GPU Algorithms

Scan, compact and sparse matrixes

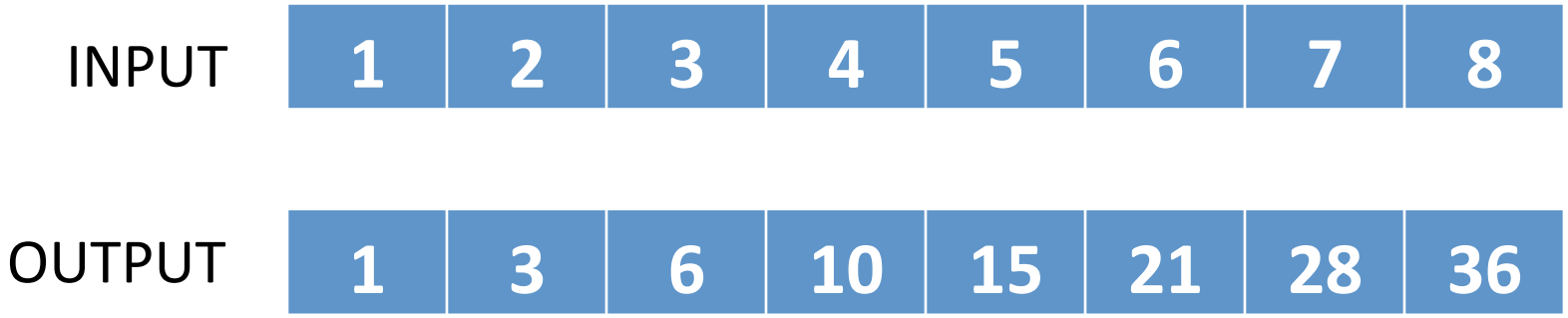
Bryan Mills, PhD

GPU Algorithms

- Reduction
- Scan (covered in more details earlier)
  - Hillis/Steele
  - Blelloch
- Compact
  - Using Sparse Matrixes
- Sorting – Next Class

