CIS 330
C++ and Unix

Lecture 4

Memory and Pointers II

iOS Dev Club

iOS

Development Club

Register

Learn to Build Apps

Are you interested in learning app development? Bring your idea to life or see if iOS development is for you!

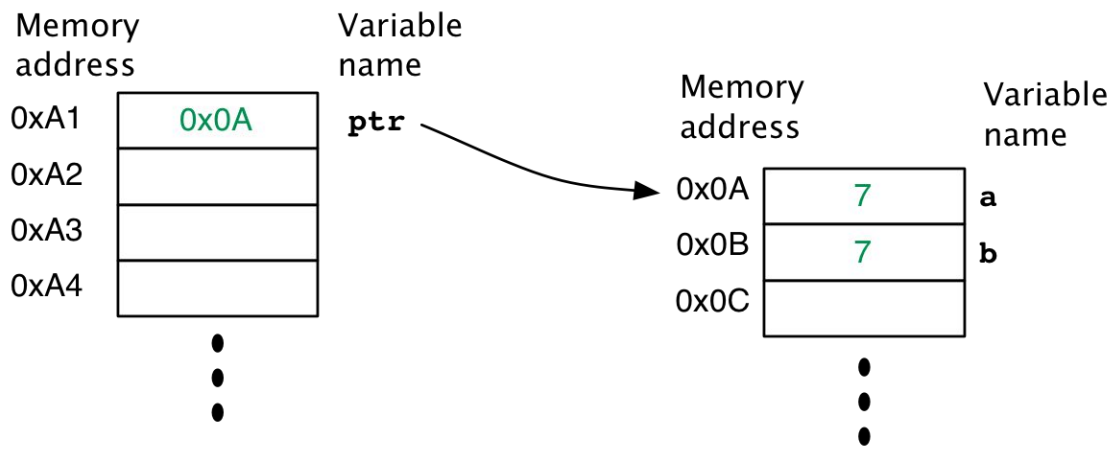
Tuesdays Deschutes 100 @ 5:00 pm

Install Xcode from the MacOS App Store for free before attending.
We encourage people of all skill levels to join.

Mentorship	Duck alumni from industry-leading companies provide support and help you navigate through iOS development
Career Advice	Special guests give advice and share stories. You have questions, they have answers.
Community	Committed members will push you to succeed, stay consistent, and reinforce good habits.

@iosdevclub https://bit.ly/3yaym7r appledevelopmentclub@gmail.com

Last Lecture

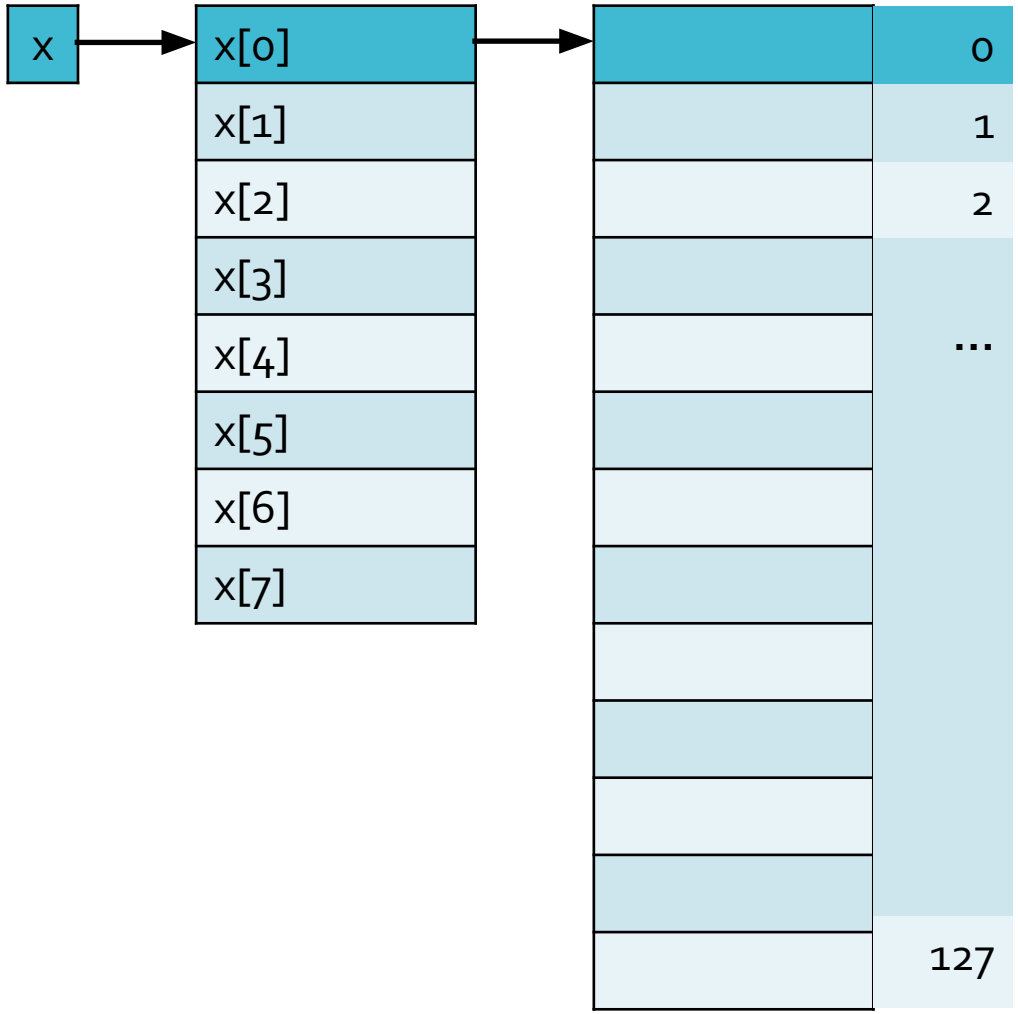



```
int main() {
  int *ptr; // pointer to integer value
  int a, b; // integer variables

  a = 7;    // assign value to a
  ptr = &a; // assign address of a (0x0A in this example) to ptr
  b = *ptr; // assign value at the address contained in ptr to b

  // Result: a and be contain the same value (7) and
  //         ptr contains the address of a.
  return 0;
}
```

Pointers to pointer (to pointer...)




```
x[0] = malloc(sizeof(int) * 128);
```

Memory address

- Represented by hexadecimal numbers

```
int* A = 0x8000;
```

Pointer arithmetic - arithmetic operations on pointers at done at data size granularity

```
A++; /* A+1 == 0x8004 since int is 4 Bytes */
```

```
double* A = 0x6000;
A++; /* A+1 == 0x6008 since double is 8 Bytes */
```

```
double* A = 0x6000;
A+=2; /* A+2 == ? */
```

