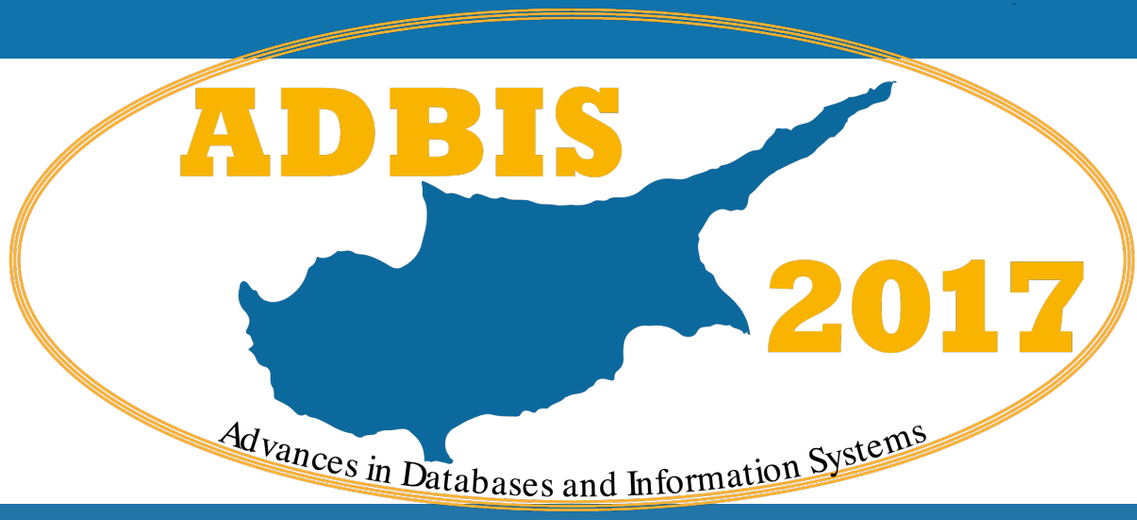
			6				
7							6
		5	16				
	5			6			77
8					2	5	



Sparse Prefix Sums

Michael Shekelyan, Anton Dignös, Johann Gamper

Free University of Bozen-Bolzano, Italy

25.09.2017, Nicosia, Cyprus

1	0	0	0	1	7	7	13
7	0	0	6	13	0	0	6
7	0	5	27	34	0	0	6
8	5	10	32	46	7	7	98
8	0	0	0	8	2	7	7
8	0	0	54	62	2	7	7
8	0	0	54	62	2	7	7

Outline

Introduction

- range sums
- prefix sums
  - technique to compute range sums in constant-time
- relative prefix sums
  - technique to achieve faster updating
- related work

Contribution

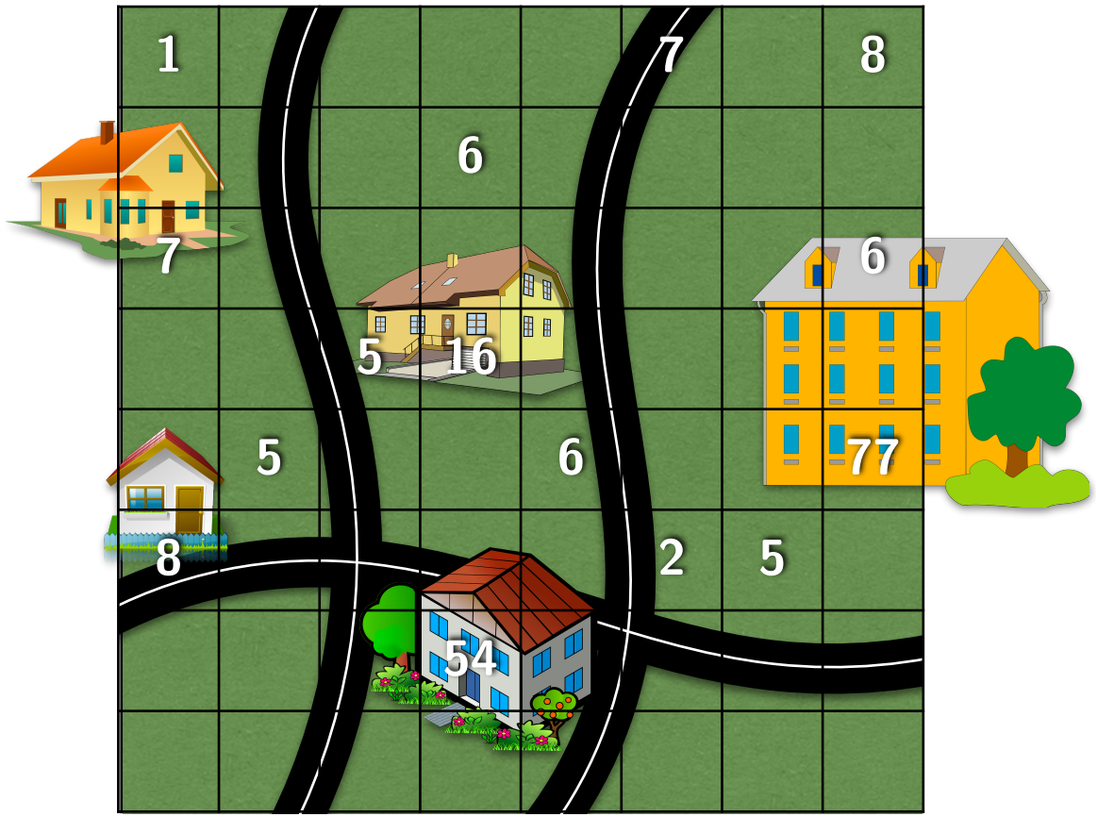
- sparse prefix sums
  - compression of relative prefix sums preserving constant query time

Experiments

- low-resolution grids
- high-resolution grids
- impact of dimensionality
- population grids based on satellite imagery

Range Sums

original data table



- e.g. each cell counts the number of inhabitants in a city

Range Sums

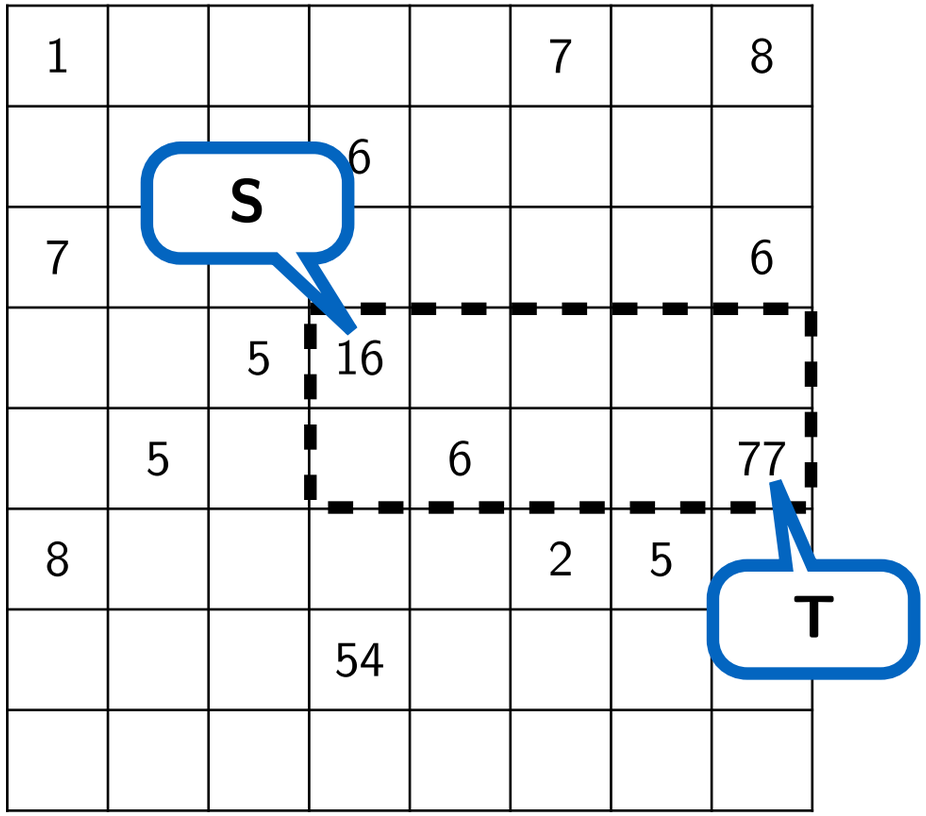
original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