Scan

- Prefix Sum Example

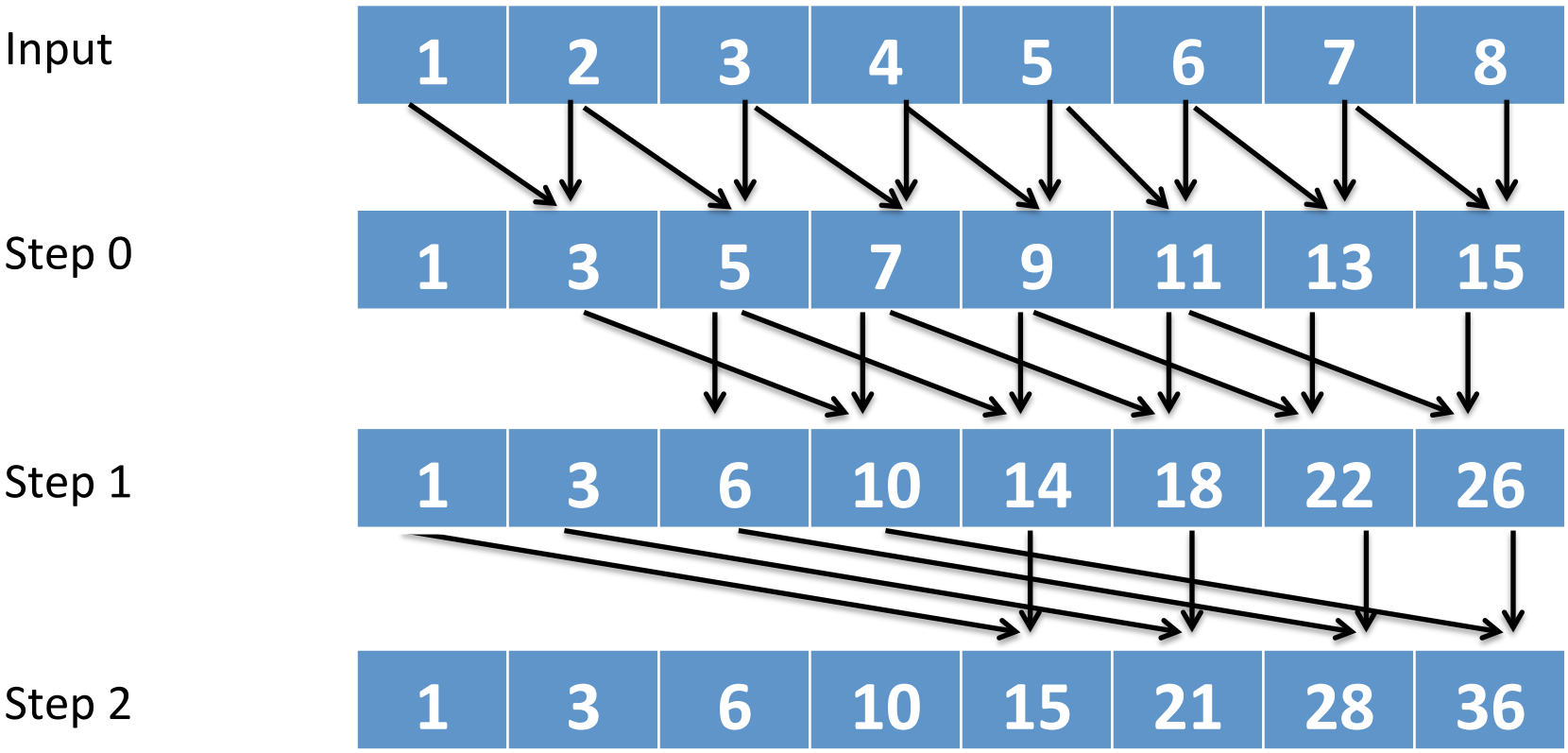


Types of Scan

- Exclusive
  - Output all elements **ex**cluding the current

- Inclusive
  - Output all elements **in**cluding the current

Hillis/Steele Inclusive Scan



You now have the inclusive scan.

Steps = O(log n)

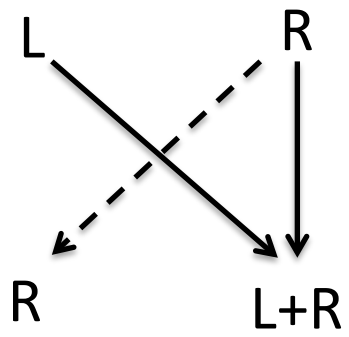
Work = O(n log n) <- dimensions of rectangle above

