu_1)C(d - \mu_1, \mu_2)C(d - \mu_1 - \mu_2, \mu_3) \dots C(d - \mu_1 - \mu_2 - \dots - \mu_{m-1}, \mu_m) \\ = \frac{d!}{(\mu_1)! (\mu_2)! \dots (\mu_m)!}$$

ways to fill a m -by- d array, where d is the sum of the row constraints and the sum of the column constraints.

- ▶ When the column constraints are allowed to be greater than 1, the problem becomes substantially more complicated.

Yes, We Can Rearrange Rows and Columns.

- ▶ Previously, adding the values in the first row from left to right, we had $\mu_1 = a_{1,1} + a_{1,2} + \cdots + a_{1,n}$

Yes, We Can Rearrange Rows and Columns.

- ▶ Previously, adding the values in the first row from left to right, we had $\mu_1 = a_{1,1} + a_{1,2} + \cdots + a_{1,n}$
- ▶ Now, adding the values in the first row from left to right, we have $\mu_1 = a_{1,2} + a_{1,1} + \cdots + a_{1,n}$

Yes, We Can Rearrange Rows and Columns.

- ▶ Previously, adding the values in the first row from left to right, we had $\mu_1 = a_{1,1} + a_{1,2} + \cdots + a_{1,n}$
- ▶ Now, adding the values in the first row from left to right, we have $\mu_1 = a_{1,2} + a_{1,1} + \cdots + a_{1,n}$
- ▶ So row sums are unchanged by switching the columns. Likewise, column sums are unchanged by switching the rows.

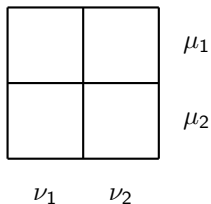
Yes, We Can Rearrange Rows and Columns.

- ▶ Previously, adding the values in the first row from left to right, we had $\mu_1 = a_{1,1} + a_{1,2} + \cdots + a_{1,n}$
- ▶ Now, adding the values in the first row from left to right, we have $\mu_1 = a_{1,2} + a_{1,1} + \cdots + a_{1,n}$
- ▶ So row sums are unchanged by switching the columns. Likewise, column sums are unchanged by switching the rows.
- ▶ Permuting the rows and columns of an array in any fashion will not fundamentally change the nature of the array-filling problem. The number of fillings in the array thus permuted will equal the number of fillings in the original array.

Yes, We Can Rearrange Rows and Columns.

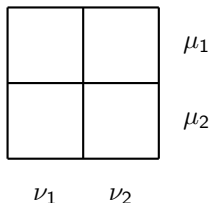
- ▶ Previously, adding the values in the first row from left to right, we had $\mu_1 = a_{1,1} + a_{1,2} + \dots + a_{1,n}$
- ▶ Now, adding the values in the first row from left to right, we have $\mu_1 = a_{1,2} + a_{1,1} + \dots + a_{1,n}$
- ▶ So row sums are unchanged by switching the columns. Likewise, column sums are unchanged by switching the rows.
- ▶ Permuting the rows and columns of an array in any fashion will not fundamentally change the nature of the array-filling problem. The number of fillings in the array thus permuted will equal the number of fillings in the original array.
- ▶ Hereafter, we will assume that our row and column constraints are arranged in descending order.

The Number of Fillings for a 2-by-2 Array



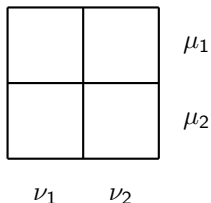
- ▶ The placement of a value in a single cell will determine the entire array. Where can we place a value to always generate a possible filling?

The Number of Fillings for a 2-by-2 Array



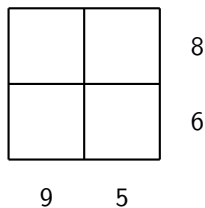
- ▶ The placement of a value in a single cell will determine the entire array. Where can we place a value to always generate a possible filling?
- ▶ We can choose any number to go into $a_{2,2}$, provided that it satisfies both the row constraint μ_2 and the column constraint ν_2 .

