Sparse array in J

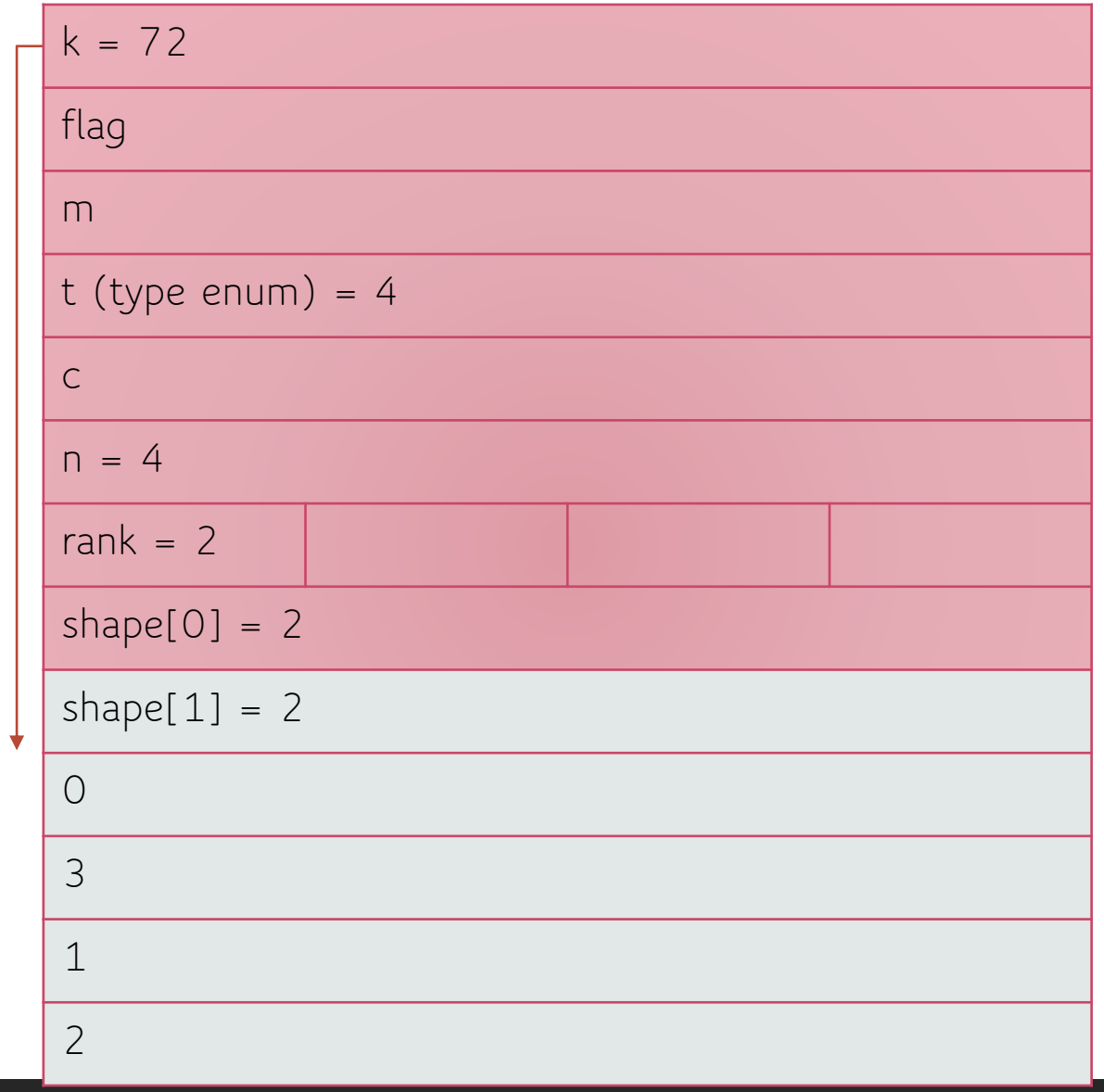
ZHIHAO YUAN

6/2/2020

2d array of integer

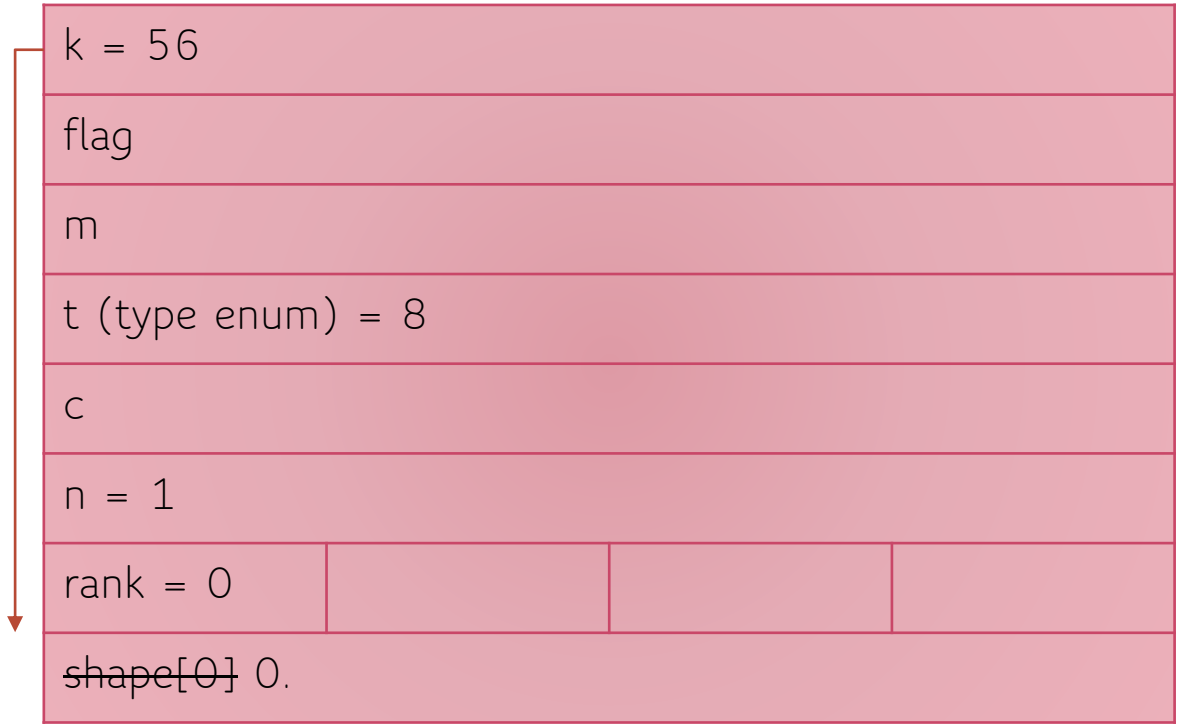
0 3

1 2



atom of integer

0.



COO (Coordinate list)

Conceptually, a list of $(row, column, value)$ triples.

0	0	0	1.5
0	0	2.2	0
0	0	0	0

(0 3 1.5)

(1 2 2.2)