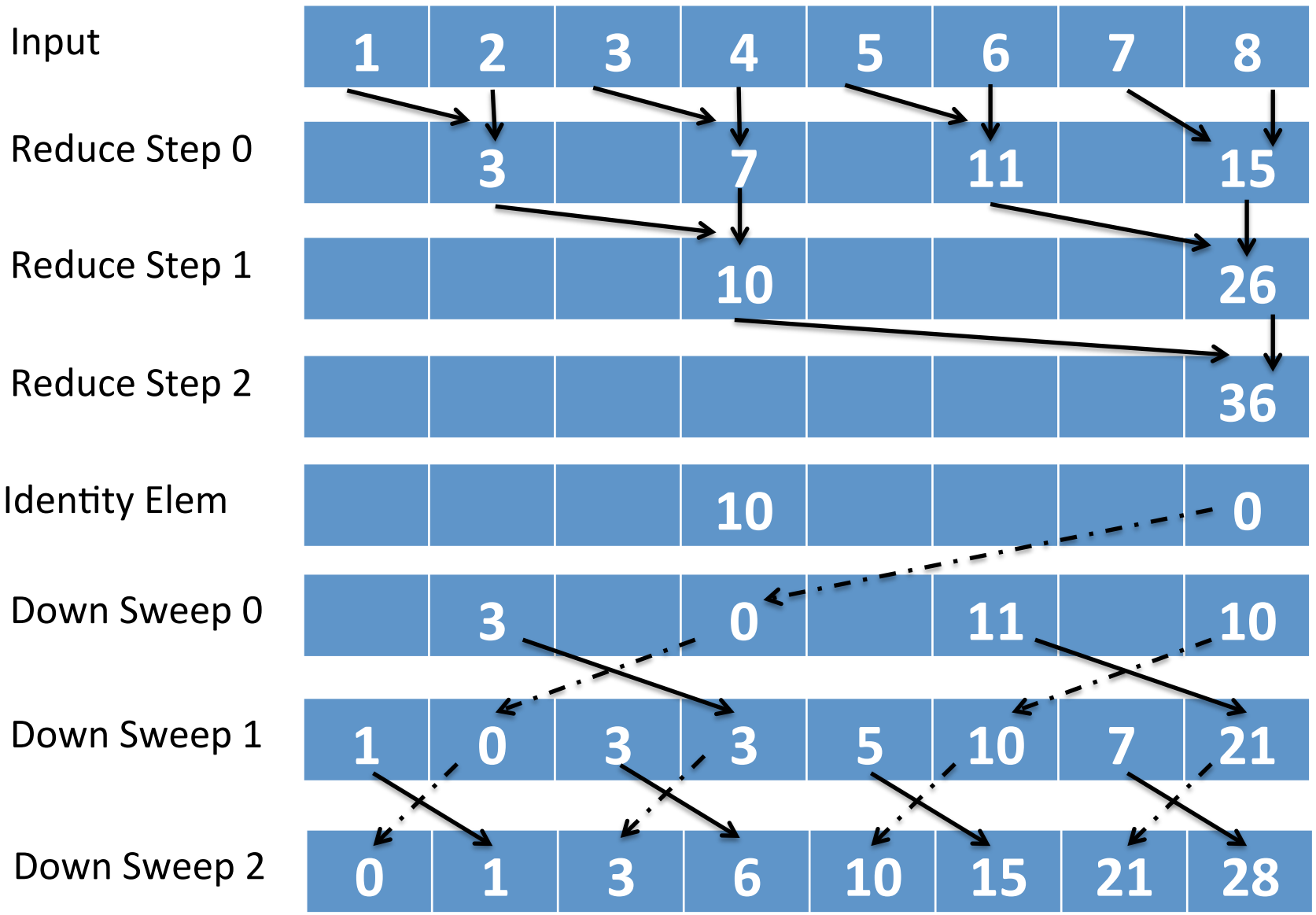
Blelloch Exclusive Scan

- Happens in two passes:
  - Reduce
    - Like previous reduce steps but keep around intermediate results
  - Down sweep
    - New operation

Blelloch Down Sweep Operation

- Reverse reduce step
  - Same inputs (left and right)
  - But two outputs also left and right
    - Add L+R and put on right
    - Copy down R and put it on left



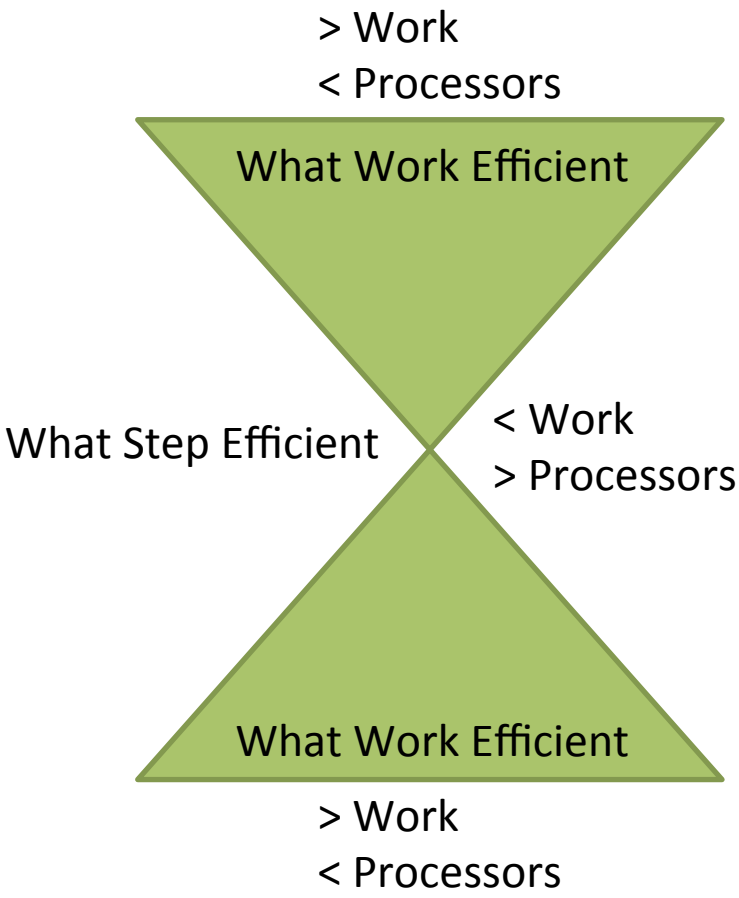

Input
1 2 3 4 5 6 7 8
Reduce Step 0
3 7 11 15
Reduce Step 1
10 26
Reduce Step 2
36
Identity Elem
10 0
Down Sweep 0
3 0 11 10
Down Sweep 1
1 0 3 3 5 10 7 21
Down Sweep 2
0 1 3 6 10 15 21 28

Blelloch Exclusive Scan

- # Steps?
  - 2 log n = O(log n)
- Work?
  - O(n)

Work Efficient or Step Efficient?

- Depends on the amount of work and amount of workers/processors.