The Number of Fillings for a 2-by-2 Array



- ▶ The placement of a value in a single cell will determine the entire array. Where can we place a value to always generate a possible filling?
- ▶ We can choose any number to go into $a_{2,2}$, provided that it satisfies both the row constraint μ_2 and the column constraint ν_2 .
- ▶ There are $\min(\mu_2, \nu_2) + 1$ fillings for a 2-by-2 array.

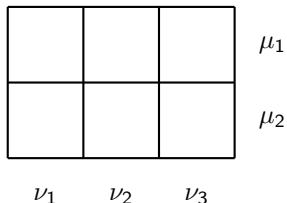
Example Illustrating The Number of Fillings for a 2-by-2 Array



This array has $\min(\mu_2, \nu_2) + 1 = \min(6, 5) + 1 = 6$ fillings:

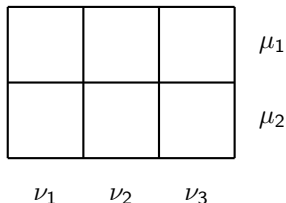
3	5	4	4	5	3	6	2	7	1	8	0
6	0	5	1	4	2	3	3	2	4	1	5

The Number of Fillings for a 2-by-3 Array



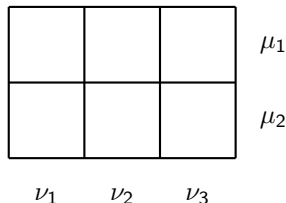
- ▶ We place $k_{2,3}$ in $a_{2,3}$, ranging from 0 to $\min(\mu_2, \nu_3)$.

The Number of Fillings for a 2-by-3 Array



- ▶ We place $k_{2,3}$ in $a_{2,3}$, ranging from 0 to $\min(\mu_2, \nu_3)$.
- ▶ We get a 2-by-2 subarray with new row constraints $\mu_1 - \nu_3 + k_{2,3}$ and $\mu_2 - k_{2,3}$.

The Number of Fillings for a 2-by-3 Array



- ▶ We place $k_{2,3}$ in $a_{2,3}$, ranging from 0 to $\min(\mu_2, \nu_3)$.
- ▶ We get a 2-by-2 subarray with new row constraints $\mu_1 - \nu_3 + k_{2,3}$ and $\mu_2 - k_{2,3}$.
- ▶ We place another value in $a_{1,2}$ or $a_{2,2}$, depending on which row is smaller. This placement determines the array.

The Number of Fillings for a 2-by-3 Array

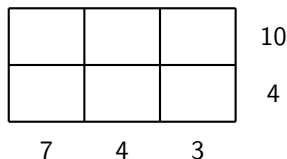
			μ_1
			μ_2
ν_1	ν_2	ν_3	

- ▶ We place $k_{2,3}$ in $a_{2,3}$, ranging from 0 to $\min(\mu_2, \nu_3)$.
- ▶ We get a 2-by-2 subarray with new row constraints $\mu_1 - \nu_3 + k_{2,3}$ and $\mu_2 - k_{2,3}$.
- ▶ We place another value in $a_{1,2}$ or $a_{2,2}$, depending on which row is smaller. This placement determines the array.
- ▶ The expression

$$\sum_{k_{2,3}=0}^{\min(\mu_2, \nu_3)} (\min(\nu_2, \mu_2 - k_{2,3}, \mu_1 - \nu_3 + k_{2,3}) + 1)$$

gives the number of fillings for a 2-by-3 array.

Example Illustrating The Number of Fillings for a 2-by-3 Array



The number of fillings is

$$\sum_{k_{2,3}=0}^{\min(\mu_2, \nu_3)} (\min(\nu_2, \mu_2 - k_{2,3}, \mu_1 - \nu_3 + k_{2,3}) + 1)$$

$$= \sum_{k_{2,3}=0}^{\min(4, 3)} (\min(4, 4 - k_{2,3}, 10 - 3 + k_{2,3}) + 1) = \sum_{k_{2,3}=0}^3 (\min(4, 4 - k_{2,3}, 7 + k_{2,3}) + 1)$$

$$= \sum_{k_{2,3}=0}^3 (4 - k_{2,3}) + 1 = [(4 - 0) + 1] + [(4 - 1) + 1] + [(4 - 2) + 1] + [(4 - 3) + 1] = 14$$

The Number of Fillings for a 2-by-4 Array

				μ_1
				μ_2
ν_1	ν_2	ν_3	ν_4	

- ▶ We place $k_{2,4}$ in $a_{2,4}$, ranging from 0 to $\min(\mu_2, \nu_4)$.

