CMSC 828S, Spring 2005

Presentation on

MULTIDIMENSIONAL LINEAR HASHING

by
B Brent Gordon

MULTIDIMENSIONAL LINEAR HASHING

- Introduction & basic principles
- Linear Hashing with Reversed Bit Interleaving (LHRBI)
- Multidimensional Extendible Hashing (MDEH)
- Comparison of LHRBI and MDEH
- Quantile Hashing
- Piecewise Linear Order Preserving (PLOP)
- Dynamic Z Hashing
- Linear Hashing with Partial Expansions (LHPE)

Linear Hashing

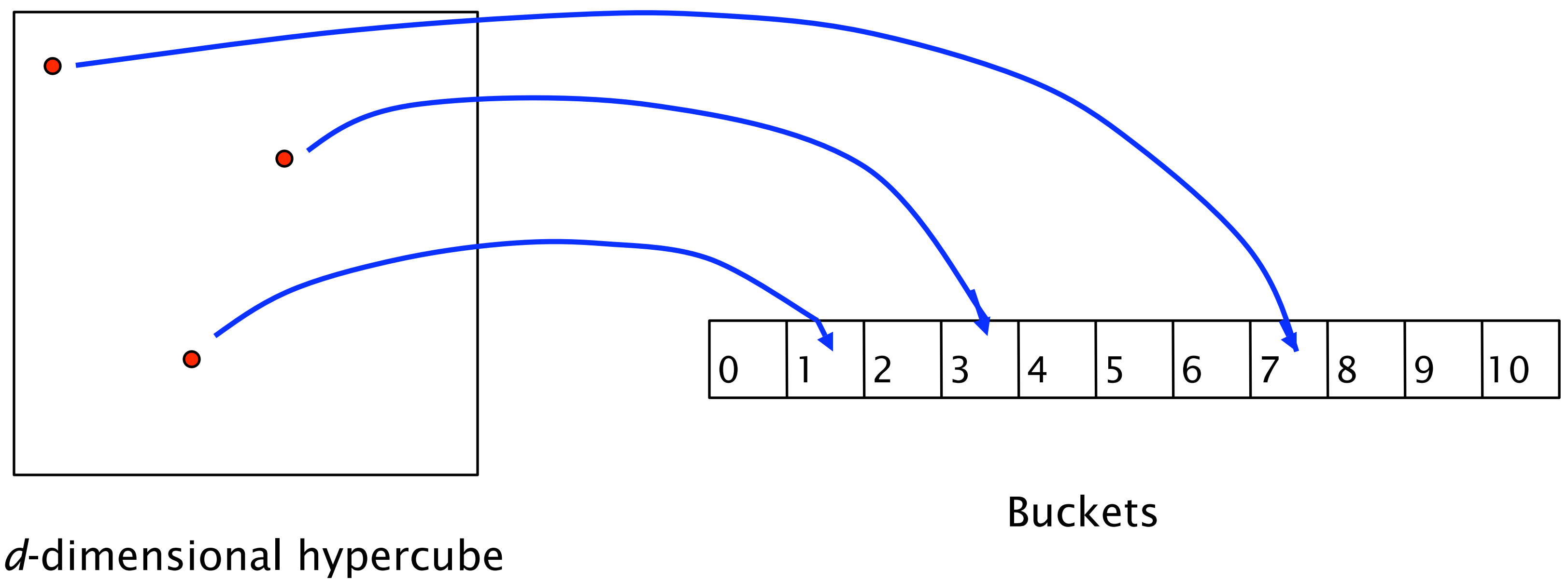
Linear Hashing in general:

- Create one additional cell at a time
- May be based on
  - Overflow of bucket or cell, or
  - Storage Utilization Factor

Linear Hashing in dimension $d > 1$

- d-dimensional hypervolume is subdivided into d-dimensional cells
- Cells are linearly ordered, typically in order of creation

The hashing function must map points in *d*-space to a unique cell/bucket number



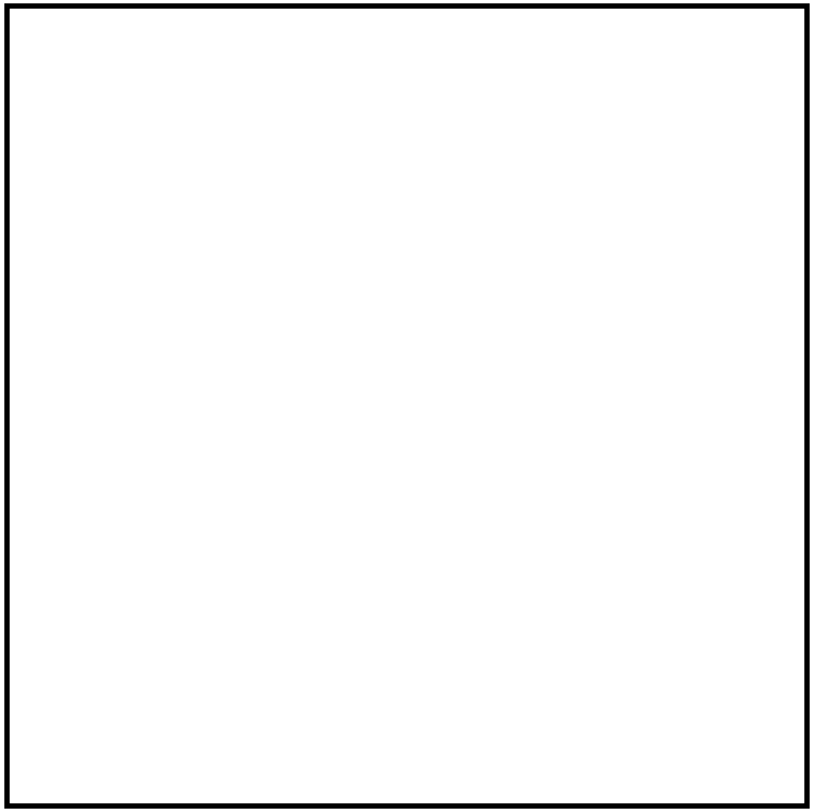
Total Grid Partition

Total grid partition: number of cells is doubled

(But cells have same shape as original only every dth total grid partition (tgp).

Example $d = 2$.

Number of cells = 1
Total grid partition 0
Cell is square

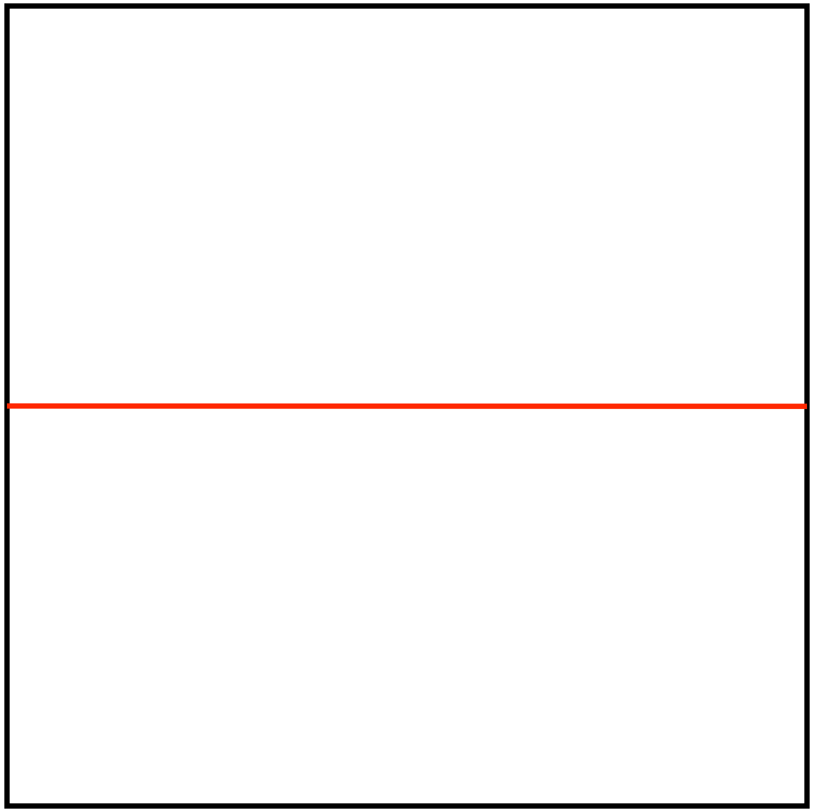


Total grid partition: number of cells is doubled

(But cells have same shape as original only every *d*th total grid partition (tgp).

Example $d = 2$.

Number of cells = 2
Total grid partition 1
Cells are not square

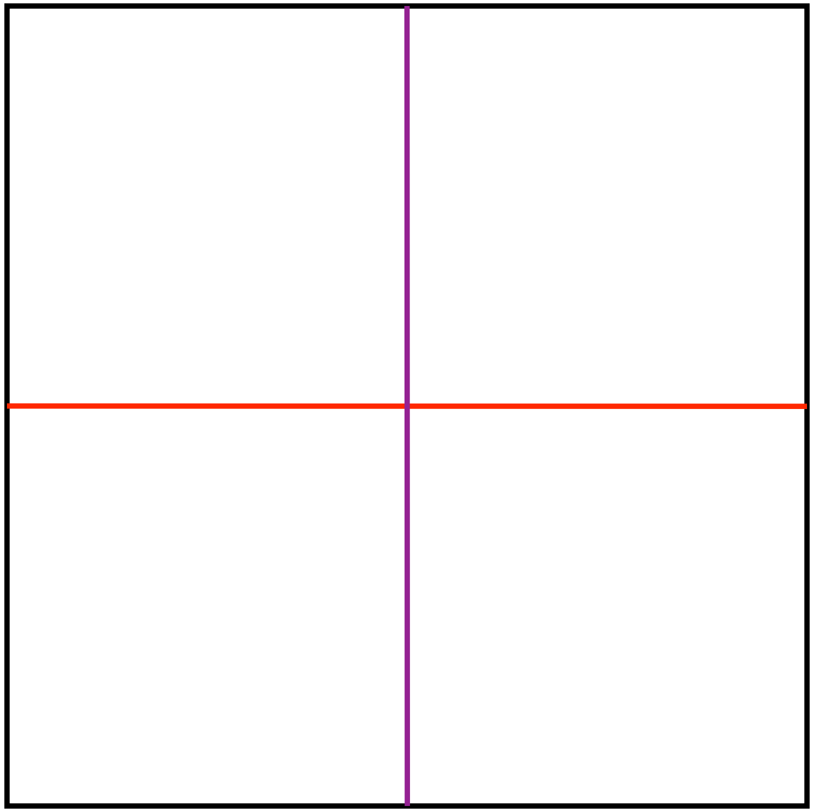


Total grid partition: number of cells is doubled

(But cells have same shape as original only every dth total grid partition (tgp).

Example $d = 2$.

Number of cells = 4
Total grid partition 2
Cells are square

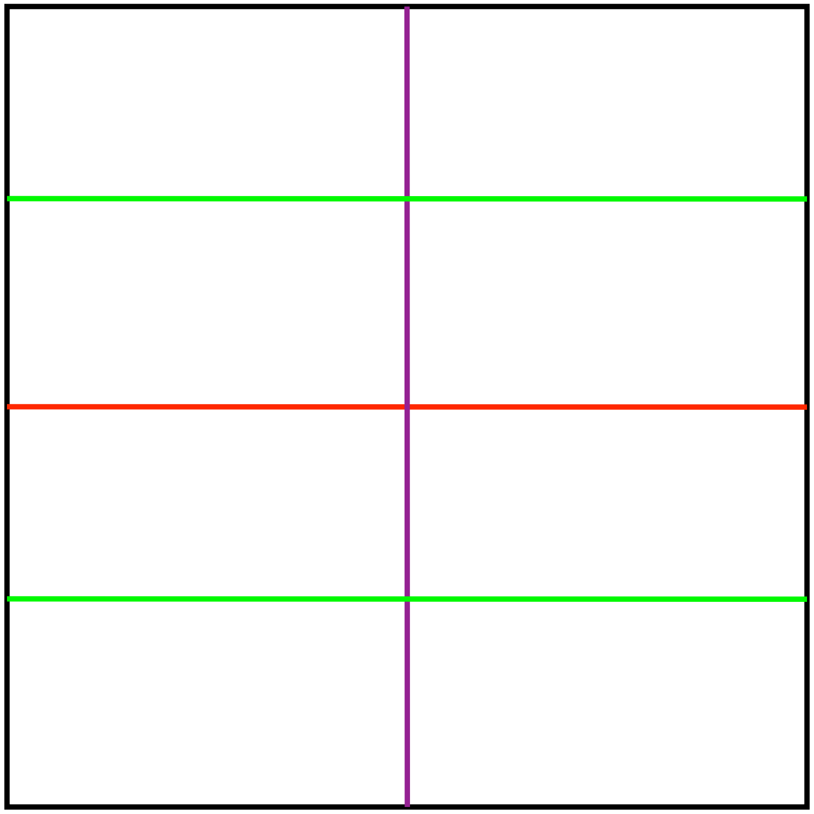


Total grid partition: number of cells is doubled

(But cells have same shape as original only every dth total grid partition (tgp).

Example $d = 2$.

Number of cells = 8
Total grid partition 3
Cells are not square

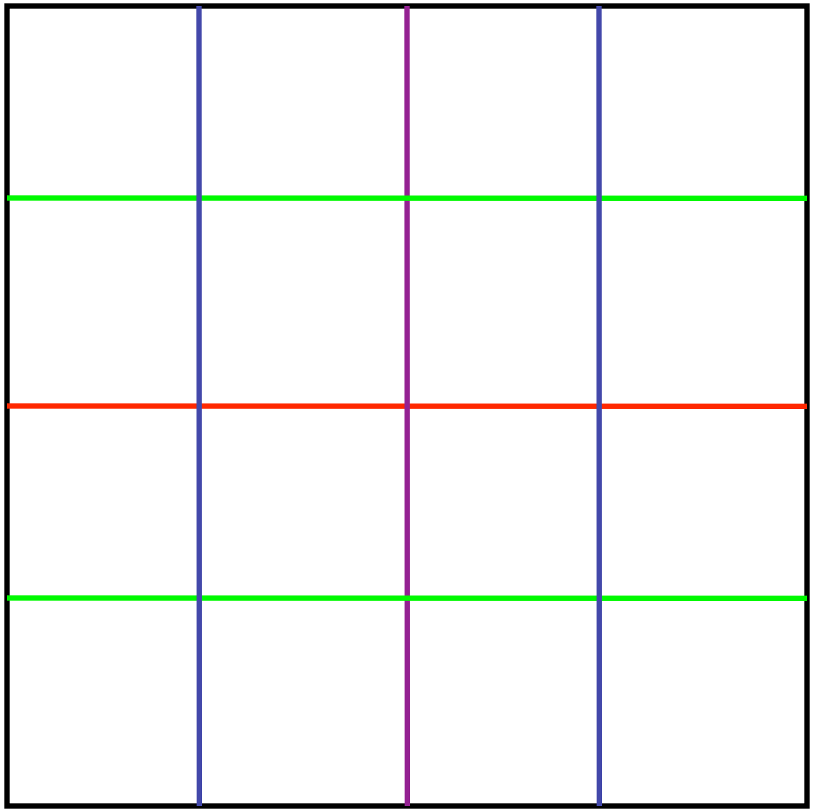


Total grid partition: number of cells is doubled

(But cells have same shape as original only every *d*th total grid partition (tgp).

Example $d = 2$.

Number of cells = 16
Total grid partition 4
Cells are square



Bit Operations

Bit Concatonation: $(4, 3)_{10} = (100, 011)_2$

$(100011)_2 = (35)_{10}$

y-first Bit Interleaving: $(4, 3)_{10} = (100, 011)_2$

$(011010)_2 = (26)_{10}$

y-first Reversed Bit Interleaving: $(4, 3)_{10} = (100, 011)_2$

(reverse the previous one) $(010110)_2 = (22)_{10}$

Outline

- Introduction, some basic principles
- Linear Hashing with Reversed Bit Interleaving (LHRBI)
- Multidimensional Extendible Hashing (MDEH)
- Comparison of LHRBI and MDEH
- Quantile Hashing
- Piecewise Linear Order Preserving (PLOP)
- Dynamic Z Hashing
- Linear Hashing with Partial Expansions (LHPE)

LHRBI

MAIN POINTS:

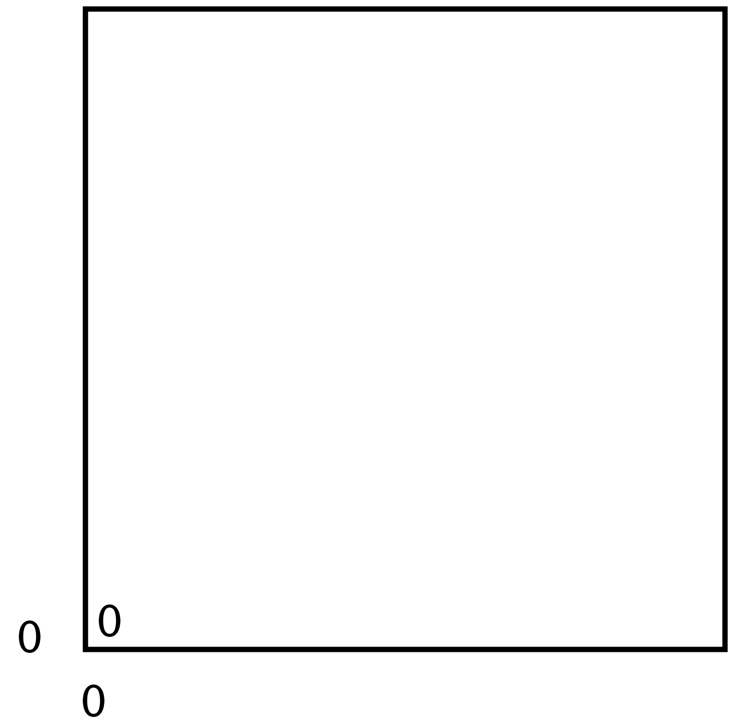
Decouple cell overflow from cell split by having chosen the order of splitting in advance.

Order preserving.

LHRBI

The Specifics:

1. Embedding space-based
2. Always split next smallest-numbered cell.
3. *y*-first reversed bit interleaving
   on partition indices
4. Count partition indices
   by distance from origin

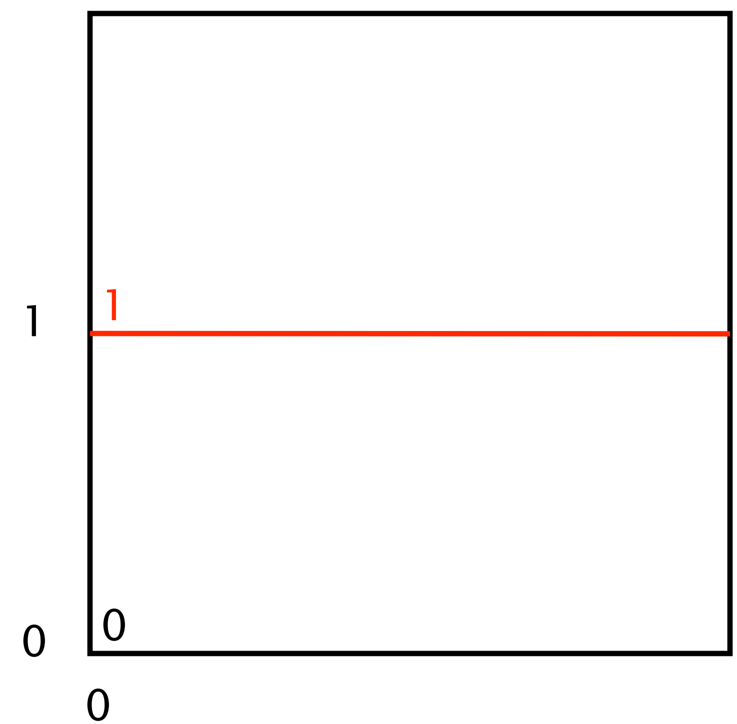


LHRBI

Key Points:

1. Order preserving, i.e., always split next smallest-numbered cell.
2. *y*-first reversed bit interleaving on partition indices
3. Count partition indices by distance from origin

Split cell 0

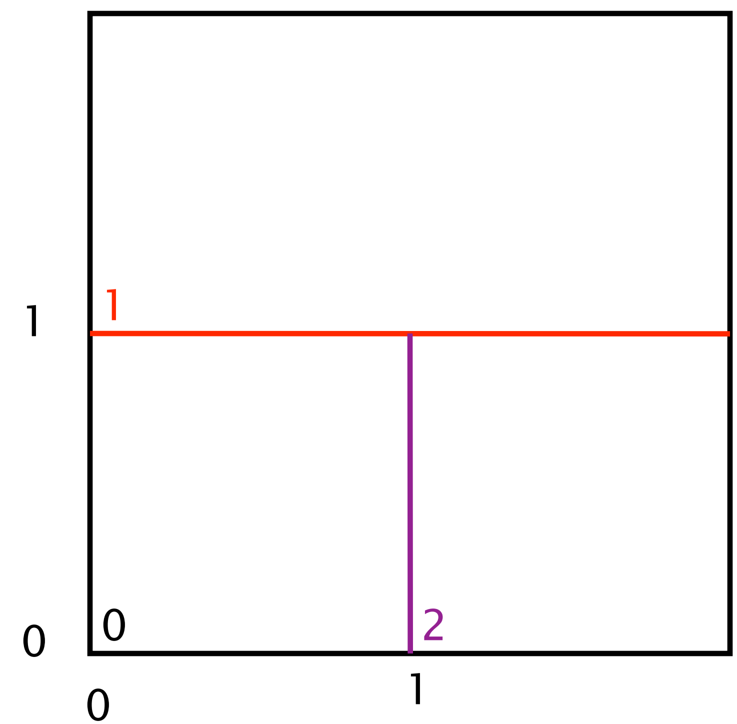


LHRBI

Key Points:

1. Order preserving, i.e., always split next smallest-numbered cell.
2. *y*-first reversed bit interleaving on partition indices
3. Count partition indices by distance from origin

Split cell 0



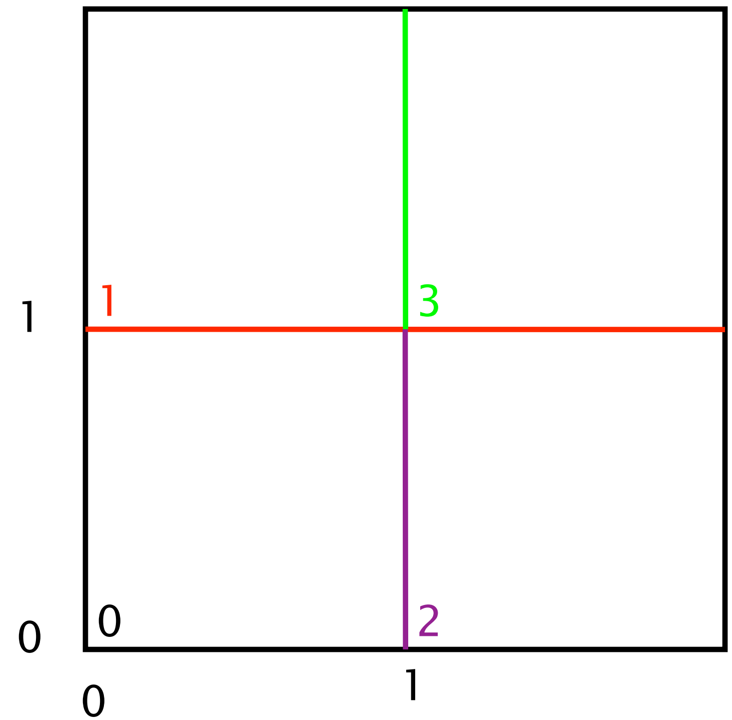
LHRBI

Key Points:

1. Order preserving, i.e., always split next smallest-numbered cell.
2. *y*-first reversed bit interleaving on partition indices
3. Count partition indices by distance from origin

Split cell 1

Finish total grid partition 2

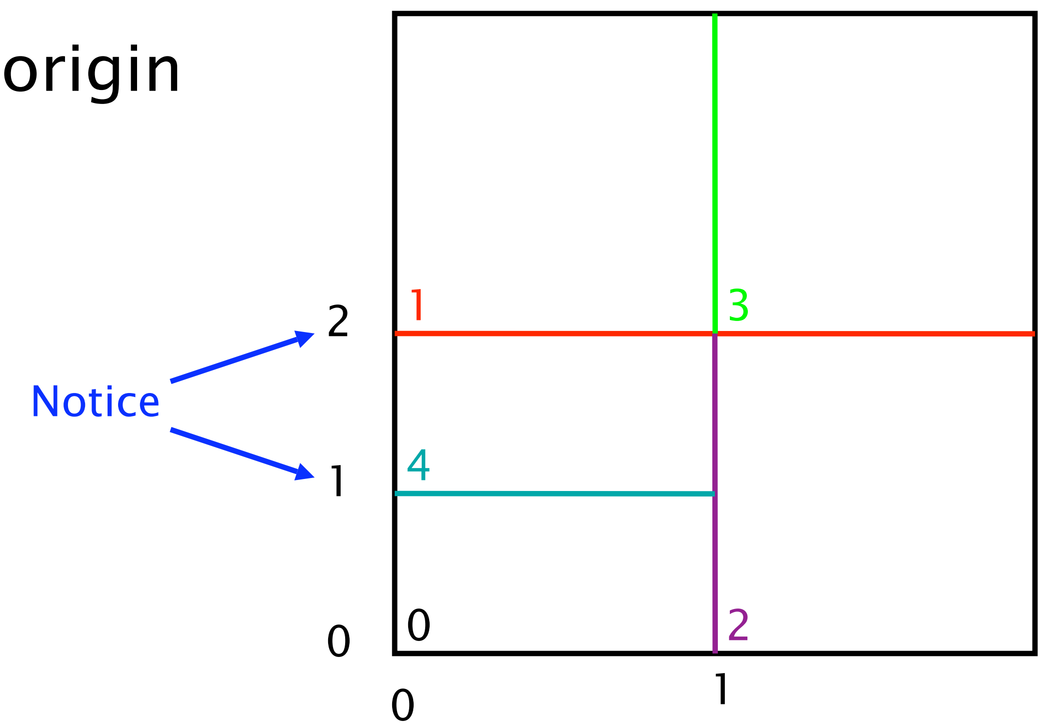


LHRBI

Key Points:

1. Order preserving, i.e., always split next smallest-numbered cell.
2. *y*-first reversed bit interleaving on partition indices
3. Count partition indices by distance from origin

Split cell 0

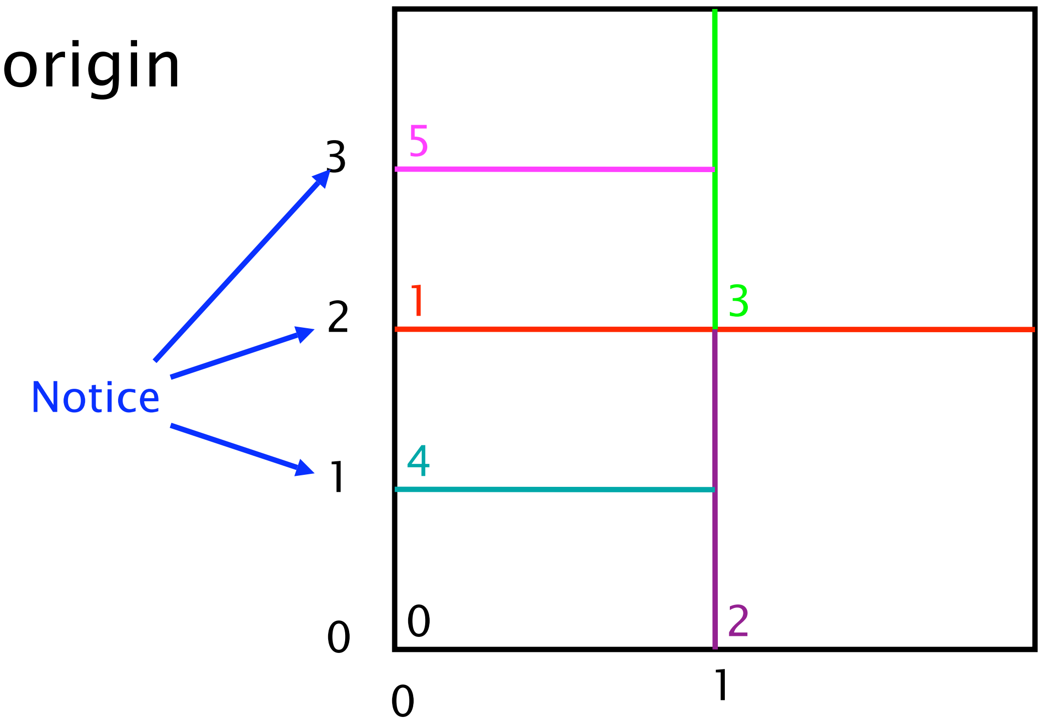


LHRBI

Key Points:

1. Order preserving, i.e., always split next smallest-numbered cell.
2. *y*-first reversed bit interleaving on partition indices
3. Count partition indices by distance from origin