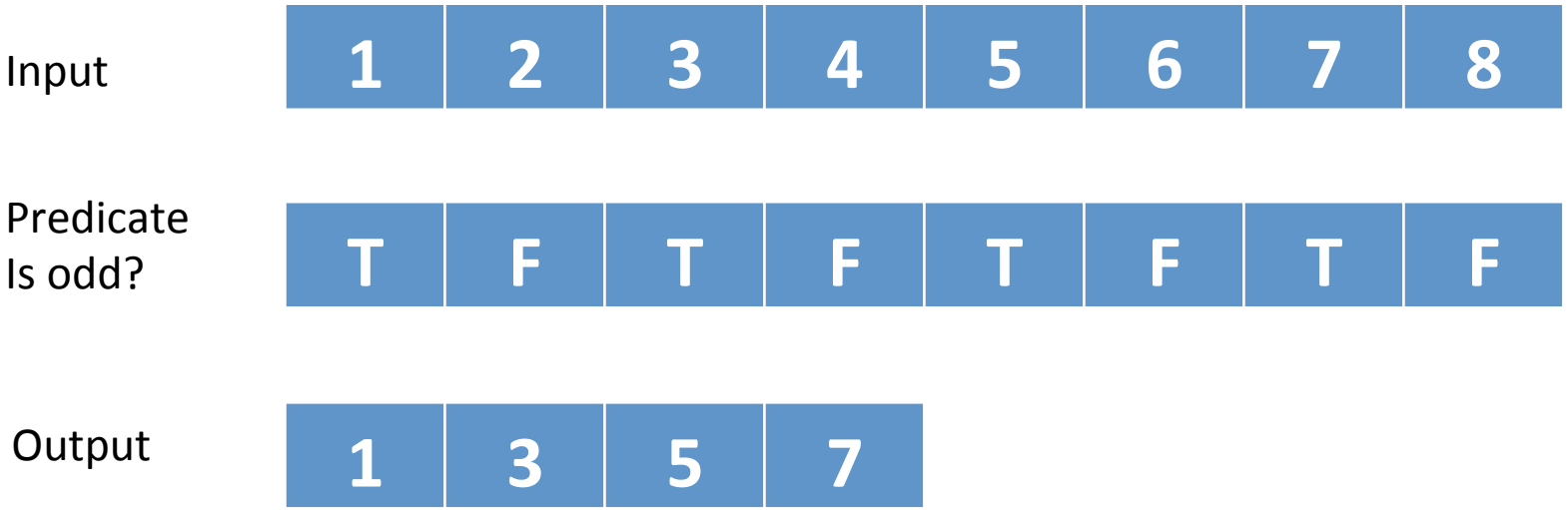


Compact

- Many times you have lots of data and you only want to perform some computation on a subset of that data.
  - Logs analysis: only look at logs containing a certain search term or type of search term
  - Graphics: only perform ray tracing on elements in the viewport
  - Big Data: Calculate histogram of incomes for everyone with a dog

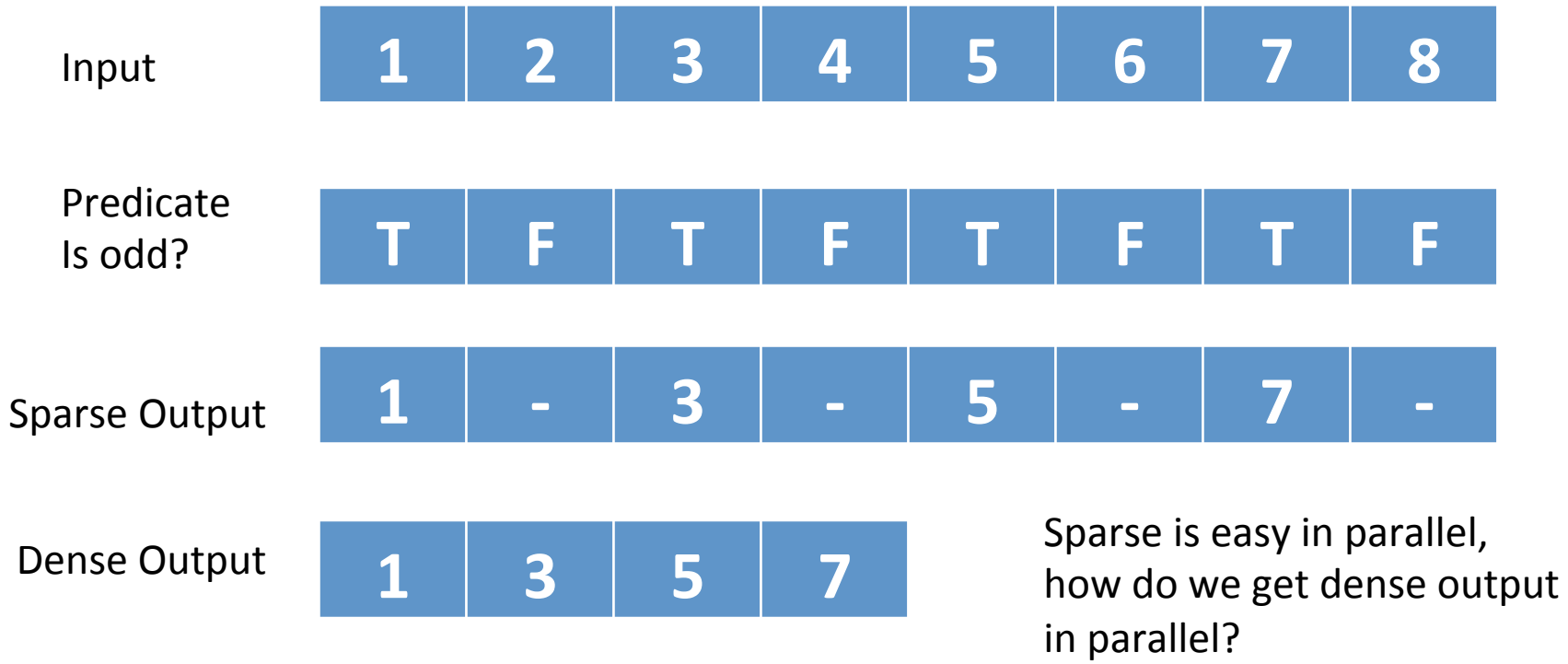
What is compact

- Given some predicate function remove those elements which return false and “squeeze” the data into the required space.