The Number of Fillings for a 2-by-4 Array

				μ_1
				μ_2
ν_1	ν_2	ν_3	ν_4	

- ▶ We place $k_{2,4}$ in $a_{2,4}$, ranging from 0 to $\min(\mu_2, \nu_4)$.
- ▶ We are left to fill a 2-by-3 array with row constraints $(\mu_1 - \nu_4 + k_{2,4}, \mu_2 - k_{2,4})$. We can use the formula for filling a 2-by-3 array, provided we can place our next subscripted k in $a_{2,3}$.

The Number of Fillings for a 2-by-4 Array

				μ_1
				μ_2
ν_1	ν_2	ν_3	ν_4	

- ▶ We place $k_{2,4}$ in $a_{2,4}$, ranging from 0 to $\min(\mu_2, \nu_4)$.
- ▶ We are left to fill a 2-by-3 array with row constraints $(\mu_1 - \nu_4 + k_{2,4}, \mu_2 - k_{2,4})$. We can use the formula for filling a 2-by-3 array, provided we can place our next subscripted k in $a_{2,3}$.
- ▶ We can show that for $n \geq 4$, it is always the case that $\nu_{n-1} \leq \mu_1 - \nu_n + k_{2,n}$. Thus, $\nu_3 \leq \mu_1 - \nu_4 + k_{2,4}$ and it is legitimate to place the next subscripted k in $a_{2,3}$.

The Number of Fillings for a 2-by-4 Array

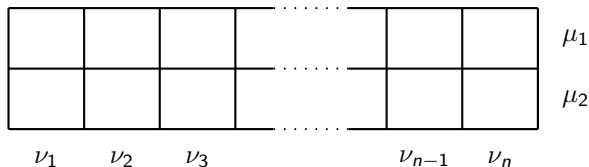
				μ_1
				μ_2
ν_1	ν_2	ν_3	ν_4	

- ▶ We place $k_{2,4}$ in $a_{2,4}$, ranging from 0 to $\min(\mu_2, \nu_4)$.
- ▶ We are left to fill a 2-by-3 array with row constraints $(\mu_1 - \nu_4 + k_{2,4}, \mu_2 - k_{2,4})$. We can use the formula for filling a 2-by-3 array, provided we can place our next subscripted k in $a_{2,3}$.
- ▶ We can show that for $n \geq 4$, it is always the case that $\nu_{n-1} \leq \mu_1 - \nu_n + k_{2,n}$. Thus, $\nu_3 \leq \mu_1 - \nu_4 + k_{2,4}$ and it is legitimate to place the next subscripted k in $a_{2,3}$.
- ▶ There are

$$\sum_{k_{2,4}=0}^{\min(\mu_2, \nu_4)} \sum_{k_{2,3}=0}^{\min(\mu_2 - k_{2,4}, \nu_3)} (\min(\nu_2, \mu_2 - k_{2,3} - k_{2,4}, \mu_1 - \nu_4 - \nu_3 + k_{2,3} + k_{2,4}) + 1)$$

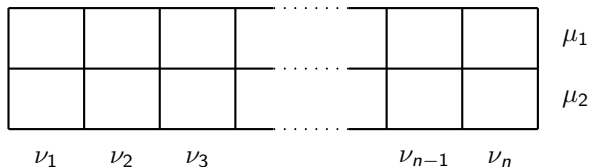
fillings for a 2-by-4 array.

The Number of Fillings for a General 2-by- n Array



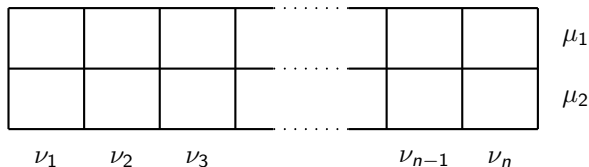
- ▶ We place $k_{2,n}$, a value anywhere between 0 and $\min(\nu_n, \mu_2)$, in cell $a_{2,n}$.