- e.g. each cell counts the number of inhabitants in a city

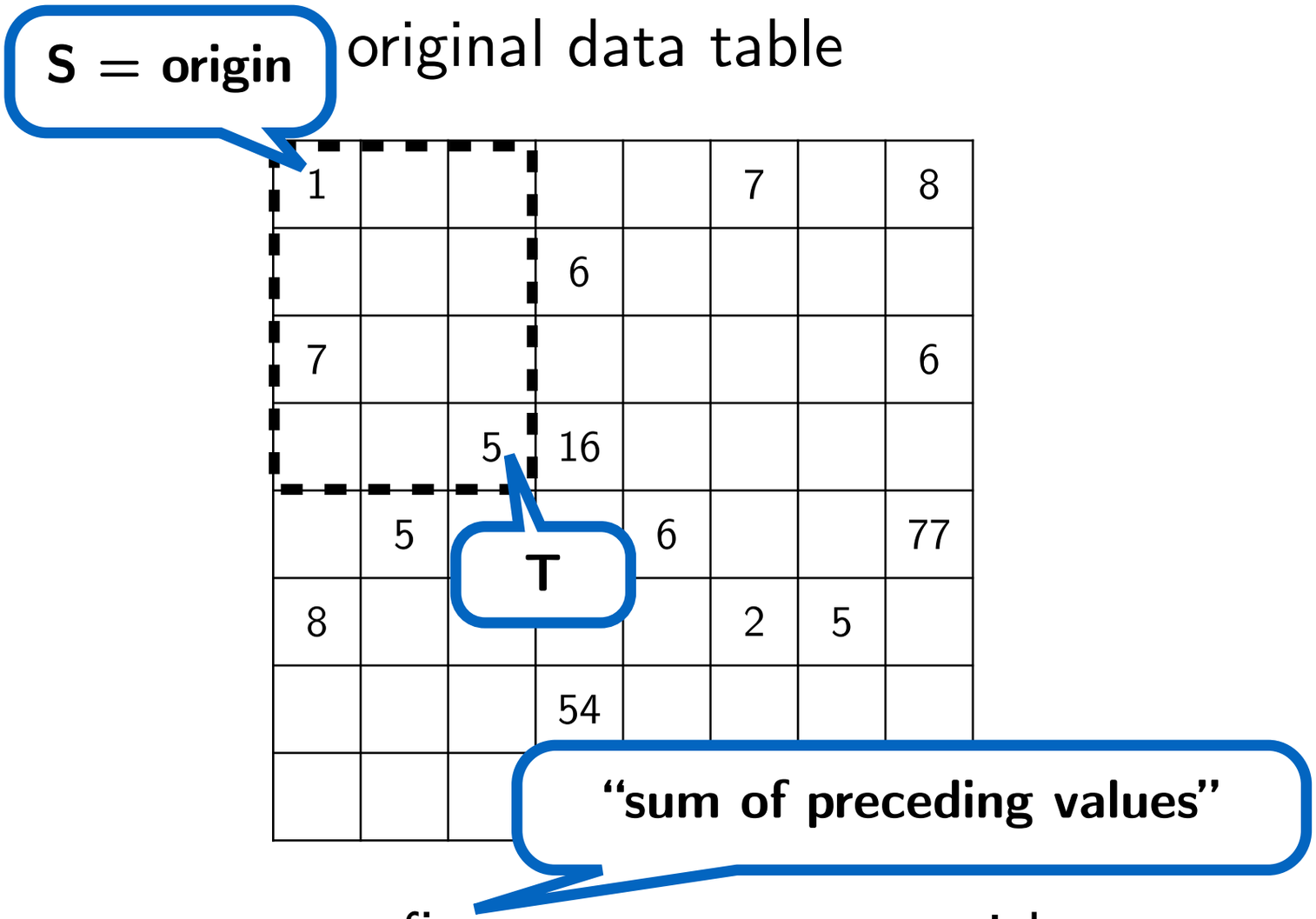
Range Sums

original data table



- range sum: sum values in a sub-matrix/tensor
- query: how many inhabitants live in range between S and T?
- answer: 16+6+77 = 99

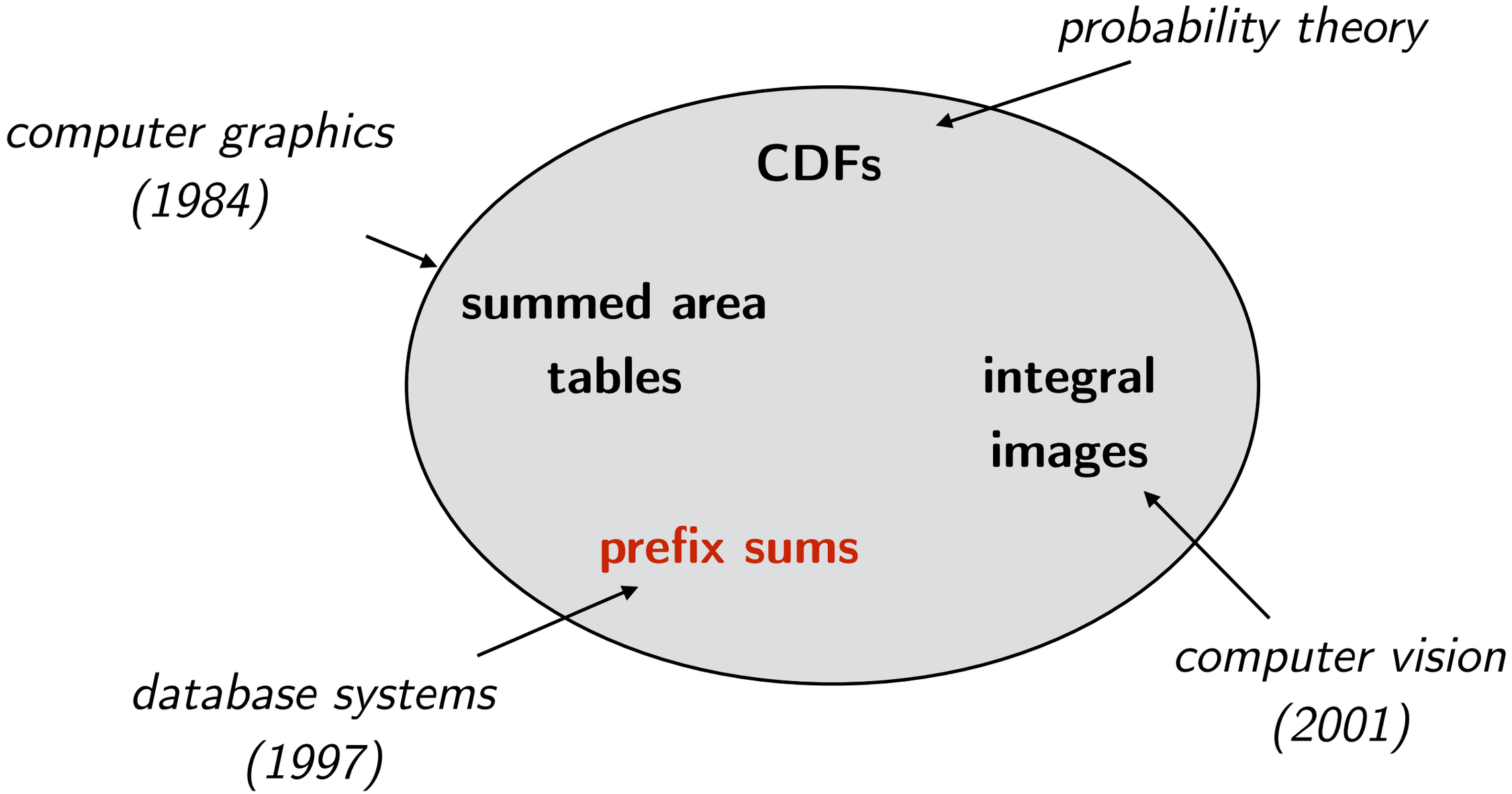
Prefix Sums



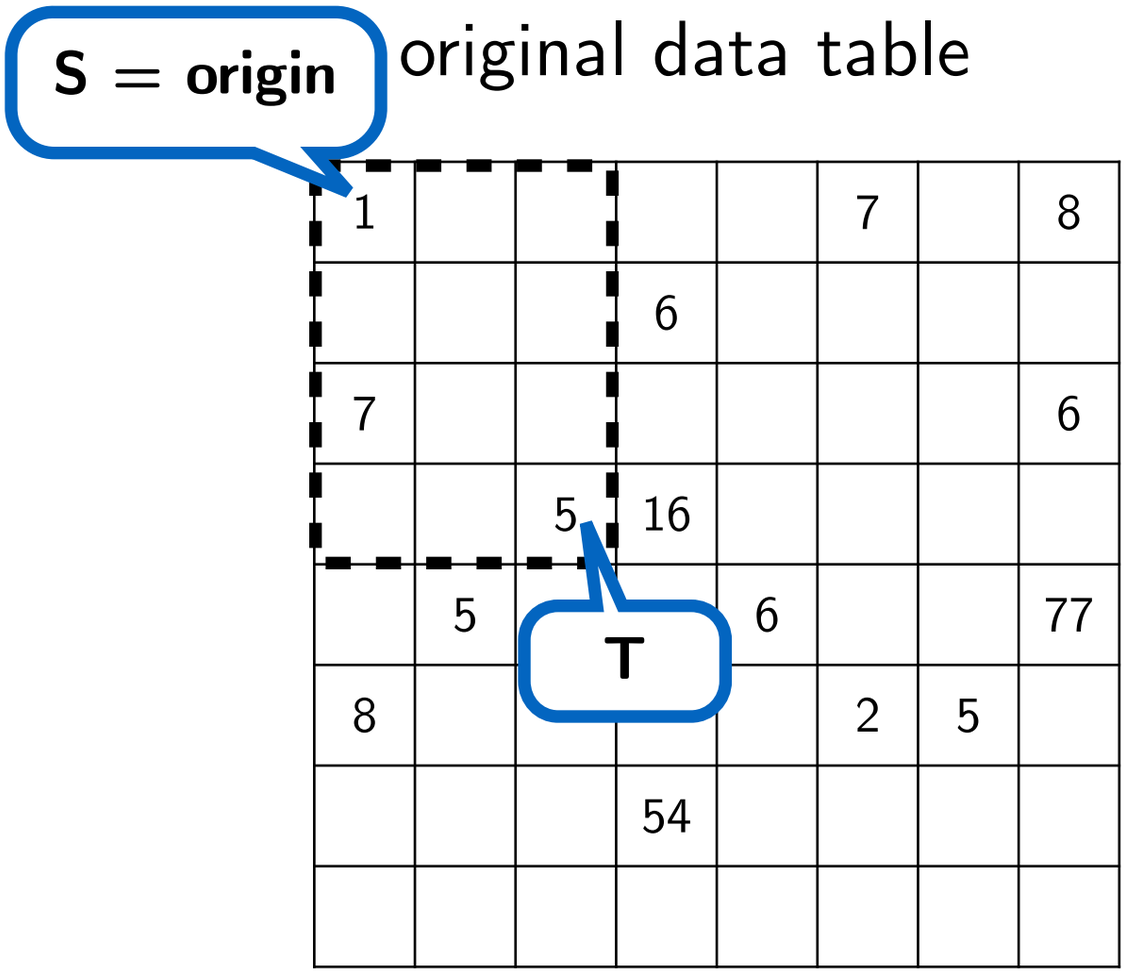
- prefix sum: range sum with one corner at origin
- query: how many inhabitants in range between origin and T?
- answer: 1+7+5 = 13

Prefix Sums

- well-known concept across many disciplines



Prefix Sums



original data table

- *prefix sum:* range sum with one corner at origin
- *query:* how many inhabitants in range between origin and T?
- *answer:* 1+7+5 = 13

Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

- data table with N cells has N prefix sums!
- *idea:* store result for each prefix sum
- *result:* O(1) querying

Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

- data table with N cells has N prefix sums!
- *idea:* store result for each prefix sum
- *result:* O(1) querying
- *example:* 13 = 1+7+5

Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	**8**	14	14	21	21	**35**
8	8	13	35	35	42	42	56
8	13	**18**	40	46	53	53	**144**
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