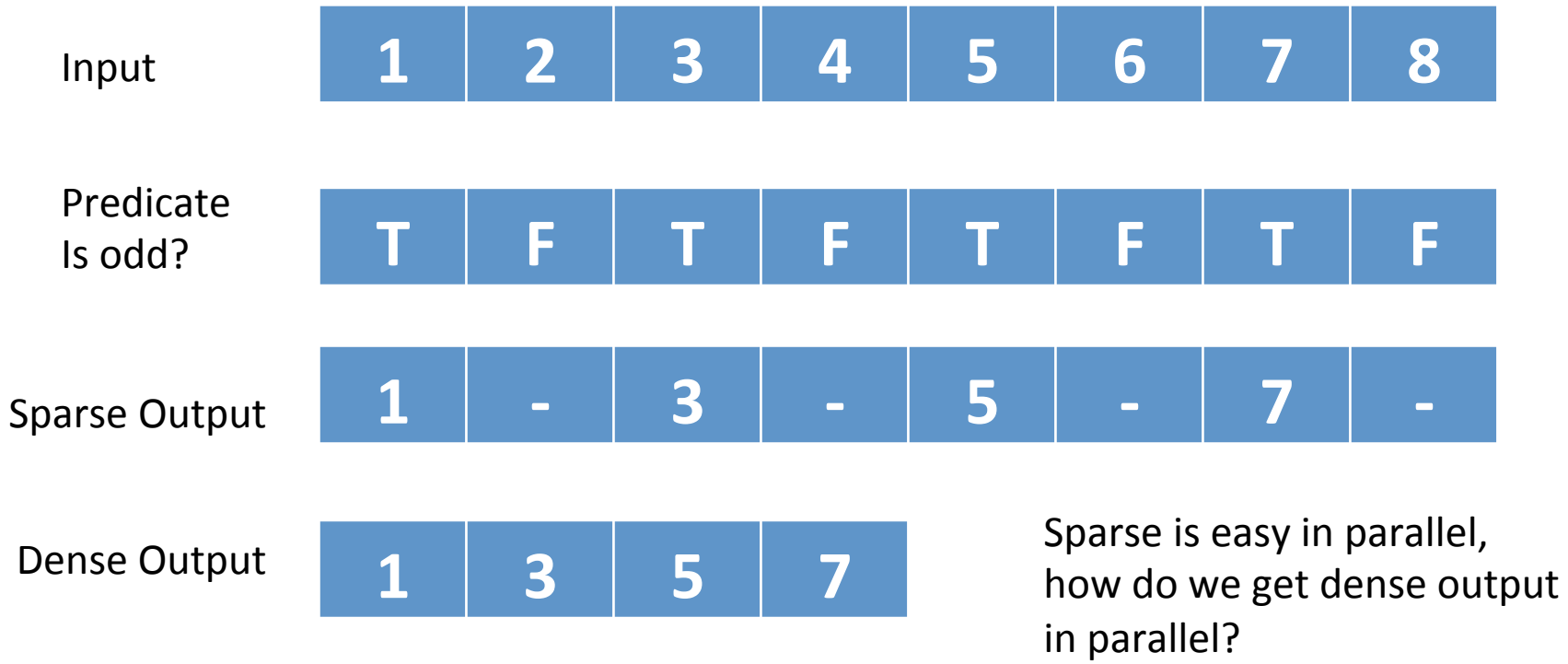
Parallel Compact

- Just have each thread evaluate predicate and copy only on true.