The Number of Fillings for a General 2-by- n Array



- ▶ We place $k_{2,n}$, a value anywhere between 0 and $\min(\nu_n, \mu_2)$, in cell $a_{2,n}$.
- ▶ The second placement will be of $k_{2,n-1}$, a value anywhere between 0 and $\min(\nu_{n-1}, \mu_2 - k_{2,n})$, in cell $a_{2,n-1}$.

The Number of Fillings for a General 2-by- n Array



- ▶ We place $k_{2,n}$, a value anywhere between 0 and $\min(\nu_n, \mu_2)$, in cell $a_{2,n}$.
- ▶ The second placement will be of $k_{2,n-1}$, a value anywhere between 0 and $\min(\nu_{n-1}, \mu_2 - k_{2,n})$, in cell $a_{2,n-1}$.
- ▶ The two outermost summations will be of the form

$$\sum_{k_{2,n}=0}^{\min(\mu_2, \nu_n)} \sum_{k_{2,n-1}=0}^{\min(\nu_{n-1}, \mu_2 - k_{2,n})}$$

The Number of Fillings for a General 2-by- n Array

The number of fillings for a 2-by- n array where $n \geq 3$ is

$$\sum_{k_{2,n}=0}^{\min(\mu_2, \nu_n)} \sum_{k_{2,n-1}=0}^{\min(\nu_{n-1}, \mu_2 - k_{2,n})} \sum_{k_{2,n-2}=0}^{\min(\nu_{n-2}, \mu_2 - k_{2,n} - k_{2,n-1})} \cdots \sum_{k_{2,3}=0}^{\min(\nu_3, \mu_2 - k_{2,n} - k_{2,n-1} - \cdots - k_{2,4}, \nu_3)}$$

$$\min(\nu_2, \mu_2 - \sum_{i=3}^n k_{2,i}, \mu_1 - \sum_{j=3}^n \nu_j + \sum_{i=3}^n k_{2,i}) + 1$$

This formula can be proved by induction on the number of columns.

A More Concise and Convenient Notation: \triangleright and \triangleleft Notation

- ▶ We will use

$$\triangleright^{\ell;a,b} \text{ to represent } \sum_{t=a}^b k_{\ell,t}.$$

The ℓ in this notation represents the row in which values are being added. Within this row, the values $k_{\ell,a}$ through $k_{\ell,b}$ are being added. For example, $\triangleright^{\ell;2,5} = k_{\ell,2} + k_{\ell,3} + k_{\ell,4} + k_{\ell,5}$.

A More Concise and Convenient Notation: \triangleright and \triangledown Notation

- ▶ We will use

$$\triangleright^{\ell;a,b} \text{ to represent } \sum_{t=a}^b k_{\ell,t}.$$

The ℓ in this notation represents the row in which values are being added. Within this row, the values $k_{\ell,a}$ through $k_{\ell,b}$ are being added. For example, $\triangleright^{\ell;2,5} = k_{\ell,2} + k_{\ell,3} + k_{\ell,4} + k_{\ell,5}$.

- ▶ We will use

$$\triangledown^{\ell;a,b} \text{ to represent } \sum_{t=a}^b k_{t,\ell}.$$

For instance, $k_{2,7} + k_{3,7} + k_{4,7} + k_{5,7} + k_{6,7} = \triangledown^{7;2,6}$.

A More Concise and Convenient Notation: \sqcup Notation

In general, we will let

$$\sqcup_{q:t;s} \sum_{k_{q,j}=0}^f$$

represent the composition of all the summations where q is any value from s to t , and j is any value from u to v . The summations are applied so that the outermost summation is the one where $q = t$ and $j = v$, where t and v are the *highest* row and column numbers within the given range. Within each summation in the composition above, the value of $k_{q,j}$ can range from 0 to some value f .