- each range sum can be computed from 2^d prefix sums

Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

- add prefix sum (144)

Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

- prefix sum (144) adds too much

Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

- subtract prefix sum (35)

Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

- still too much added on the left (5+5)

Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

- subtract prefix sum (18)

Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

- top-left (1+7) was first added once and then subtracted twice

Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

- add top-left again to balance out additions/subtractions

Prefix Sums

original data table

1							8
7							6
		5	16				
	5			6			77
8					2	5	
			54				

16+6+77 = 99

prefix sums

1	1						16
1	1						22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

144-35-18+8 = 99

- range sum computed with prefix sums at corner cells of range

Prefix Sum Updating

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

update complexity

- O(1) with dense storage

update complexity

- O(N) where N is # of cells

Prefix Sum Updating

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

relative prefix sums

1	0	0	0	1	7	7	15
0	0	0	6	6	0	0	0
7	0	0	6	13	0	0	6
7	0	5	27	34	0	0	6
8	5	10	32	46	7	7	98
8	0	0	0	8	2	7	7
8	0	0	54	62	2	7	7
8	0	0	54	62	2	7	7

update complexity

- O(1) with dense storage

update complexity

- O(N) where N is # of cells

update complexity

- $O(N^{1/2})$

Relative Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

relative prefix sums

1	0	0	0	1	7	7	15
0	0	0	6	6	0	0	0
7	0	0	6	13	0	0	6
7	0	5	27	34	0	0	6
8	5	10	32	46	7	7	98
8	0	0	0	8	2	7	7
8	0	0	54	62	2	7	7
8	0	0	54	62	2	7	7

relative prefix sums

- split into around $N^{1/2}$ blocks
- each block has an anchor cell storing the prefix sum
- each block has border cells storing prefix sum minus anchor cell
- each block has local cells storing local prefix sum

Relative Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	**1**	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	**46**	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

relative prefix sums

1	0	0	0	**1**	7	7	15
0	0	0	6	6	0	0	0
7	0	0	6	13	0	0	6
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0	6
8	5	10	32	**46**	7	7	98
8	0	0	0	8	2	7	7
8	0	0	54	62	2	7	7
8	0	0	54	62	2	7	7

anchor cell

relative prefix sums

- split into around $N^{1/2}$ blocks
- each block has an anchor cell storing the prefix sum

Relative Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

relative prefix sums

1	0	0	0	1	7	7	15
0	0	0	6	6	0	0	0
7	0	0	6	13	0	0	6
1	0	0	6	4	0	0	6
8	5	10	32	46	7	7	98
8	0	0	0	8	2	7	7
8	0	0	54	62	2	7	7
8	0	0	54	62	2	7	7

anchor cell

relative prefix sums

- split into around $N^{1/2}$ blocks
- each block has an anchor cell storing the prefix sum

Relative Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

relative prefix sums

1	0	0	0	1	7	7	15
0	0	0	6	6	0	0	0
7	0	0	6	13	0	0	6
					0	0	6
					7	7	98
8	0	0	0	8	2	7	7
8	0	0	54	62	2	7	7
8	0	0	54	62	2	7	7

overlay cell

relative prefix sums

- split into around $N^{1/2}$ blocks
- each block has an anchor cell storing the prefix sum
- each block has overlay cells storing prefix sum minus anchor cell

Relative Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	**46**	53	53	144
16	21	26	48	**54**	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

relative prefix sums

1	0	0	0	1	7	7	15
0	0	0	6	6	0	0	0
7	0	0	6	13	0	0	6
					0	0	6
					7	7	98
8	0	0	0	**8**	2	7	7
8	0	0	54	62	2	7	7
8	0	0	54	62	2	7	7

overlay cell

relative prefix sums

- split into around $N^{1/2}$ blocks
- each block has an anchor cell storing the prefix sum
- each block has overlay cells storing prefix sum minus anchor cell

Relative Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	
8	8	8	14	14	21	21	
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

relative prefix sums

1	0	0	0	1	7	7	15
			6	6	0	0	0
			6	13	0	0	6
7	0	5	27	34	0	0	6
8	5	10	32	46	7	7	98
8	0	0	0	8	2	7	7
8	0	0	54	62	2	7	7
8	0	0	54	62	2	7	7

local cell

relative prefix sums

- split into around $N^{1/2}$ blocks
- each block has an anchor cell storing the prefix sum
- each block has overlay cells storing prefix sum minus anchor cell
- each block has local cells storing local prefix sum

Relative Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	**1**	1	1	8	8	16
1	1	1	7	7	14	14	
8	8	8	14	14	21	21	
8	8	**13**	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

relative prefix sums

1	0	0	0	1	7	7	15
			6	6	0	0	0
			6	13	0	0	6
7	0	**5**	27	34	0	0	6
8	5	10	32	46	7	7	98
8	0	0	0	8	2	7	7
8	0	0	54	62	2	7	7
8	0	0	54	62	2	7	7

local cell

relative prefix sums

- split into around $N^{1/2}$ blocks
- each block has an anchor cell storing the prefix sum
- each block has overlay cells storing prefix sum minus anchor cell
- each block has local cells storing local prefix sum

Relative Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	
8	8	8	14	14	21	21	
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

relative prefix sums

1	0	0	0	1	7	7	15
			6	6	0	0	0
			6	13	0	0	6
7	0	5	27	34	0	0	6
8	5	10	32	46	7	7	98
8	0	0	0	8	2	7	7
8	0	0	54	62	2	7	7
8	0	0	54	62	2	7	7

local cell

relative prefix sums

- split into around $N^{1/2}$ blocks
- each block has an anchor cell storing the prefix sum
- each block has overlay cells storing prefix sum minus anchor cell
- each block has local cells storing local prefix sum

Relative Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

A	1	**B**	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
C	8	**X**	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

relative prefix sums

A	0	B-A	0	1	7	7	15
0	0	0	6	6	0	0	0
7	0	0	6	13	0	0	6
C-A	0	**Y**	27	34	0	0	6
8	5	10	32	46	7	7	98
8	0	0	0	8	2	7	7
8	0	0	54	62	2	7	7
8	0	0	54	62	2	7	7

reconstruct prefix sums from relative prefix sums

- clearly Y = X+A-B-C

Relative Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

A	1	**B**	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
C	8	**X**	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

relative prefix sums

A	0	B-A	0	1	7	7	15
0	0	0	6	6	0	0	0
7	0	0	6	13	0	0	6
C-A	0	**Y**	27	34	0	0	6
8	5	10	32	46	7	7	98
8	0	0	0	8	2	7	7
8	0	0	54	62	2	7	7
8	0	0	54	62	2	7	7

reconstruct prefix sums from relative prefix sums

- clearly Y = X+A-B-C
- solving for X results in X = Y-A+B+C
- which can be rewritten as X = Y+A+(B-A)+(C-A)

Relative Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

relative prefix sums

1	0	0	0	1	7	7	15
0	0	0	6	6	0	0	0
7	0	0	6	13	0	0	6
7	0	5	27	34	0	0	6
8	5	10	32	46	7	7	98
8	0	0	0	8	2	7	7
8	0	0	54	62	2	7	7
8	0	0	54	62	2	7	7

relative prefix sums

- problem: O(N) storage inefficient for sparse tables
- clearly a lot of redundancy as some column/rows repeat themselves
  - (due to zeros in the original data table)

Related Work

- objective: constant-time querying and sub-linear storage for sparse matrices

approach	reference	query	update	storage
convent. prefix sums	SIGMOD'97	O(1)	O(N)	O(N)
relative prefix sums	ICDE'99	O(1)	O(N	O(N)
prefix cube pool	DSS'04	O(1)	O(N)	O(N)
sparse prefix sums	ADBIS'17	O(1)	O(N	O(N
range trees	SIAM'88	O(log(S)	O(log(S)	O(S log(S)
double rel. prefix sums	DKE'00	O(N	O(N	O(N)
dynamic data cube	EDBT'00	O(log(N)	O(log(N)	O(N)
pCube	SSDBM'00	O(S)	O(S)	O(S)

SIGMOD'97: Ho et al. "Range queries in OLAP data cubes."
ICDE'99: Geffner et al. "Relative prefix sums: an efficient approach for querying dynamic OLAP data cubes."
DSS'04: Chun et al. "Space-efficient cubes for OLAP range-sum queries."
ADBIS'17: Shekelyan et. al. "Sparse Prefix Sums."
SIAM'88: Chazelle. "A functional approach to data structures and its use in multidimensional searching."
DKE'00: Liang et al. "Range queries in dynamic OLAP data cubes"
EDBT'00: Geffner et al."The dynamic data cube."
SSDBM'00: Riedewald et al. "pCUBE: update-efficient online aggregation with progressive feedback and error bounds."

Outline

Introduction

- range sums
- prefix sums
  - technique to compute range sums in constant-time
- relative prefix sums
  - technique to achieve faster updating
- related work

Contribution

- sparse prefix sums
  - compression of relative prefix sums preserving constant query time

Experiments

- low-resolution grids
- high-resolution grids
- impact of dimensionality
- population grids based on satellite imagery

Contribution

sparse prefix sums for low-dimensional data cubes

- based on relative prefix sums (RPS)
- unlike RPS lossless compression exploiting sparsity
  - orders of magnitude storage cost reduction in case of sparsity
  - use of look-up tables keeps query costs at O(1) (µsecond-fast)
- up to order of magnitude construction cost reduction

Sparse Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

sparse prefix sums

1	0	0	0	1	7	7	15
0	0	0	6	6	0	0	0
7	0	0	6	13	0	0	6
7	0	5	27	34	0	0	6
8	5	10	32	46	7	7	98
8	0	0	0	8	2	7	7
8	0	0	54	62	2	7	7
8	0	0	54	62	2	7	7

sparse prefix sums

- based on relative prefix sums, but exploits sparsity in original table
- no significant query time overhead due to the use of look-up tables