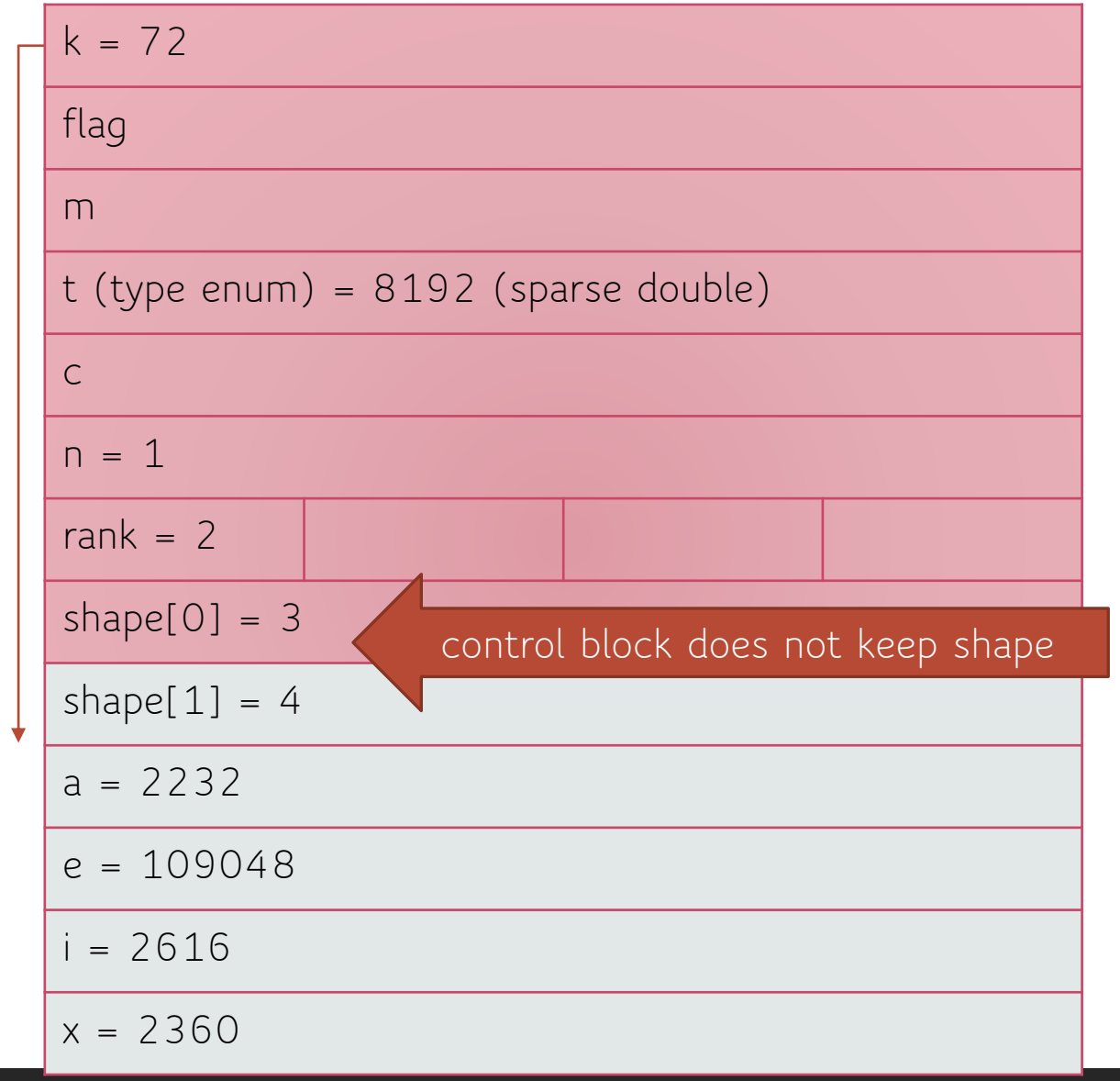
COO in J

1. Gather indices in a list, one tuple of indices per row
2. Gather data (nonzero entries) into a separated array
3. Indices may be 1-N dimensional
4. Nonzero entries may be 0-Any dimensional as well