Pointer arithmetic

- Dereferencing a pointer
- int X = 10;
- int* A = &X;
- printf("%d\n", *A); /* this prints 10 */

- Another way to dereference memory – []
- A[0] <-> *A
- A[5] <-> *(A + 5)
- B = &(A[0]) <-> B = A
- B = &(A[5]) <-> B = A + 5

Allocating a Pointer Array

- 3D Array

```
d3_array = malloc(sizeof(int**) * ARR_SIZE);
for(int i = 0; i < ARR_SIZE; i++) {
    d3_array[i] = malloc(sizeof(int*) * ARR_SIZE);
    for(int j = 0; j < ARR_SIZE; j++) {
        d3_array[i][j] = malloc(sizeof(int) * ARR_SIZE);
    }
}
```

Freeing a Pointer Array

```
for(int i = 0; i < ARR_SIZE; i++) {
    for(int j = 0; j < ARR_SIZE; j++) {
        free(d3_array[i][j]);
    }
    free(d3_array[i]);
}
free(d3_array);
```

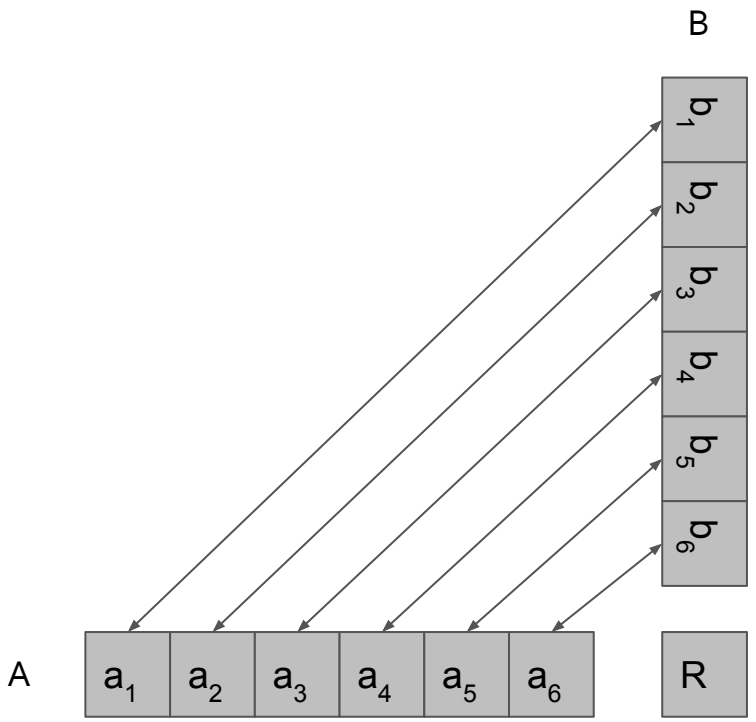
Questions?

Homework 3

Homework 3

- Dot-product
- Matrix multiplication
  - sparse matrix-vector multiply
- Histogram
- Prefix-sum

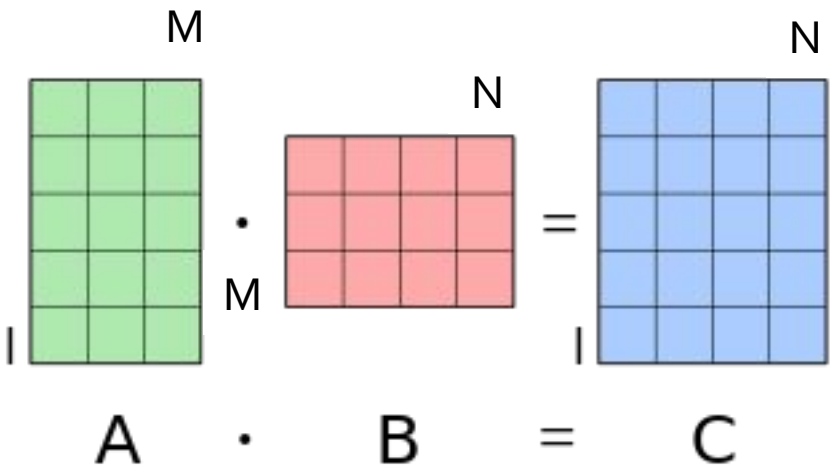
Dot Product



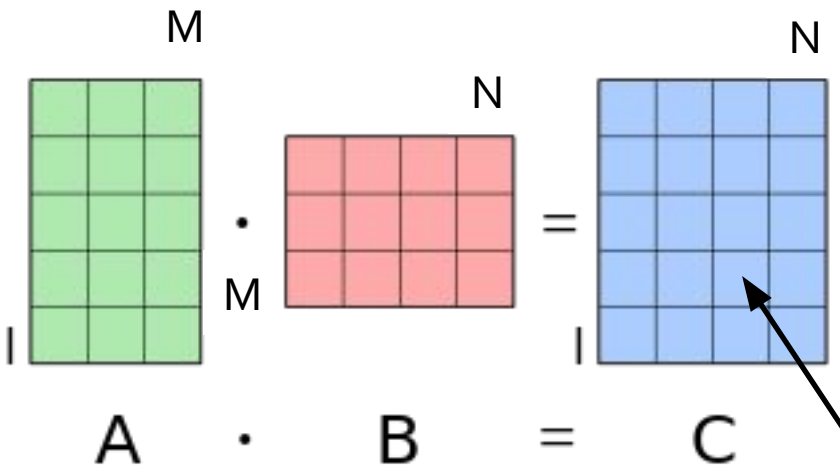
$$R = a_1 * b_1 + a_2 * b_2 + \ldots + a_6 * b_6$$