A More Concise and Convenient Notation: \sqcup Notation: Examples



$$\bigsqcup_{q:9;8} \sum_{k_q,j=0}^f = \sum_{k_9,6=0}^f \sum_{k_9,5=0}^f \sum_{k_9,4=0}^f \sum_{k_9,3=0}^f \sum_{k_9,2=0}^f \sum_{k_8,6=0}^f \sum_{k_8,5=0}^f \sum_{k_8,4=0}^f \sum_{k_8,3=0}^f \sum_{k_8,2=0}^f$$

A More Concise and Convenient Notation: \sqcup Notation: Examples



$$\sqcup_{q:9;8}^{j:6;2} \sum_{k_{q,j}=0}^f = \sum_{k_{9,6}=0}^f \sum_{k_{9,5}=0}^f \sum_{k_{9,4}=0}^f \sum_{k_{9,3}=0}^f \sum_{k_{9,2}=0}^f \sum_{k_{8,6}=0}^f \sum_{k_{8,5}=0}^f \sum_{k_{8,4}=0}^f \sum_{k_{8,3}=0}^f \sum_{k_{8,2}=0}^f$$



$$\sqcup_{q:5;3}^{j:6;4} \sum_{k_{q,j}=0}^{\min(\nu_j - \nabla^{j;q+1,m}, \mu_q - \triangleright^{q;j+1,n})}$$

is the composition of all the summations for picking the values of cells in the rectangle whose corner cells are $a_{3,4}$, $a_{3,6}$, $a_{5,4}$, and $a_{5,6}$. The outermost summation will be the one in which $(q, j) = (5, 6)$, and the innermost summation will be the one in which $(q, j) = (3, 4)$.

A More Concise and Convenient Notation: \sqcup Notation: More Examples

- ▶ In the formulas we will be working with, instead of writing

$$\min(\nu_j - \nabla^{j;q+1,m}) \sum_{k_{q,j}=0}$$

after each \sqcup operator, we will write the overall composition as

$$\begin{array}{c} j:v_1;u_1 \\ \sqcup \\ q:t_1;s_1 \end{array} \begin{array}{c} j:v_2;u_2 \\ \sqcup \\ q:t_2;s_2 \end{array} \min(\nu_j - \nabla^{j;q+1,m}) \sum_{k_{q,j}=0}$$

A More Concise and Convenient Notation: \sqcup Notation: More Examples

- ▶ In the formulas we will be working with, instead of writing

$$\min(\nu_j - \nabla^{j;q+1,m}) \sum_{k_{q,j}=0}$$

after each \sqcup operator, we will write the overall composition as

$$\begin{array}{ccc} j:v_1;u_1 & j:v_2;u_2 & \min(\nu_j - \nabla^{j;q+1,m}) \\ \sqcup & \sqcup & \sum \\ q:t_1;s_1 & q:t_2;s_2 & k_{q,j}=0 \end{array}$$

- ▶ The number of fillings for a 2-by- n array where $n \geq 3$ is

$$\begin{array}{ccc} j:n;3 & \min(\nu_j, \mu_q - \nabla^{q;j+1,n}) & \\ \sqcup & \sum & \\ q:2;2 & k_{q,j}=0 & \min(\nu_2, \mu_2 - \nabla^{2;3,n}, \mu_1 - \sum_{j=3}^n \nu_j + \nabla^{2;3,n}) + 1. \end{array}$$

The Flexibility of Cell Placements for Arrays With More Than 2 Rows and 2 Columns

- ▶ We consider some cell $a_{i,j}$ of an array for which all cells to the right and all cells below $a_{i,j}$ have been filled. We fill $a_{i,j}$ with $k_{i,j}$.

The Flexibility of Cell Placements for Arrays With More Than 2 Rows and 2 Columns

- ▶ We consider some cell $a_{i,j}$ of an array for which all cells to the right and all cells below $a_{i,j}$ have been filled. We fill $a_{i,j}$ with $k_{i,j}$.
- ▶ Next, we can place either $k_{i,j-1}$ in $a_{i,j-1}$ or $k_{i-1,j}$ in $a_{i-1,j}$, without regard to the order of their placement. This is because the row and column constraints affecting cell $a_{i,j-1}$ are independent of the row and column constraints affecting cell $a_{i-1,j}$.