Sparse Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

sparse prefix sums

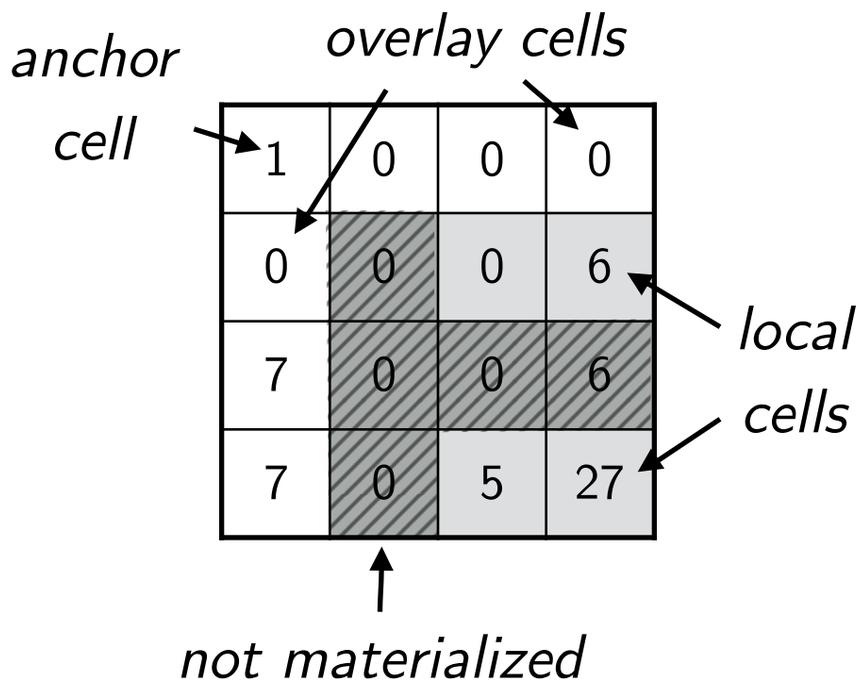
1	0	0	0	1	7	7	15
0	0	0	6	6	0	0	0
7	0	0	6	13	0	0	6
7	0	5	27	34	0	0	6
8	5	10	32	46	7	7	98
8	0	0	0	8	2	7	7
8	0	0	54	62	2	7	7
8	0	0	54	62	2	7	7

sparse prefix sums

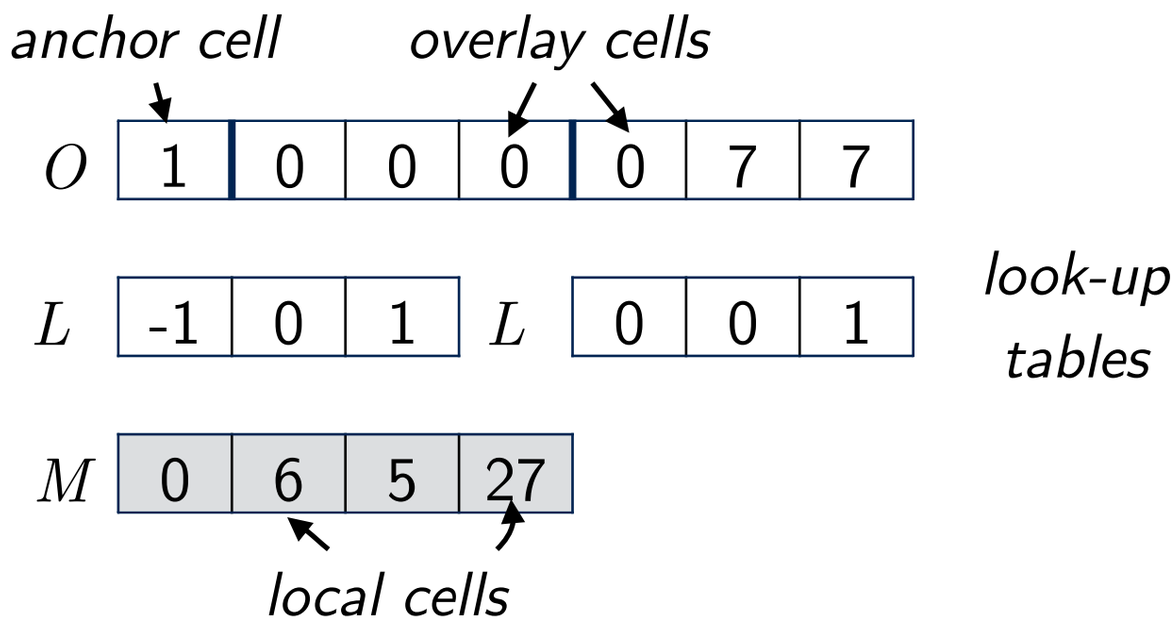
- based on relative prefix sums, but exploits sparsity in original table
- no significant query time overhead due to the use of look-up tables

Sparse Prefix Sums

conceptually



representation

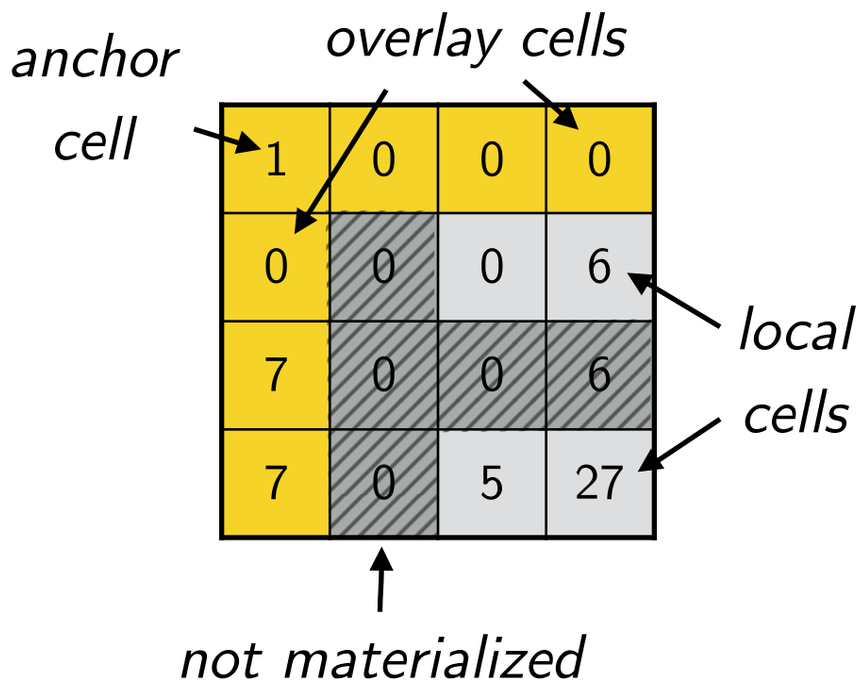


sparse prefix sums

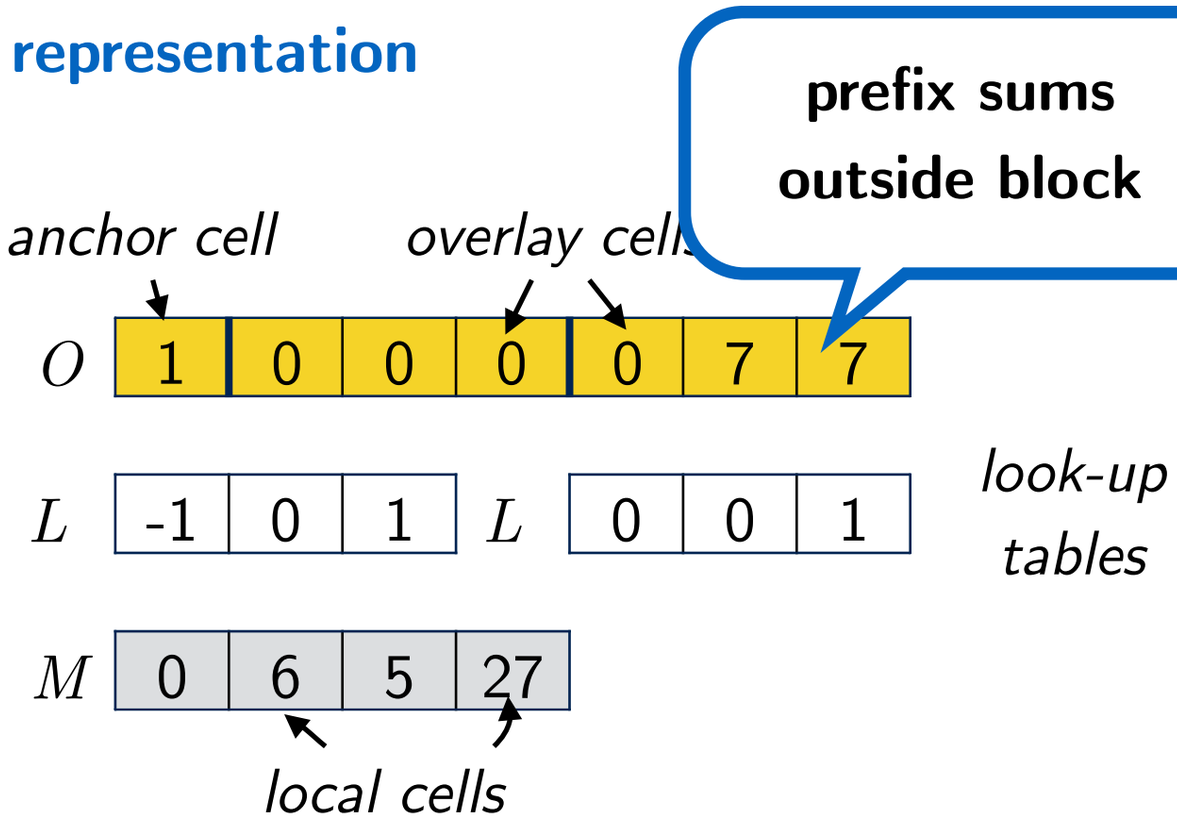
- stores each block as 2+d arrays
- from stored arrays all prefix sums of the block can be reconstructed