Split cell 1

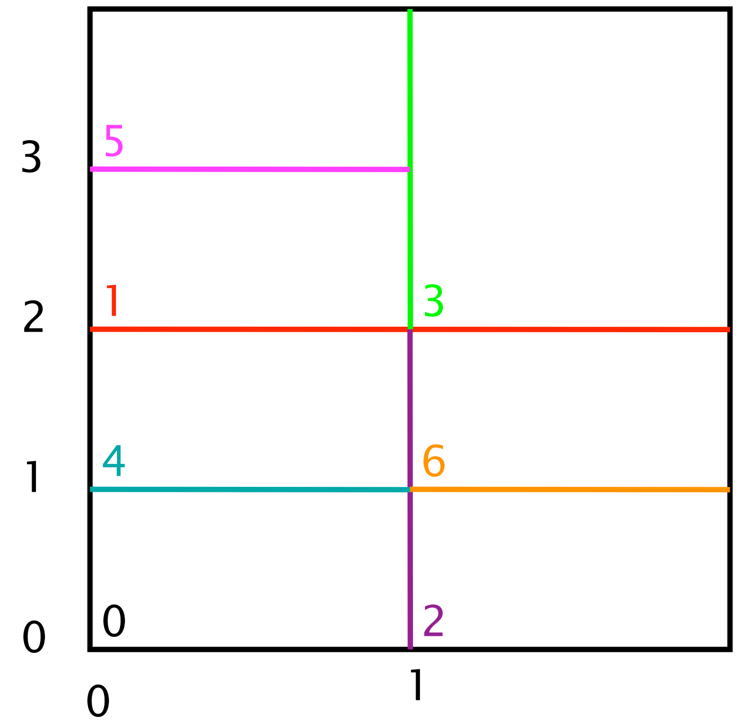


LHRBI

Key Points:

1. Order preserving, i.e., always split next smallest-numbered cell.
2. *y*-first reversed bit interleaving on partition indices
3. Count partition indices by distance from origin

Split cell 2



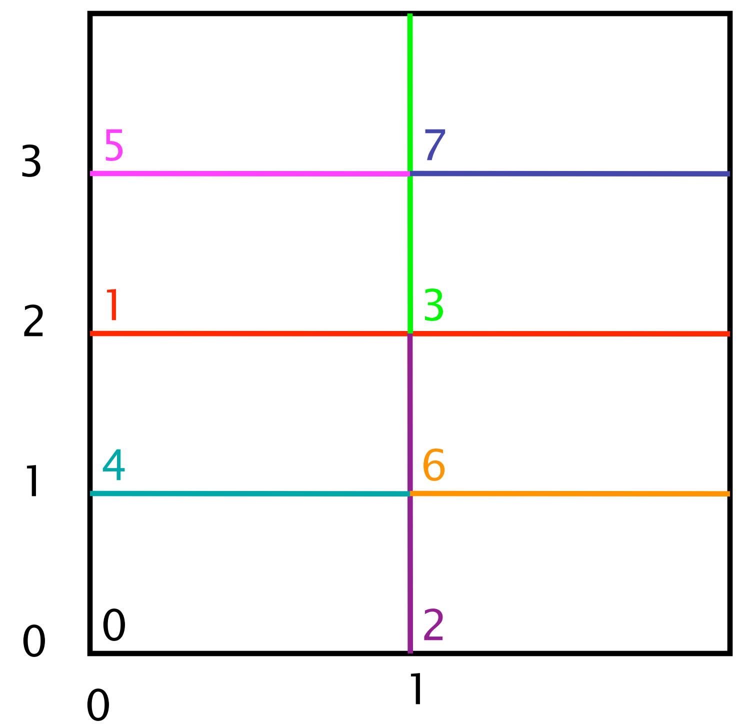
LHRBI

Key Points:

1. Order preserving, i.e., always split next smallest-numbered cell.
2. *y*-first reversed bit interleaving on partition indices
3. Count partition indices by distance from origin

Split cell 3

Finish total grid partition 3

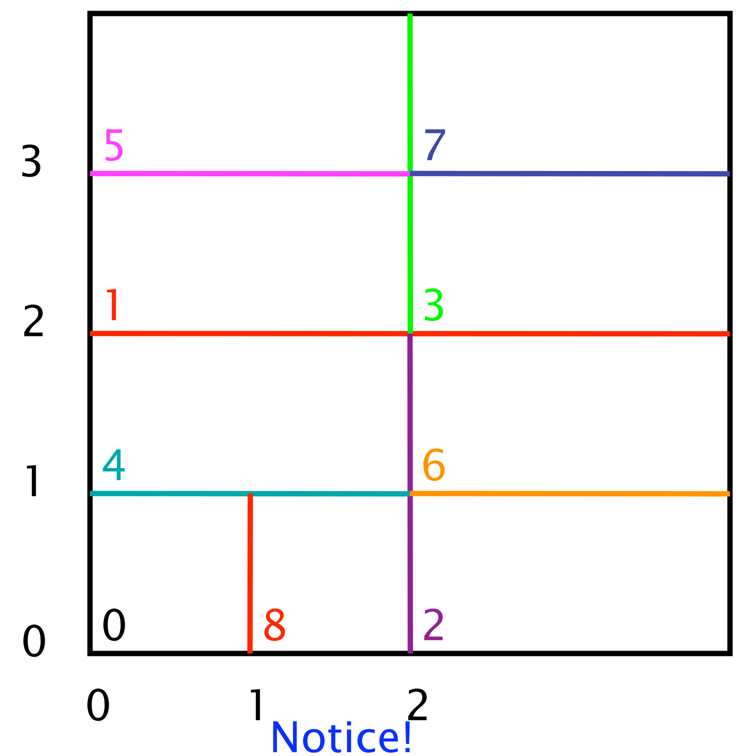


LHRBI

Key Points:

1. Order preserving, i.e., always split next smallest-numbered cell.
2. *y*-first reversed bit interleaving on partition indices
3. Count partition indices by distance from origin

Split cell 0

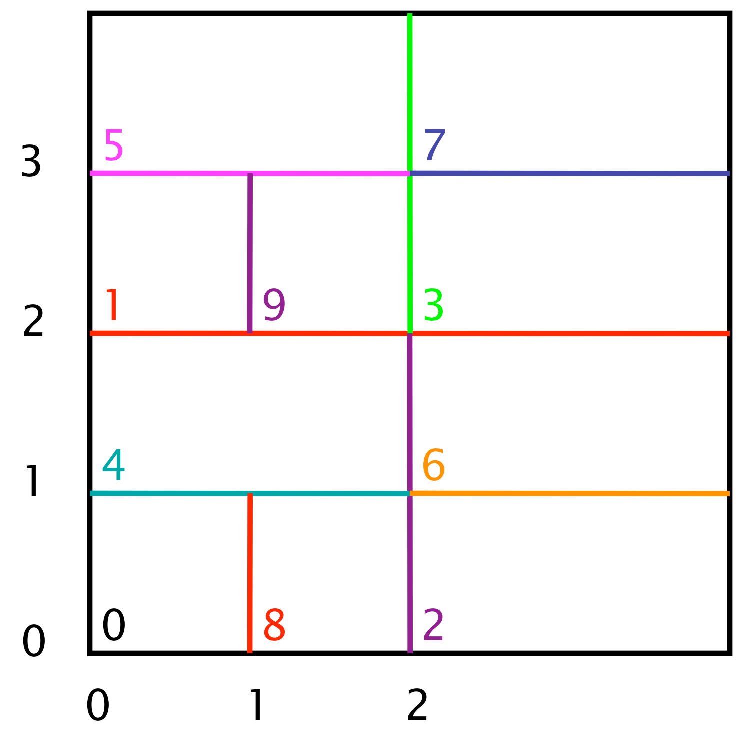


LHRBI

Key Points:

1. Order preserving, i.e., always split next smallest-numbered cell.
2. *y*-first reversed bit interleaving on partition indices
3. Count partition indices by distance from origin

Split cell 1

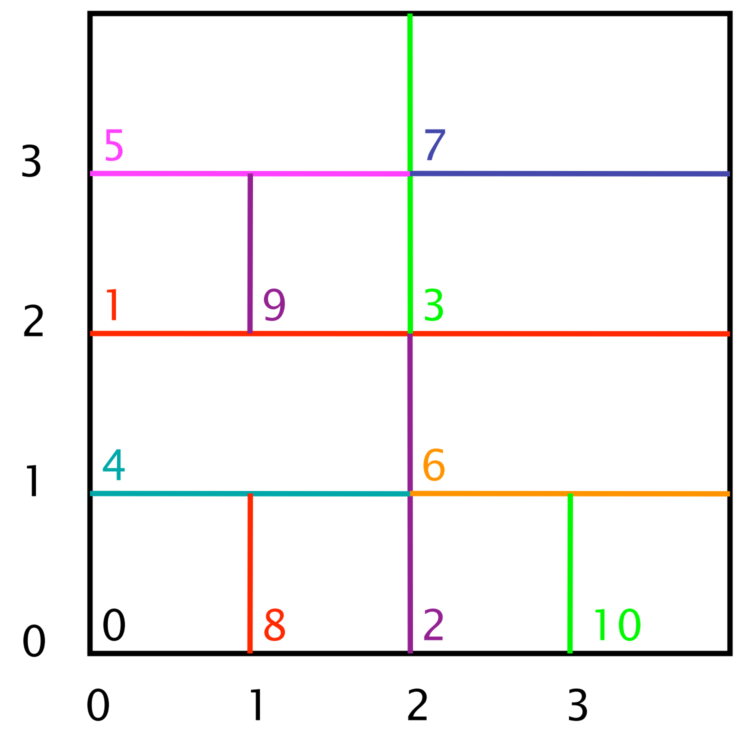


LHRBI

Key Points:

1. Order preserving, i.e., always split next smallest-numbered cell.
2. *y*-first reversed bit interleaving on partition indices
3. Count partition indices by distance from origin

Split cell 2

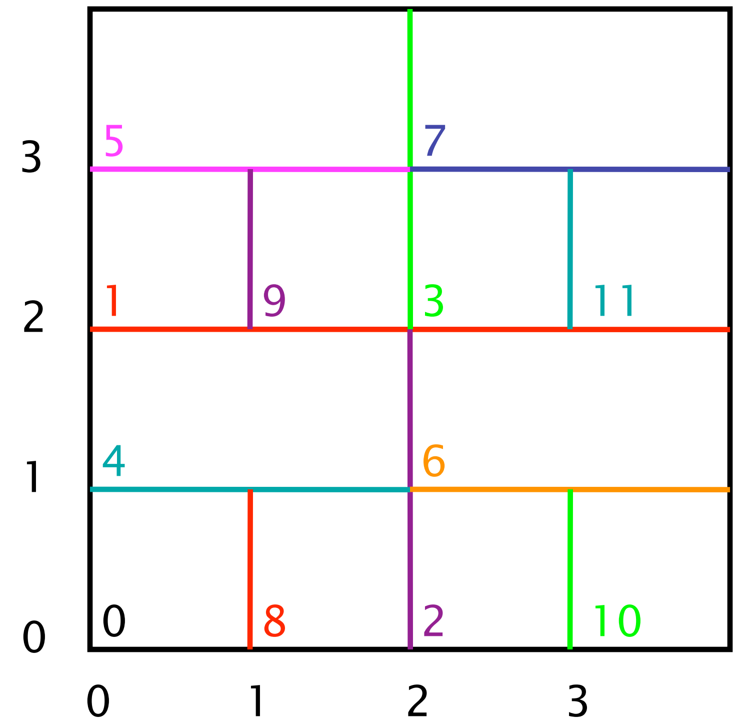


LHRBI

Key Points:

1. Order preserving, i.e., always split next smallest-numbered cell.
2. *y*-first reversed bit interleaving on partition indices
3. Count partition indices by distance from origin

Split cell 3

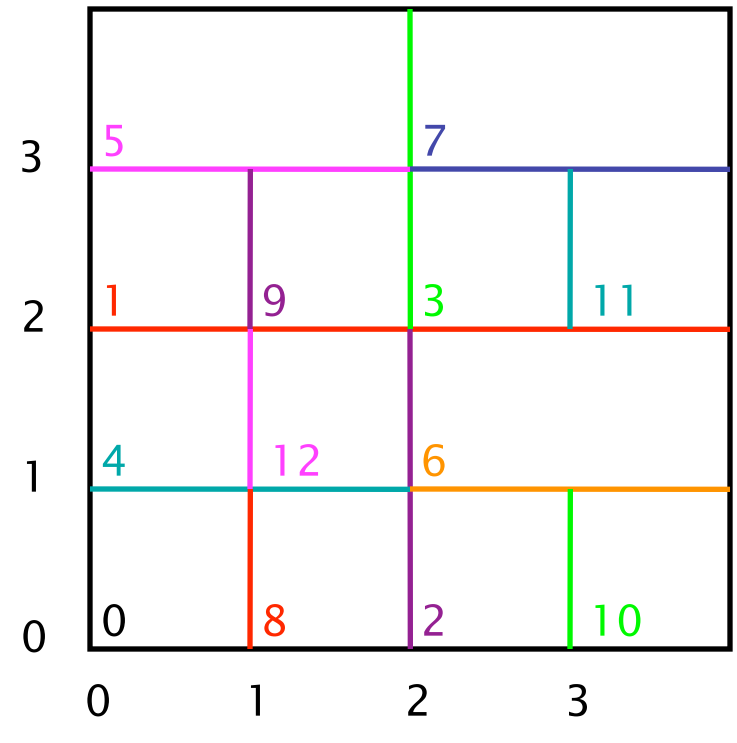


LHRBI

Key Points:

1. Order preserving, i.e., always split next smallest-numbered cell.
2. *y*-first reversed bit interleaving on partition indices
3. Count partition indices by distance from origin

Split cell 4

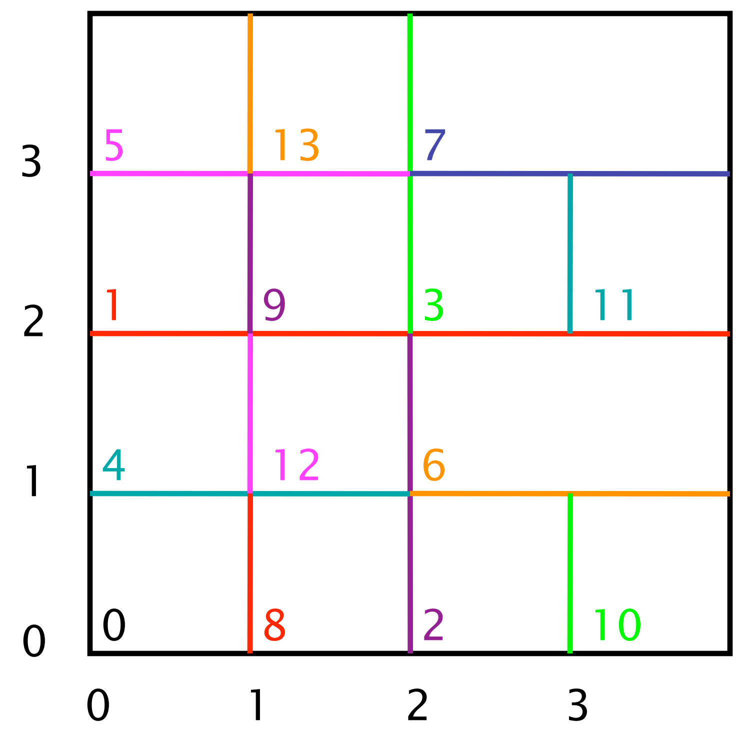


LHRBI

Key Points:

1. Order preserving, i.e., always split next smallest-numbered cell.
2. *y*-first reversed bit interleaving on partition indices
3. Count partition indices by distance from origin

Split cell 5

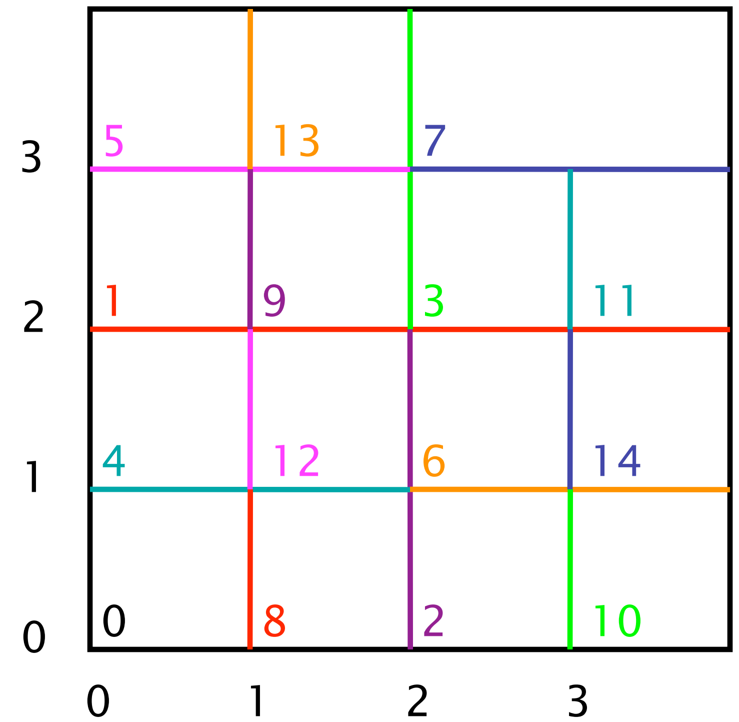


LHRBI

Key Points:

1. Order preserving, i.e., always split next smallest-numbered cell.
2. *y*-first reversed bit interleaving on partition indices
3. Count partition indices by distance from origin

Split cell 6



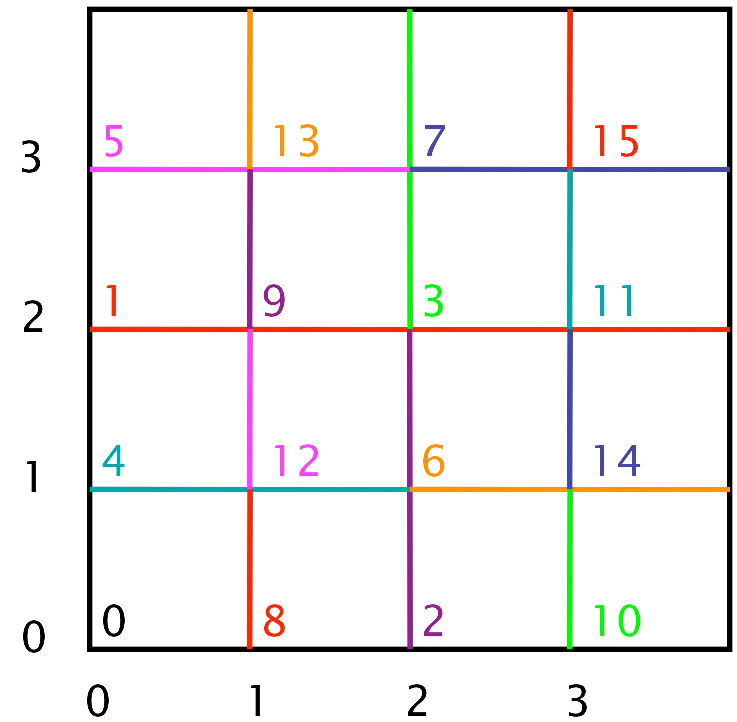
LHRBI

Key Points:

1. Order preserving, i.e., always split next smallest-numbered cell.
2. *y*-first reversed bit interleaving on partition indices
3. Count partition indices by distance from origin

Split cell 7

Total grid partition 4



LHRBI

Key Points:

1. Order preserving, i.e., always split next smallest-numbered cell.
2. *y*-first reversed bit interleaving on partition indices
3. Count partition indices by distance from origin

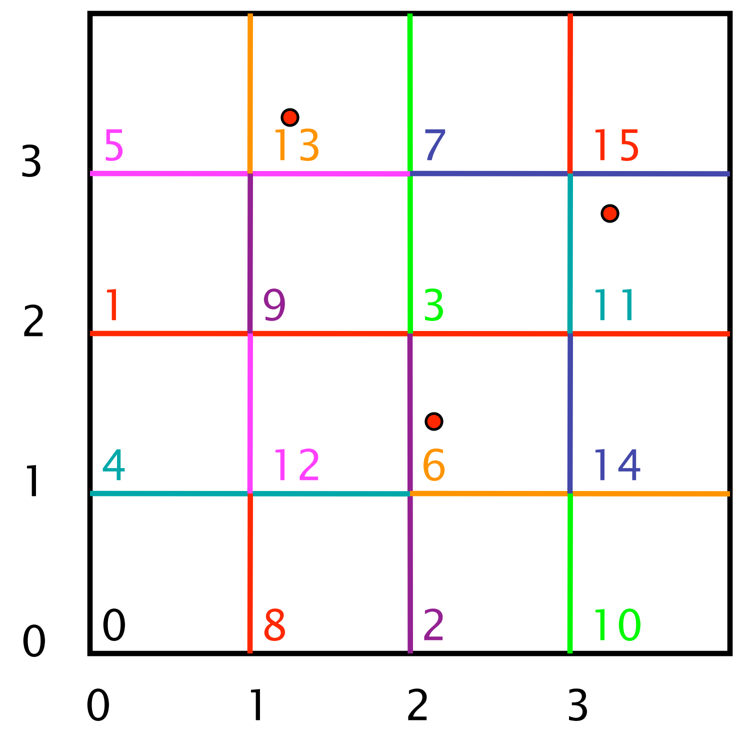
In what cell does (0.8, 0.7) lie?

x-partition index = $(3)_{10} = (11)_2$

y-partition index = $(2)_{10} = (10)_2$

Reverse bit interleave

$= (1011)_2 = (11)_{10}$



Outline

- Introduction, some basic principles
- Linear Hashing with Reversed Bit Interleaving (LHRBI)
- Multidimensional Extendible Hashing (MDEH)
- Comparison of LHRBI and MDEH
- Quantile Hashing
- Piecewise Linear Order Preserving (PLOP)
- Dynamic Z Hashing
- Linear Hashing with Partial Expansions (LHPE)

Multdimensional Extendible Hashing (MDEH)

MAIN POINTS:

Variant of Linear Hashing

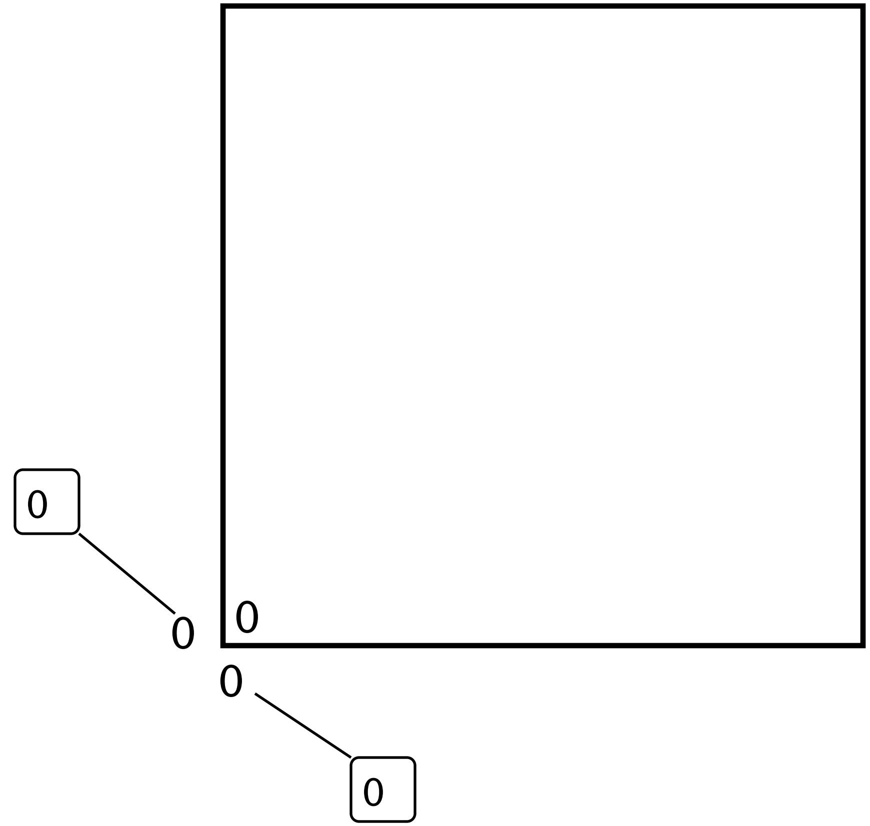
Split a slice completely before any cell
in another slice.

Multdimensional Extendible Hashing (MDEH)

Properties:

1. Linear hashing, split one cell at a time.
2. Complete one slice before starting the next.
3. Complete splitting along an axis before starting next.
4. Cycle through axes.
5. Embedding space based decomposition.
6. Split based on storage utilization factor threshold.
7. Partition Indices don’t change from order of creation.
8. Use Linear Scales (tries) to track partition indices.
9. An *Expansion Index* points to next cell to be split.

Linear scale on y-axis

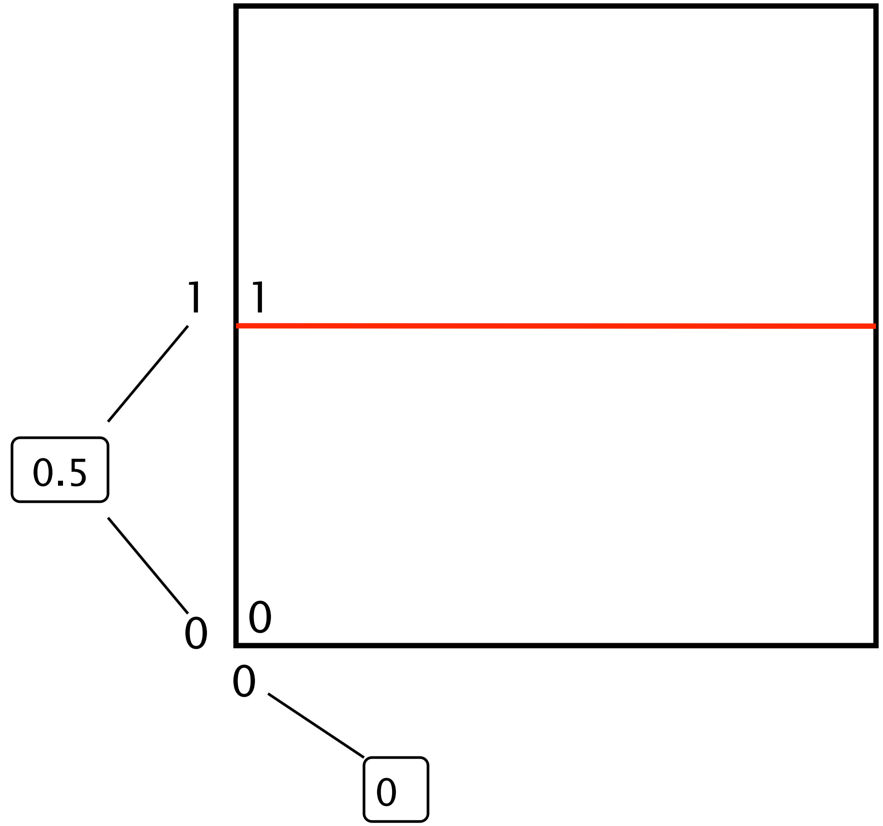


Expansion index $(e_y, e_x) = (0, 0)$

Next to be split is cell 0.

Linear scale on x-axis

Linear scale on y-axis



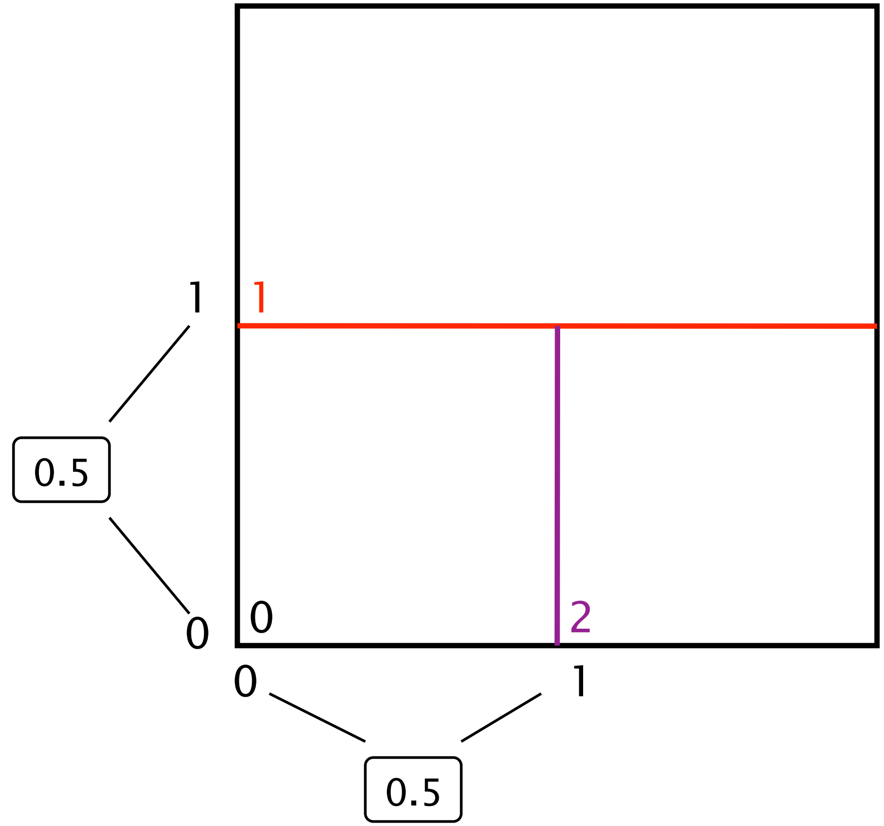
Expansion index $(e_y, e_x) = (0, 0)$

Next to be split is cell 0.