Multiply elements from corresponding positions and then sum them all up

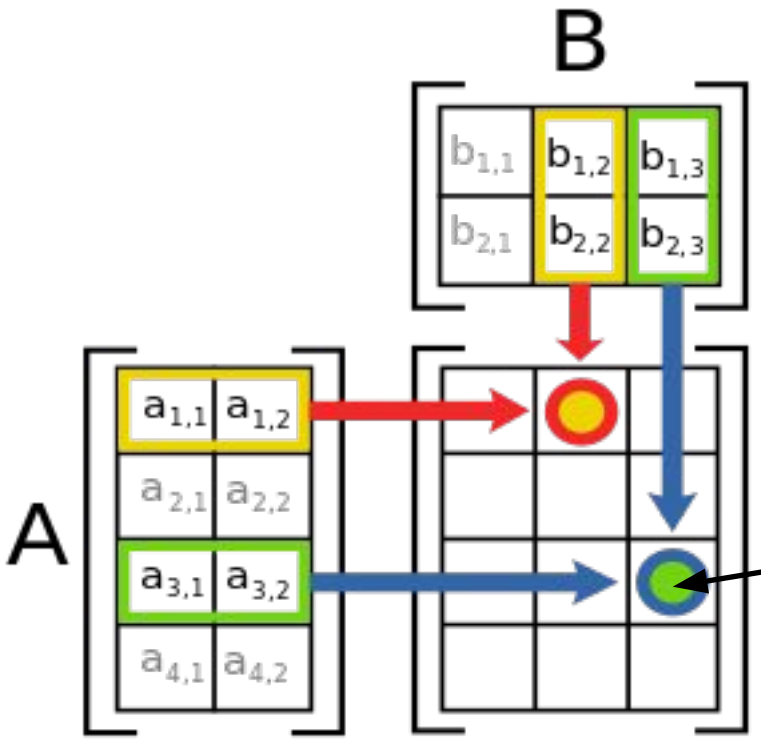
Matrix Multiplication



Matrix Multiplication



$$c_{i,n} = \sum_{m=1}^{M} a_{i,m} * b_{m,n}$$



$$c_{3,3} = a_{3,1} * b_{1,3} + a_{3,2} * b_{2,3}$$

Matrix Multiplication Algorithm

Let's say you want to multiply A (3x4 matrix) with B (4x2 matrix)

What should you get?

Matrix Multiplication Algorithm

Let's say you want to multiply A (3x4 matrix) with B (4x2 matrix)

What should you get?

C (3x2 matrix)

How do we calculate it?

Matrix Multiplication Algorithm

Let's say you want to multiply A (3x4 matrix) with B (4x2 matrix)

What should you get?

C (3x2 matrix)

How do we calculate it?

3x2 = 6 numbers need to be calculated

```
i = 3;
j = 2;
k = 4;
```

Matrix Multiplication Algorithm

Let's say you want to multiply A (3x4 matrix) with B (4x2 matrix)

What should you get?

C (3x2 matrix)

How do we calculate it?

3x2 = 6 numbers need to be calculated

```
i = 3;
j = 2;
k = 4;
for(int x = 0; x < i; x++) {
  for(int y = 0; y < j; y++) {



  }
}
```

Matrix Multiplication Algorithm

Let's say you want to multiply A (3x4 matrix) with B (4x2 matrix)

What should you get?

C (3x2 matrix)

How do we calculate it?

3x2 = 6 numbers need to be calculated

```
i = 3;
j = 2;
k = 4;
for(int x = 0; x < i; x++) {
  for(int y = 0; y < j; y++) {
    for(int z = 0; z < k; z++) {

    }
  }
}
```

Matrix Multiplication Algorithm

Let's say you want to multiply A (3x4 matrix) with B (4x2 matrix)

What should you get?

C (3x2 matrix)

How do we calculate it?

3x2 = 6 numbers need to be calculated

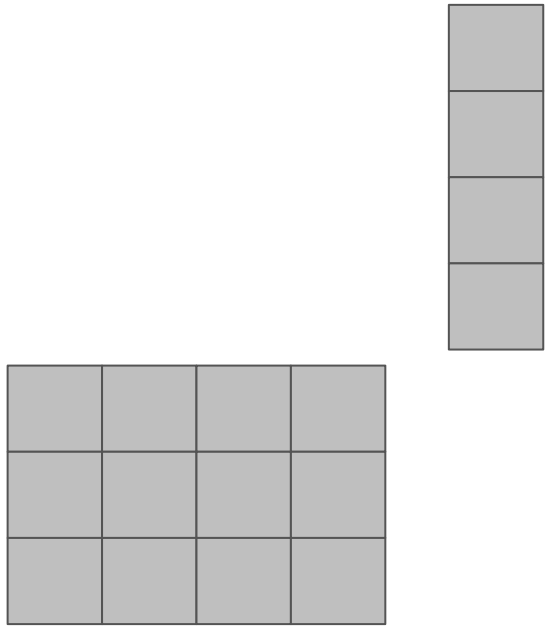
```
i = 3;
j = 2;
k = 4;
for(int x = 0; x < i; x++) {
  for(int y = 0; y < j; y++) {
    for(int z = 0; z < k; z++) {
      C[x][y] += A[x][z] * B[z][y];
    }
  }
}
```

Matrix-vector Multiply

What if the second matrix was a vector instead?

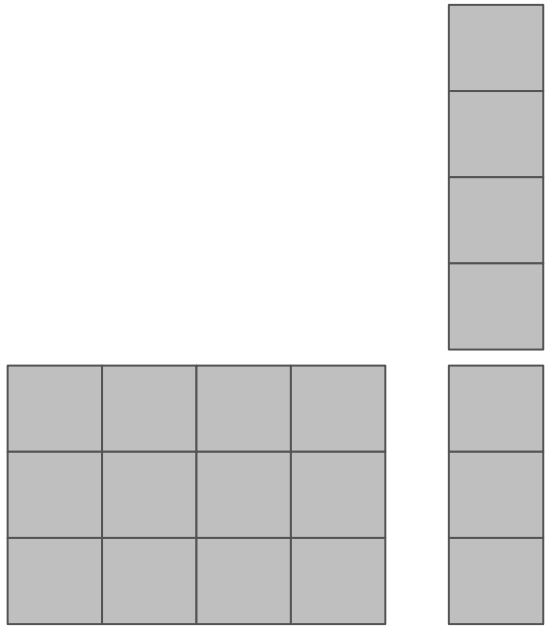
Matrix-vector Multiply

What if the second matrix was a vector instead?



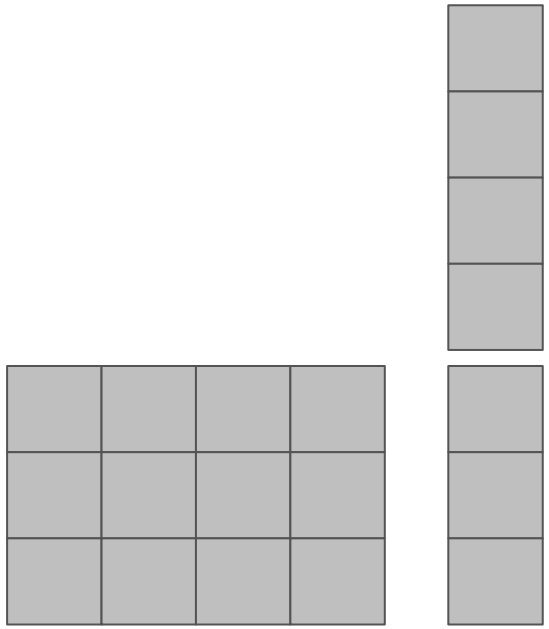
Matrix-vector Multiply

What if the second matrix was a vector instead?