Sparse Prefix Sums

conceptually



representation

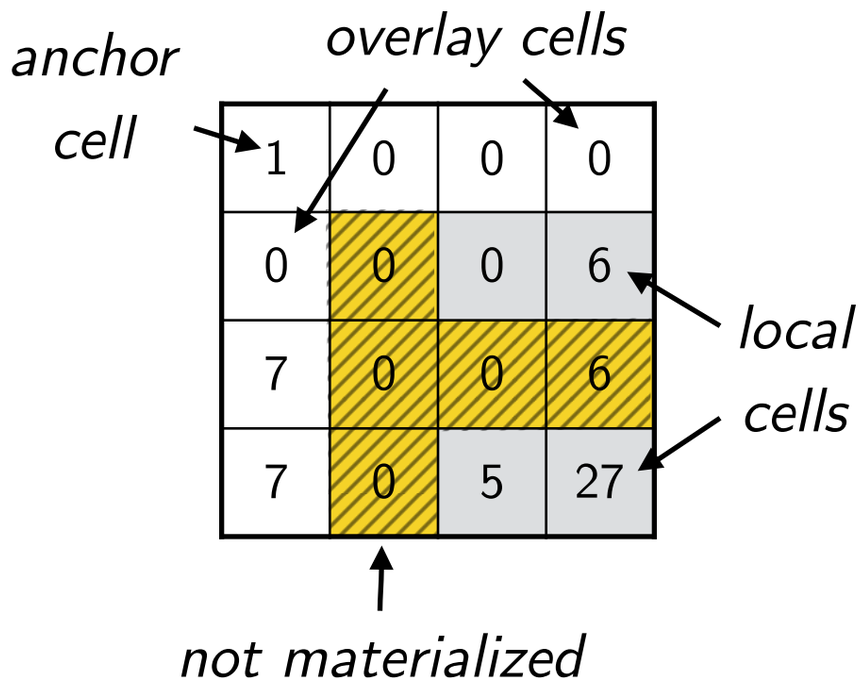


sparse prefix sums

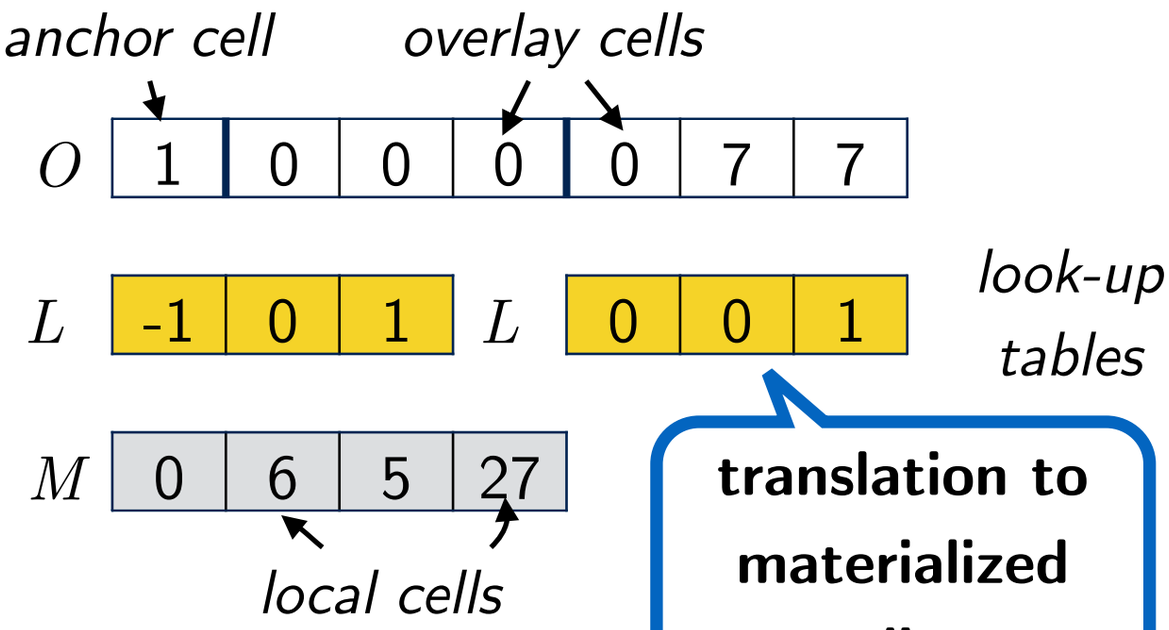
- stores each block as 2+d arrays
- from stored arrays all prefix sums of the block can be reconstructed
  - anchor/overlay cells express prefix sums along the upper border

Sparse Prefix Sums

conceptually



representation

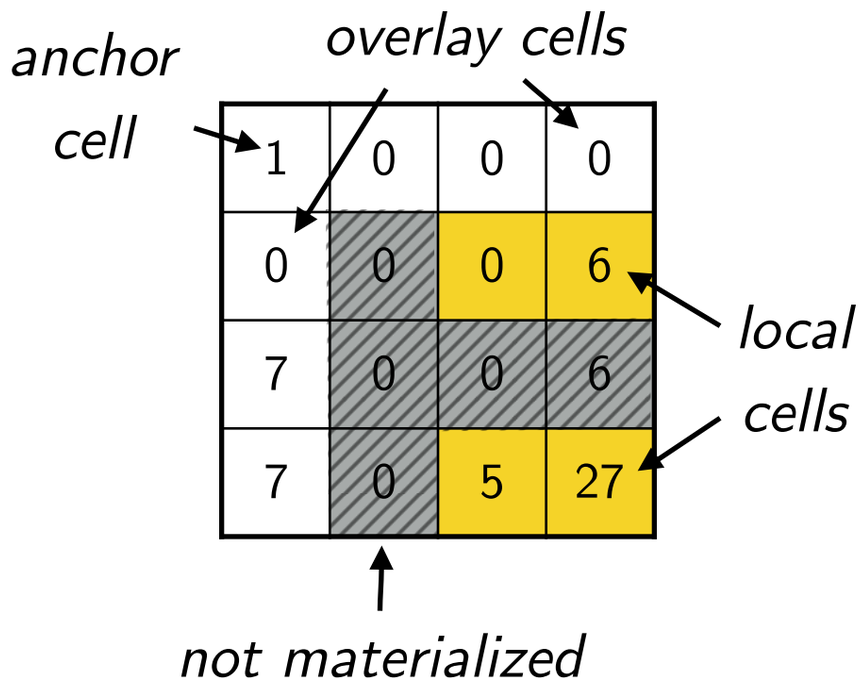


sparse prefix sums

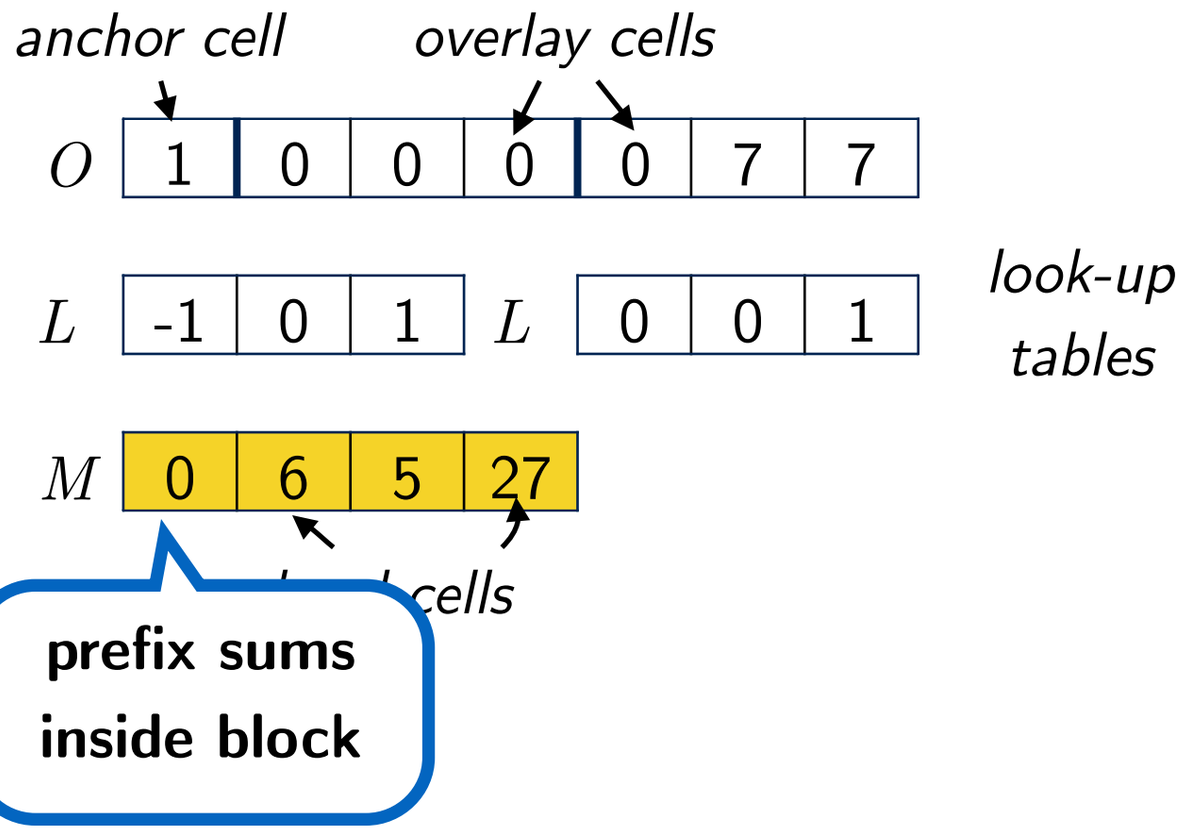
- stores each block as 2+d arrays
- from stored arrays all prefix sums of the block can be reconstructed
  - anchor/overlay cells express prefix sums along the upper border
  - look-up tables express empty rows/column in a way that avoids overhead

Sparse Prefix Sums

conceptually



representation



sparse prefix sums

- stores each block as 2+d arrays
- from stored arrays all prefix sums of the block can be reconstructed
  - anchor/overlay cells express prefix sums along the upper border
  - look-up tables express empty rows/column in a way that avoids overhead
  - local cells express prefix sums over non-upper border cells

Sparse Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

sparse prefix sums

1	0	0	0	1	7	7	15
0	0	0	6	6	0	0	0
7	0	0	6	13	0	0	6
7	0	5	27	34	0	0	6
8	5	10	32	46	7	7	98
8	0	0	0	8	2	7	7
8	0	0	54	62	2	7	7
8	0	0	54	62	2	7	7

sparse prefix sums

Sparse Prefix Sums

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

sparse prefix sums

1	0	0	0	1	7	7	15
0	0	0	6	6	0	0	0
7	0	0	6	13	0	0	6
7	0	5	27	34	0	0	6
8	5	10	32	46	7	7	98
8	0	0	0	8	2	7	7
8	0	0	54	62	2	7	7
8	0	0	54	62	2	7	7

