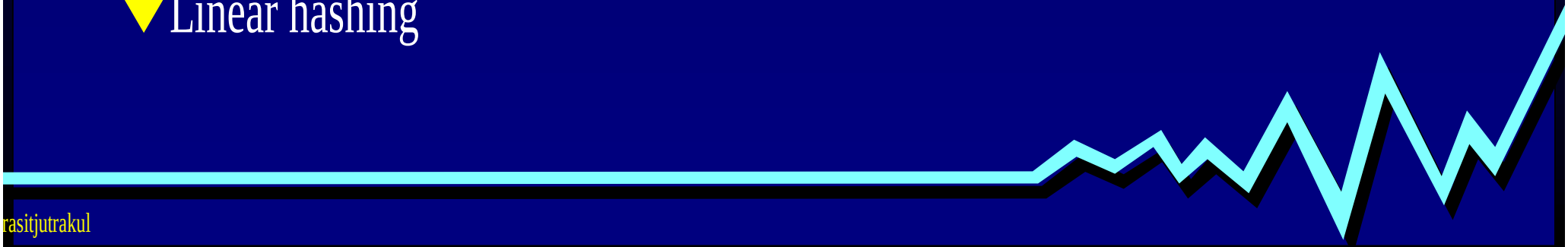
Overflow Management Techniques

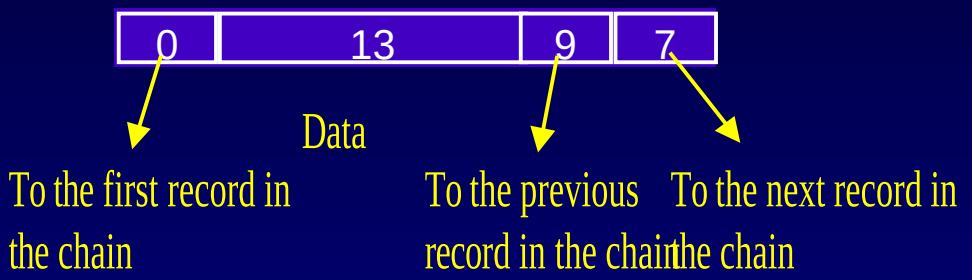
- Open addressing
  - Linear probing (primary clustering)
  - Quadratic probing (secondary clustering)
  - Double hashing (≈ random hashing)
- Chaining
  - Coalesced chaining
  - Separate overflow area
- Extendible hashing
- Linear hashing



Coalesced Chaining

- Buckets in the hash table are used for both home address and overflow records.

0	0	13	9	7
1	1	1	1	9
2	2	15	x	x
3	3	16	x	x
4	4	17	x	x
5	5	31	x	x
6	6	19	x	x
7	x	26	0	8
8	x	0	7	x
9	x	14	1	x
10	x		12	11
11	x		10	12
12	x		11	10



Coalesced Chaining

FreeChain 0

0	x		x	1
1	x		x	2
2	x		x	3
3	x		x	4
4	x		x	5
5	x		x	6
6	x		x	7
7	x		x	8
8	x		x	9
9	x		x	10
10	x		x	11
11	x		x	12
12	x		x	x

FreeChain 1

0	0	13	x	x
1	x		x	2
2	x		x	3
3	x		x	4
4	x		x	5
5	x		x	6
6	x		x	7
7	x		x	8
8	x		x	9
9	x		x	10
10	x		x	11
11	x		x	12
12	x		x	x

$h(key) = key \bmod 13$

13, 26, 0, 1, 14, 17, 15, 16, 19, 31

Coalesced Chaining (w/o chain repair)

FreeChain 3

0	0	13	x	1
1	x	26	0	2
2	x	0	1	x
3	x		x	4
4	x		x	5
5	x		x	6
6	x		x	7
7	x		x	8
8	x		x	9
9	x		x	10
10	x		x	11
11	x		x	12
12	x		x	x

FreeChain 10

0	0	13	x	x
1	3	26	0	2
2	6	0	1	x
3	7	1	x	4
4	5	14	3	x
5	9	17	x	x
6	8	15	x	x
7	x	16	x	x
8	x	19	x	x
9	x	31	x	x
10	x		x	11
11	x		x	12
12	x		x	x

$h(key) = key \bmod 13$

13, 26, 0, 1, 14, 17, 15, 16, 19, 31

Coalesced Chaining (w/ chain repair)

FreeChain 3

0	0	13	x	1
1	x	26	0	2
2	x	0	1	x
3	x		x	4
4	x		x	5
5	x		x	6
6	x		x	7
7	x		x	8
8	x		x	9
9	x		x	10
10	x		x	11
11	x		x	12
12	x		x	x

FreeChain 4

0	0	13	x	3
1	1	1	x	x
2	x	0	3	x
3	x	26	0	2
4	x		x	5
5	x		x	6
6	x		x	7
7	x		x	8
8	x		x	9
9	x		x	10
10	x		x	11
11	x		x	12
12	x		x	x

$h(key) = key \bmod 13$

13, 26, 0, 1, 14, 17, 15, 16, 19, 31

Coalesced Chaining (w/ chain repair)

FreeChain 5

0	0	13	x	3
1	1	1	x	4
2	x	0	3	x
3	x	26	0	2
4	x	14	1	x
5	x		x	6
6	x		x	7
7	x		x	8
8	x		x	9
9	x		x	10
10	x		x	11
11	x		x	12
12	x		x	0