Linear scale on x-axis

MDEH

Linear scale on *y*-axis



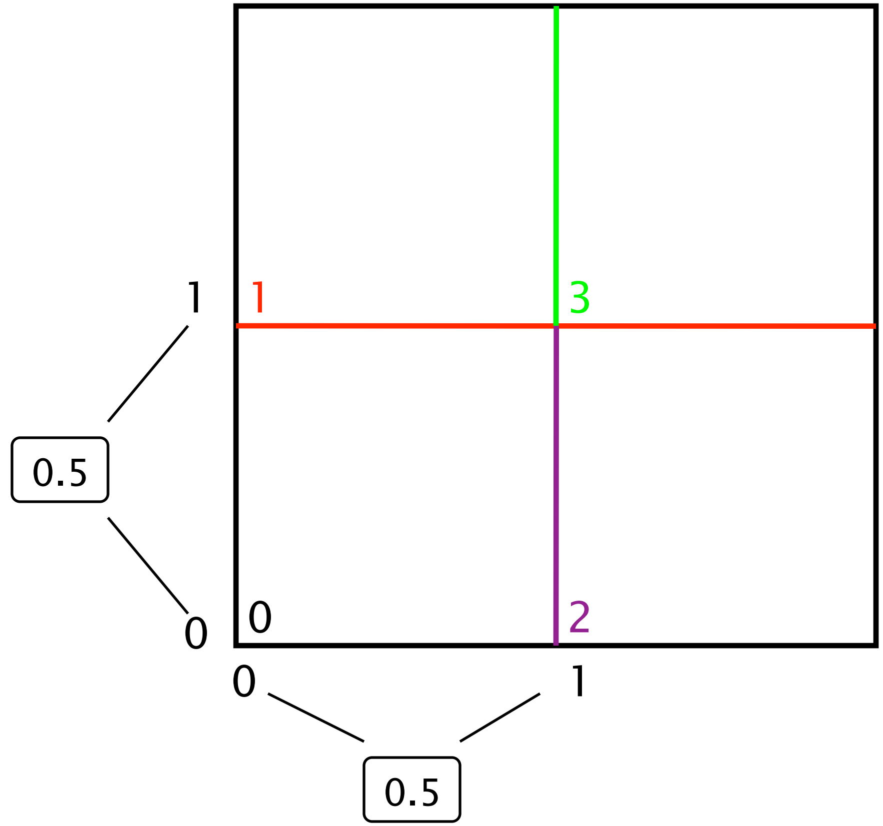
Expansion index $(e_y, e_x) = (1, 0)$

Next to be split is cell 1.

Linear scale on *x*-axis

MDEH

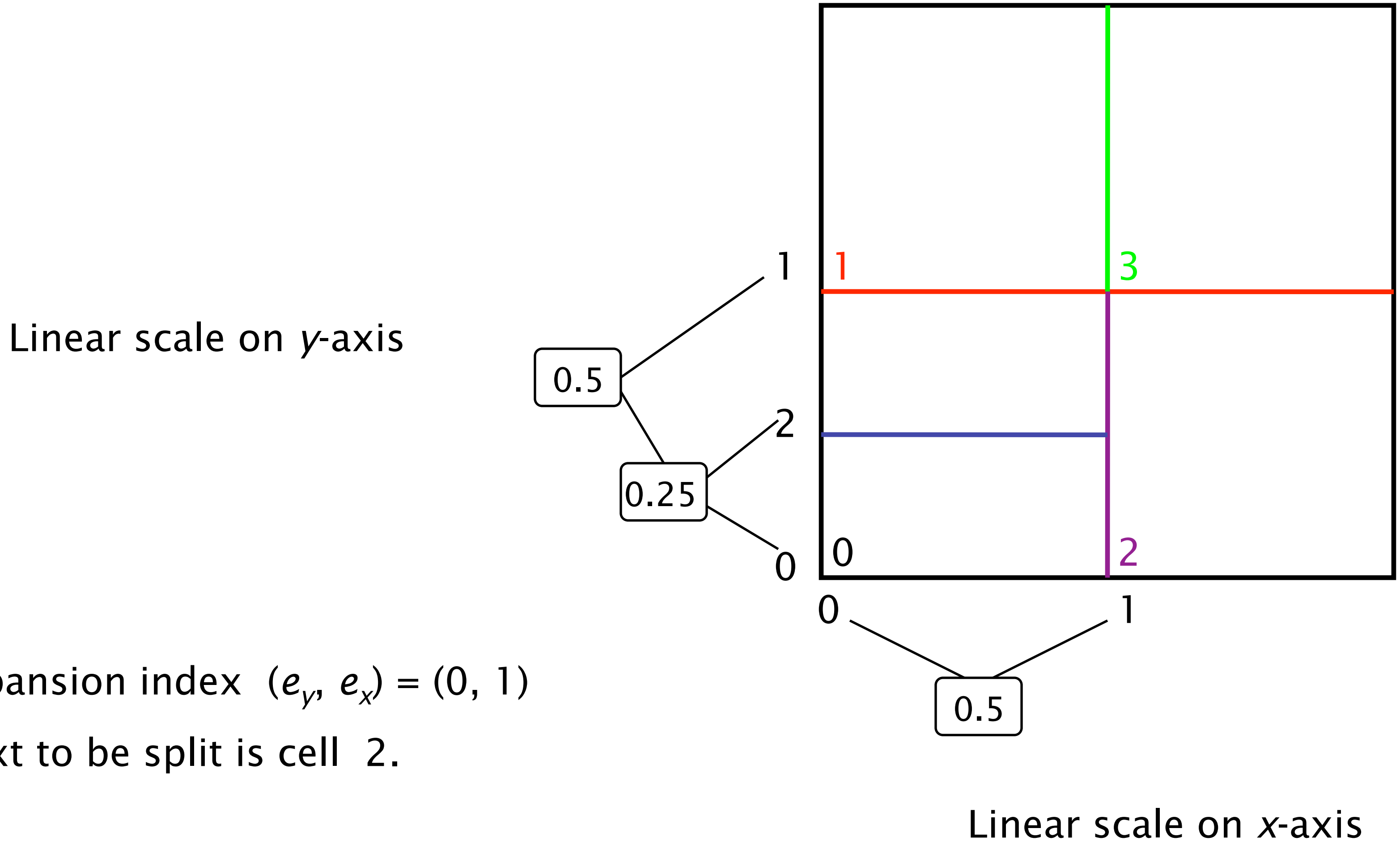
Linear scale on y-axis



Linear scale on x-axis

Expansion index $(e_y, e_x) = (0, 0)$

Next to be split is cell 0.

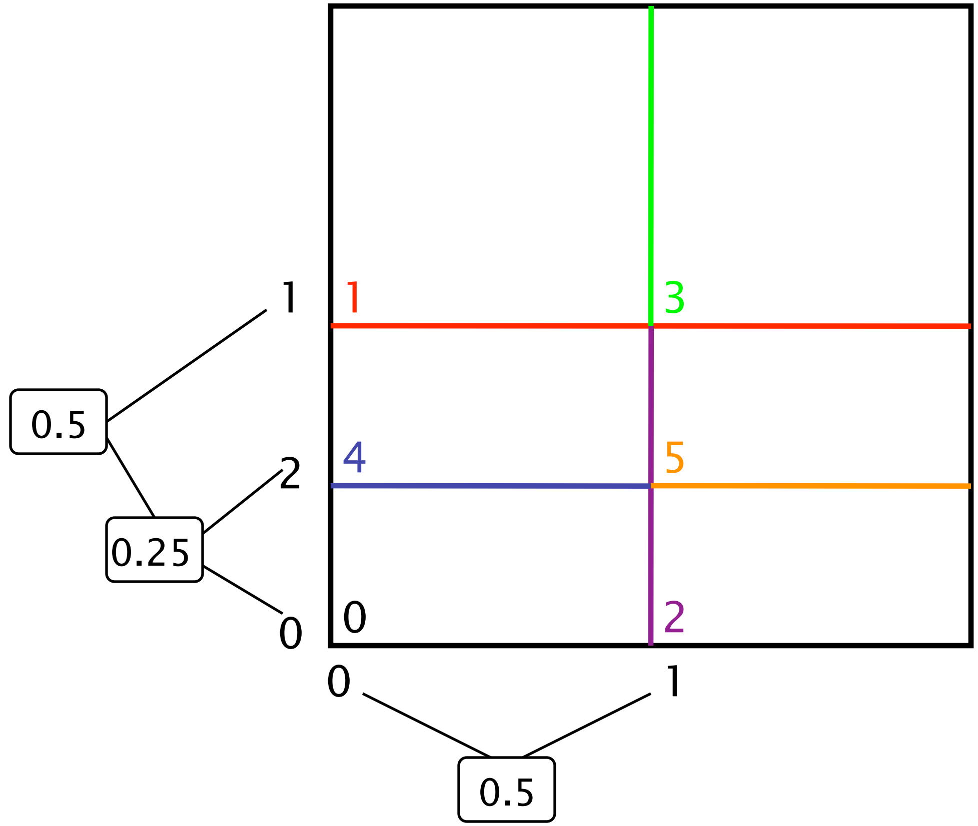


Expansion index $(e_y, e_x) = (0, 1)$

Next to be split is cell 2.

MDEH

Linear scale on y-axis



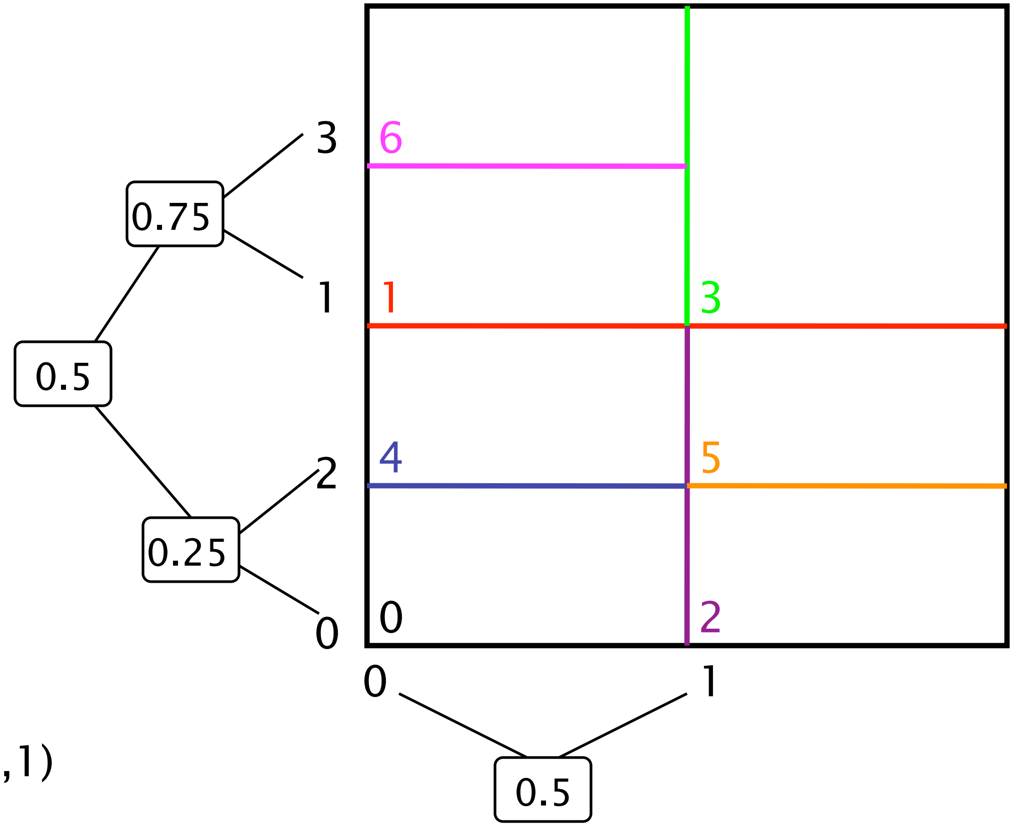
Expansion index $(e_y, e_x) = (1,0)$

Next to be split is cell 1.

Linear scale on x-axis

MDEH

Linear scale on y-axis



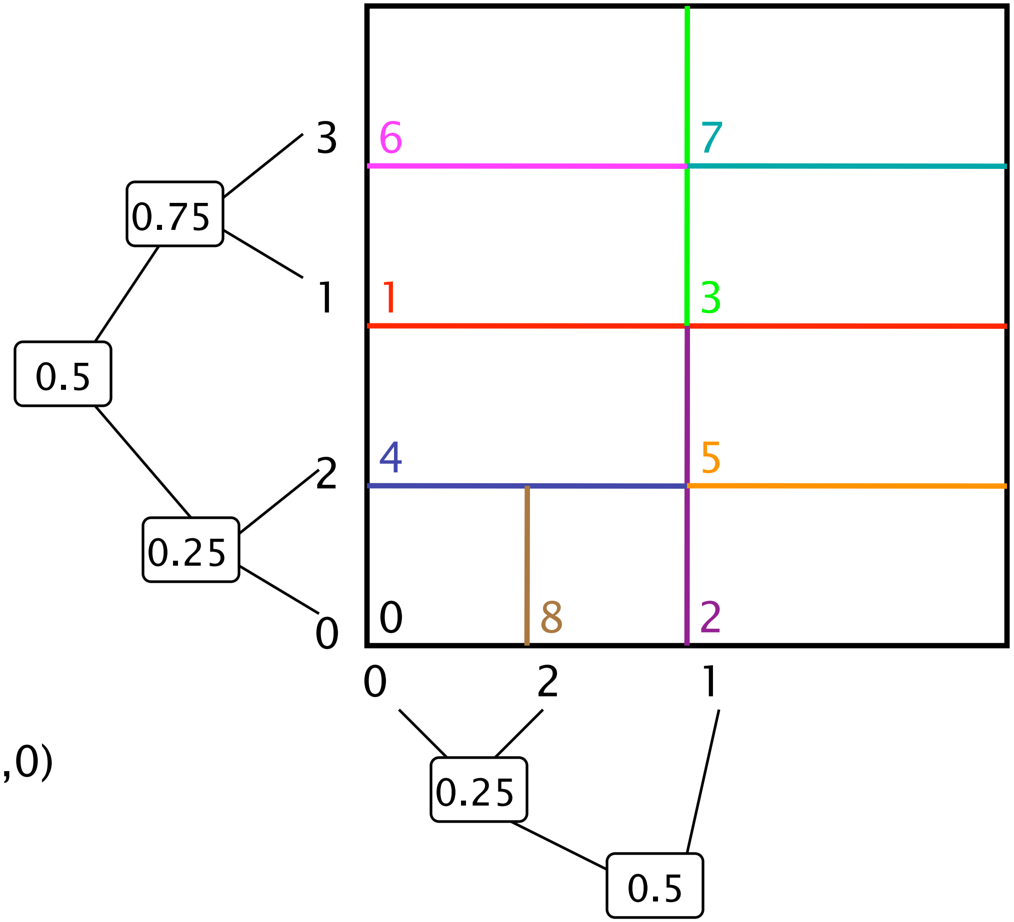
Expansion index $(e_y, e_x) = (01,1)$

Next to be split is cell 3.

Linear scale on x-axis

MDEH

Linear scale on y-axis



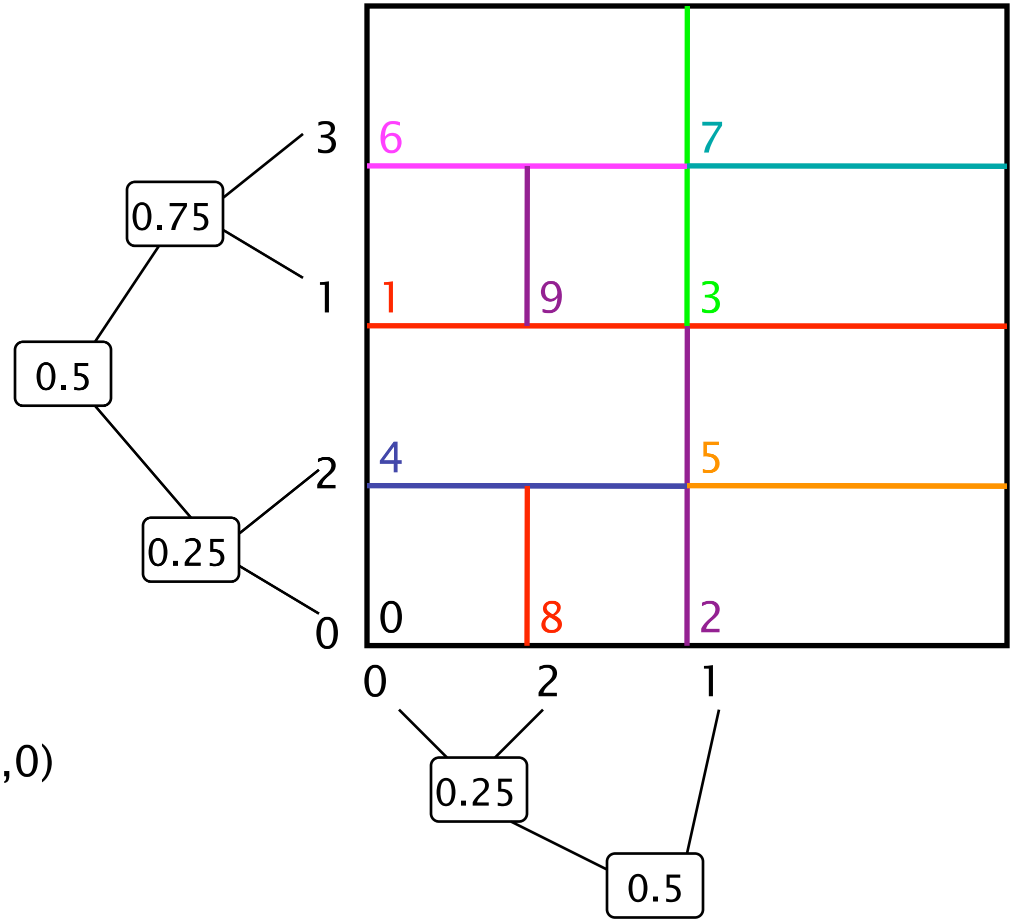
Expansion index $(e_y, e_x) = (01,0)$

Next to be split is cell 1.

Linear scale on x-axis

MDEH

Linear scale on y-axis



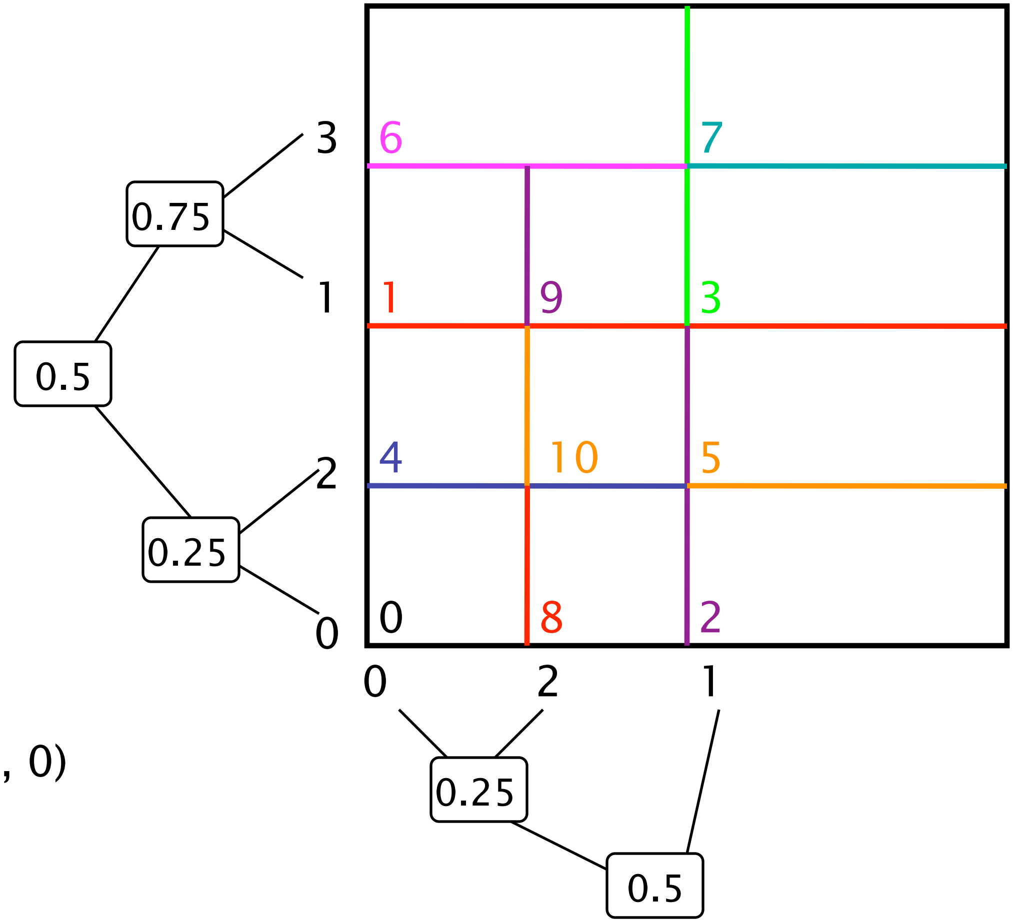
Expansion index $(e_y, e_x) = (10,0)$

Next to be split is cell 4.

Linear scale on x-axis

MDEH

Linear scale on y-axis



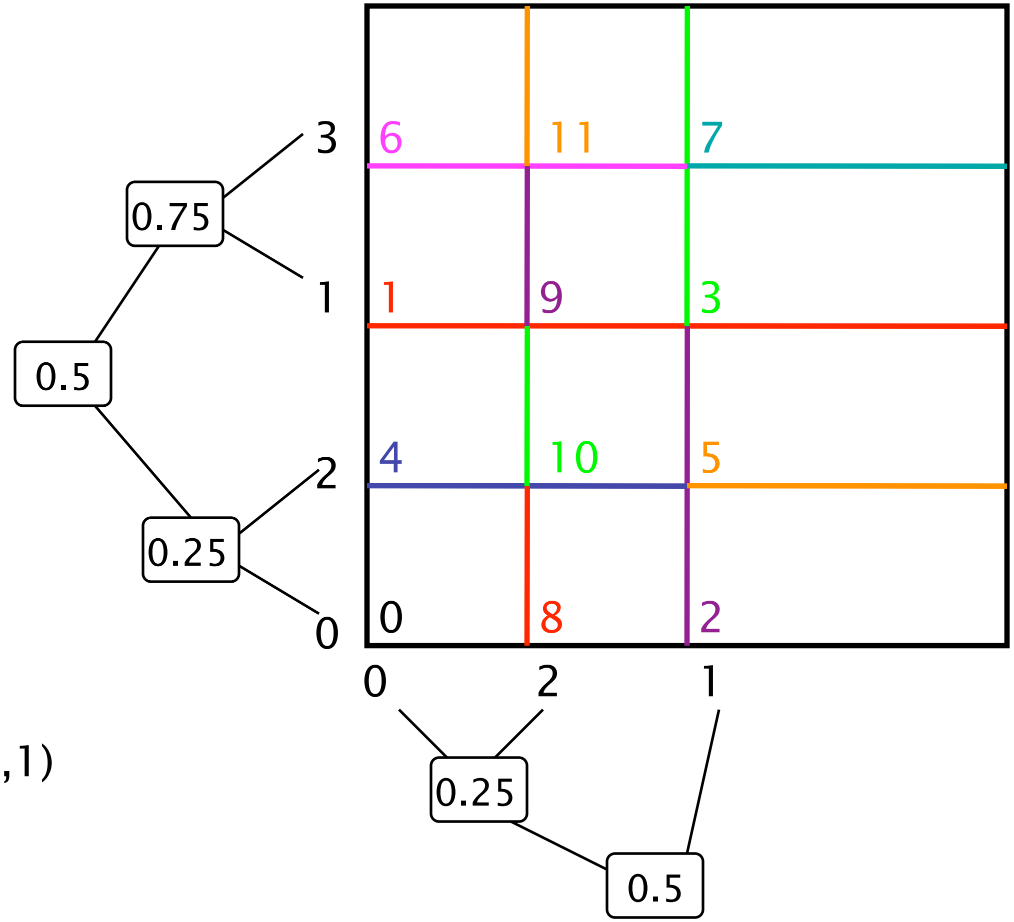
Expansion index $(e_y, e_x) = (11, 0)$

Next to be split is cell 6.

Linear scale on x-axis

MDEH

Linear scale on *y*-axis



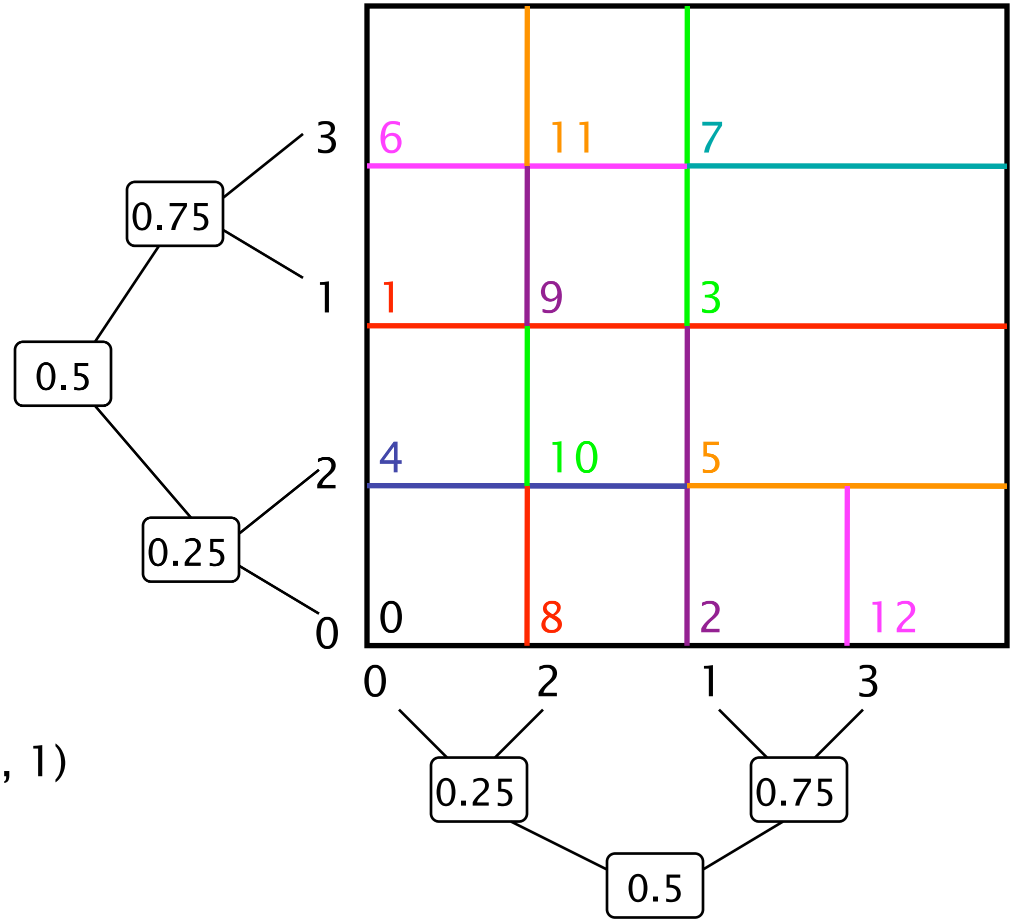
Linear scale on *x*-axis

Expansion index $(e_y, e_x) = (00,1)$

Next to be split is cell 2.

MDEH

Linear scale on y-axis



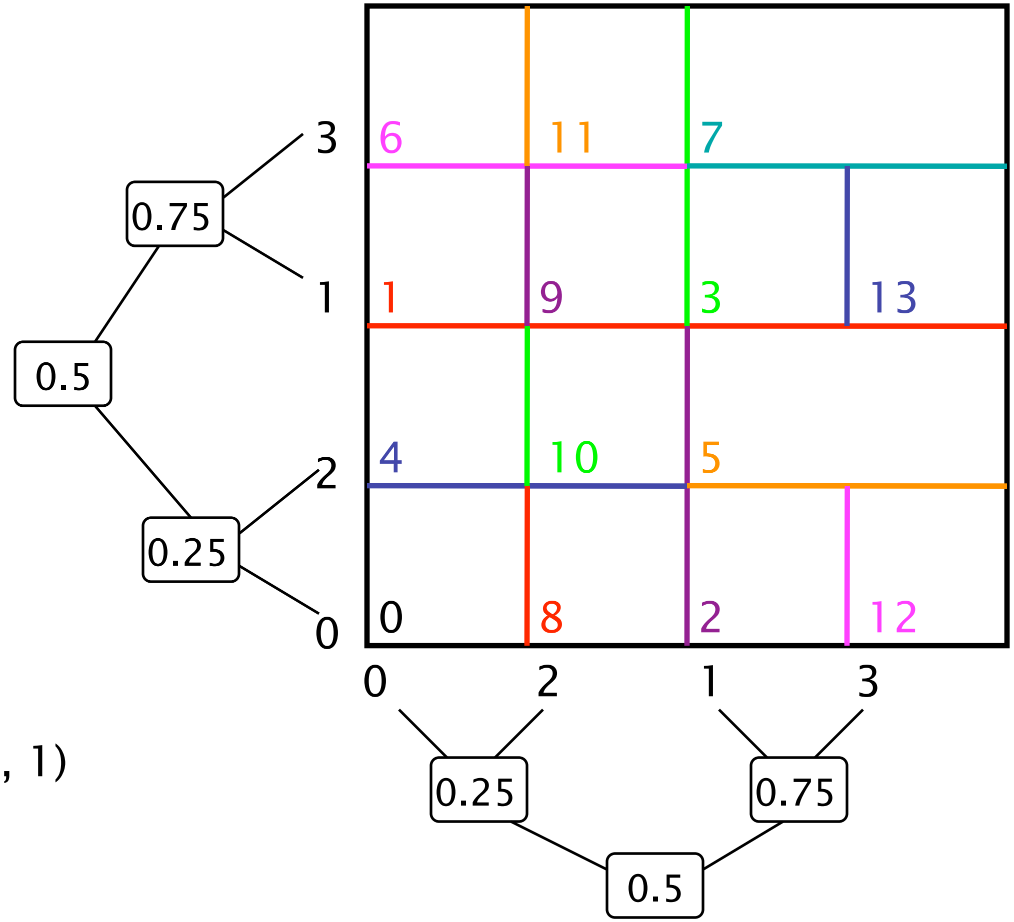
Expansion index $(e_y, e_x) = (01, 1)$

Next to be split is cell 3.