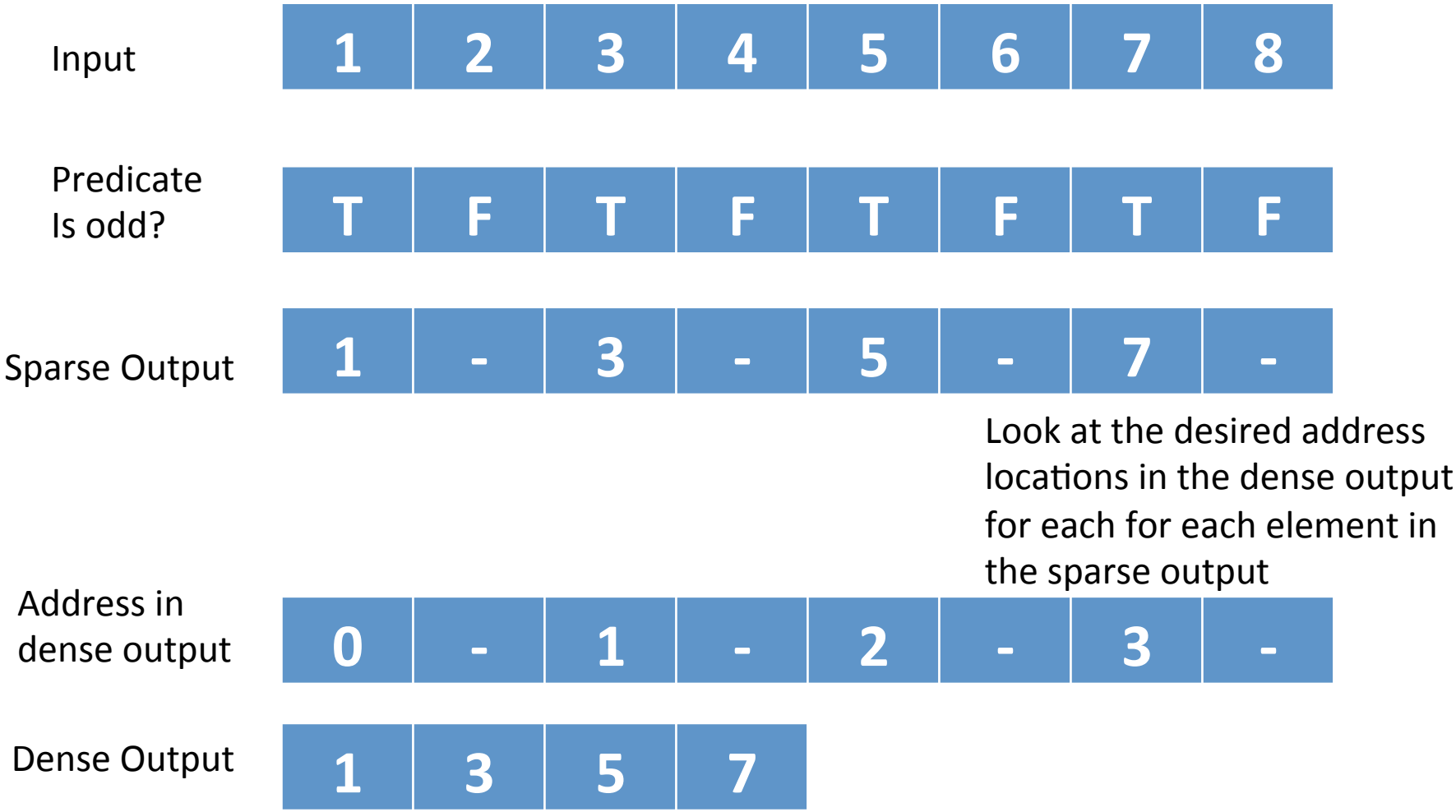


Parallel Compact

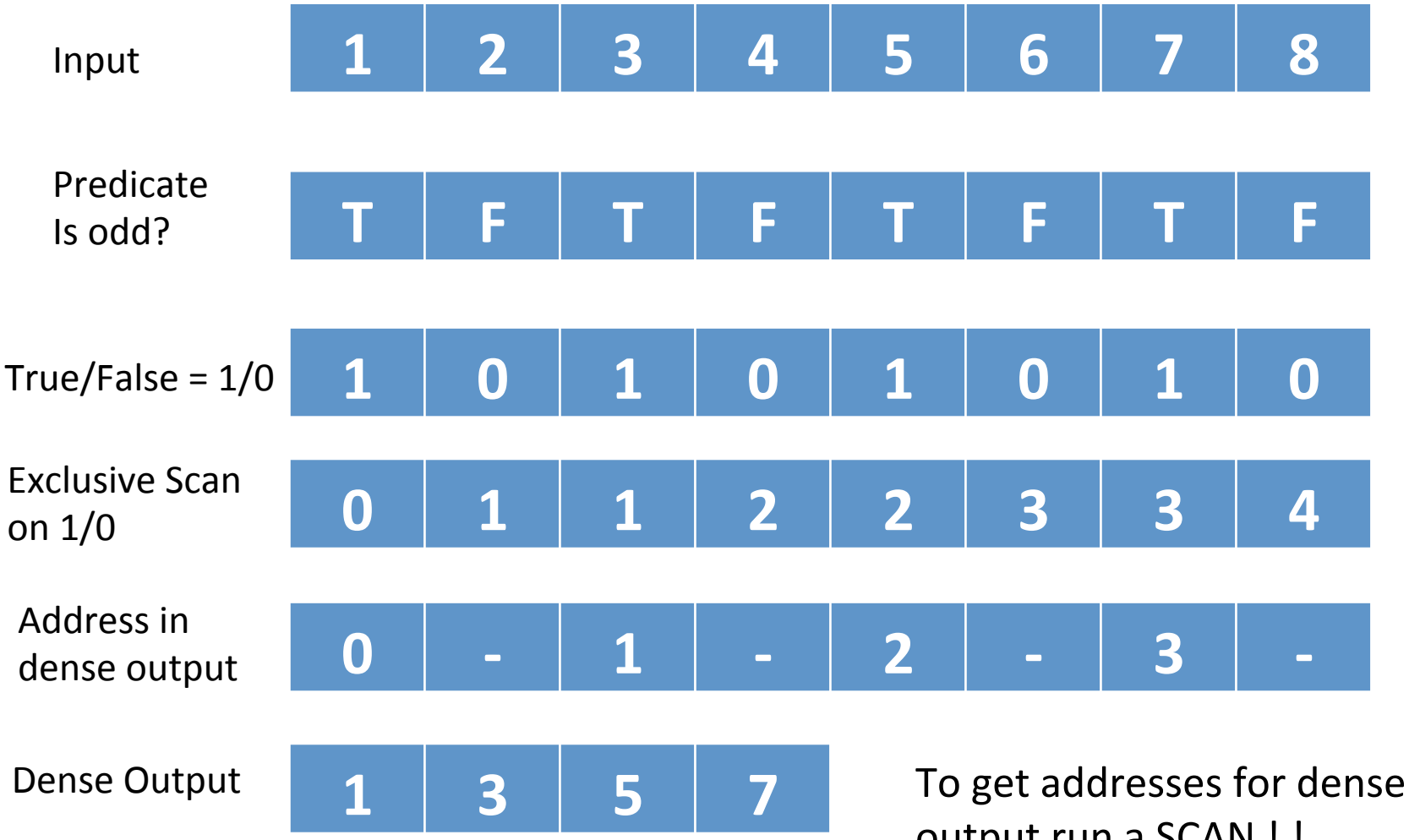
- Just have each thread evaluate predicate and copy only on true.