Matrix-vector Multiply

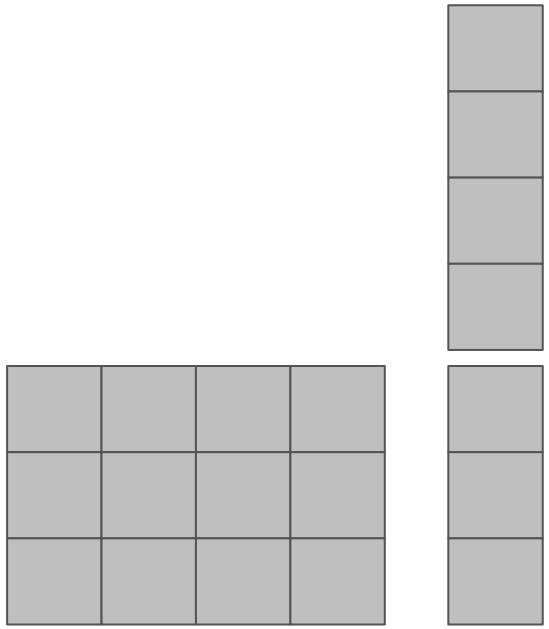
What if the second matrix was a vector instead?



```
i = 3;
j = 2;
k = 4;
for(int x = 0; x < i; x++) {
  for(int y = 0; y < j; y++) {
    for(int z = 0; z < k; z++) {
      C[x][y] += A[x][z] * B[z][y];
    }
  }
}
```

Matrix-vector Multiply

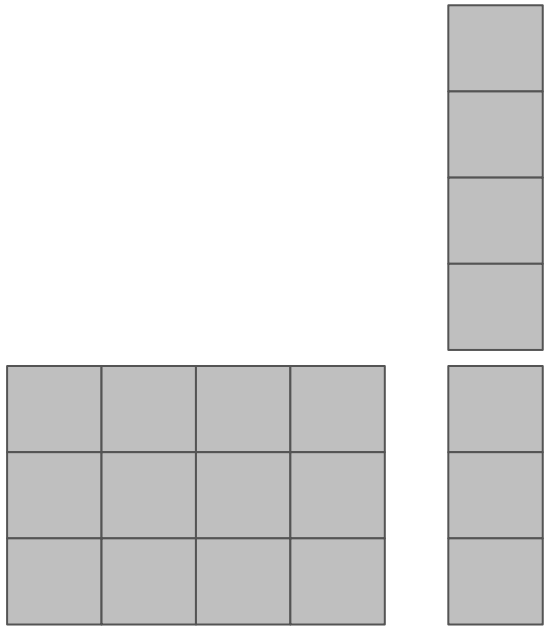
What if the second matrix was a vector instead?



```
i = 3;
j = 1;
k = 4;
for(int x = 0; x < i; x++) {
  for(int y = 0; y < 1; y++) {
    for(int z = 0; z < k; z++) {
      C[x][0] += A[x][z] * B[z][0];
    }
  }
}
```

Matrix-vector Multiply

What if the second matrix was a vector instead?



```
i = 3;
j = 1;
k = 4;
for(int x = 0; x < i; x++) {

    for(int z = 0; z < k; z++) {
      C[x] += A[x][z] * B[z];
    }

}
```

What if the matrix is "sparse?"

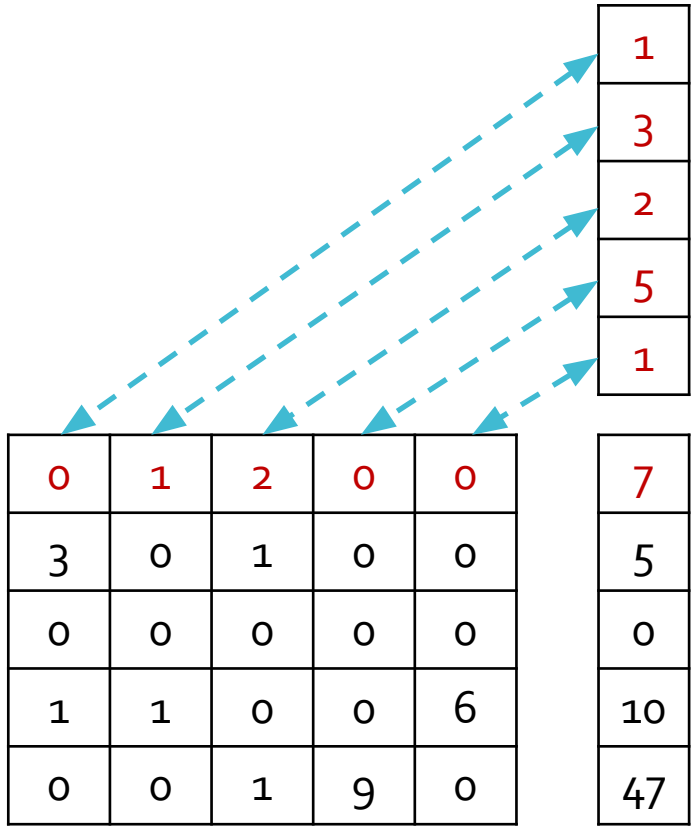
- Sparse matrix – matrix with only a small number of values that are not zeros
- SpMV – Sparse Matrix Vector Multiply
  - Solve a systems of equations
  - Graph algorithms

Example Sparse Matrices

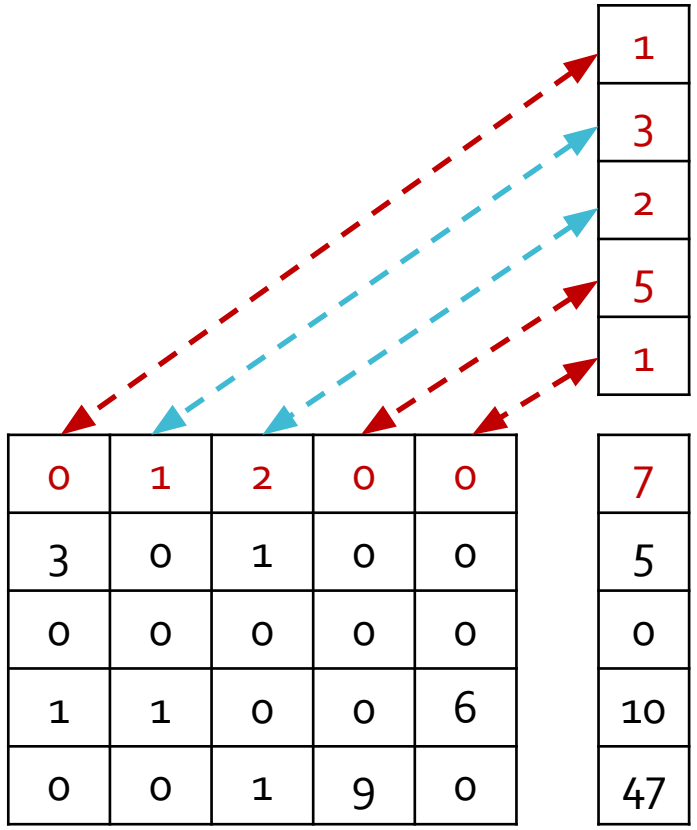
Name	Dimensions	Non-zeros	Description
Dense	2K×2K	4.0M	Dense
Protein	36K×36K	4.3M	Protein data bank 1HY2S
QCD	49K×49K	1.9M	Quark propagation
Cantilever	62K×62K	4.0M	Cantilever
Spheres	83K×83K	6.0M	Concentric spheres
Harbor	47K×47K	2.37M	3D CFD of Charleston Harbor
Ship	141K×141K	3.98M	Ship section
Wind Tunnel	218K×218K	11.6M	Pressurized wind tunnel
Cop	121K×121k	2.6M	Accelerator
Epidemiology	526K×526K	2.1M	2D Markov model of epidemic
Economics	207K×207K	1.27M	Macro-economics model
Circuit	171K×171K	959K	Motorola circuit simulation
Webbase	1M×1M	3.1M	Web connectivity
LP	4K×1M	11.3M	Linear programming