Linear scale on x-axis

MDEH

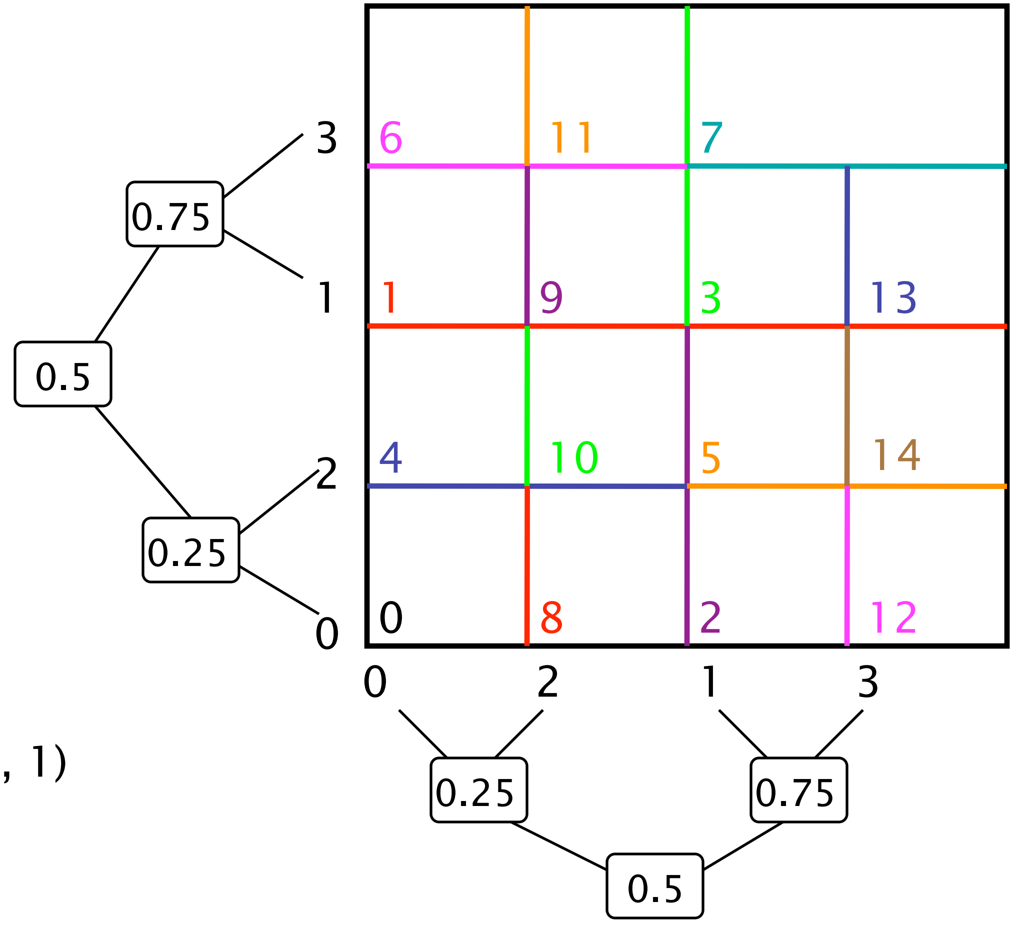
Linear scale on y-axis



Linear scale on x-axis

Expansion index $(e_y, e_x) = (10, 1)$

Next to be split is cell 5.



Linear scale on *y*-axis

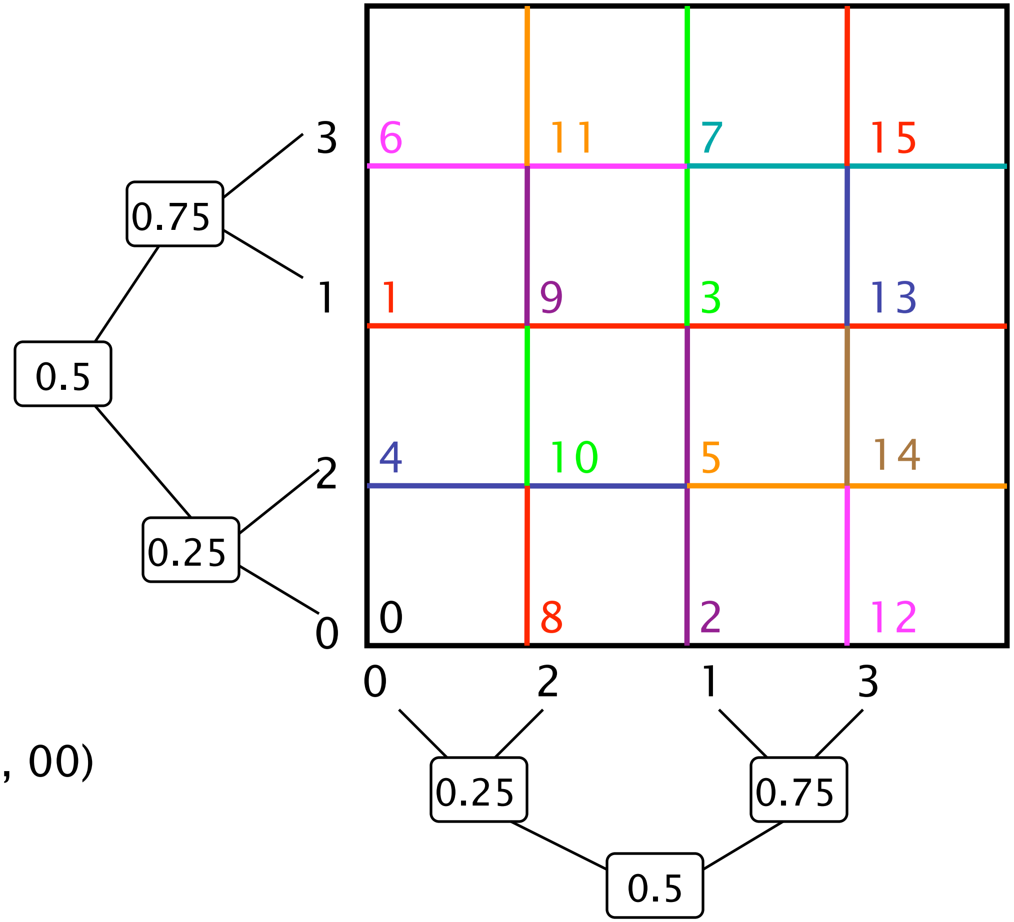
Linear scale on *x*-axis

Expansion index $(e_y, e_x) = (11, 1)$

Next to be split is cell 7.

MDEH

Linear scale on y-axis



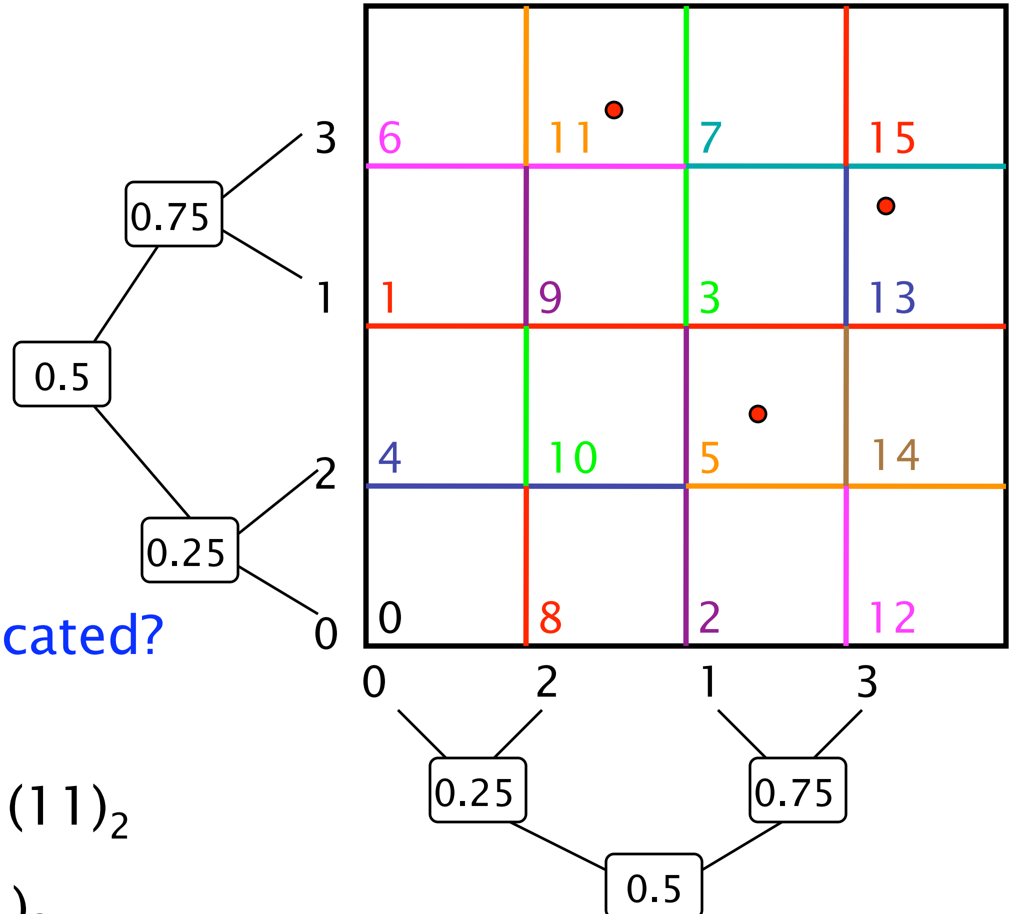
Expansion index $(e_y, e_x) = (00, 00)$

Next to be split is cell 0.

Linear scale on x-axis

MDEH

Linear scale on y-axis



Linear scale on x-axis

In what cell is (0.8, 0.7) located?

Linear scales lead to

x-partition index = $(3)_{10} = (11)_2$

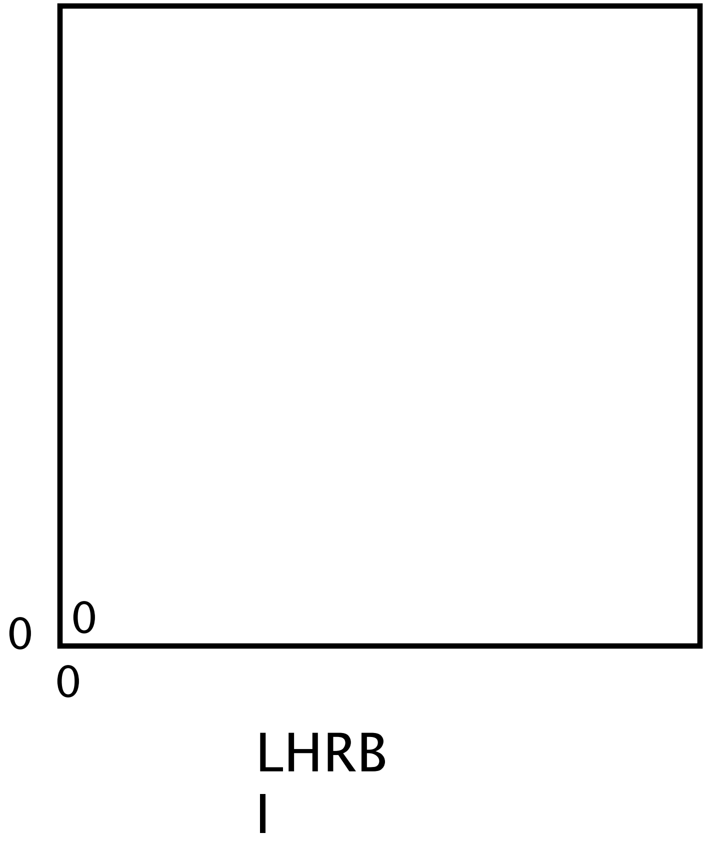
y-partition index = $1 = (01)_2$

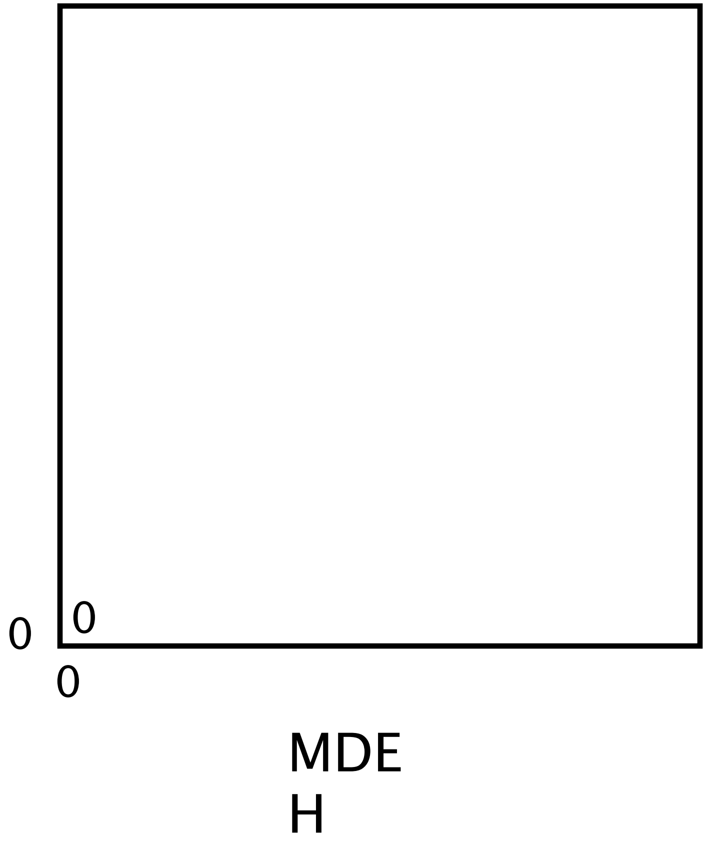
Concatonate: $(1101)_2 = (13)_{10}$

Outline

- Introduction, some basic principles
- Linear Hashing with Reversed Bit Interleaving (LHRBI)
- Multidimensional Extendible Hashing (MDEH)
- Comparison of LHRBI and MDEH
- Quantile Hashing
- Piecewise Linear Order Preserving (PLOP)
- Dynamic Z Hashing
- Linear Hashing with Partial Expansions (LHPE)

LHRBI vs. MDEH

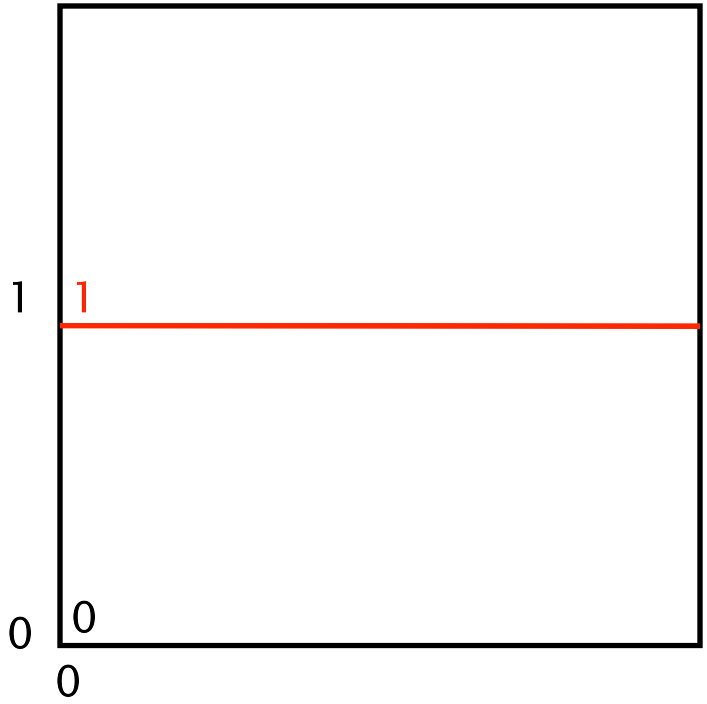


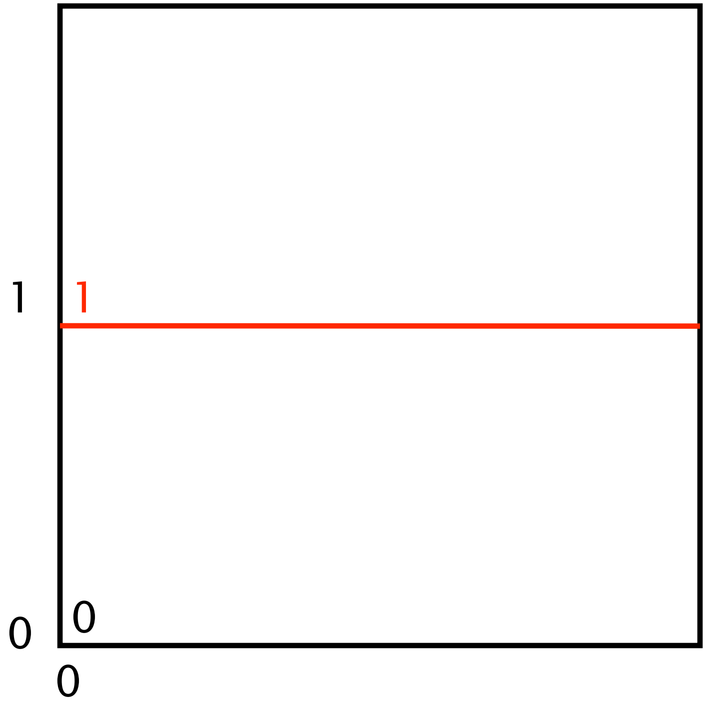


Key Point:

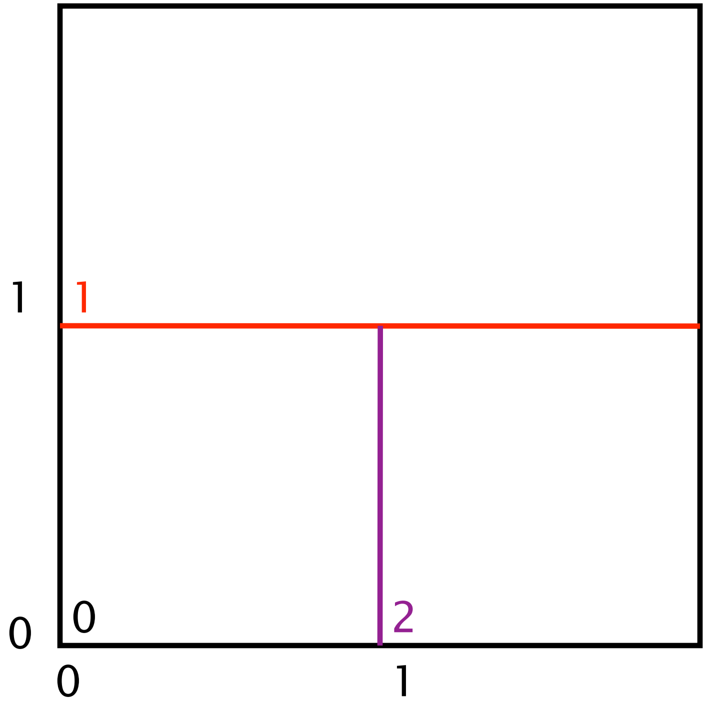
Differences due to choice of how to order cells/splittings.