Parallel Compact



Parallel Compact

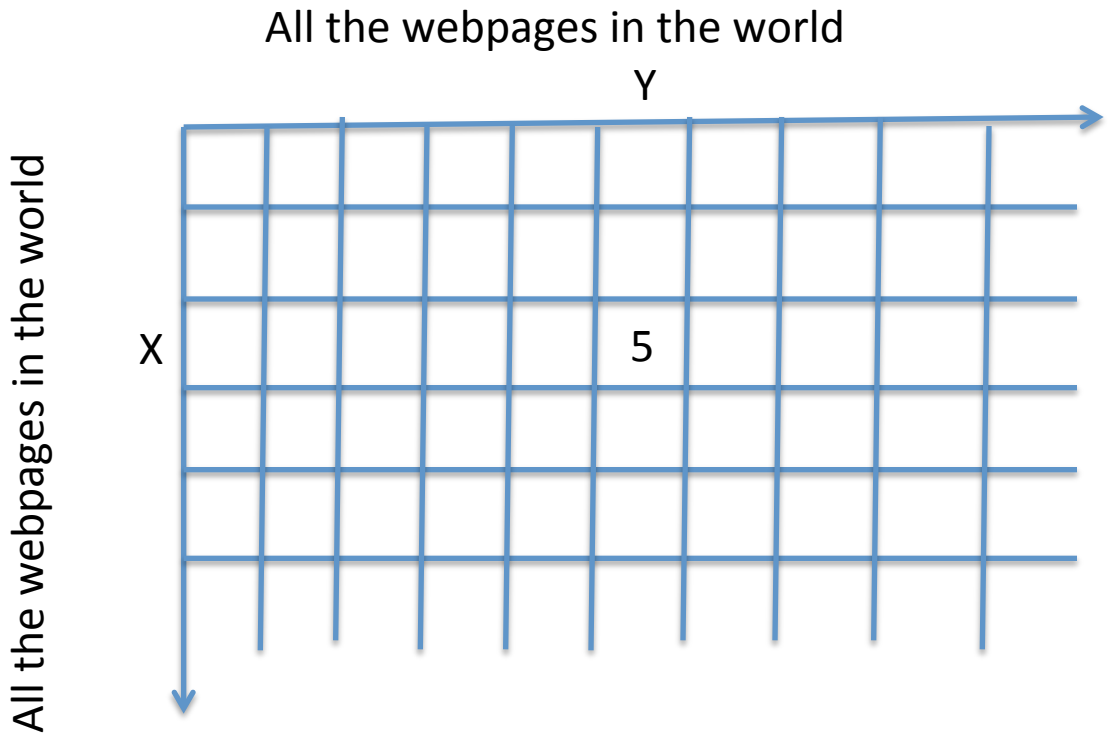


Parallel Compact Steps

1. Run Predicate
2. Create a scan-in array
   - True = 1
   - False = 0
3. Run exclusive scan over scan-in array
   - Output is the scatter addresses for input
4. Scatter the input into output addresses