Sparse Prefix Sums

original data table

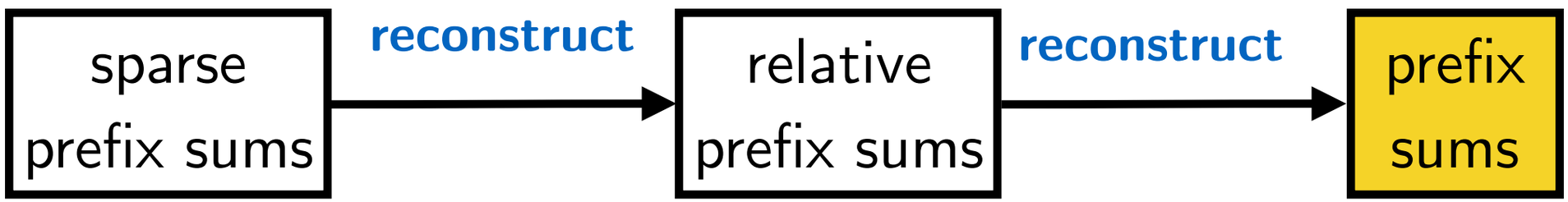
1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

sparse prefix sums

1	0	0	0	1	7	7	15
0	0	0	6	6	0	0	0
7	0	0	6	13	0	0	6
7	0	5	27	34	0	0	6
8	5	10	32	46	7	7	98
8	0	0	0	8	2	7	7
8	0	0	54	62	2	7	7
8	0	0	54	62	2	7	7



Sparse Prefix Sums

original data table

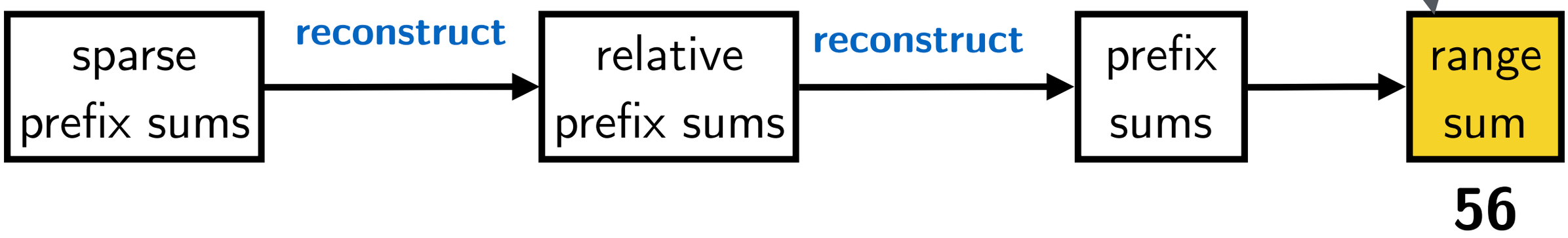
1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

sparse prefix sums

1	0	0	0	1	7	7	15
0	0	0	6	6	0	0	0
7	0	0	6	13	0	0	6
7	0	5	27	34	0	0	6
8	5	10	32	46	7	7	98
8	0	0	0	8	2	7	7
8	0	0	54	62	2	7	7
8	0	0	54	62	2	7	7



Reconstruct Relative Prefix Sums

original data table

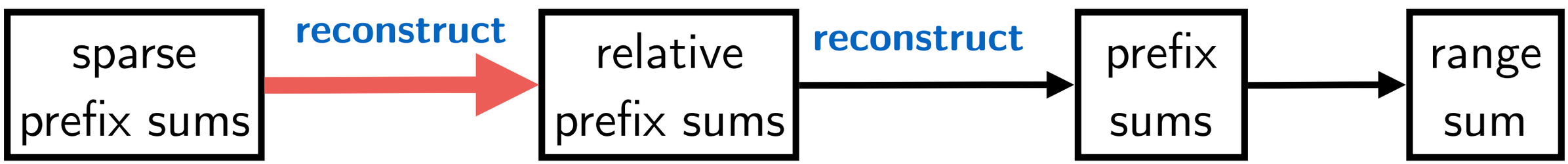
1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

sparse prefix sums

1	0	0	0	1	7	7	15
0	0	0	6	6	0	0	0
7	0	0	6	13	0	0	6
7	0	5	27	34	0	0	6
8	5	10	32	46	7	7	98
8	0	0	0	8	2	7	7
8	0	0	54	62	2	7	7
8	0	0	54	62	2	7	7



Reconstruct Relative Prefix Sums

coordinates

	0	*1*	*2*
0	0	0	6
1	0	0	6
2	0	5	27

local prefix sums

exploiting redundancy

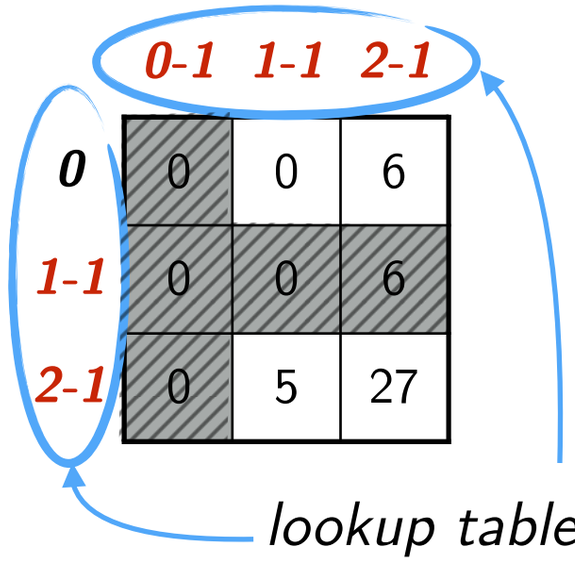
- repeated column/row can be reconstructed from predecessor column/row
- (same applies to first columns/rows that are zero)

Reconstruct Relative Prefix Sums

coordinates

	0	*1*	*2*
0	0	0	6
1	0	0	6
2	0	5	27

local prefix sums



lookup tables

	0	*1*
0	0	6
1	5	27

materialized cells

non-materialization of repeated rows/columns

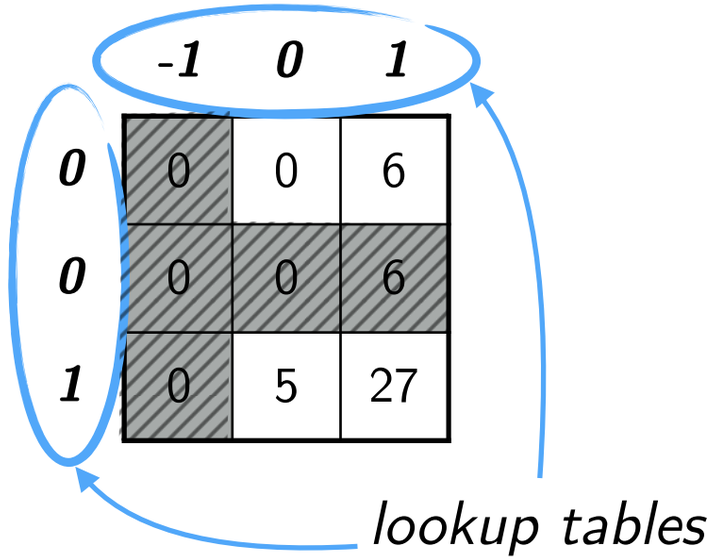
- translate coordinates to space without the repeated rows/columns
- use look-up tables to avoid query cost overhead

Reconstruct Relative Prefix Sums

coordinates

	0	*1*	*2*
0	0	0	6
1	0	0	6
2	0	5	27

local prefix sums



lookup tables

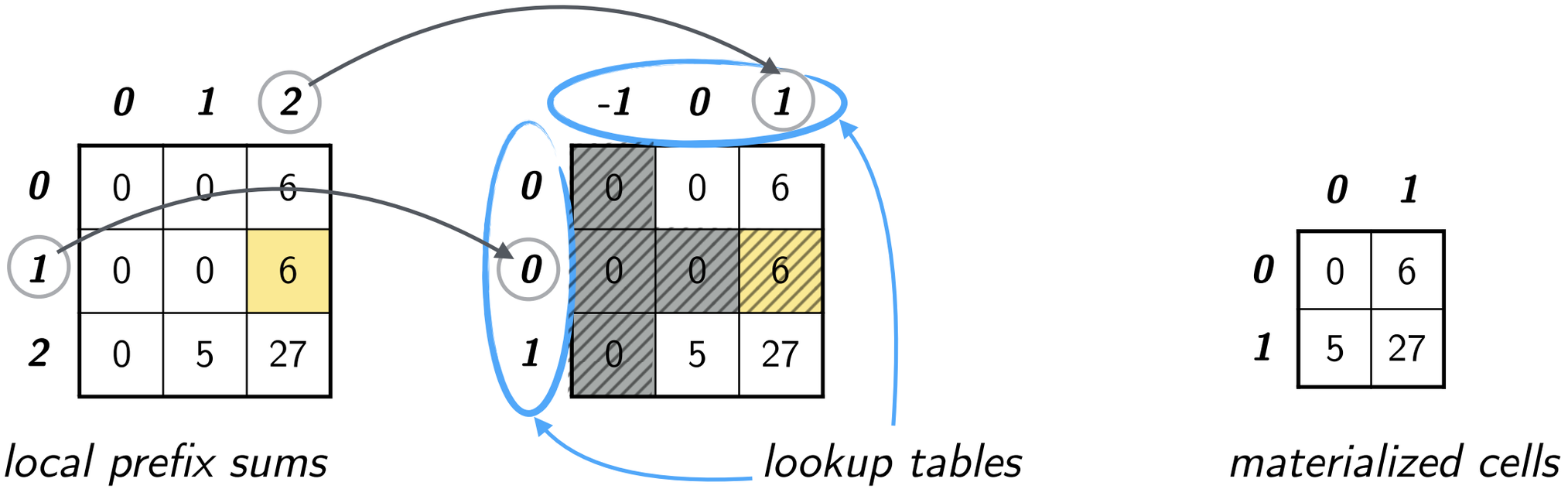
	0	*1*
0	0	6
1	5	27

materialized cells

non-materialization of repeated rows/columns

- translate coordinates to space without the repeated rows/columns
- use look-up tables to avoid query cost overhead