LHRBI vs. MDEH

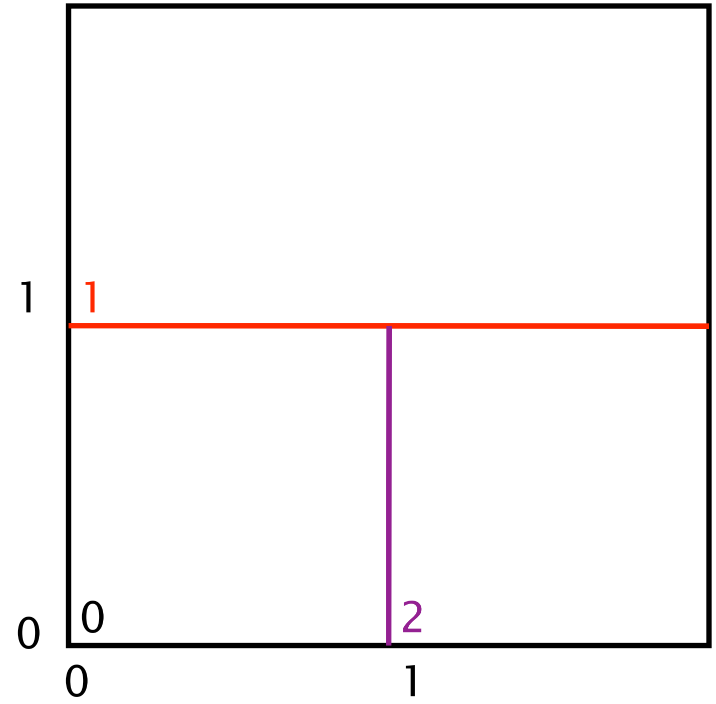




LHRBI vs. MDEH

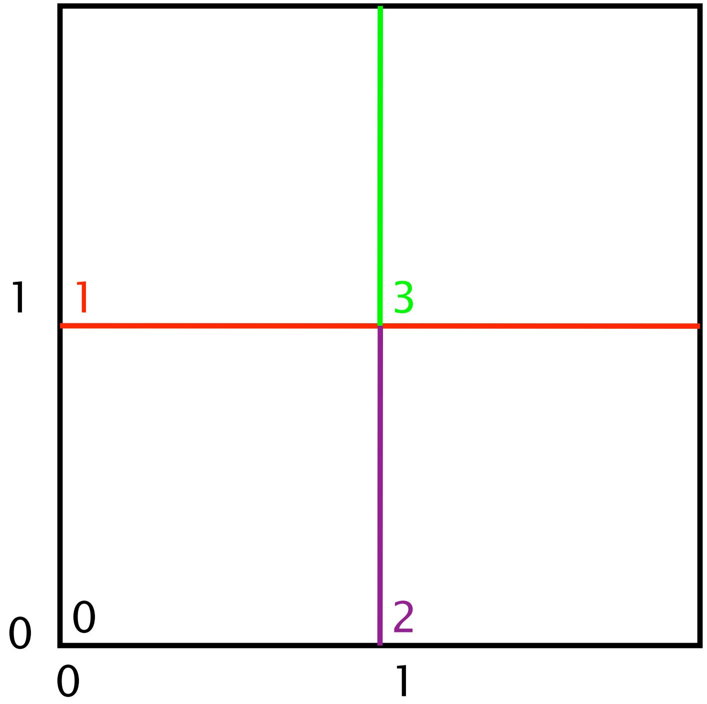


LHRB
I

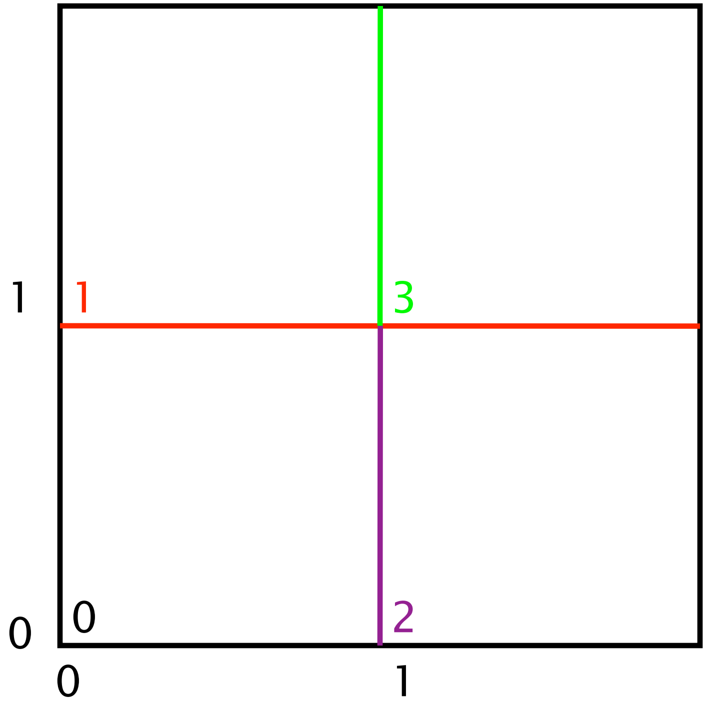


MDE
H

LHRBI vs. MDEH

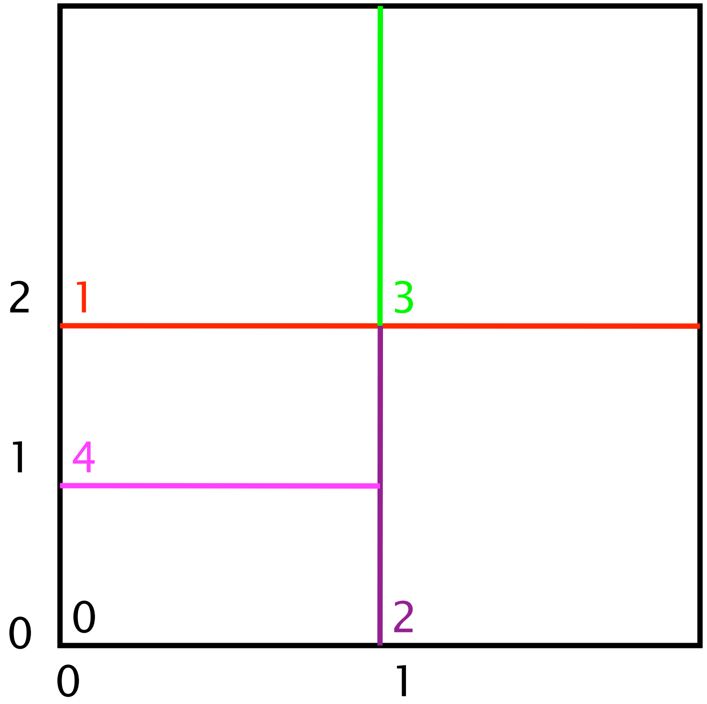


LHRB
I

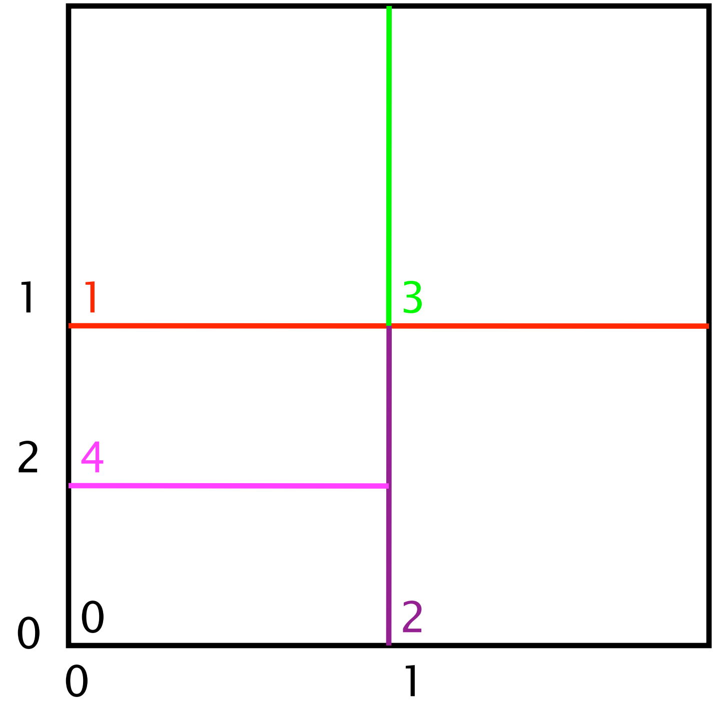


MDE
H

LHRBI vs. MDEH

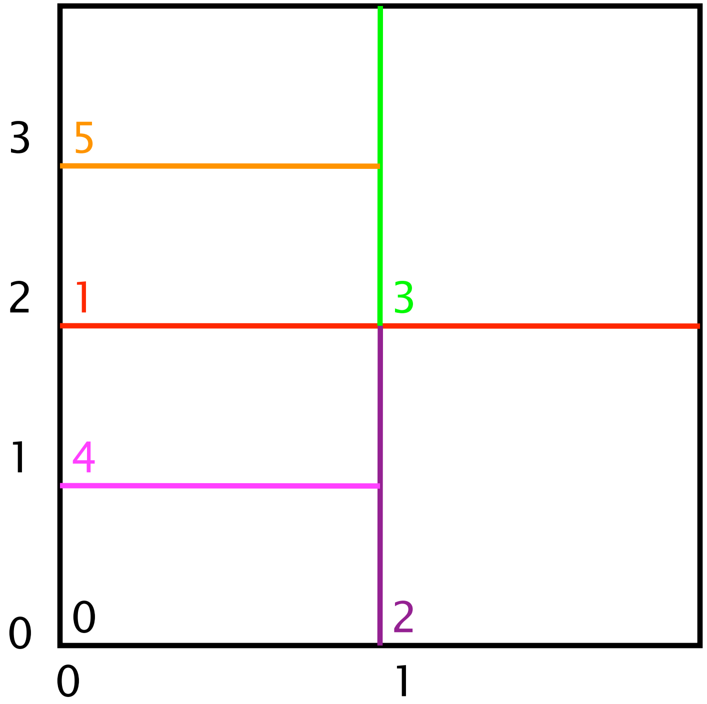


LHRB
I

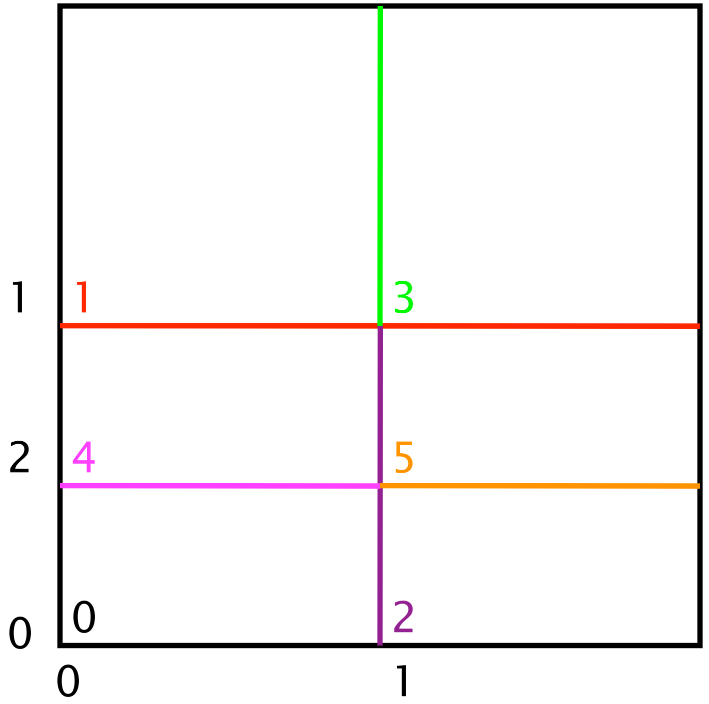


MDE
H

LHRBI vs. MDEH

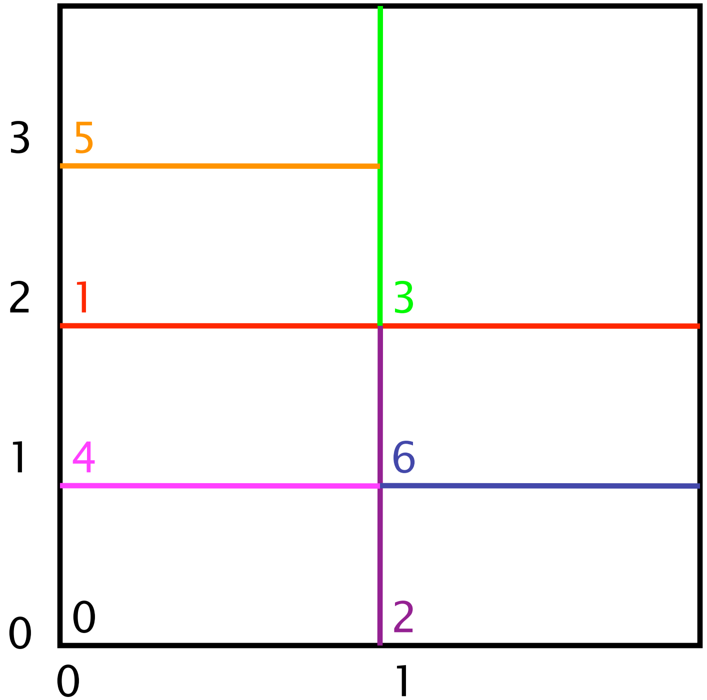


LHRB
I

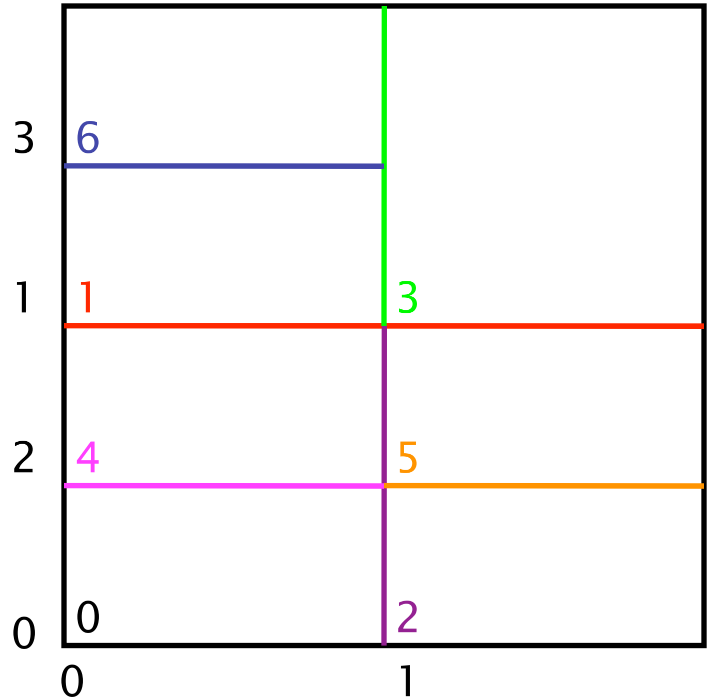


MDE
H

LHRBI vs. MDEH

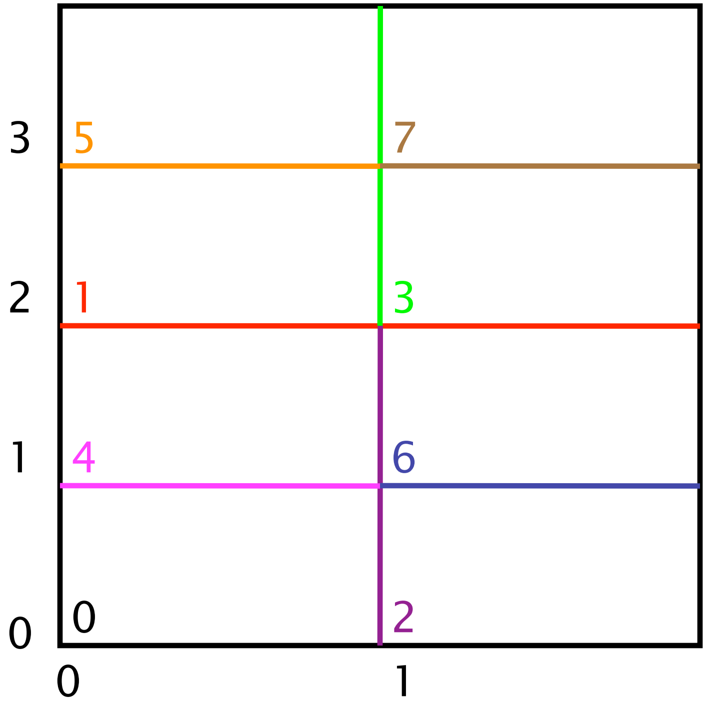


LHRB
I

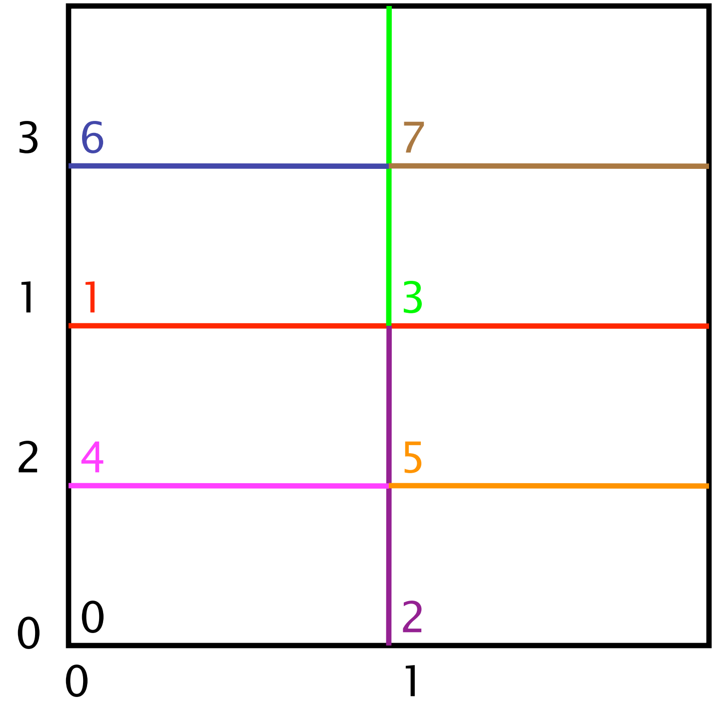


MDE
H

LHRBI vs. MDEH

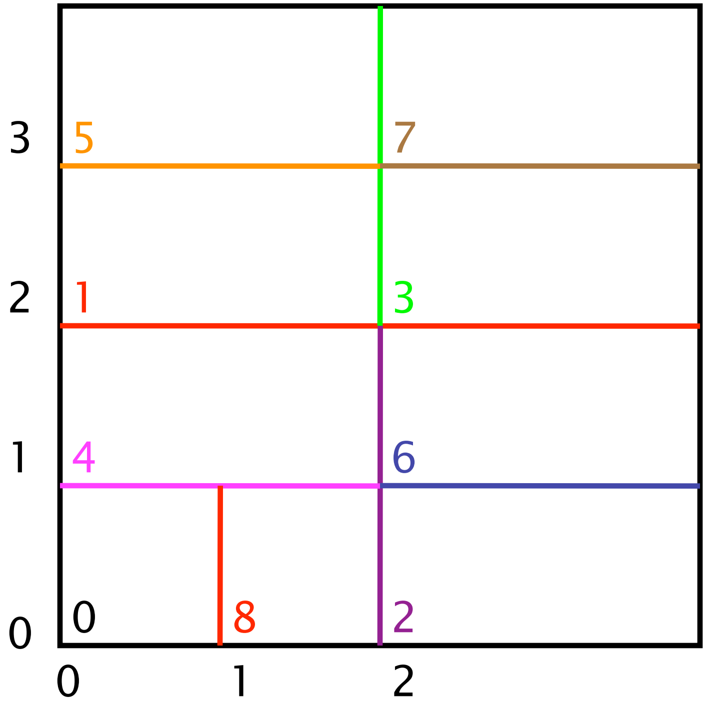


LHRB
I

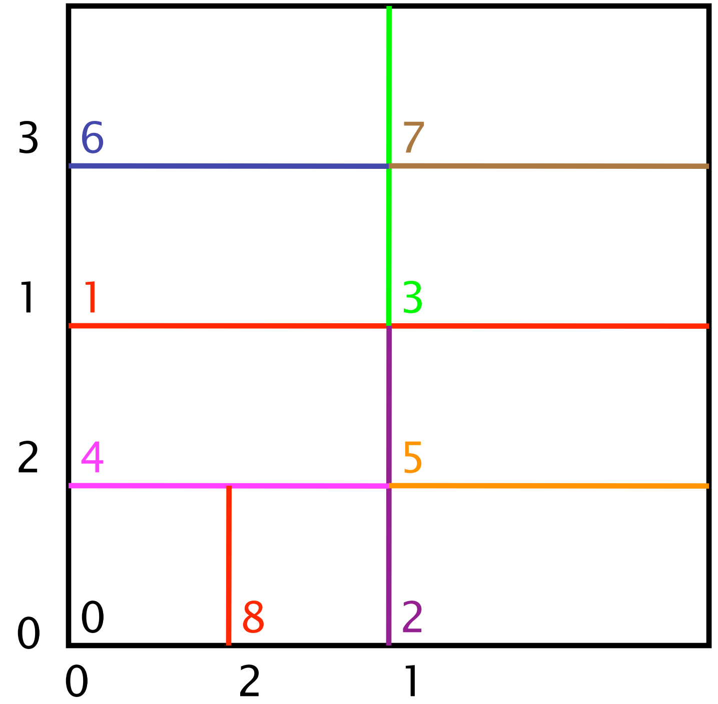


MDE
H

LHRBI vs. MDEH

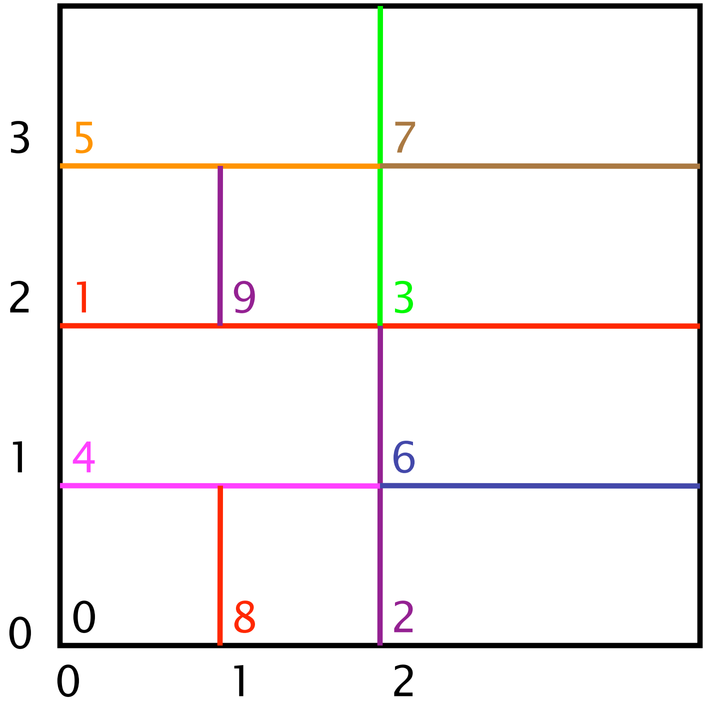


LHRB
I

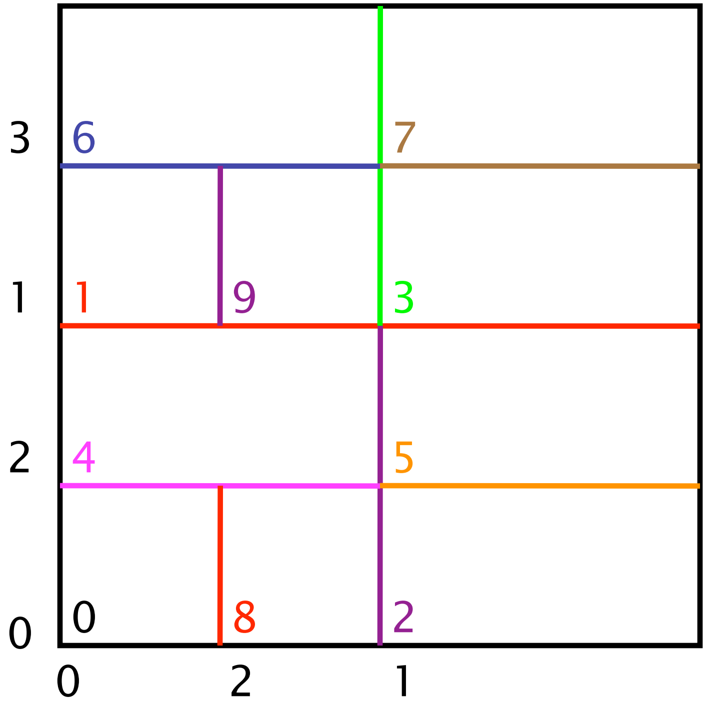


MDE
H

LHRBI vs. MDEH

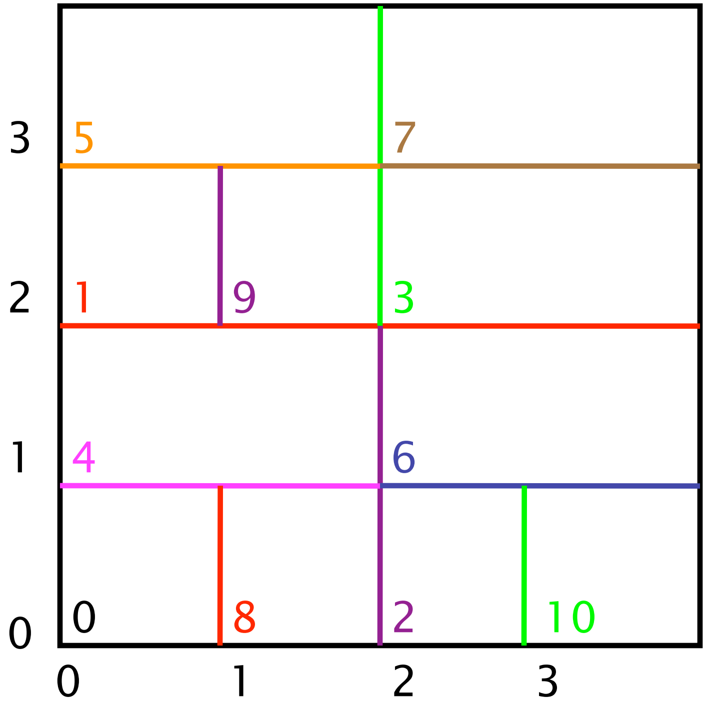


LHRB
I

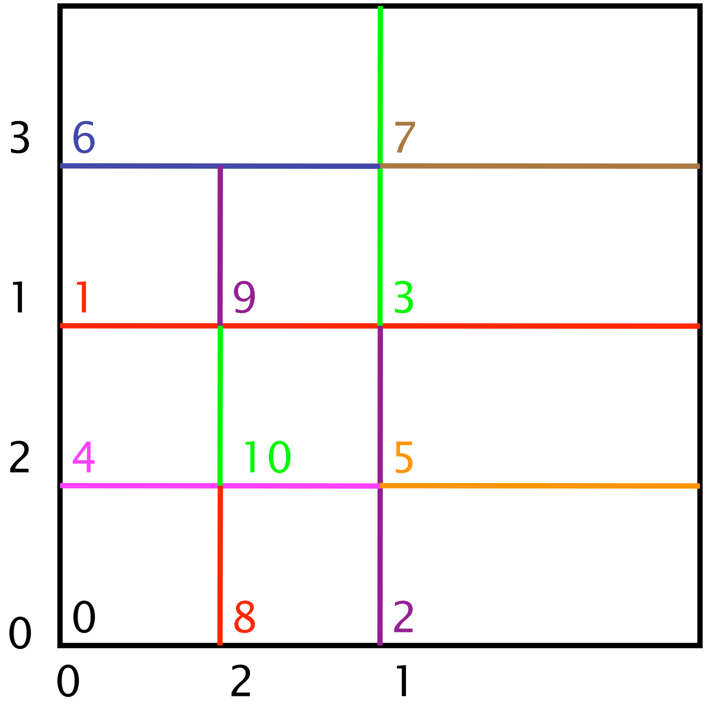


MDE
H

LHRBI vs. MDEH

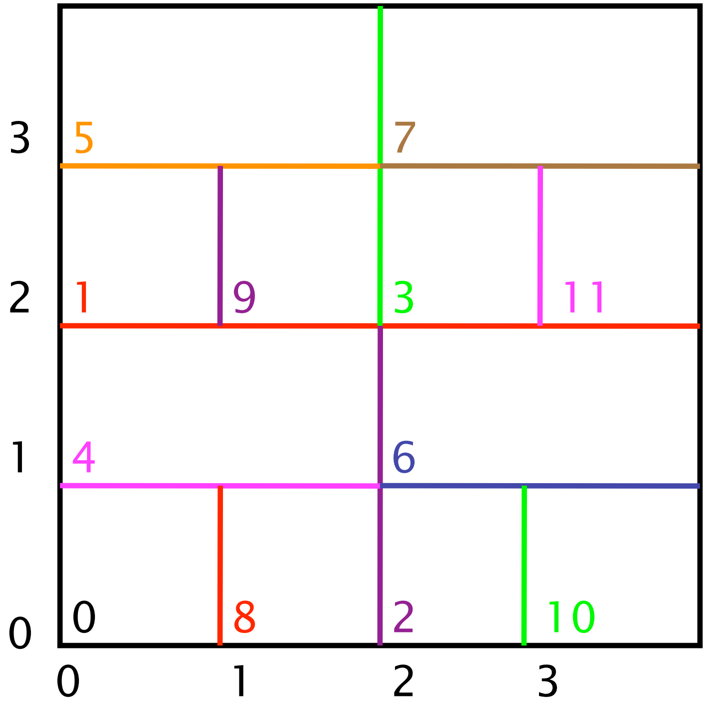


LHRB
I

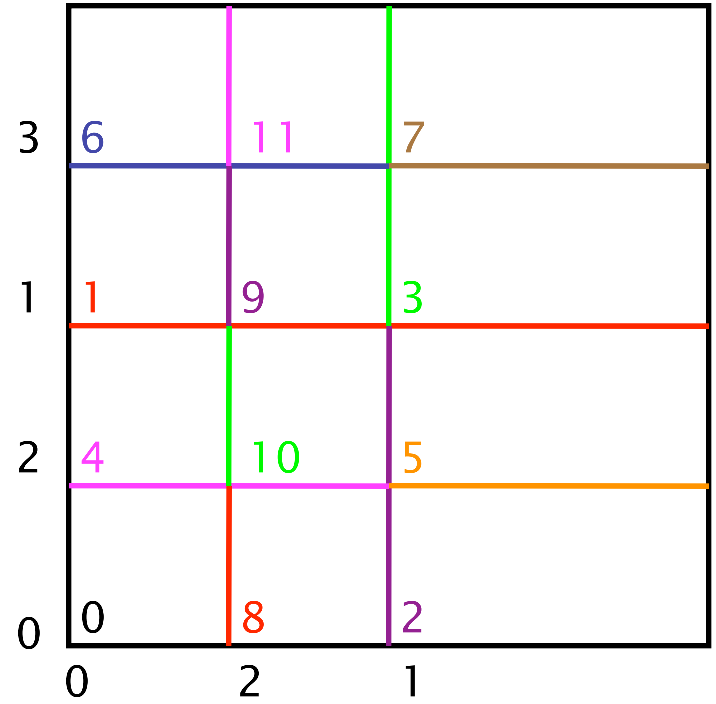


MDE
H

LHRBI vs. MDEH

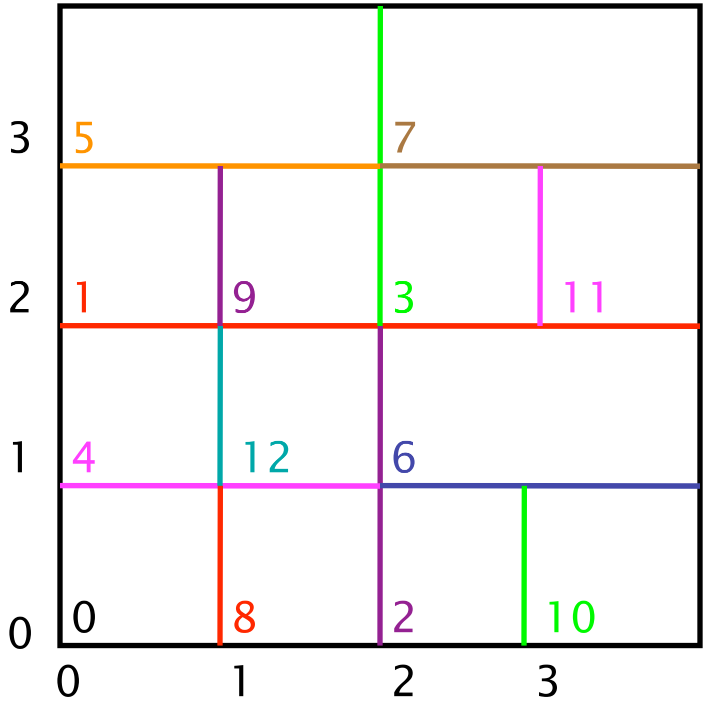


LHRB
I

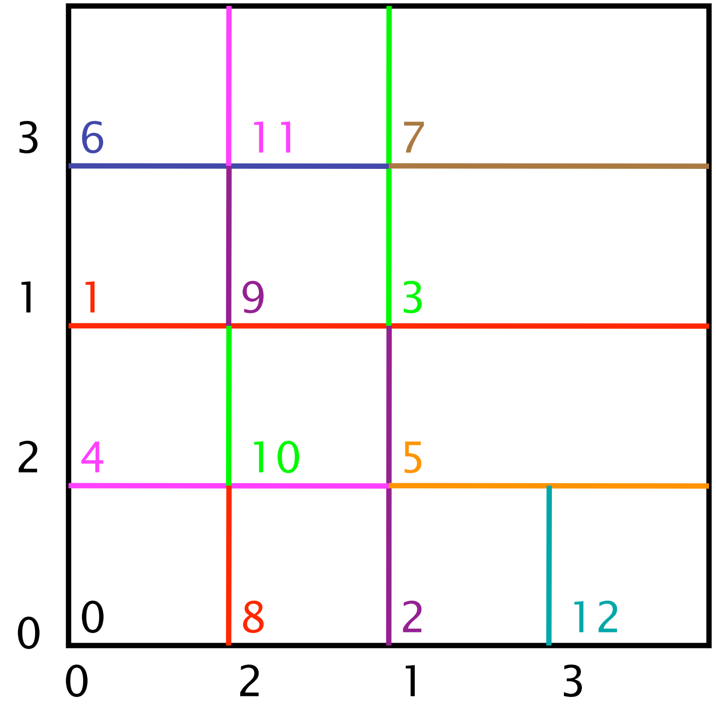


MDE
H

LHRBI vs. MDEH

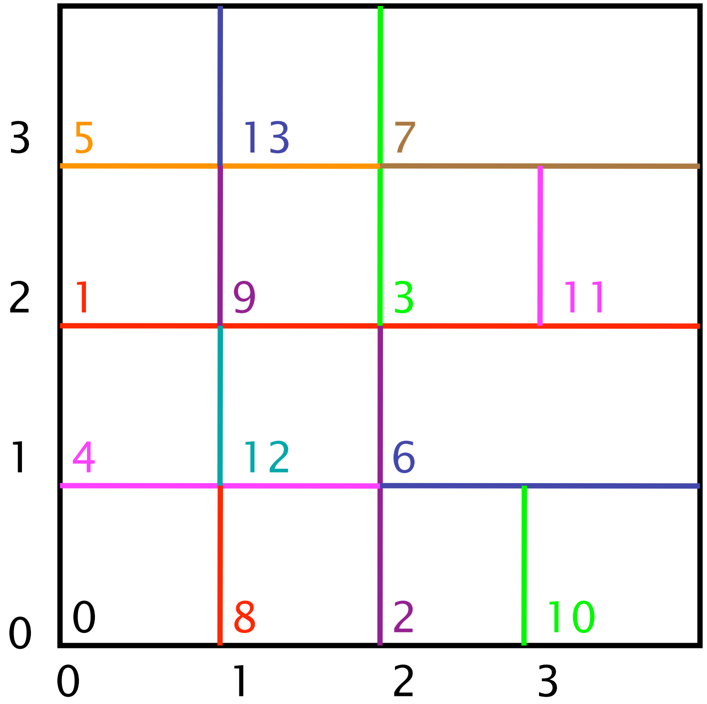


LHRB
I

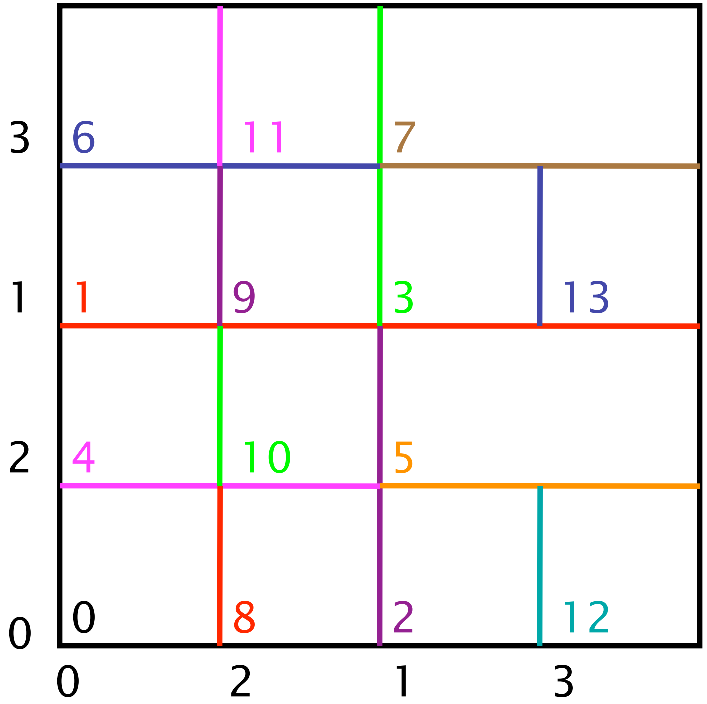


MDE
H

LHRBI vs. MDEH

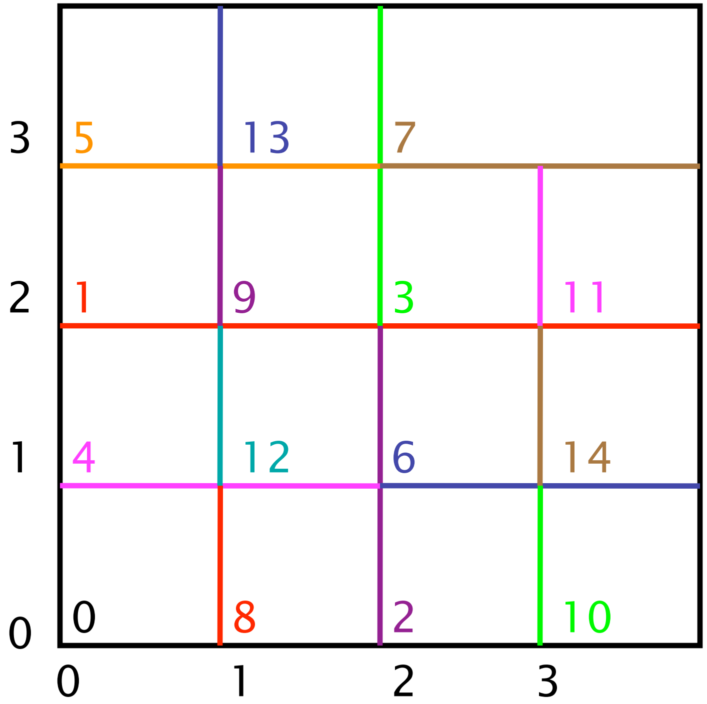


LHRB
I

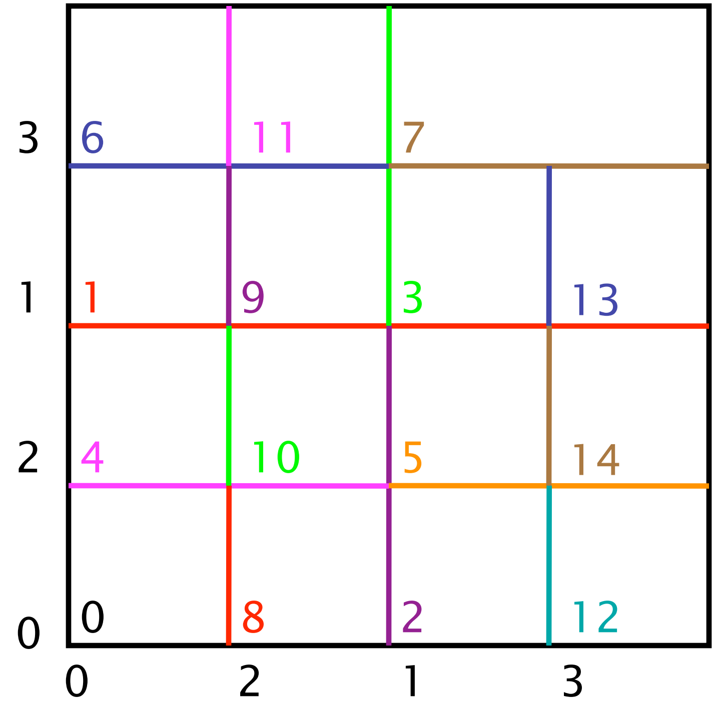


MDE
H

LHRBI vs. MDEH

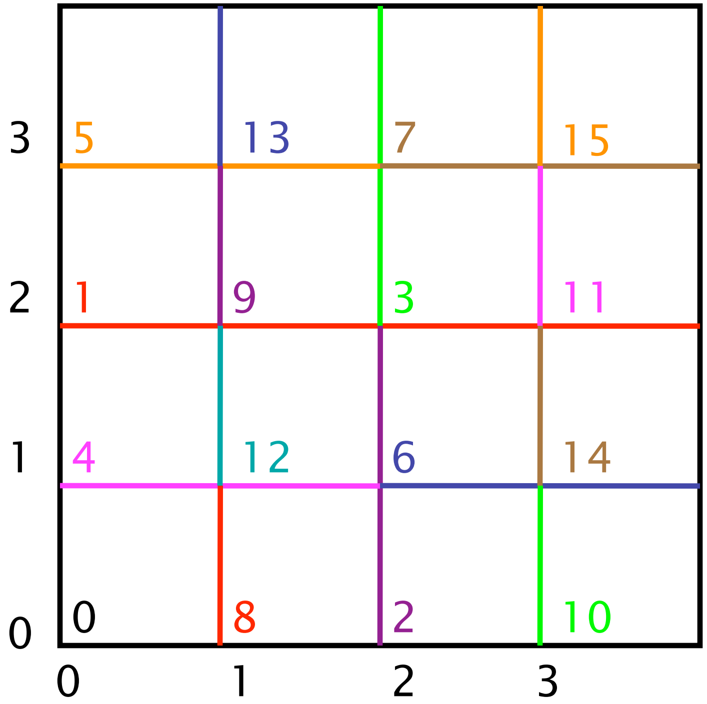


LHRB
I

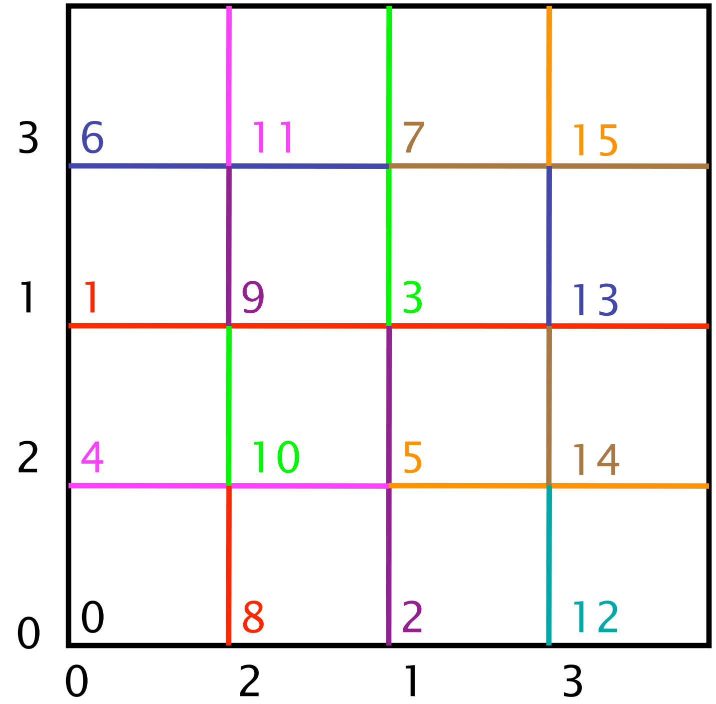


MDE
H

LHRBI vs. MDEH



LHRB
I



MDE
H

Outline