sparsity = 0.33%

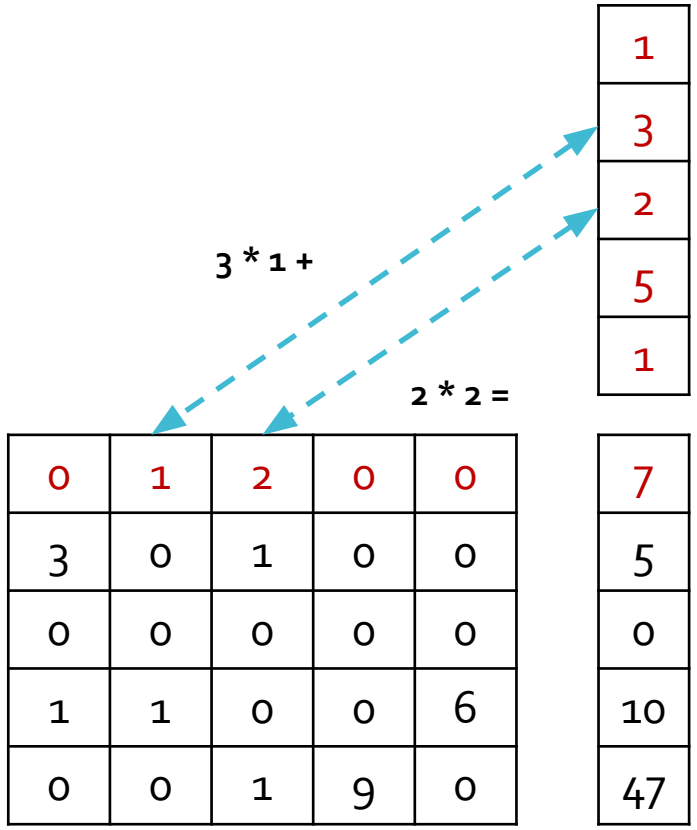
SpMV



SpMV



SpMV



SpMV in COO

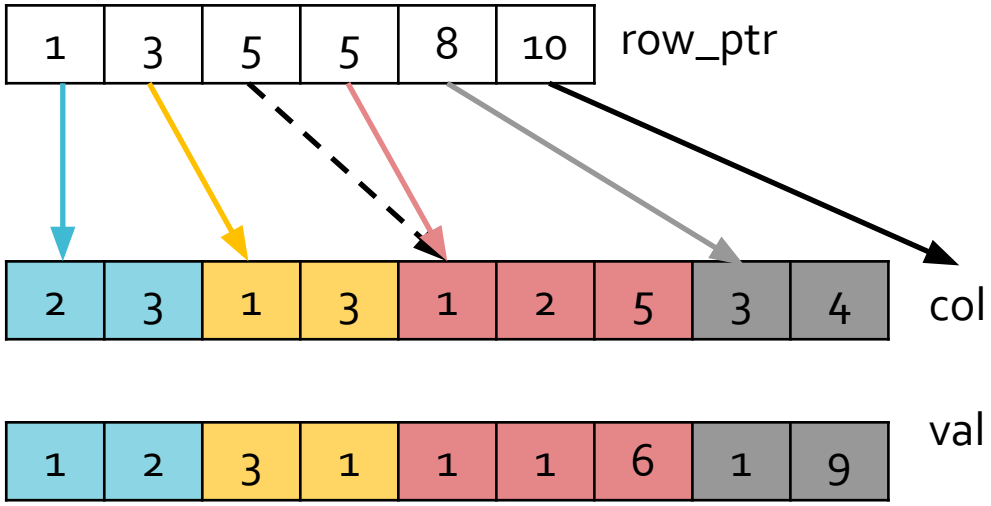
0	**1**	2	0	0
3	0	1	0	0
0	0	0	0	0
1	1	0	0	6
0	0	1	9	0

row	col	val
1	**2**	**1**
1	3	2
2	1	3
2	3	1
4	1	1
4	2	1
4	5	6
5	3	1
5	4	9

Compressed Sparse Row (CSR)

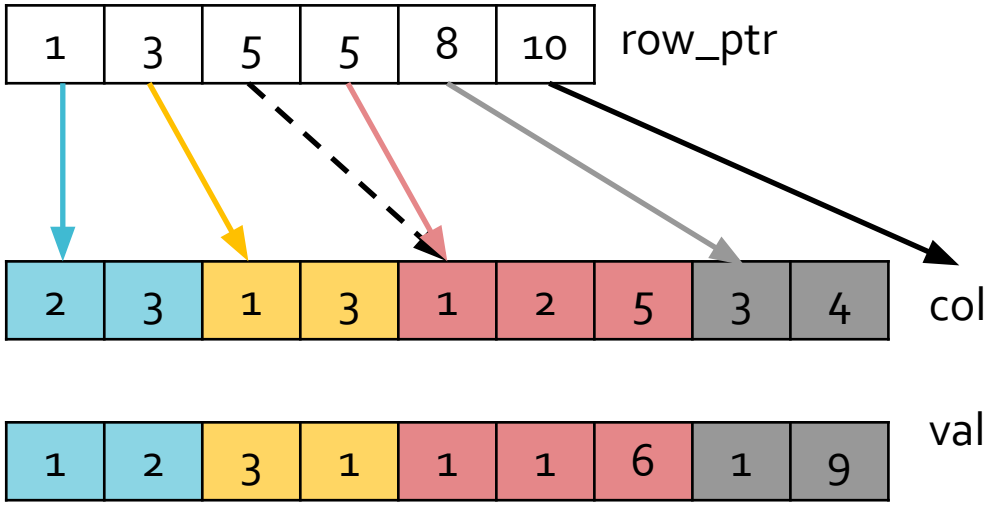
row	col	val
1	2	1
1	3	2
2	1	3
2	3	1
4	1	1
4	2	1
4	5	6
5	3	1
5	4	9