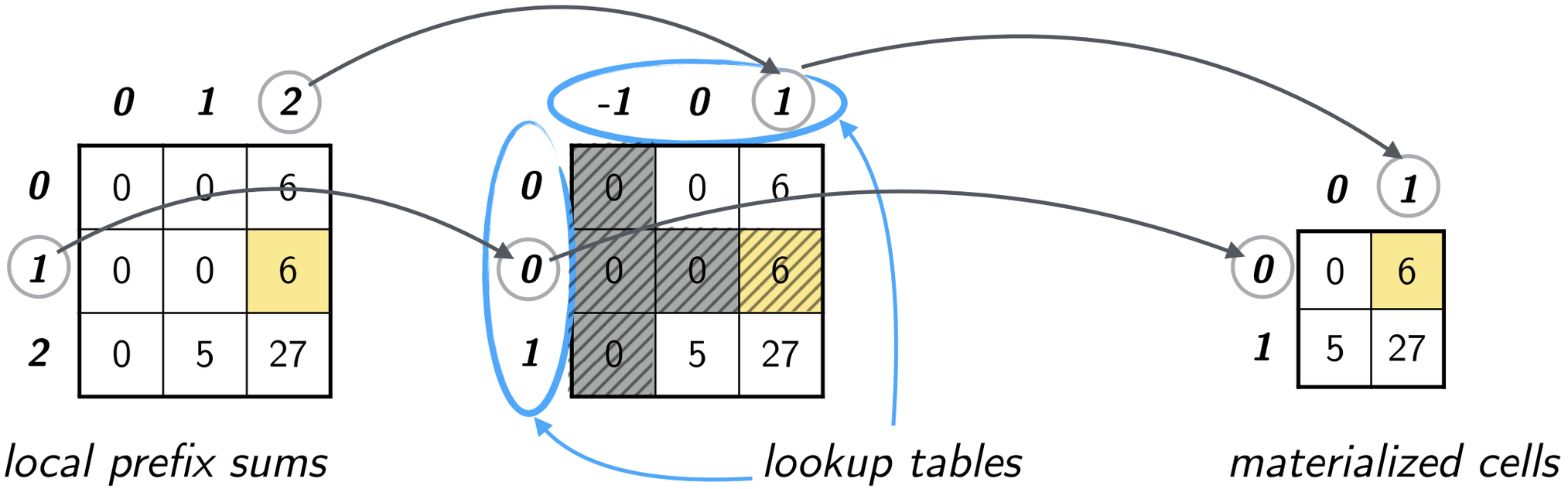
Reconstruct Relative Prefix Sums



reconstructing local prefix sums

- lookup tables translate (1,2) to (0,1)
- local prefix sum is then stored at (0,1) in materialized table

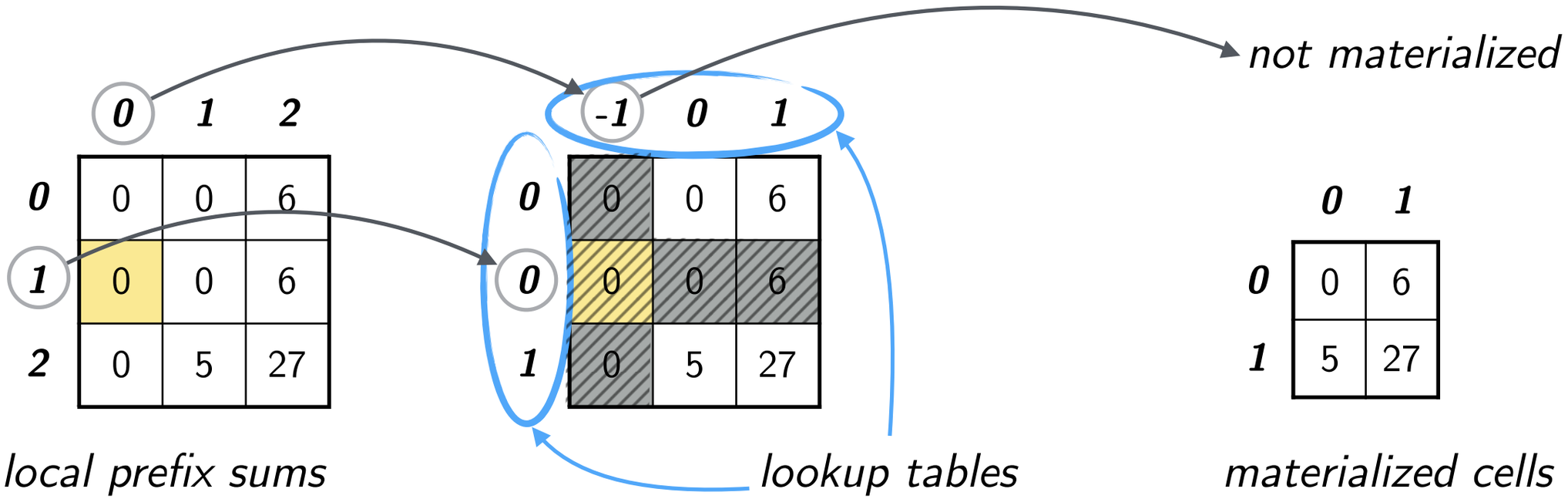
Reconstruct Relative Prefix Sums



reconstructing local prefix sums

- lookup tables translate (1,2) to (0,1)
- local prefix sum is then stored at (0,1) in materialized table

Reconstruct Relative Prefix Sums



reconstructing local prefix sums

- lookup tables translate (1,0) to (0,-1)
- when one coordinate is equal to (-1) local prefix sum is simply equal to zero

Theoretical Results

original data table

1					7		8
			6				
7							6
		5	16				
	5			6			77
8					2	5	
			54				

prefix sums

1	1	1	1	1	8	8	16
1	1	1	7	7	14	14	22
8	8	8	14	14	21	21	35
8	8	13	35	35	42	42	56
8	13	18	40	46	53	53	144
16	21	26	48	54	63	68	159
16	21	26	54	108	117	122	213
16	21	26	102	108	117	122	213

sparse prefix sums

1	0	0	0	1	7	7	15
0	0	0	6	6	0	0	0
7	0	0	6	13	0	0	6
7	0	5	27	34	0	0	6
8	5	10	32	46	7	7	98
8	0	0	0	8	2	7	7
8	0	0	54	62	2	7	7
8	0	0	54	62	2	7	7

- Let N=64 be number of grid cells and S=15 be number of non-zero grid cells
- storage complexity of sparse prefix sums are dominated by
  - $O(N^{1-1/(2d)})$ overlay cells
  - $O(SN^{1/2-1/(2d)})$ materialized local cells (assuming worst-case of only diagonals non-zero)
- storage complexity: $O(N^{1-1/(2d)}+SN^{1/2-1/(2d)})$

Outline

Introduction

- range sums
- prefix sums
  - technique to compute range sums in constant-time
- relative prefix sums
  - technique to achieve faster updating
- related work

Contribution

- sparse prefix sums
  - compression of relative prefix sums preserving constant query time

Experiments

- low-resolution grids
- high-resolution grids
- impact of dimensionality
- population grids based on satellite imagery

Experiments

baselines

- conventional prefix sums (CPS)
- relative prefix sums (RPS)

experiment 1

- matrix/tensor created from counting number of points along grid
- impact of grid resolution on storage/construction/query costs

experiment 2

- real-world matrices (gridded population data)
- storage reduction for real world data sets

Low-resolution grids

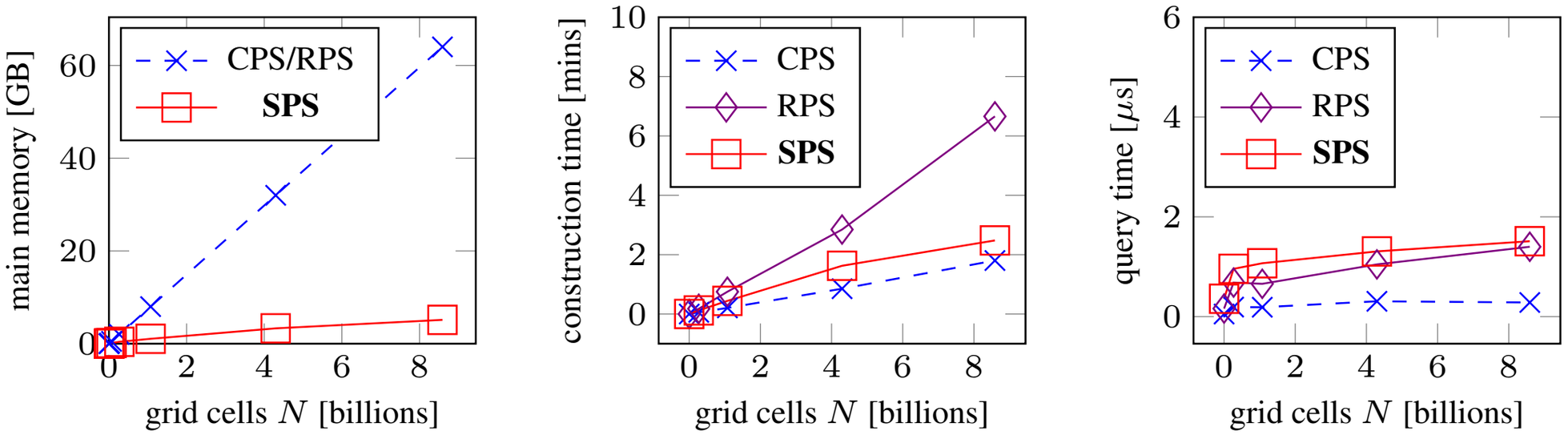


Fig. 6: Impact of grid resolution (**OSM** dataset).