- Introduction, some basic principles
- Linear Hashing with Reversed Bit Interleaving (LHRBI)
- Multidimensional Extendible Hashing (MDEH)
- Comparison of LHRBI and MDEH
- **Quantile Hashing**
- Piecewise Linear Order Preserving (PLOP)
- Dynamic Z Hashing
- Linear Hashing with Partial Expansions (LHPE)

Quantile Hashing

MAIN POINTS.

Adaptive Variant of MDEH

Each Slice-Cut Divides Data in Half

Quantile Hashing

The Specifics.

1. Splits and slices in same relative position and order.

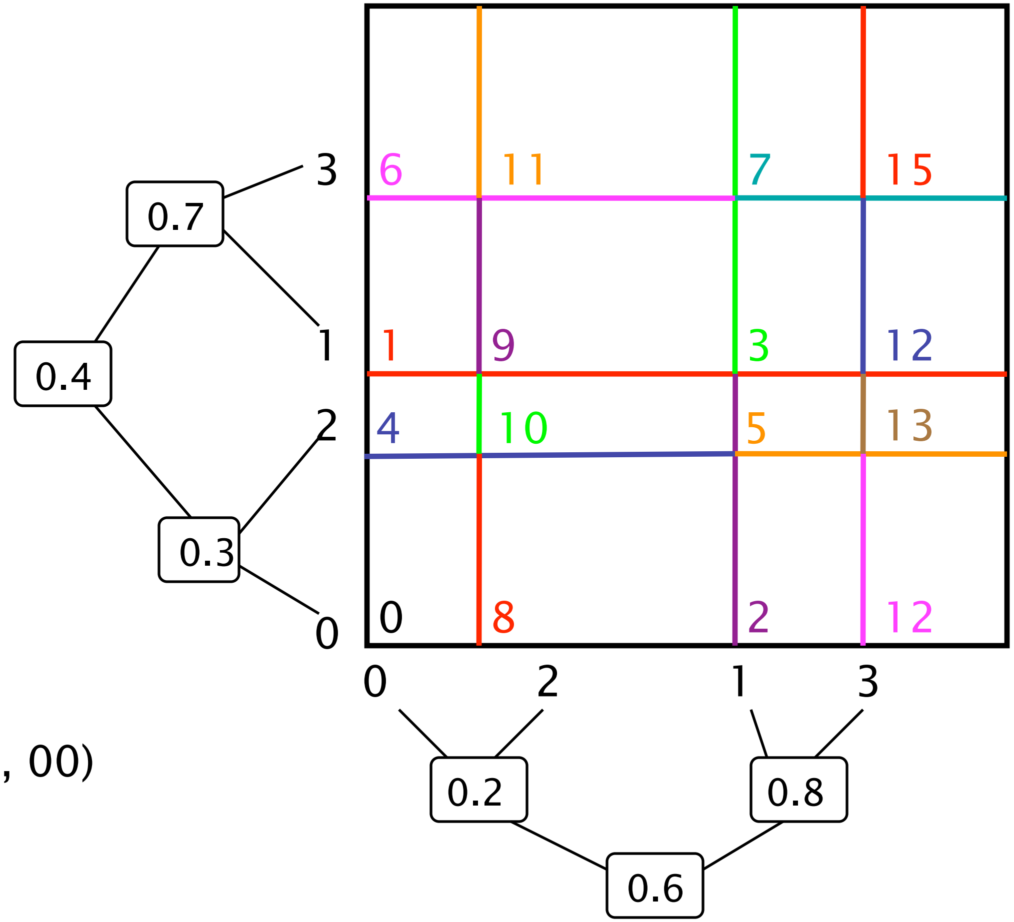
2. Instead of midpoint, cut at estimated data median.

3. Cuts’ locations in non-leaf nodes of linear scale.

4. Data based decomposition.

Quantile Hashing

Linear scale on y-axis



Expansion index $(e_y, e_x) = (00, 00)$

Next to be split is cell 0.

Linear scale on x-axis

Outline

- Introduction, some basic principles
- Linear Hashing with Reversed Bit Interleaving (LHRBI)
- Multidimensional Extendible Hashing (MDEH)
- Comparison of LHRBI and MDEH
- Quantile Hashing
- Piecewise Linear Order Preserving (PLOP)
- Dynamic Z Hashing
- Linear Hashing with Partial Expansions (LHPE)

Piecewise Linear Order Preserving Hashing

MAIN POINTS.

Adaptive Generalization of Quantile Hashing hence of MDEH.

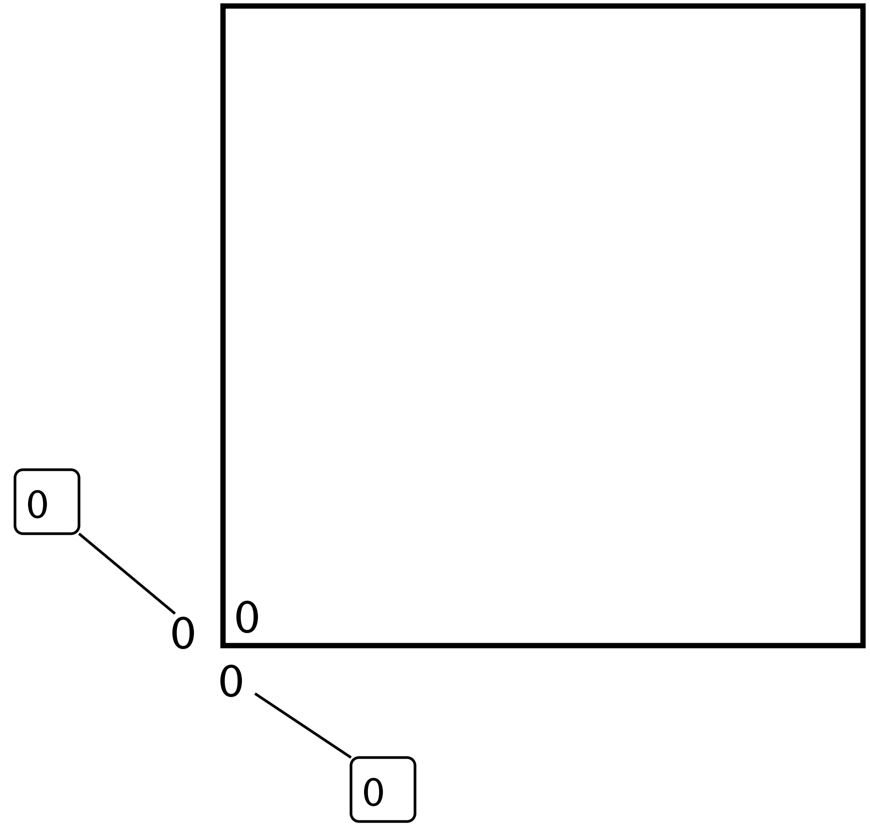
May Split the Same Slice More than Once During a Cycle

Piecewise Linear Order Preserving Hashing

1. Data based decomposition.
2. Generalization of Quantile Hashing, more adaptive
3. Cycle through axes.
4. Each axis-cycle doubles number of cells.
5. Finish a slices before starting another (like MDEH).
6. Otherwise, slice-cuts are allowed anywhere.