Here, row pointer is **not** an array of C pointers - it's a pointer in the general sense that it points to something (i.e., points to a particular location in another array)



Compressed Sparse Row (CSR)

row	col	val
1	2	1
1	3	2
2	1	3
2	3	1
4	1	1
4	2	1
4	5	6
5	3	1
5	4	9



After creating the row_ptr array, copy the col and val from COO (left table) to CSR (right arrays)
It must work even if the elements are **NOT** sorted by rows

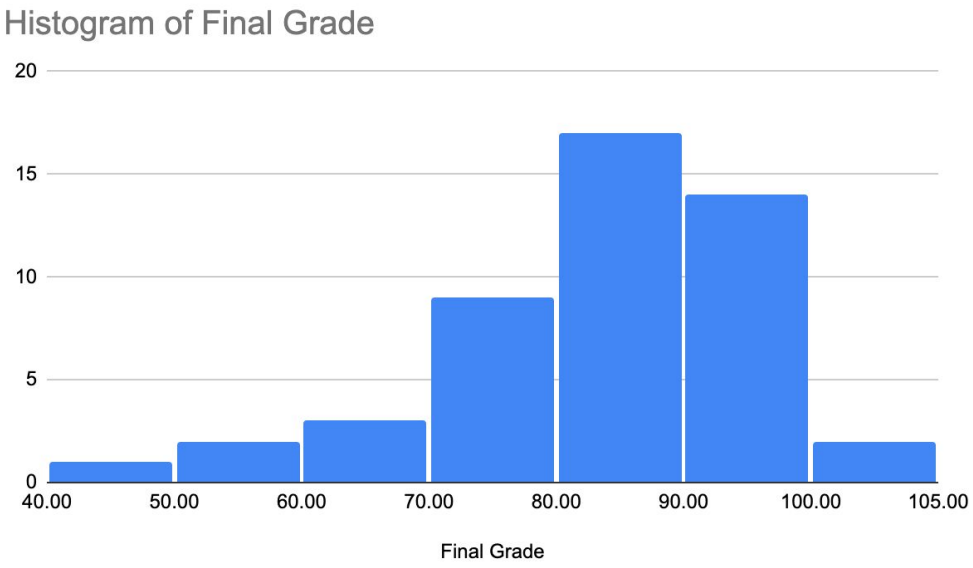
Algorithm

- Input: $A \in \mathbb{R}^{I \times J}$, $x \in \mathbb{R}^{J}$
- Output: $Ax = y \in \mathbb{R}^{I}$
- for i = 1 to I /* number of rows */
- get *begin* and *end* index into col and val for row I
- for j = *begin* to *end*
- y[i] += val[j] * x[col[j]]
- end
- end

Converting COO to CSR

Two algorithms are required:

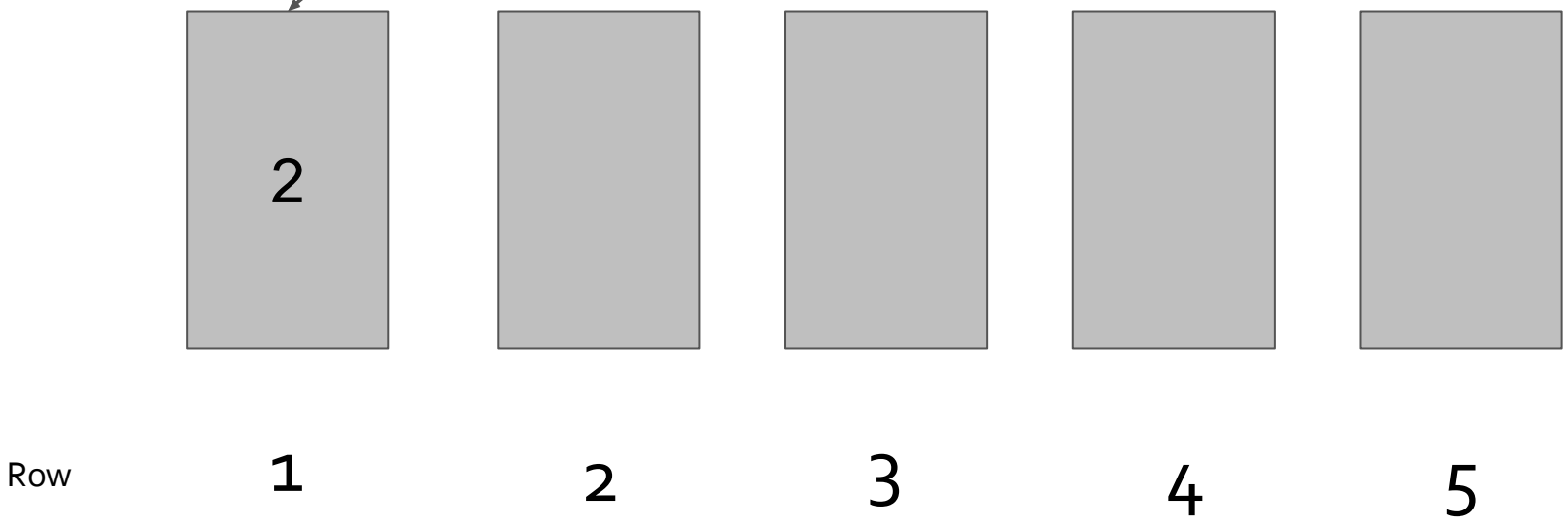
- Histogram
- prefix-sum



Histogram

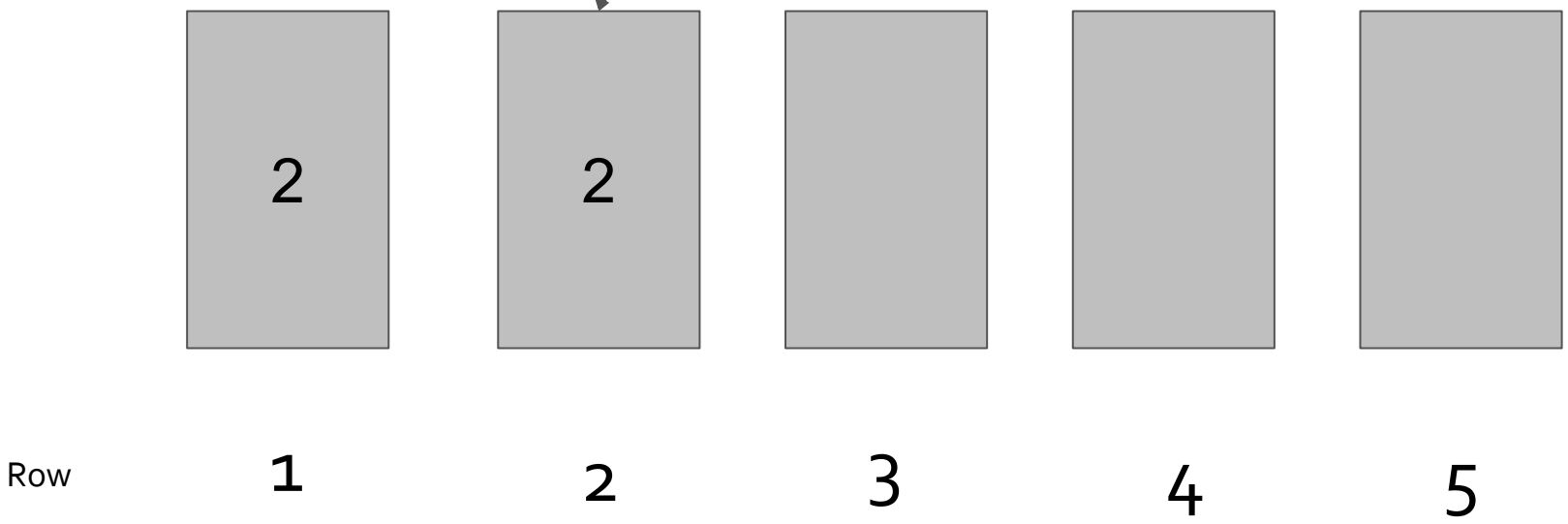
Histogram of how many non-zero elements are in each row