The Flexibility of Cell Placements for Arrays With More Than 2 Rows and 2 Columns

- ▶ We consider some cell $a_{i,j}$ of an array for which all cells to the right and all cells below $a_{i,j}$ have been filled. We fill $a_{i,j}$ with $k_{i,j}$.
- ▶ Next, we can place either $k_{i,j-1}$ in $a_{i,j-1}$ or $k_{i-1,j}$ in $a_{i-1,j}$, without regard to the order of their placement. This is because the row and column constraints affecting cell $a_{i,j-1}$ are independent of the row and column constraints affecting cell $a_{i-1,j}$.
- ▶ We can therefore start filling any array by first filling the bottom-right corner cell $a_{m,n}$ and filling either one row at a time or one column at a time. By using this procedure, it is always possible to get a filling of the array.

The Flexibility of Cell Placements for Arrays With More Than 2 Rows and 2 Columns

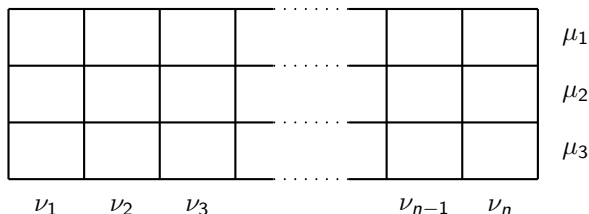
- ▶ We consider some cell $a_{i,j}$ of an array for which all cells to the right and all cells below $a_{i,j}$ have been filled. We fill $a_{i,j}$ with $k_{i,j}$.
- ▶ Next, we can place either $k_{i,j-1}$ in $a_{i,j-1}$ or $k_{i-1,j}$ in $a_{i-1,j}$, without regard to the order of their placement. This is because the row and column constraints affecting cell $a_{i,j-1}$ are independent of the row and column constraints affecting cell $a_{i-1,j}$.
- ▶ We can therefore start filling any array by first filling the bottom-right corner cell $a_{m,n}$ and filling either one row at a time or one column at a time. By using this procedure, it is always possible to get a filling of the array.
- ▶ Each $k_{s,t}$ thus placed is subject to no restrictions besides the minimum of the revised row constraint and the revised column constraint of cell $a_{s,t}$.

The Flexibility of Cell Placements for Arrays With More Than 2 Rows and 2 Columns

- ▶ We consider some cell $a_{i,j}$ of an array for which all cells to the right and all cells below $a_{i,j}$ have been filled. We fill $a_{i,j}$ with $k_{i,j}$.
- ▶ Next, we can place either $k_{i,j-1}$ in $a_{i,j-1}$ or $k_{i-1,j}$ in $a_{i-1,j}$, without regard to the order of their placement. This is because the row and column constraints affecting cell $a_{i,j-1}$ are independent of the row and column constraints affecting cell $a_{i-1,j}$.
- ▶ We can therefore start filling any array by first filling the bottom-right corner cell $a_{m,n}$ and filling either one row at a time or one column at a time. By using this procedure, it is always possible to get a filling of the array.
- ▶ Each $k_{s,t}$ thus placed is subject to no restrictions besides the minimum of the revised row constraint and the revised column constraint of cell $a_{s,t}$.
- ▶ This procedure of filling the array can be performed until the original array has been reduced to a 2-by-2 subarray.

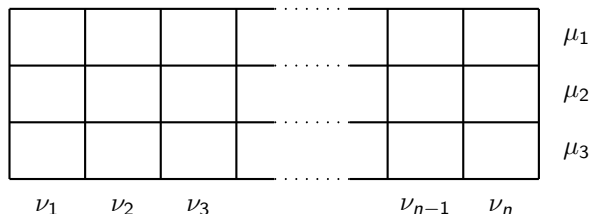
The Number of Fillings for a 3-by- n Array

We consider the following 3-by- n array with $\mu_1 \geq \mu_2 \geq \mu_3$ and $\nu_1 \geq \nu_2 \geq \dots \geq \nu_n$:



We first find out the number of ways to fill in the third row of the array, so that each filling generates a distinct 2-by- n array. For counting the number of ways to fill this array, we can apply the formula derived earlier.