Sparse array control block

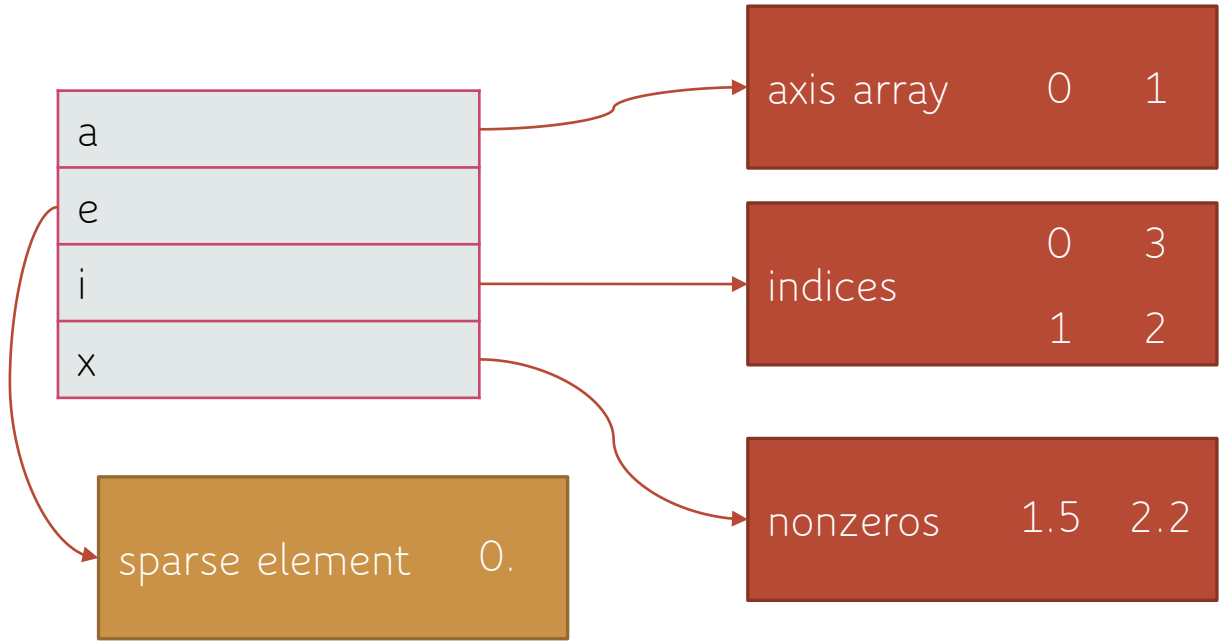
```
typedef struct   // offsets to 4 components in J arrays
{
    int64_t a;
    int64_t e;
    int64_t i;
    int64_t x;
} P;
```

2d sparse array object



Sparse array in memory (2d)

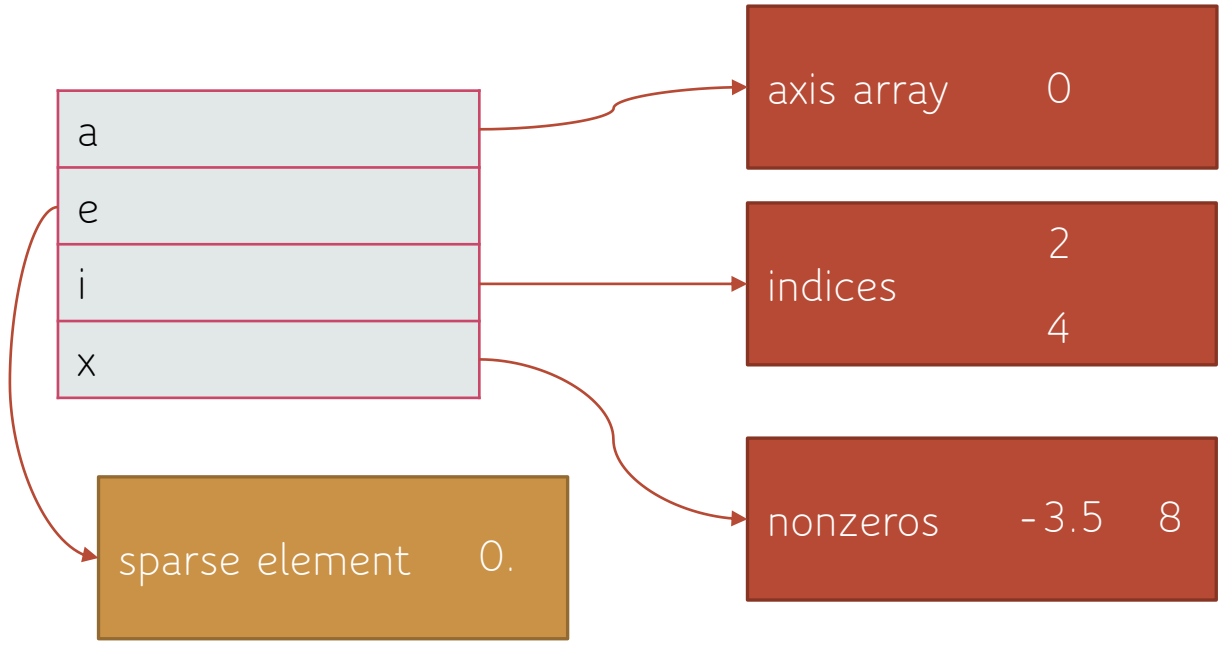
The sparse element may be in static memory if it is a predefined J constant