- sparse prefix sums significantly lower storage (8GB instead of 60GB)
- insignificant construction/query time overhead

High-resolution grids

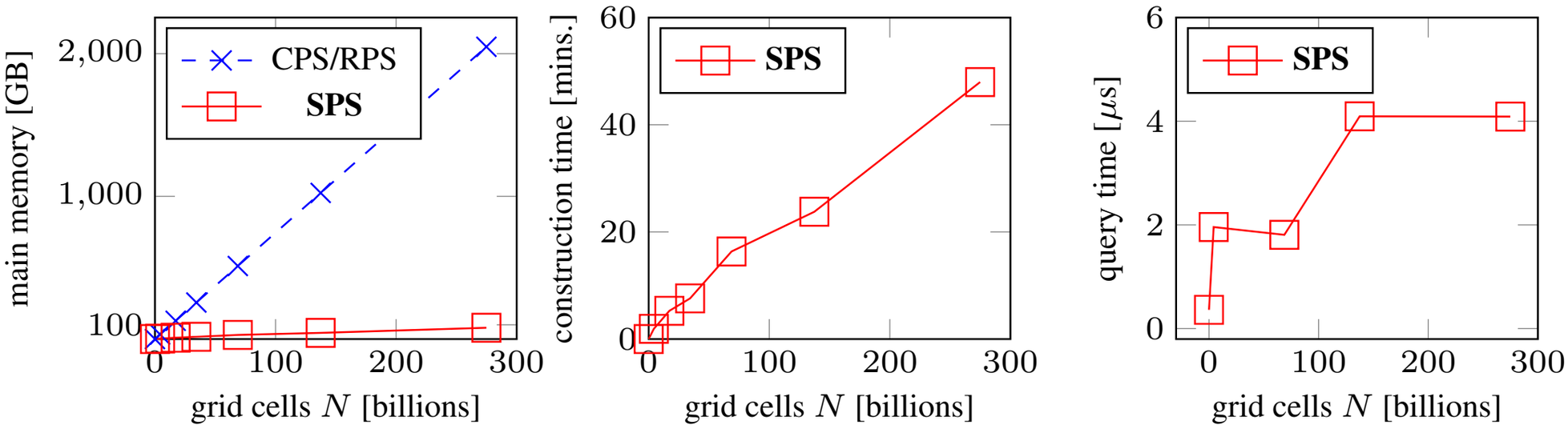


Fig. 7: Impact of a very high grid resolution (**OSM** dataset).

- sparse prefix sums make it feasible to use very high grid resolutions
- feasible construction (minutes) and ultra-fast query time (μseconds)

Grid dimensionality

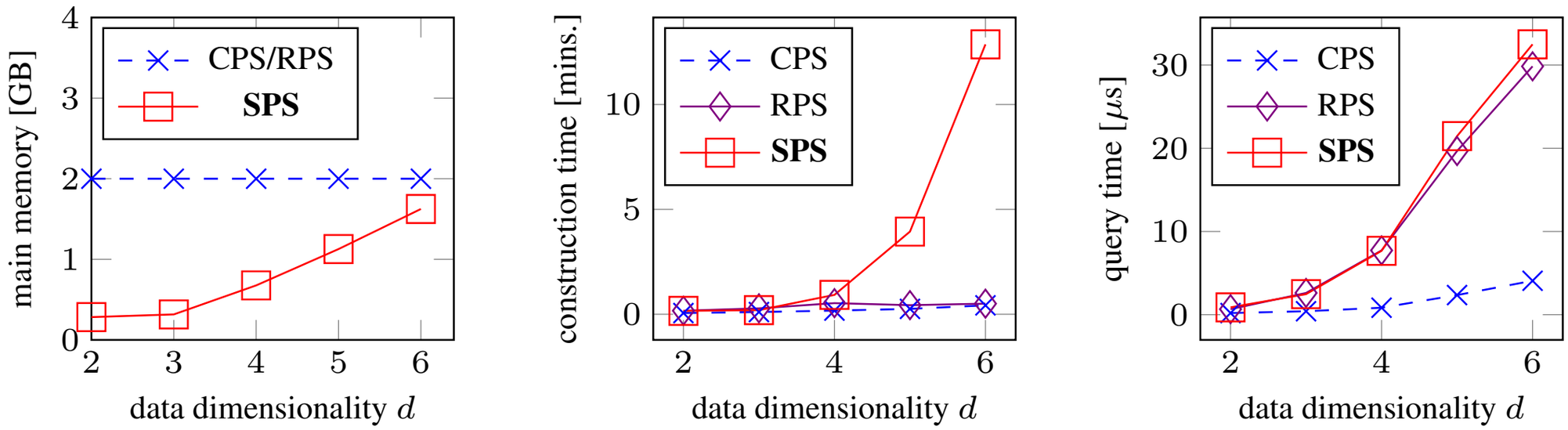
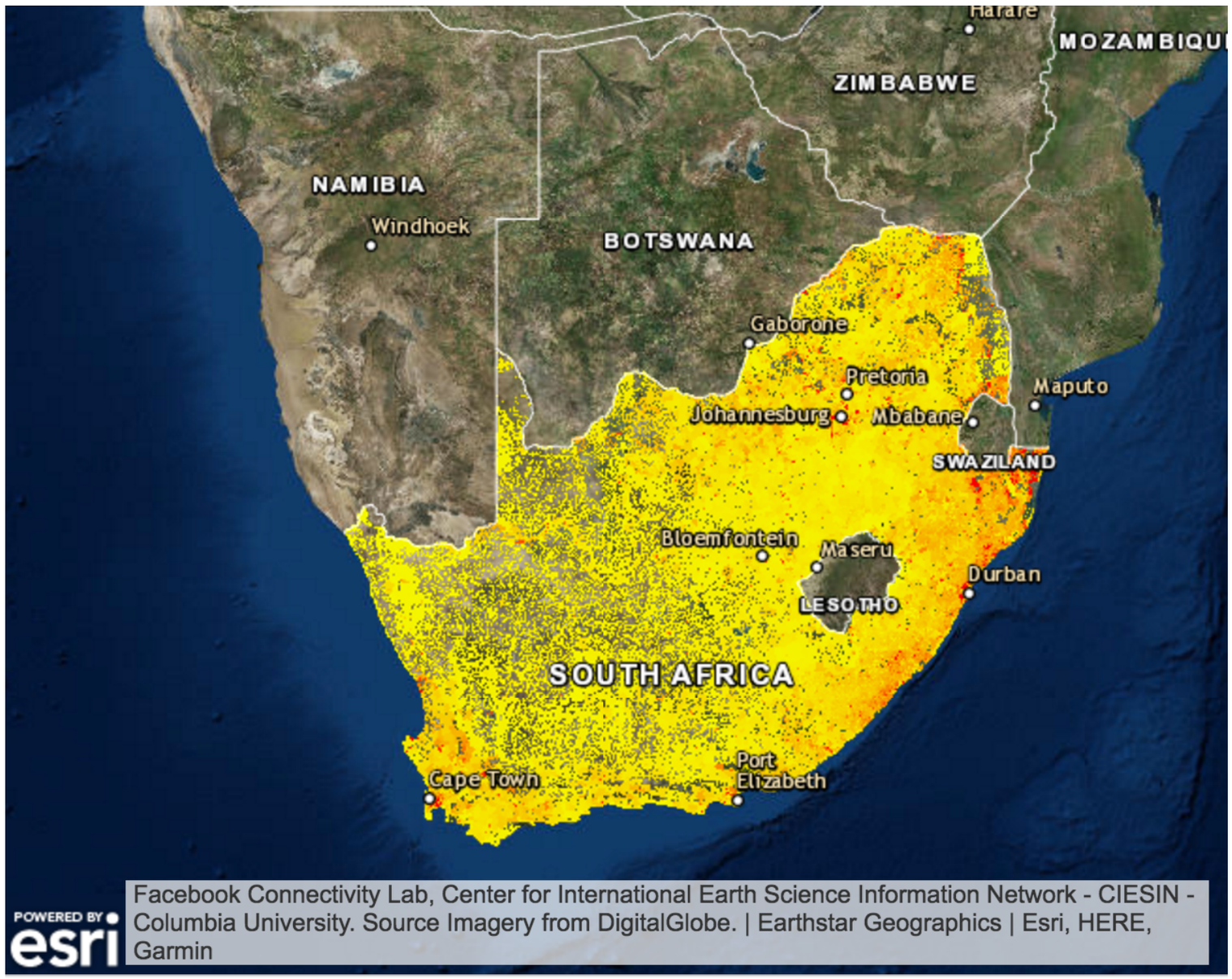


Fig. 8: Impact of dimensionality (**ZIPF** dataset).

- sparse prefix sums effective up to four dimensions
- in more dimensions % of overlay cells increases

Satellite-imagery based population grids



Population Count Low High

".tif"-file

- 0.7 GB (baseline)
- slow range sums

relative prefix sums

- 22.7 GB (32x larger)
- µs-fast range sums

sparse prefix sums

- 1.7 GB (2.4x larger)
- µs-fast range sums

Satellite-imagery based population grids

	Data		Storage (GB)		Construction			Query (micros.)		
	Res.	Sparsity	CPS/RPS	SPS	CPS	RPS	SPS	CPS	RPS	SPS
South Africa	66612x45748	99.55%	22.7GB	1.7GB	52s	197s	52s	0.3µs	0.5µs	1.4µs
Madagascar	28311x49159	99.78%	10GB	0.7GB	17s	68s	5s	0.2µs	0.6µs	1.6µs
Burkina Faso	28521x20442	99.75%	4.3GB	0.5GB	7s	35s	5s	0.2µs	0.6µs	1.4µs
Ivory Coast	23663x23147	99.64%	3.8GB	0.3GB	8s	29s	4s	0.2µs	0.5µs	1.7µs
Ghana	17639x23151	99.44%	2.9GB	0.4GB	5s	25s	10s	0.2µs	0.6µs	1.4µs
Malawi	11606x27931	96.89%	2.4GB	0.4GB	4s	15s	3s	0.2µs	0.7µs	0.9µs
Sri Lanka	8757x14103	96.89%	0.9GB	0.3GB	2s	6s	4s	0.2µs	0.5µs	1.3µs
Haiti	12473x7513	98.15%	0.6GB	0.1GB	1s	3s	1s	0.2µs	0.6µs	1.0µs

- PRO: up to an order of magnitude less storage and faster construction
- CONTRA: up to 3.4x slower query time than RPS

Summary

sparse prefix sums

- based on relative prefix sums (RPS)
- theoretical results
  - constant query costs, sub-linear storage costs, sub-linear update costs
- experimental results
  - for mid-resolution grids 8x size reduction (8GB instead of 60GB)
  - for high-resolution grids 25x size reduction (79GB instead of 2TB)
  - effective up to four grid dimensions
  - for real-world grids 13x size reduction (1.7GB instead of 22.7 GB)
  - µ-second-fast query time

Future Work

improvement

- match query time of RPS (overlay cell storage, cache-friendliness)
- trade query time for storage by using multiple sparse prefix sums
- more sophisticated strategies for block splitting

evaluation

- compare speed/storage to range tree indices

applications

- dynamic low-dimensional histograms
- constant-time containment check for convex polytopes

Thank you for your attention!