PLOP

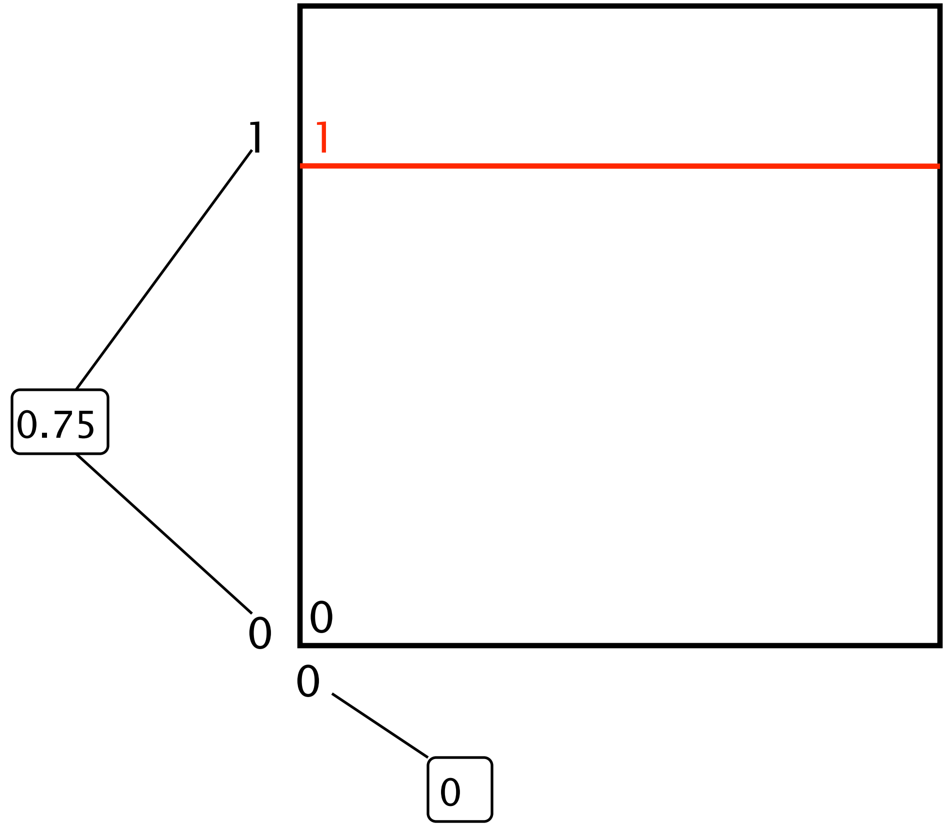
Linear scale on *y*-axis



Linear scale on *x*-axis

PLOP

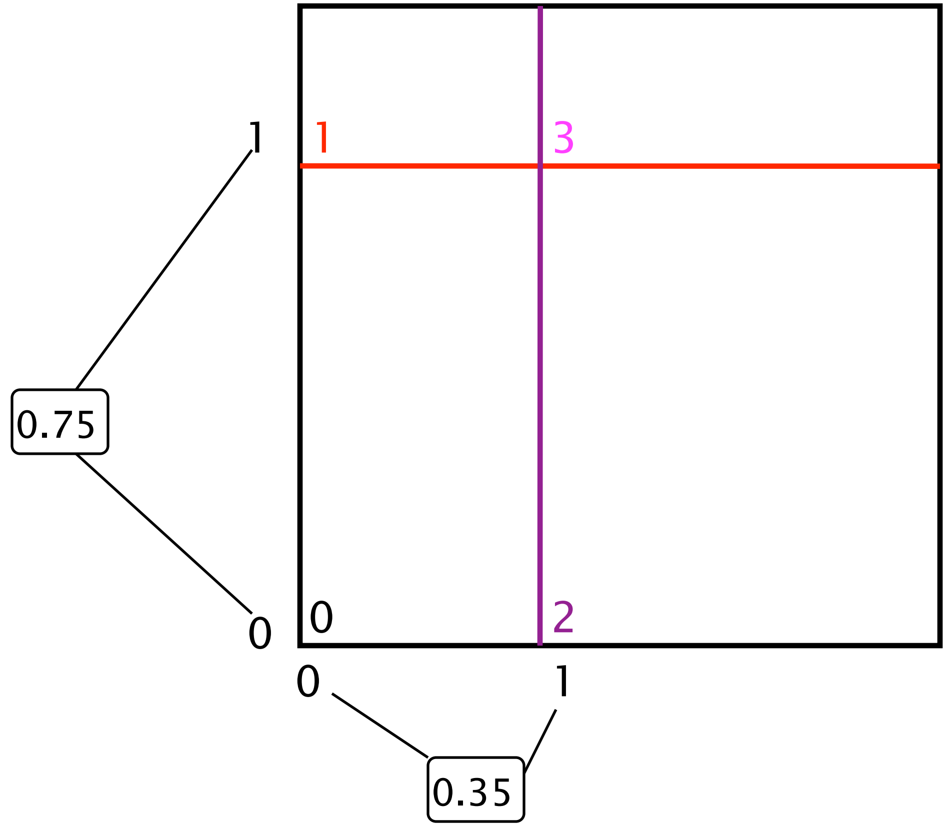
Linear scale on y-axis



Linear scale on x-axis

PLOP

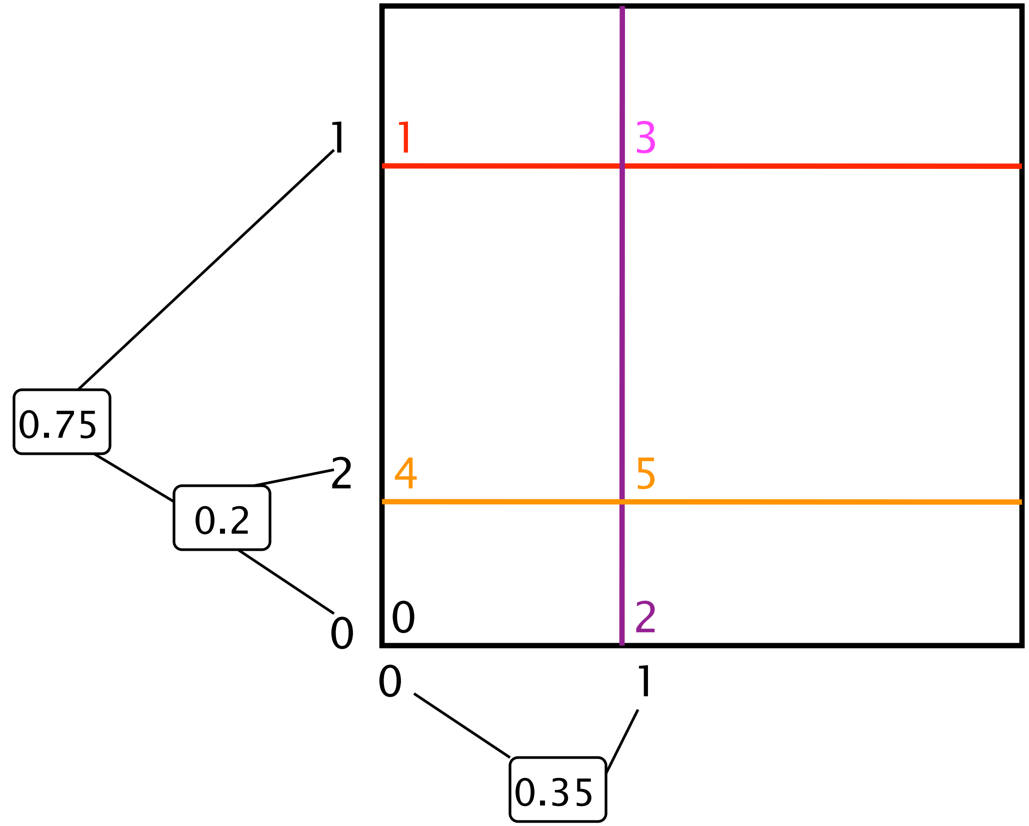
Linear scale on *y*-axis



Linear scale on *x*-axis

PLOP

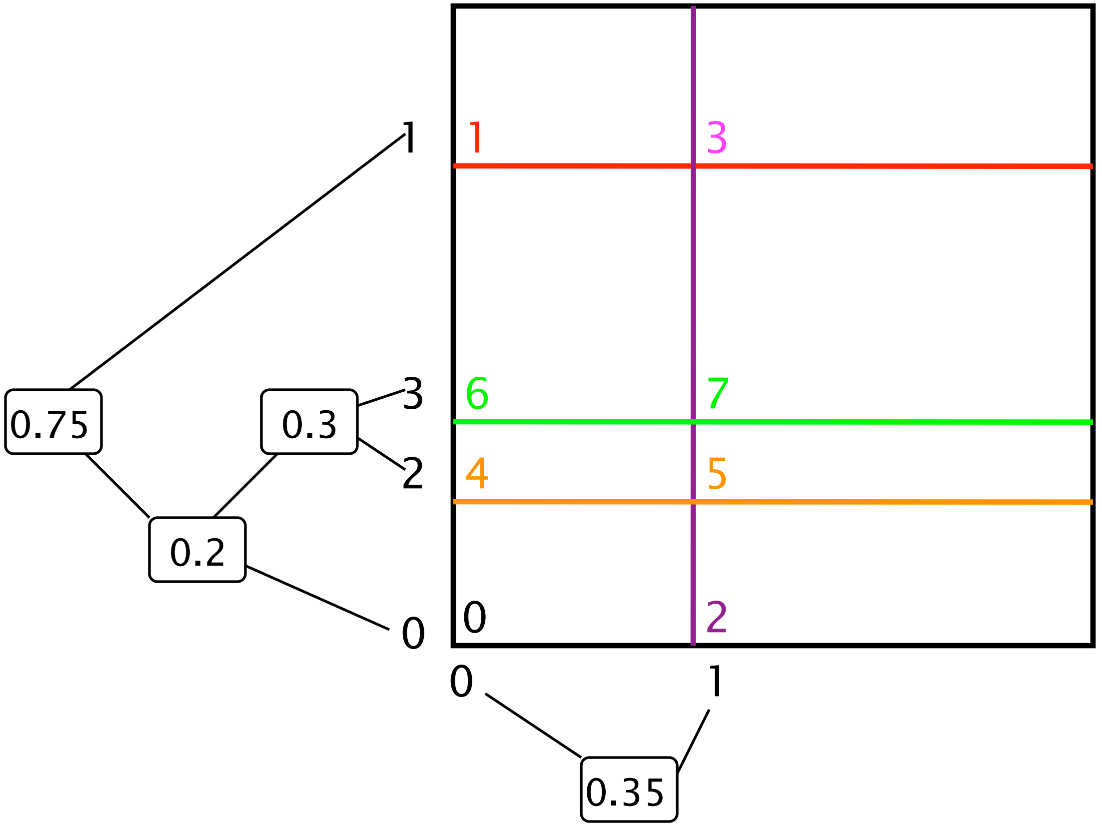
Linear scale on y-axis



Linear scale on x-axis

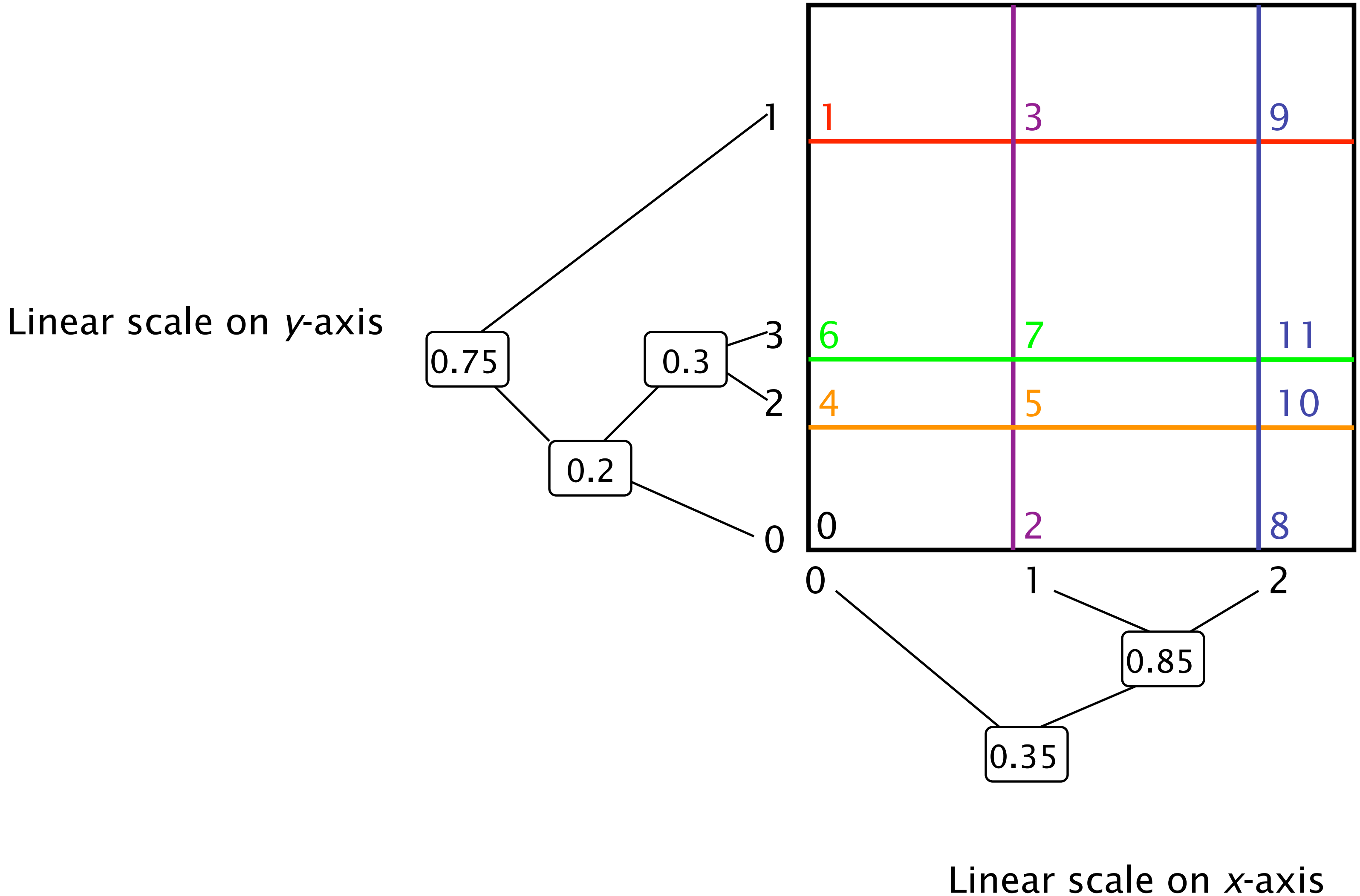
PLOP

Linear scale on *y*-axis



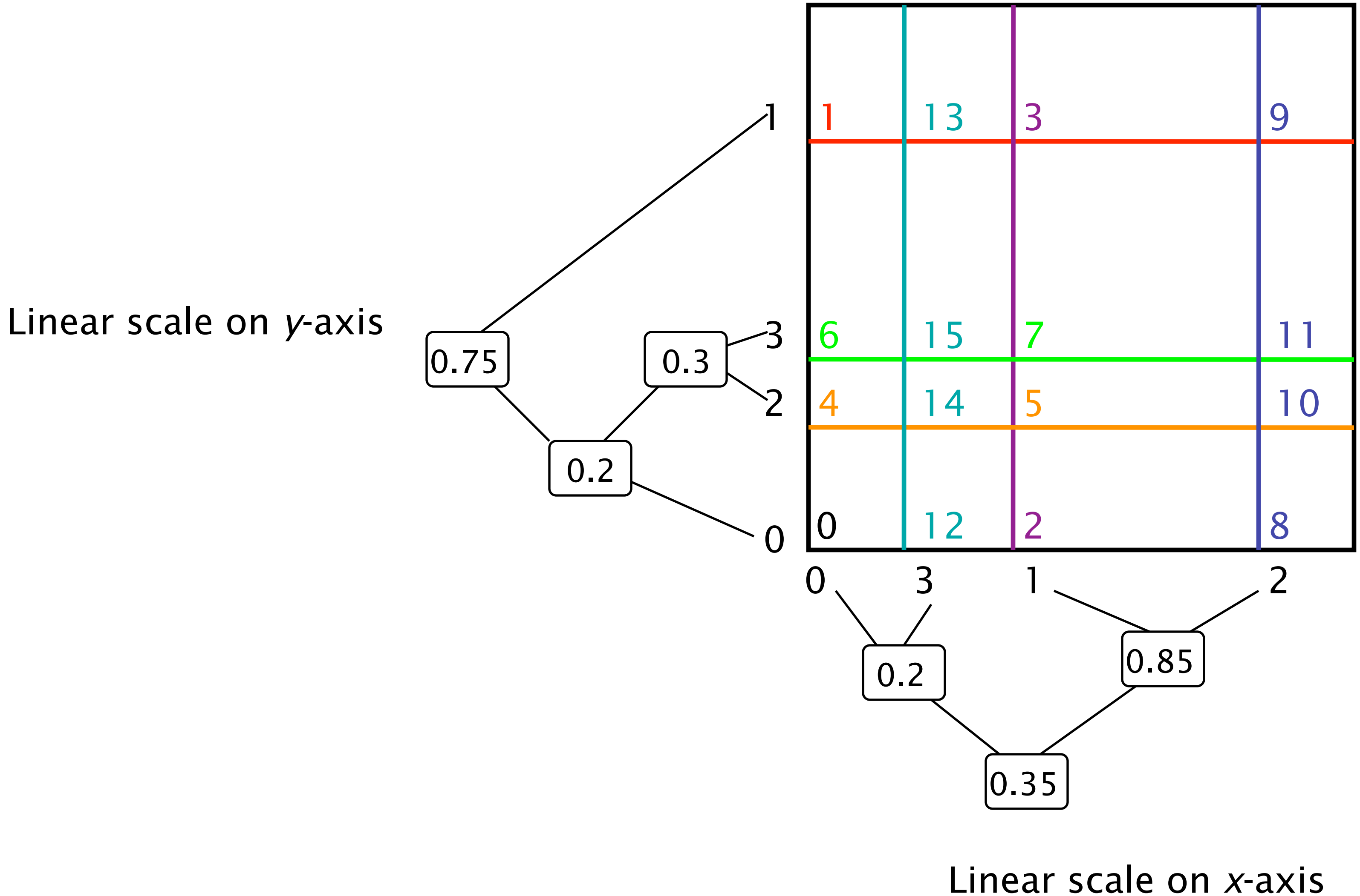
Linear scale on *x*-axis

PLOP



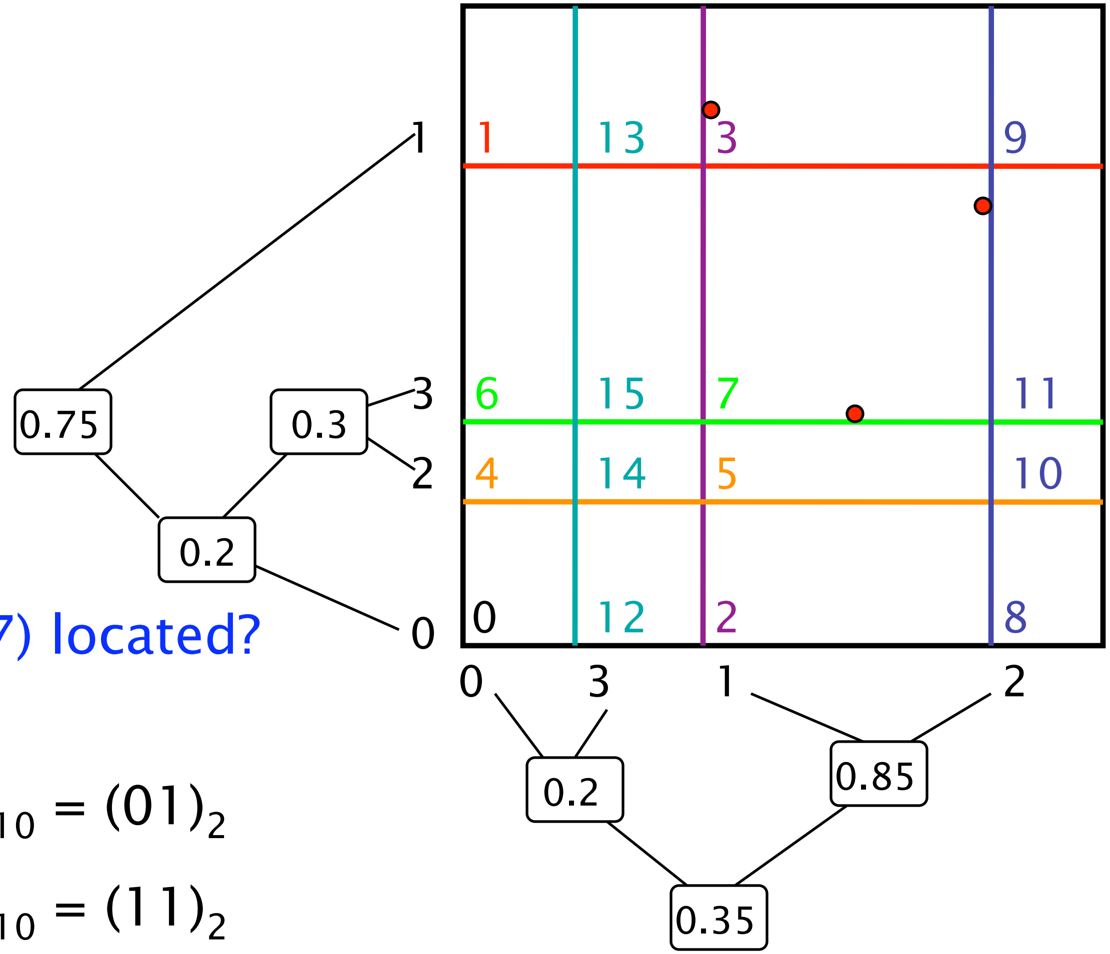
Linear scale on y-axis
0.75
0.3
0.2
1
3
2
0
1
3
9
6
7
11
4
5
10
0
2
8
0
1
2
0.85
0.35
Linear scale on x-axis

PLOP



Linear scale on y-axis
1
3
2
0
0.75
0.3
0.2
1
13
3
9
6
15
7
11
4
14
5
10
0
12
2
8
0
3
1
2
0.2
0.85
0.35
Linear scale on x-axis

PLOP



In what cell is (0.8, 0.7) located?

Linear scales lead to

x-partition index = $(1)_{10} = (01)_2$

y-partition index = $(3)_{10} = (11)_2$

Concatonate: $(0111)_2 = (7)_{10}$

Outline

- Introduction, some basic principles
- Linear Hashing with Reversed Bit Interleaving (LHRBI)
- Multidimensional Extendible Hashing (MDEH)
- Comparison of LHRBI and MDEH
- Quantile Hashing
- Piecewise Linear Order Preserving (PLOP)
- Dynamic Z Hashing
- Linear Hashing with Partial Expansions (LHPE)

Dynamic Z Hashing

MAIN POINTS.

Spatially Nearby Cells have Sequentially Nearby Numbers/Addresses.

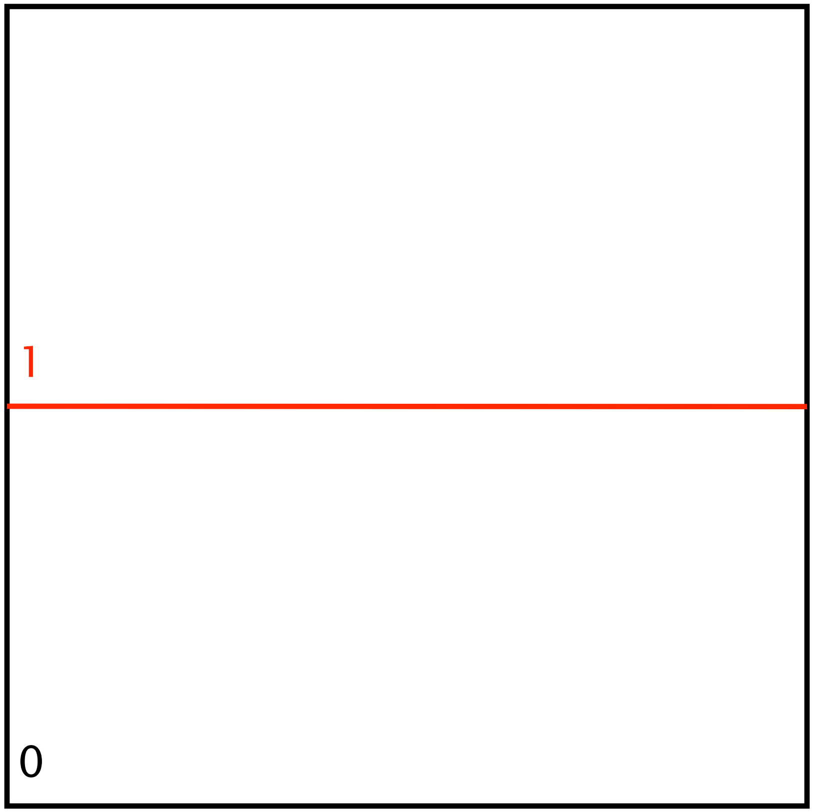
Each Round (2-power number of cells) is Order-Preserving.

Dynamic Z Hashing

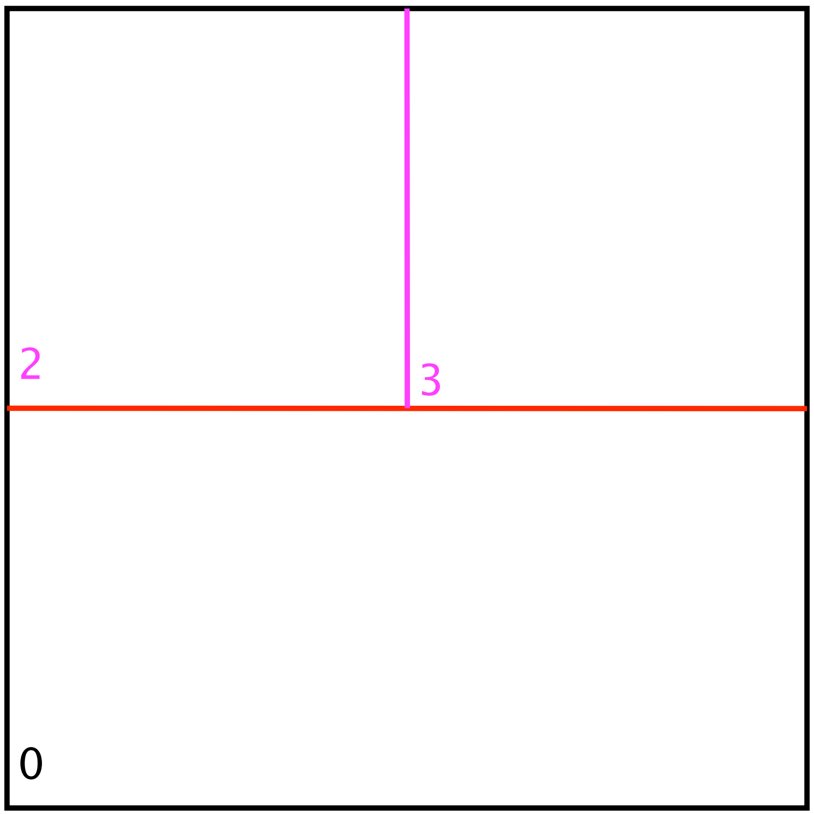
1. Embedding space-based decomposition.
2. Better than LHRBI or MDEH at keeping spacially close cells sequentially close.
3. Splitting process creates gaps.
4. Hence lower storage utilization factor.
5. With $4t$ cells currently, split cell $2t$ then $2t+1$, then cell t.

Dynamic Z

To start $t = 0$, so we split
$2t = 0$,

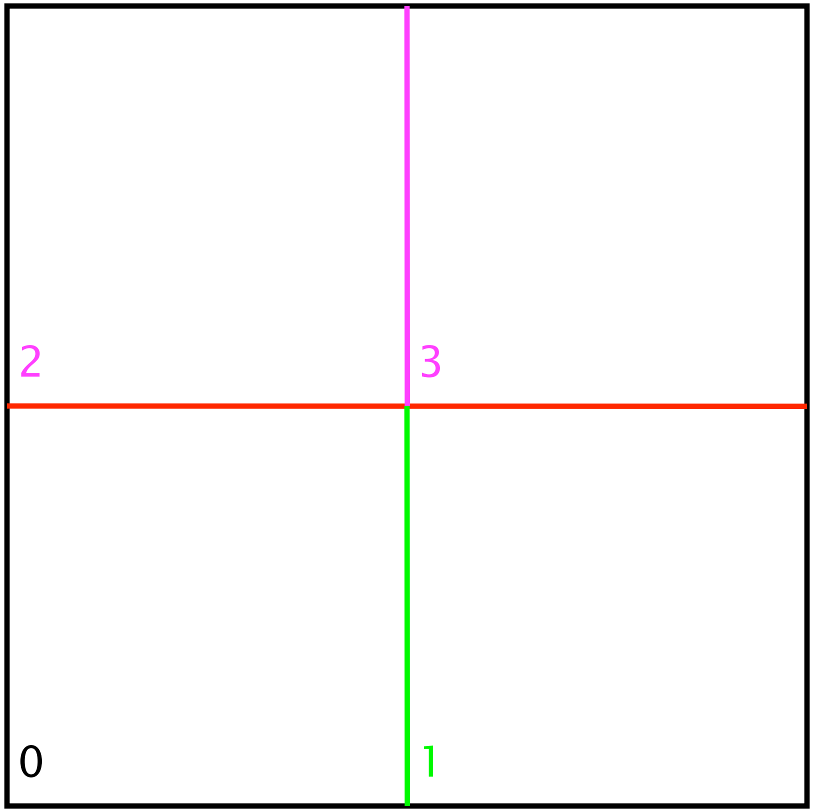


To start $t = 0$, so we split $2t = 0$, then $2t+1=1$, and renumber.



Dynamic Z

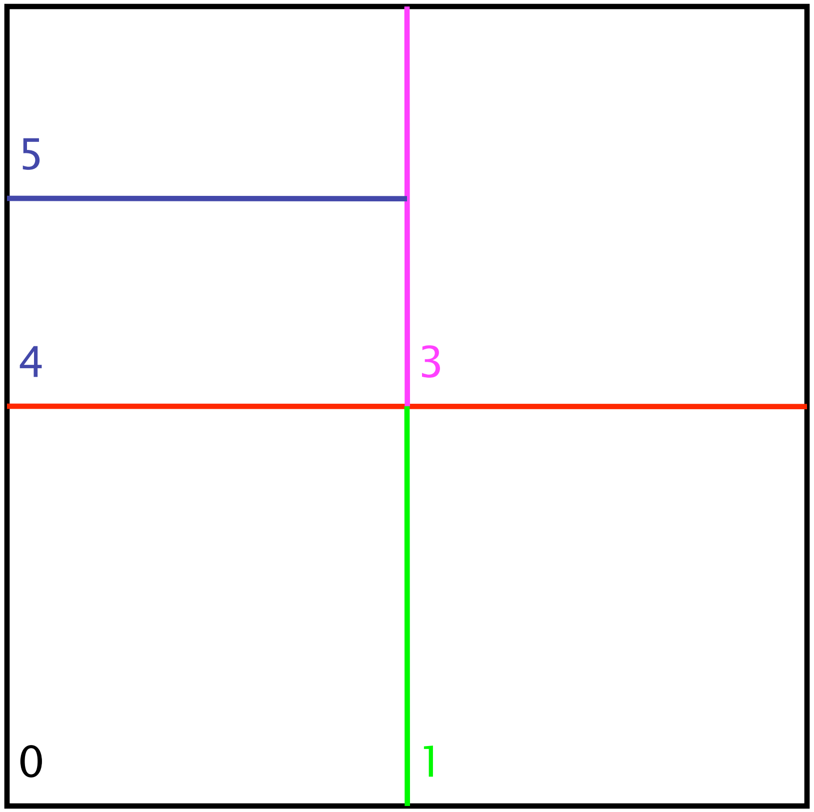
To start $t = 0$, so we split $2t = 0$, then $2t+1=1$, and renumber. Next split t, and return unused number 1 to use.



Dynamic Z

To start t = 0, so we split $2t$ = 0, then $2t$+1=1, and renumber. Next split t, and return unused number 1 to use.

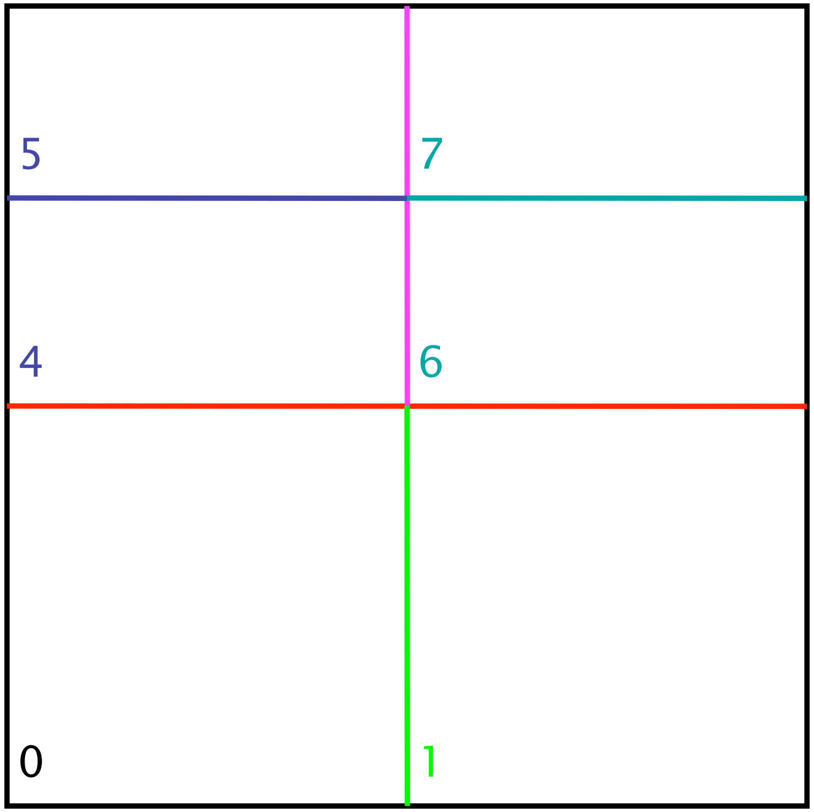
Now t = 1, so we split $2t$ = 2 first, and renumber.



Dynamic Z

To start $t = 0$, so we split $2t = 0$, then $2t+1=1$, and renumber. Next split t, and return unused number 1 to use.

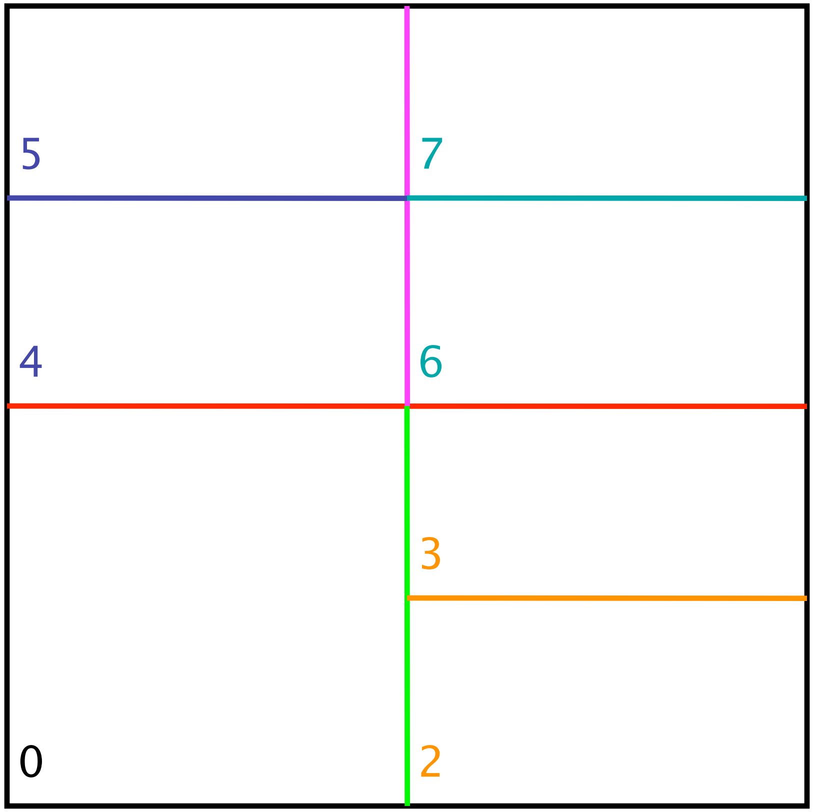
Now $t = 1$, so we split $2t = 2$ first, and renumber. Then split $2t+1 = 3$ and renumber.



Dynamic Z

To start $t = 0$, so we split $2t = 0$, then $2t+1=1$, and renumber. Next split t, and return unused number 1 to use.

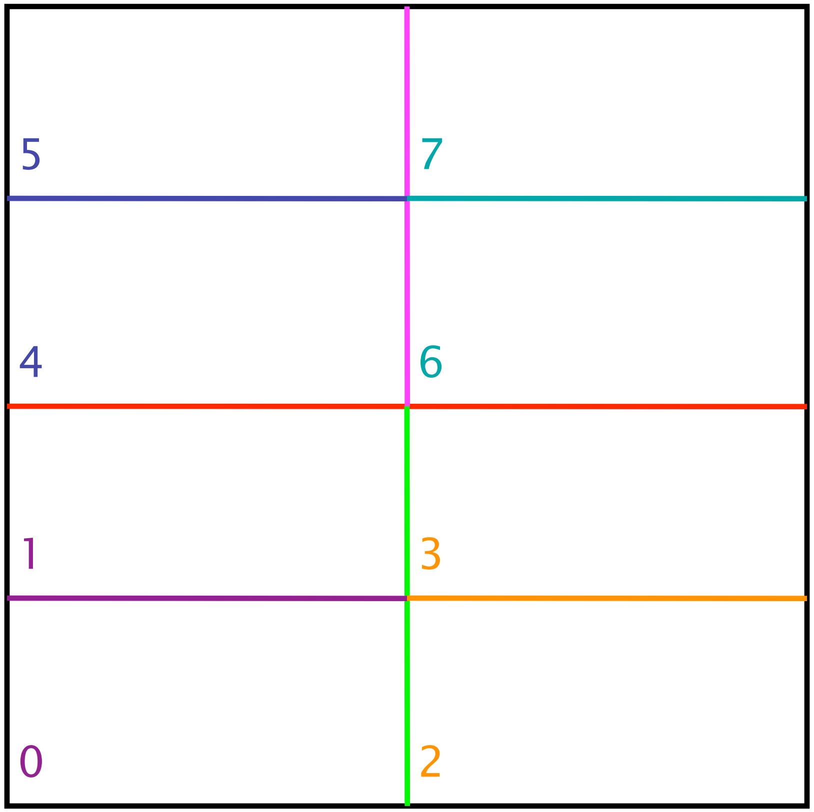
Now $t = 1$, so we split $2t = 2$ first, and renumber. Then split $2t+1 = 3$ and renumber. Next split cell $t = 1$



Dynamic Z

To start $t = 0$, so we split $2t = 0$, then $2t+1=1$, and renumber. Next split t, and return unused number 1 to use.

Now $t = 1$, so we split $2t = 2$ first, and renumber. Then split $2t+1 = 3$ and renumber. Next split cell $t = 1$, and finally cell 0.

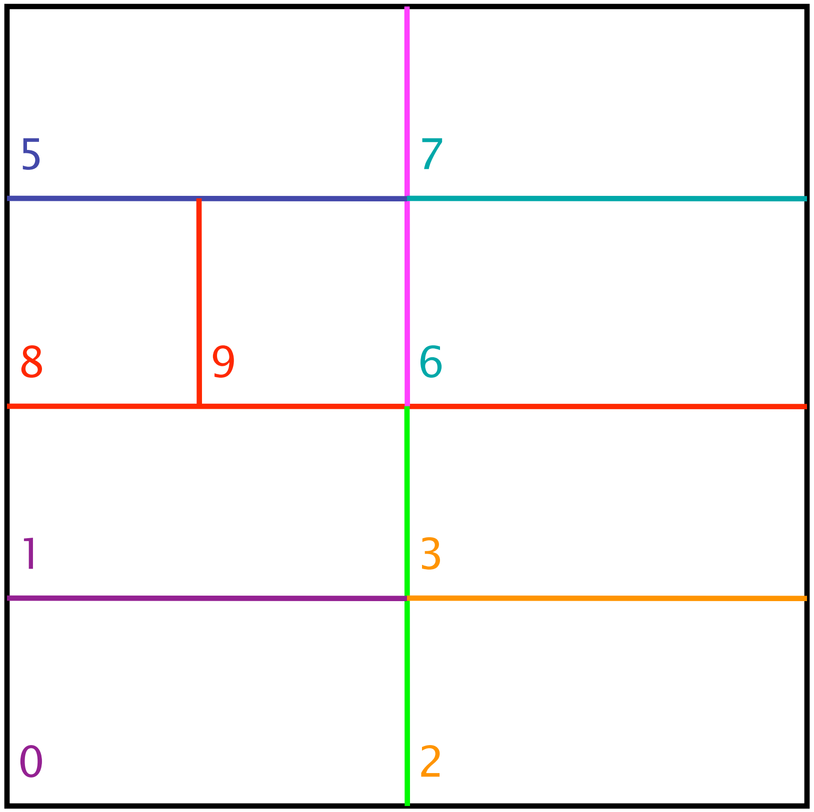


Dynamic Z

To start $t = 0$, so we split $2t = 0$, then $2t+1=1$, and renumber. Next split t, and return unused number 1 to use.

Now $t = 1$, so we split $2t = 2$ first, and renumber. Then split $2t+1 = 3$ and renumber. Next split cell $t = 1$, and finally cell 0.

Next t = 2, so we split cell $2t = 4$.

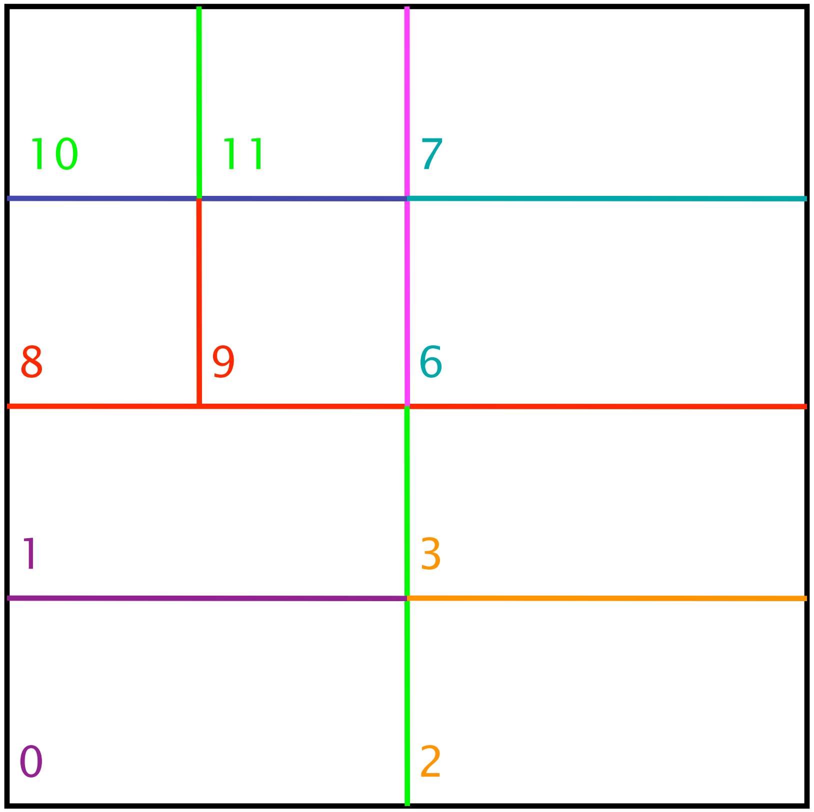


Dynamic Z

To start $t = 0$, so we split $2t = 0$, then $2t+1=1$, and renumber. Next split t, and return unused number 1 to use.

Now $t = 1$, so we split $2t = 2$ first, and renumber. Then split $2t+1 = 3$ and renumber. Next split cell $t = 1$, and finally cell 0.

Next t = 2, so we split cell $2t = 4$. Then $2t+1 = 5$.