The Number of Fillings for a 3-by- n Array: Continued



The number of fillings of a 3-by- n array, where $n \geq 3$, is

$$\sum_{\substack{j:n;2 \\ q:3;3}} \sum_{\substack{j:n;3 \\ q:2;2}} \sum_{k_{q,j}=0}^{\min(\nu_j - \nabla^{j;q+1,3}, \mu_q - \triangleright^{q;j+1,n})} \min(\nu_2 - k_{3,2}, \mu_2 - \triangleright^{2;3,n}, \mu_1 - \sum_{j=3}^n \nu_j + \triangleright^{2;3,n} + \triangleright^{3;3,n}) + 1$$

This formula can be proved by mathematical induction on the number of columns.

The Number of Fillings for a m -by- n Array: Some Considerations

- ▶ From now on, we assume that we are given an m -by- n array in which $m \geq 3$ and $n \geq 3$.

The Number of Fillings for a m -by- n Array: Some Considerations

- ▶ From now on, we assume that we are given an m -by- n array in which $m \geq 3$ and $n \geq 3$.
- ▶ The second row requires the selection of $n - 2$ values (of subscripted k 's) before a 2-by- n array becomes reduced to a 2-by-2 array.

The Number of Fillings for a m -by- n Array: Some Considerations

- ▶ From now on, we assume that we are given an m -by- n array in which $m \geq 3$ and $n \geq 3$.
- ▶ The second row requires the selection of $n - 2$ values (of subscripted k 's) before a 2-by- n array becomes reduced to a 2-by-2 array.
- ▶ The third row requires the selection of $n - 1$ values before a 3-by- n array becomes reduced to a 2-by- n array. Likewise, for any $m \geq 3$, the m th row requires the selection of $n - 1$ values before a m -by- n array becomes reduced to a $(m - 1)$ -by- n array.

The Number of Fillings for a m -by- n Array: Some Considerations

- ▶ From now on, we assume that we are given an m -by- n array in which $m \geq 3$ and $n \geq 3$.
- ▶ The second row requires the selection of $n - 2$ values (of subscripted k 's) before a 2-by- n array becomes reduced to a 2-by-2 array.
- ▶ The third row requires the selection of $n - 1$ values before a 3-by- n array becomes reduced to a 2-by- n array. Likewise, for any $m \geq 3$, the m th row requires the selection of $n - 1$ values before a m -by- n array becomes reduced to a $(m - 1)$ -by- n array.
- ▶ Thus, a m -by- n array will require choosing $(n - 2) + (m - 2)(n - 1) = n - 2 + mn - 2n - m + 2 = mn - n - m$ values of subscripted k 's before the array is reduced to a 2-by-2 array. Thus, the formula for the number of fillings of a m -by- n array will have $mn - n - m$ summations in it.

The Number of Fillings for a m -by- n Array: Some Considerations

- ▶ From now on, we assume that we are given an m -by- n array in which $m \geq 3$ and $n \geq 3$.
- ▶ The second row requires the selection of $n - 2$ values (of subscripted k 's) before a 2-by- n array becomes reduced to a 2-by-2 array.
- ▶ The third row requires the selection of $n - 1$ values before a 3-by- n array becomes reduced to a 2-by- n array. Likewise, for any $m \geq 3$, the m th row requires the selection of $n - 1$ values before a m -by- n array becomes reduced to a $(m - 1)$ -by- n array.
- ▶ Thus, a m -by- n array will require choosing $(n - 2) + (m - 2)(n - 1) = n - 2 + mn - 2n - m + 2 = mn - n - m$ values of subscripted k 's before the array is reduced to a 2-by-2 array. Thus, the formula for the number of fillings of a m -by- n array will have $mn - n - m$ summations in it.
- ▶ The formula for the number of fillings of a 2-by-3 array involves $2 \times 3 - 3 - 2 = 1$ summation.