Sparse Matrix

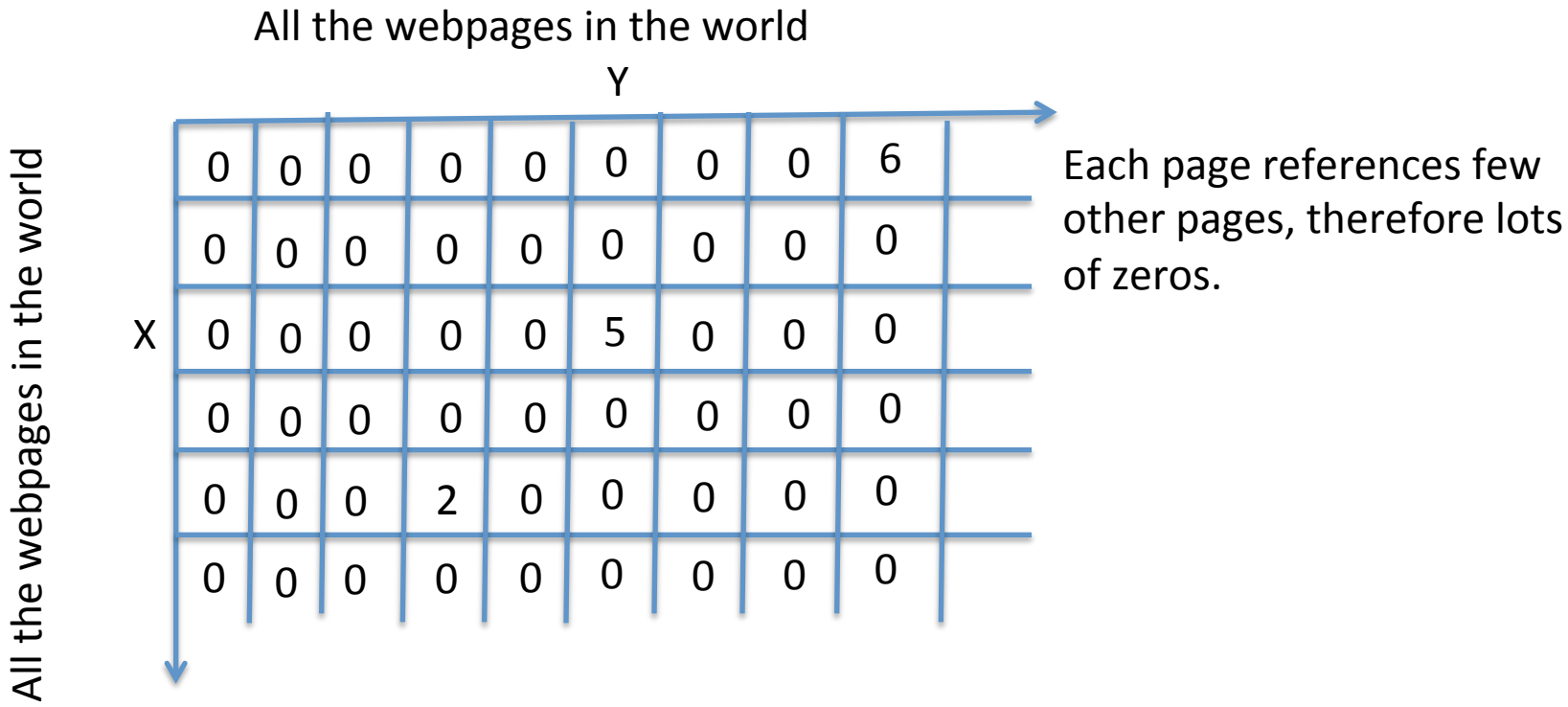
- Often matrixes are full of zeros, we want a way to squeeze out the zeros.
  - Example: Page Rank



Assume each entry is the number of times page X references page Y

Sparse Matrix

- Often matrixes are full of zeros, we want a way to squeeze out the zeros.
  - Example: Page Rank



Sparse Matrix Representation

- Compressed Sparse Row (CSR) Format

$$\begin{bmatrix} a & 0 & b \\ c & d & e \\ 0 & 0 & f \end{bmatrix}$$

Row	a	b	c	d	e	f
Column	0	2	0	1	2	2
Row Pointer	0	2	5			

Represent a matrix using three vectors

- Row = Simply all non-zero elements written in order they appear in matrix Left to Right Top to Bottom
- Column = For each element in row indicate which column the element appears in the original matrix
- Row Pointer = Indicate where each row starts in the row vector

Sparse Matrix Vector Multiplication

$$\begin{bmatrix} a & 0 & b \\ c & d & e \\ 0 & 0 & f \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} ax + 0x + by \\ cy + dy + ez \\ 0x + 0y + fz \end{bmatrix}$$

Row	a	b	c	d	e	f
Column	0	2	0	1	2	2
Row Pointer	0	2	5			

1. Create Segmented Scan
   [a b | c d e | f]
2. Gather from vector using column
   [x z x y z z]
3. Map vectors in (1) and (2) using product
   [ax bz cx dy ez fz]
4. Perform segmented scan
   [ax +bz cx + dy + ez fz]