(0 3	1.5)
(1 2	2.2)

Sparse array in memory (1d)

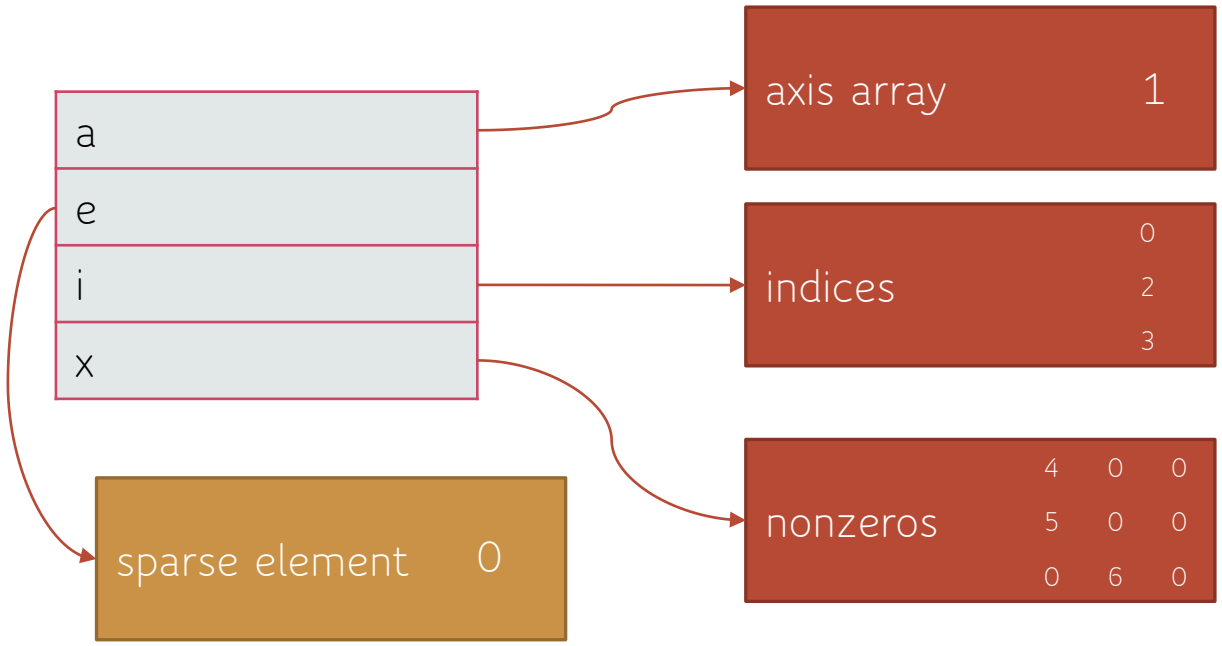
a[N] is indices column N's dimension



(2	_3.5)
(4	8)

Non-atomic sparse element

The sparse element may designate a dimension to be sparse



dimension 0 is dense, only dimension 1 is sparse

(0	[4 0 0])
(2	[5 0 0])
(3	[0 6 0])

Dimension sparsity has no implied major

Given

(0	[4 0 0])
(2	[5 0 0])
(3	[0 6 0])

If axis array is [1], this matrix is

4	0	5	0
0	0	0	6
0	0	0	0

If axis array is [0], this matrix is

4	0	0
0	0	0
5	0	0
0	6	0

Advantages of COO

Easy to construct, grow, and delete

- To grow: append coordinates to the end, sort when finishes
- To turn a single cell to nonzero: `lower_bound` + `insert`
- To delete a row: `erase_if`

COO has the right semantics even if the indices are unordered

Easy to construct CSR or CSC from a COO