The Number of Fillings for a m -by- n Array: Some Considerations

- ▶ From now on, we assume that we are given an m -by- n array in which $m \geq 3$ and $n \geq 3$.
- ▶ The second row requires the selection of $n - 2$ values (of subscripted k 's) before a 2-by- n array becomes reduced to a 2-by-2 array.
- ▶ The third row requires the selection of $n - 1$ values before a 3-by- n array becomes reduced to a 2-by- n array. Likewise, for any $m \geq 3$, the m th row requires the selection of $n - 1$ values before a m -by- n array becomes reduced to a $(m - 1)$ -by- n array.
- ▶ Thus, a m -by- n array will require choosing $(n - 2) + (m - 2)(n - 1) = n - 2 + mn - 2n - m + 2 = mn - n - m$ values of subscripted k 's before the array is reduced to a 2-by-2 array. Thus, the formula for the number of fillings of a m -by- n array will have $mn - n - m$ summations in it.
- ▶ The formula for the number of fillings of a 2-by-3 array involves $2 \times 3 - 3 - 2 = 1$ summation.
- ▶ The formula for the number of fillings of a 3-by-7 array has $3 \times 7 - 7 - 3 = 11$ summations in it.

The Number of Fillings for a m -by- n Array: Further Considerations

- ▶ When filling the cell $a_{q,j}$ of a m -by- n array the row constraint will need to be modified by subtracting $\triangleright^{q:j+1,n}$ from μ_q .

The Number of Fillings for a m -by- n Array: Further Considerations

- ▶ When filling the cell $a_{q,j}$ of a m -by- n array the row constraint will need to be modified by subtracting $\triangleright^{q;j+1,n}$ from μ_q .
- ▶ When filling the q th row of a m -by- n array, the column constraints will need to be modified by subtracting $\nabla^{j;q+1,m}$ from each ν_j to account for the cells already filled below $a_{q,j}$.

The Number of Fillings for a m -by- n Array: Further Considerations

- ▶ When filling the cell $a_{q,j}$ of a m -by- n array the row constraint will need to be modified by subtracting $\triangleright^{q;j+1,n}$ from μ_q .
- ▶ When filling the q th row of a m -by- n array, the column constraints will need to be modified by subtracting $\nabla^{j;q+1,m}$ from each ν_j to account for the cells already filled below $a_{q,j}$.
- ▶ The number of options for filling cell $a_{q,j}$ with some value $k_{q,j}$ corresponds to the summation

$$\min(\nu_j - \nabla^{j;q+1,m}, \mu_q - \triangleright^{q;j+1,n}) \sum_{k_{q,j}=0}^{\cdot} .$$

The Number of Fillings for a m -by- n Array: Further Considerations

- ▶ When filling the cell $a_{q,j}$ of a m -by- n array the row constraint will need to be modified by subtracting $\triangleright^{q;j+1,n}$ from μ_q .
- ▶ When filling the q th row of a m -by- n array, the column constraints will need to be modified by subtracting $\nabla^{j;q+1,m}$ from each ν_j to account for the cells already filled below $a_{q,j}$.
- ▶ The number of options for filling cell $a_{q,j}$ with some value $k_{q,j}$ corresponds to the summation

$$\min(\nu_j - \nabla^{j;q+1,m}, \mu_q - \triangleright^{q;j+1,n}) \sum_{k_{q,j}=0}^{\cdot} .$$

- ▶ After the array has been reduced to a 2-by-2 case, we will need to consider the quantity within the composition of summations. This quantity will be

$$\min(\nu_2 - \nabla^{2;3,m}, \mu_2 - \triangleright^{2;3,n}, \mu_1 - \sum_{j=3}^n (\nu_j - \nabla^{j;2,m})) + 1.$$

Formula for the Number of Fillings for a m -by- n Array

The number of fillings of an m -by- n array, where $m \geq 3$ and $n \geq 3$, is

$$\prod_{q:m;3}^{j:n;2} \prod_{q:2;2}^{j:n;3} \min(\nu_j - \nabla^{j;q+1,m}, \mu_q - \triangleright^{q;j+1,n}) \sum_{k_{q,j}=0} \min(\nu_2 - \nabla^{2;3,m}, \mu_2 - \triangleright^{2;3,n}, \mu_1 - \sum_{j=3}^n (\nu_j - \nabla^{j;2,m})) + 1.$$

This formula can be proved by mathematical induction on the number of rows.

Example of Formula's Application

We consider the following 6-by-5 array.

					3
					3
					3
					3
					2
					1
5	4	3	2	1	

Example of Formula's Application

We consider the following 6-by-5 array.

					3
					3
					3
					3
					2
					1
5	4	3	2	1	

Using our formula, there are

121,050

possible fillings.