row					
1	0	1	2	0	0
2	3	0	1	0	0
3	0	0	0	0	0
4	1	1	0	0	6
5	0	0	1	9	0

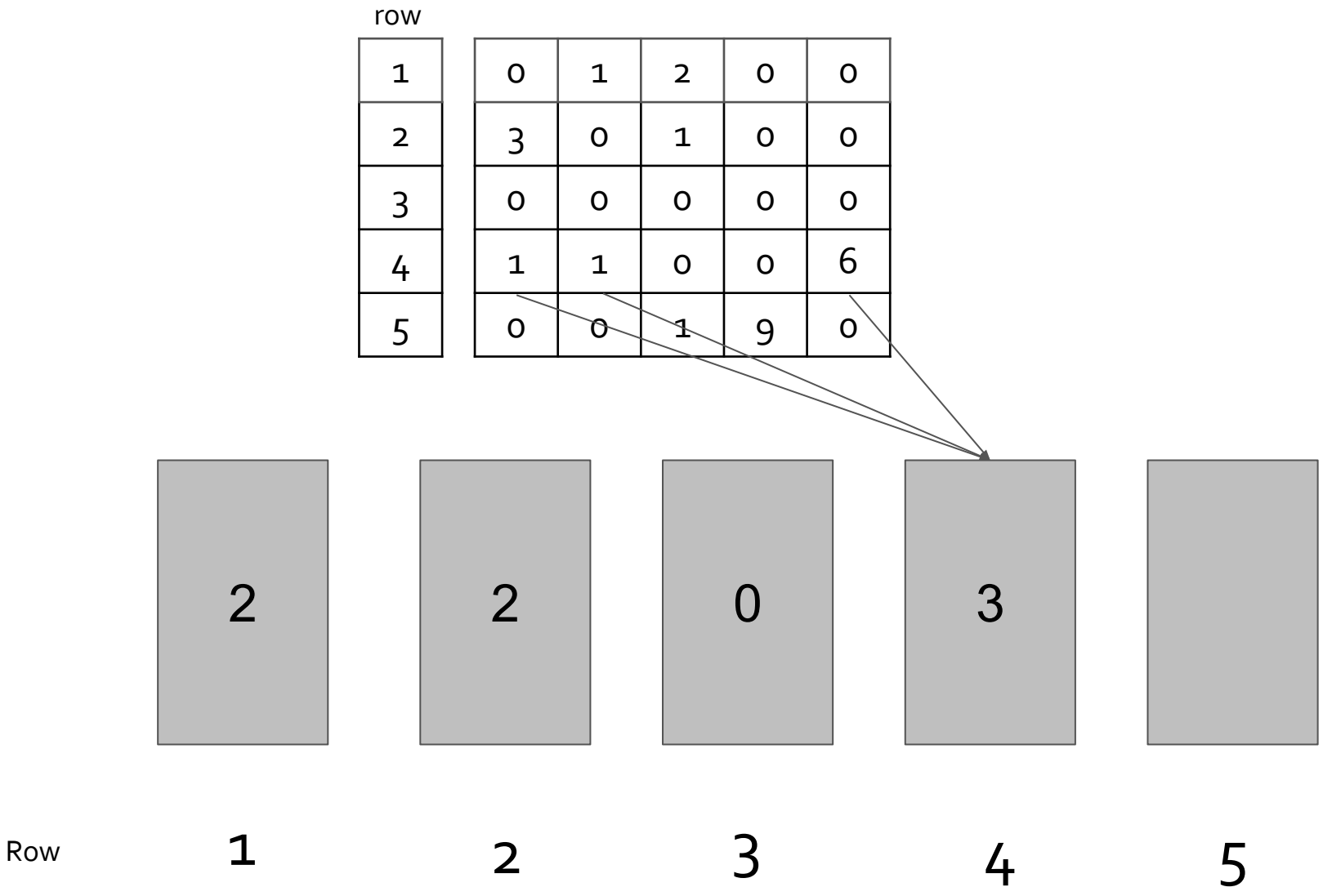


Histogram

row					
1	0	1	2	0	0
2	3	0	1	0	0
3	0	0	0	0	0
4	1	1	0	0	6
5	0	0	1	9	0

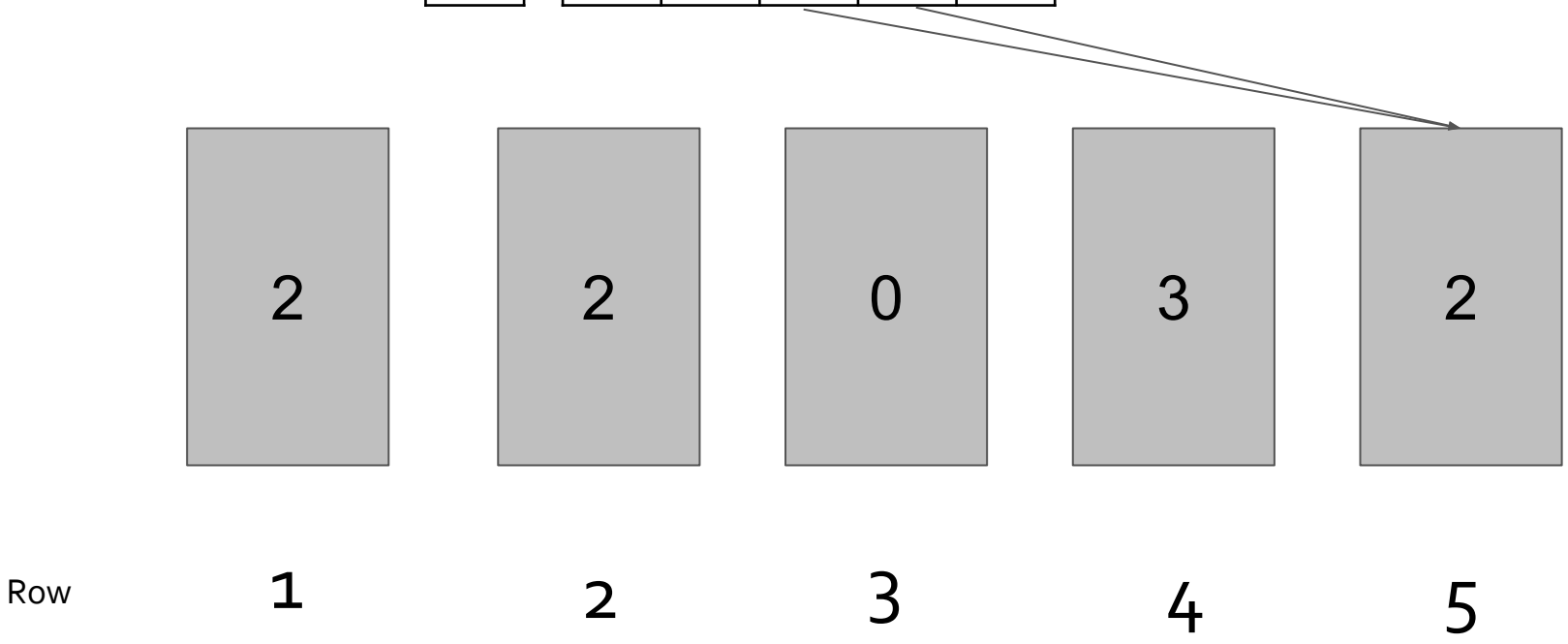


Histogram



Histogram

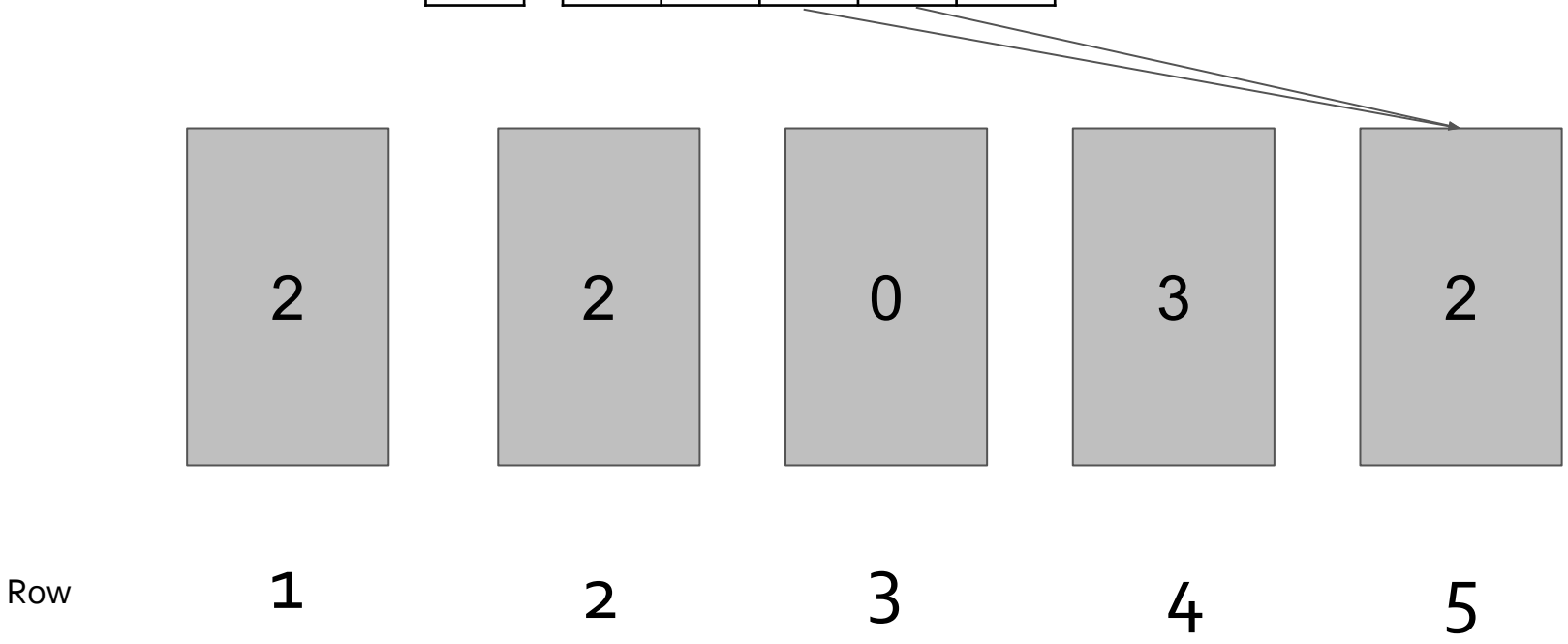
row					
1	0	1	2	0	0
2	3	0	1	0	0
3	0	0	0	0	0
4	1	1	0	0	6
5	0	0	1	9	0



Complexity O(?)

Histogram

row					
1	0	1	2	0	0
2	3	0	1	0	0
3	0	0	0	0	0
4	1	1	0	0	6
5	0	0	1	9	0



Complexity O(?) -> O(n)

Prefix Sum

$$y_0 = x_0$$
$$y_1 = x_0 + x_1$$
$$y_2 = x_0 + x_1 + x_2$$
$$\ldots$$

x	2	2	0	3	2
y	2	4	4	7	9

This implementation currently requires O(?)

Prefix Sum

$$y_0 = x_0$$
$$y_1 = x_0 + x_1$$
$$y_2 = x_0 + x_1 + x_2$$
$$\ldots$$

x	2	2	0	3	2
y	2	4	4	7	9

This implementation currently requires O((n-1)*(n)/2) = O((n^2 - n)/2) = ~O(n^2)

Can we do it faster?

Yes O(n)

For example, **y2 = y1 + x2**

Converting COO to CSR

Two algorithms are required:

- Histogram
- prefix-sum

How can you use these two algorithms to create the CSR row pointer array?

SpMV

- The concept is simple when you understand it, but initially difficult to grasp the indexing
- Questions?