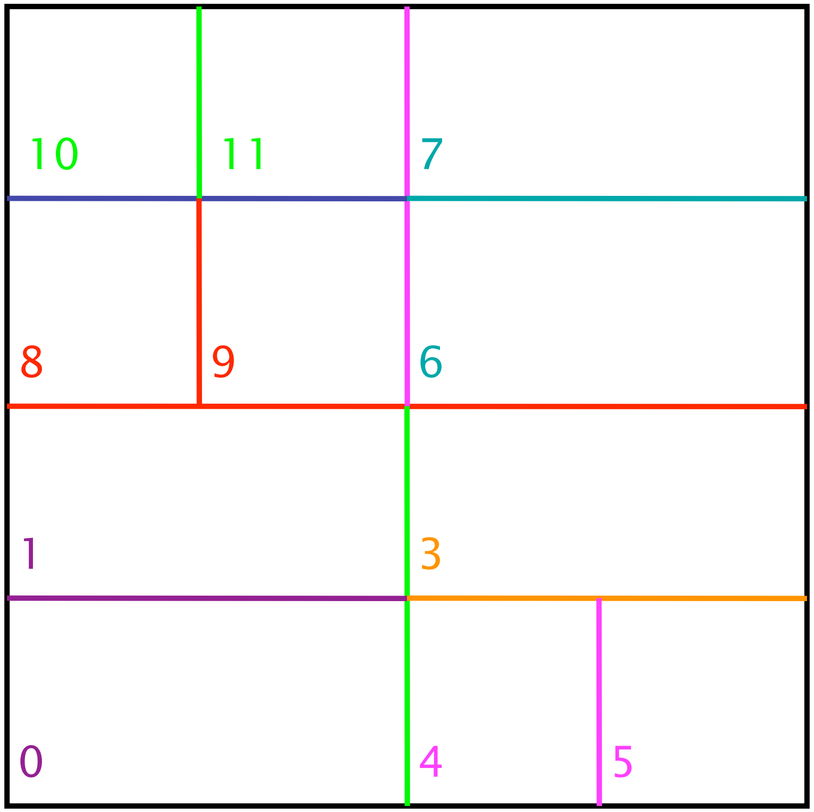


Dynamic Z

To start $t = 0$, so we split $2t = 0$, then $2t+1=1$, and renumber. Next split t, and return unused number 1 to use.

Now $t = 1$, so we split $2t = 2$ first, and renumber. Then split $2t+1 = 3$ and renumber. Next split cell $t = 1$, and finally cell 0.

Next t = 2, so we split cell $2t = 4$. Then $2t+1 = 5$. Then $t = 2$.

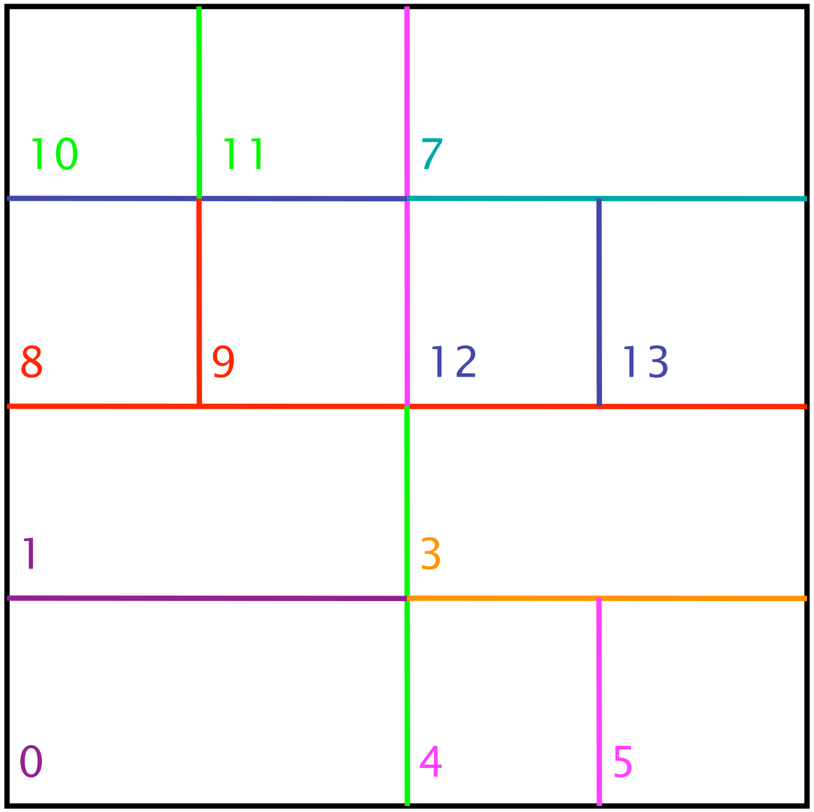


Dynamic Z

To start $t = 0$, so we split $2t = 0$, then $2t+1=1$, and renumber. Next split t, and return unused number 1 to use.

Now $t = 1$, so we split $2t = 2$ first, and renumber. Then split $2t+1 = 3$ and renumber. Next split cell $t = 1$, and finally cell 0.

Next t = 2, so we split cell $2t = 4$. Then $2t+1 = 5$. Then $t = 2$. Cell 6 is split next.

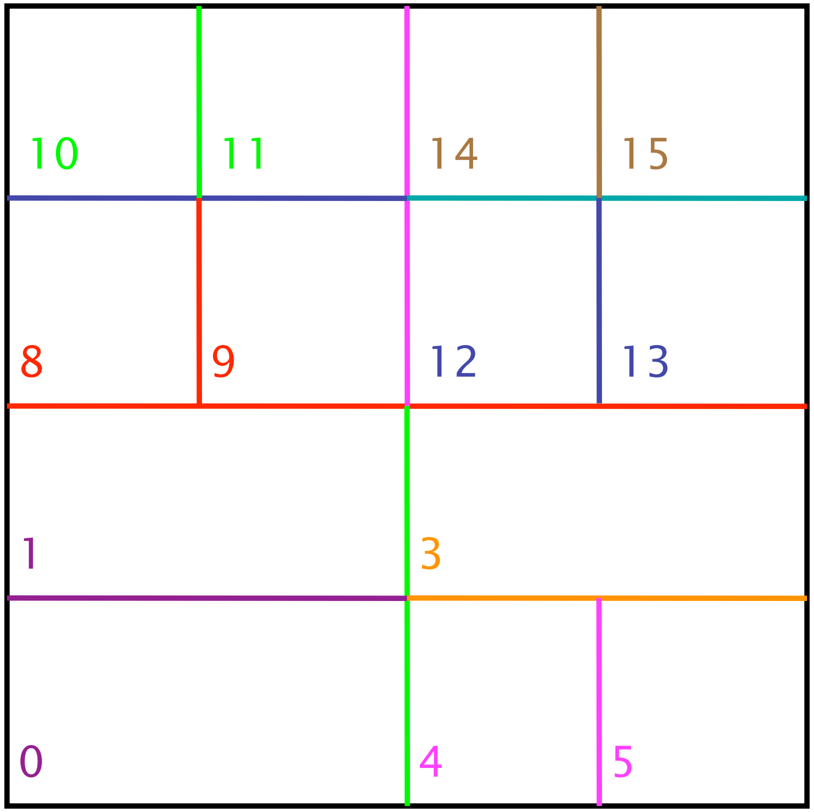


Dynamic Z

To start $t = 0$, so we split $2t = 0$, then $2t+1=1$, and renumber. Next split t, and return unused number 1 to use.

Now $t = 1$, so we split $2t = 2$ first, and renumber. Then split $2t+1 = 3$ and renumber. Next split cell $t = 1$, and finally cell 0.

Next t = 2, so we split cell $2t = 4$. Then $2t+1 = 5$. Then $t = 2$. Cell 6 is split next, then cell 7.

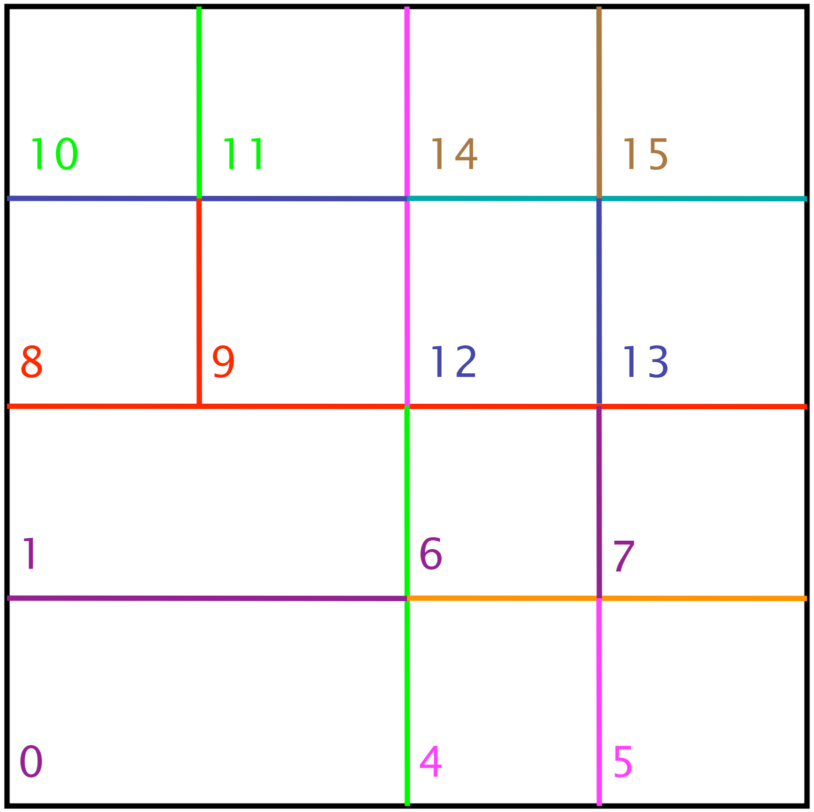


Dynamic Z

To start $t = 0$, so we split $2t = 0$, then $2t+1=1$, and renumber. Next split t, and return unused number 1 to use.

Now $t = 1$, so we split $2t = 2$ first, and renumber. Then split $2t+1 = 3$ and renumber. Next split cell $t = 1$, and finally cell 0.

Next t = 2, so we split cell $2t = 4$. Then $2t+1 = 5$. Then $t = 2$. Cell 6 is split next, then cell 7. Now 3 can be split.

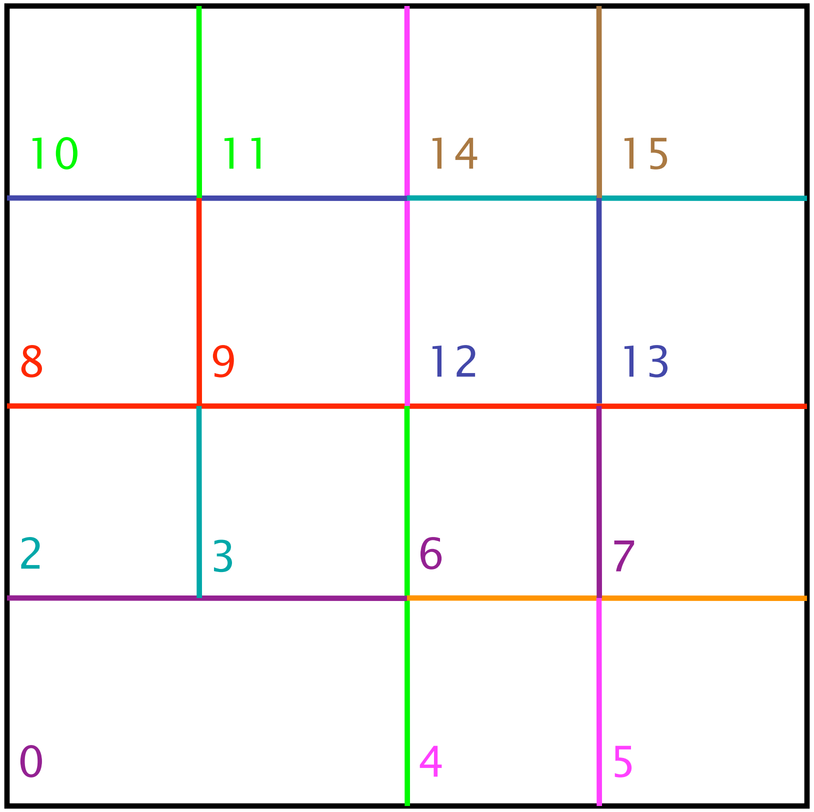


Dynamic Z

To start $t = 0$, so we split $2t = 0$, then $2t+1=1$, and renumber. Next split t, and return unused number 1 to use.

Now $t = 1$, so we split $2t = 2$ first, and renumber. Then split $2t+1 = 3$ and renumber. Next split cell $t = 1$, and finally cell 0.

Next t = 2, so we split cell $2t = 4$. Then $2t+1 = 5$. Then $t = 2$. Cell 6 is split next, then cell 7. Now 3 can be split, then 1.

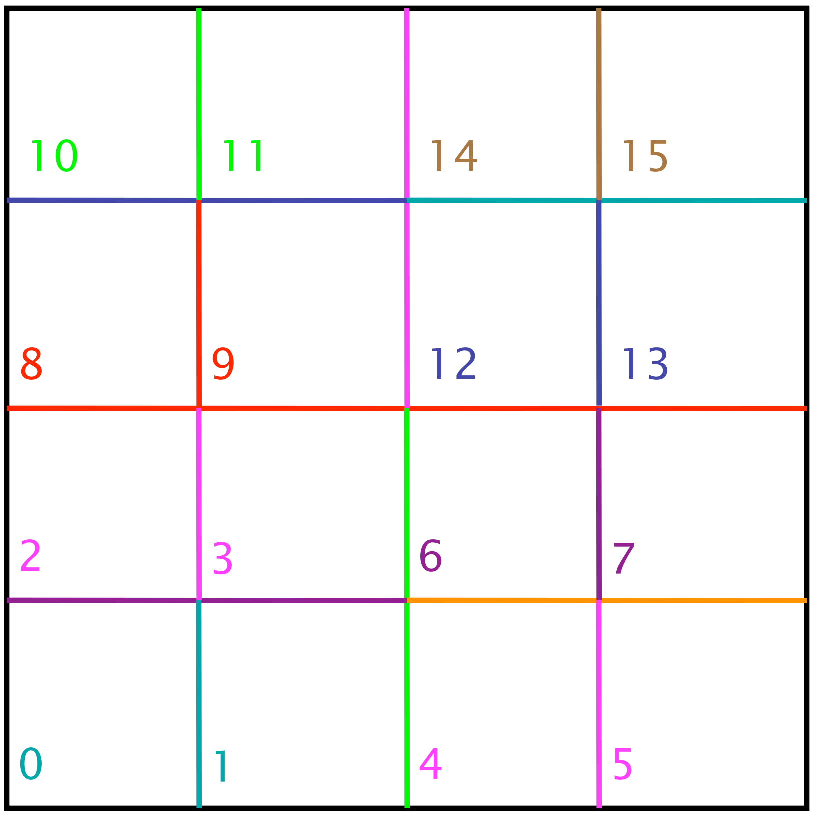


Dynamic Z

To start t = 0, so we split $2t$ = 0, then $2t$+1=1, and renumber. Next split t, and return unused number 1 to use.

Now t = 1, so we split $2t$ = 2 first, and renumber. Then split $2t$+1 = 3 and renumber. Next split cell t = 1, and finally cell 0.

Next t = 2, so we split cell $2t$ = 4. Then $2t$+1 = 5. Then t = 2. Cell 6 is split next, then cell 7. Now 3 can be split, then 1, and finally 0.



Dynamic Z

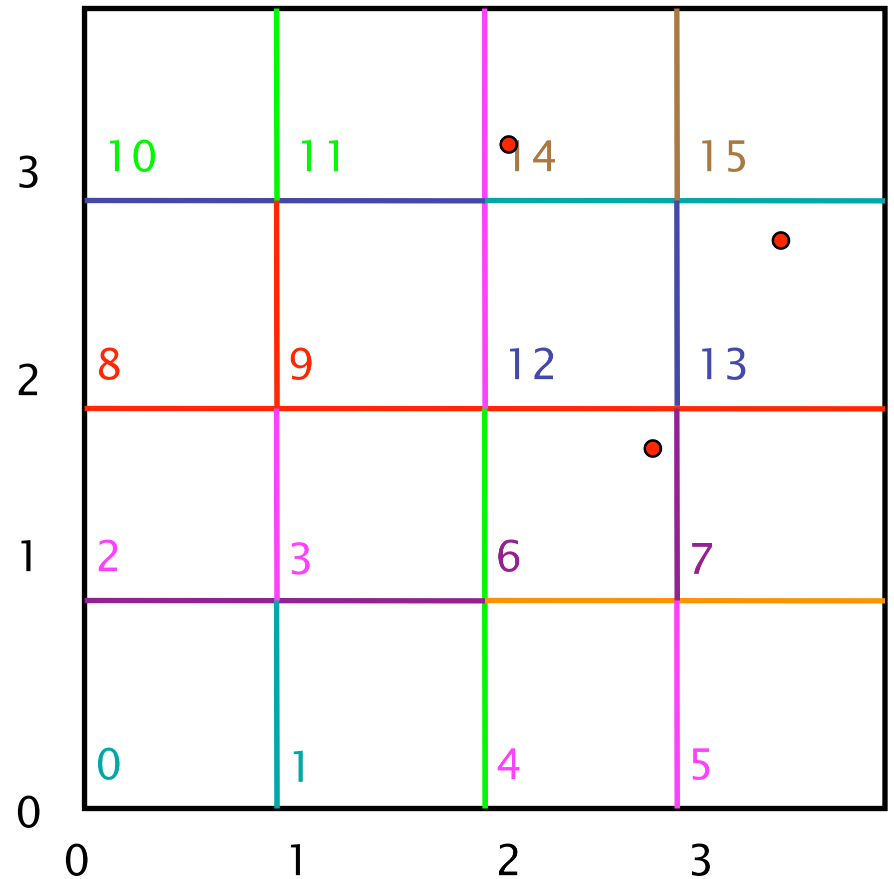
In what cell is (0.87, 0.7) located?

With partition indices ordered by distance from origin,

x-partition index = $(11)_2$,

y-partition index = $(10)_2$,

y-first interleaving implies $(1101)_2 = (13)_{10}$.



Outline

- Introduction, some basic principles
- Linear Hashing with Reversed Bit Interleaving (LHRBI)
- Multidimensional Extendible Hashing (MDEH)
- Comparison of LHRBI and MDEH
- Quantile Hashing
- Piecewise Linear Order Preserving (PLOP)
- Dynamic Z Hashing
- Linear Hashing with Partial Expansions (LHPE)

Linear Hashing with Partial Expansion (LHPE)

MAIN POINTS.

A Variation on Linear Hashing that May Be Used with Any (space-based) Method.

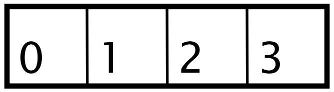
Number of Cells Doubles More Gradually.

Linear Hashing with Partial Expansion (LHPE)

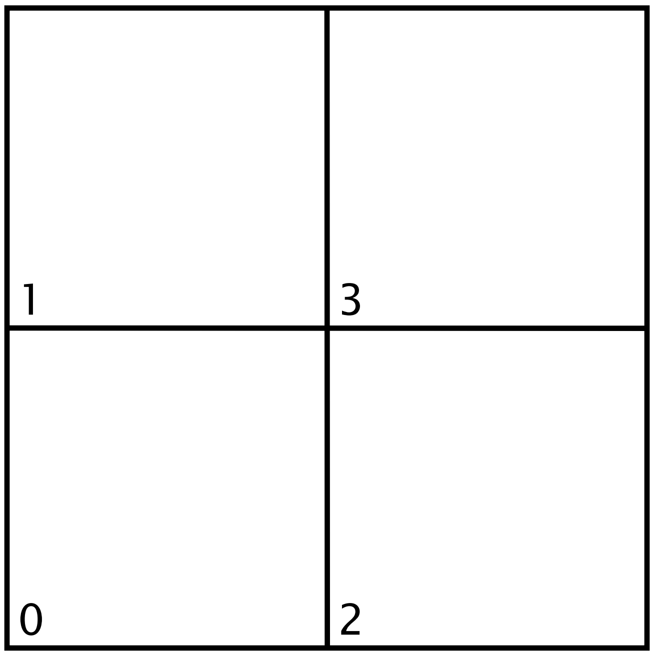
The Specifics.

1. Not itself a hashing function.
2. Not so relevant to data-based splitting approaches
3. Used with order-preserving (or other) linear hashing.
4. Doubles in two steps
    a. First split 2 cells into 3.
    b. Then 3 cells into 4.

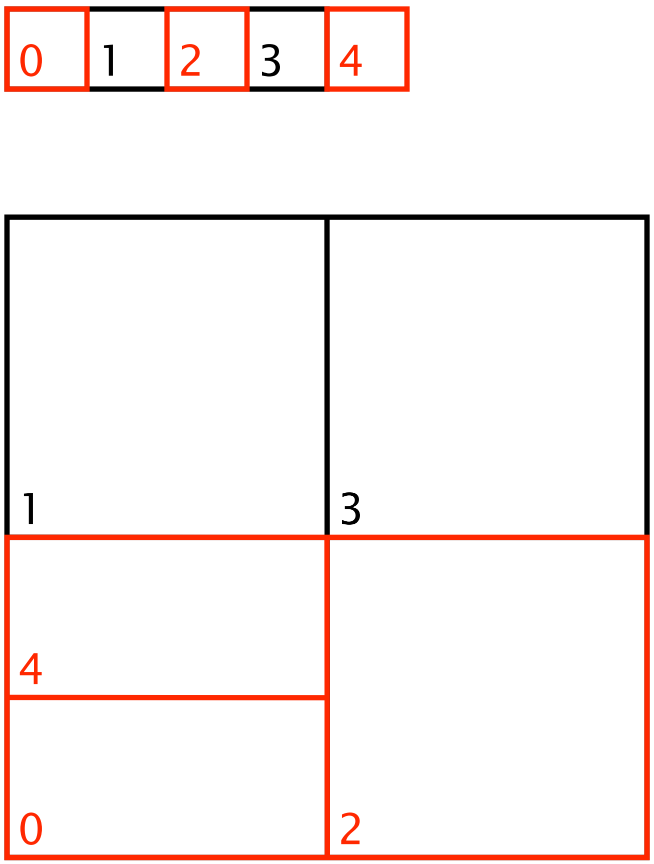
Example.



Starting with 4

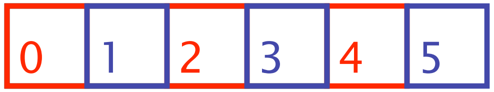


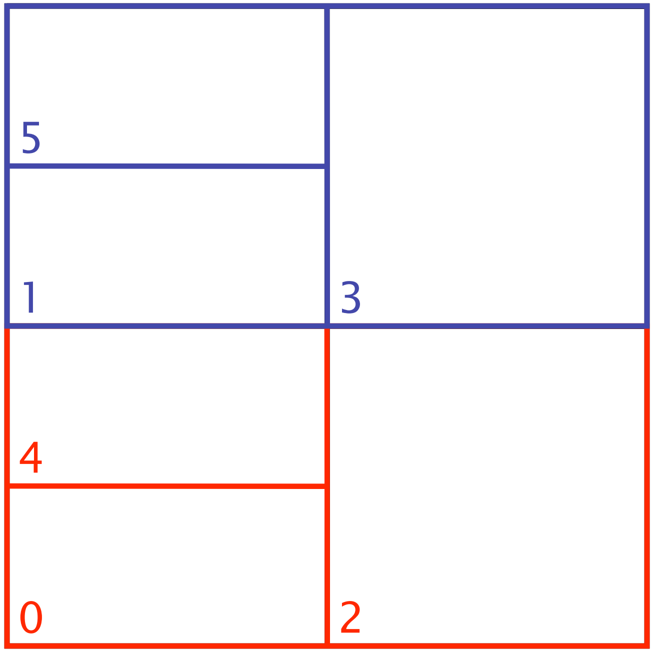
Example.



Split 2 into 3

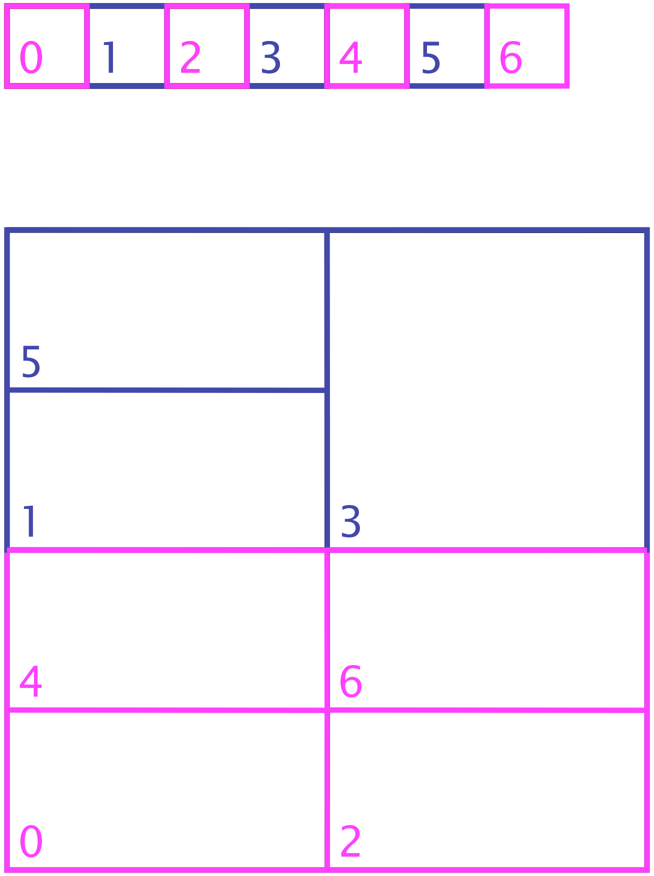
Example.





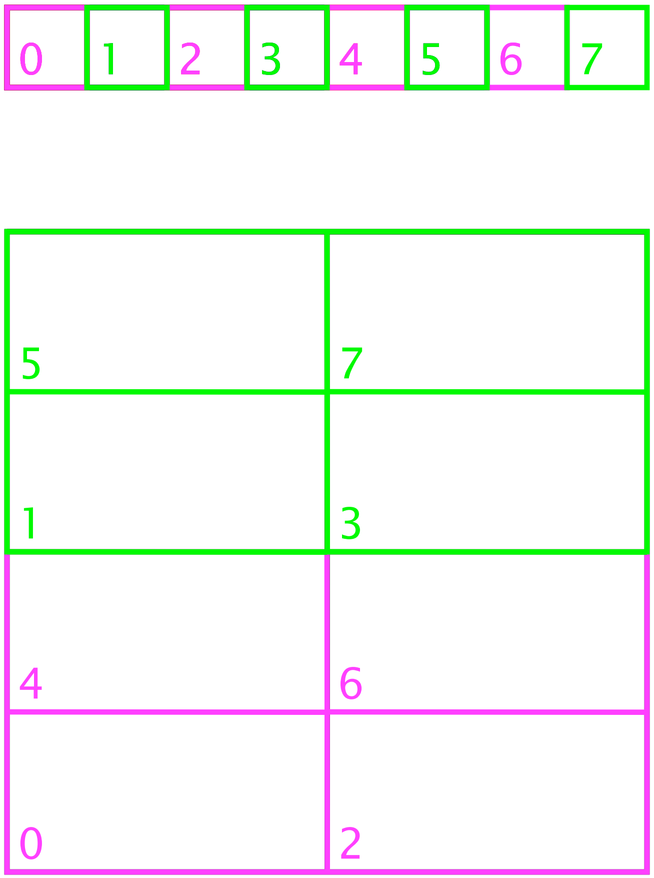
Split 2 into 3

Example.



Split 3 into 4

Example.



Split 3 into 4

Notice that 2D decomposition-construction is that of LHRBI

And so forth

SUMMARY

- Linear Hashing—add one cell/bucket at a time
    - Total grid partion
    - Bit operations—concatonation, (reversed) interleaving

- LHRBI Linear Hashing with Reversed Bit Interleaving
    - Embedding space-based decomposition
    - Order preserving
    - Partition indices ordered by distance from origin
    - Hash function—*y*-first reversed bit interleaving on partion indices

- Multidimensional Extendible Hashing
  - Embedding space-based decomposition
  - Cycle through axes
  - Complete each slice and each axis before going to next
  - Partition indices numbered in order of creation, and tracked using linear scales
  - Expansion Index points to next split
  - Hash function—*x*-first concatonation on partiton indices

Summary

- Quantile Hashing
  - Data-based decomposition
  - Just like MDEH, except
  - Slice-cut location may vary
  - Each slice-cut attempts to divide enclosed data in half
- Piecewise-Linear Order-Preserving Hashing
  - Data-based decomposition
  - Slice location is arbitrary
  - Complete slice and axis before doing next
  - Cycle through axes
  - More adaptive than Quantile Hashing

- Dynamic Z Hashing
  - Embedding space-based decomposition
  - Spacially close cells are numerically close
  - Splitting proces produces gaps, hence lower storage utilization factor
  - Hash function—*y*-first interleaving

- Linear Hashing with Partial Expansion
  - Use with embedding space decompositions
  - Double in two steps—split 2 into 3, then 3 into 4