FreeChain 6

0	0	13	x	3
1	1	1	x	5
2	x	0	3	x
3	x	26	0	2
4	4	17	x	x
5	x	14	1	x
6	x		x	7
7	x		x	8
8	x		x	9
9	x		x	10
10	x		x	11
11	x		x	12
12	x		x	0

$h(key) = key \bmod 13$

13, 26, 0, 1, 14, 17, 15, 16, 19, 31

Coalesced Chaining (w/ chain repair)

FreeChain 7

0	0	13	x	3
1	1	1	x	6
2	2	15	x	x
3	x	26	0	6
4	4	17	x	x
5	x	14	1	x
6	x	0	3	x
7	x		x	8
8	x		x	9
9	x		x	10
10	x		x	11
11	x		x	12
12	x		x	0

FreeChain 10

0	0	13	x	7
1	1	1	x	9
2	2	15	x	x
3	3	16	x	x
4	4	17	x	x
5	5	31	x	x
6	6	19	x	x
7	x	26	0	8
8	x	0	7	x
9	x	14	1	x
10	x		x	11
11	x		x	12
12	x		x	0

$h(key) = key \bmod 13$

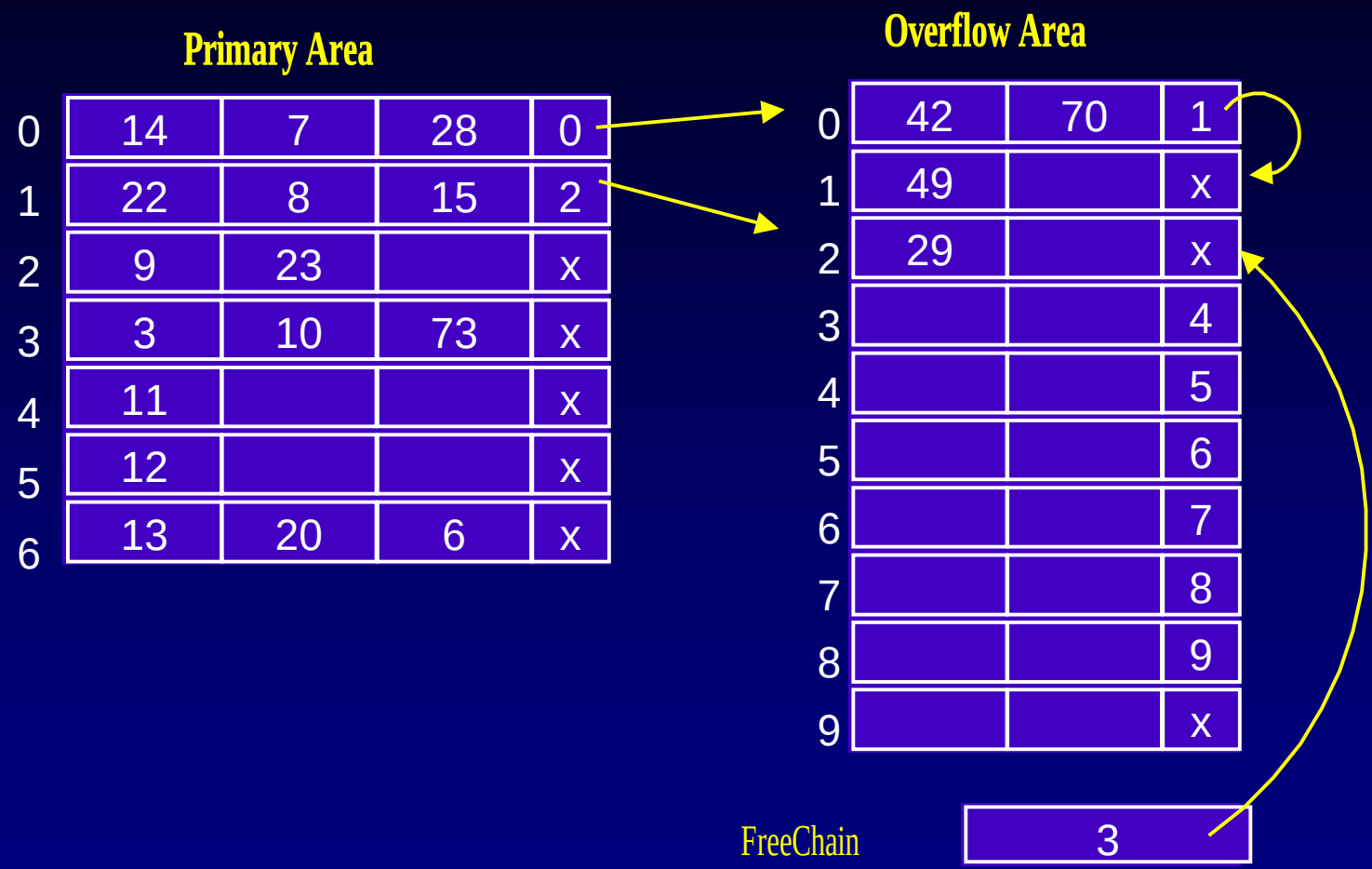
13, 26, 0, 1, 14, 17, 15, 16, 19, 31

Coalesced Chaining : *SL*

- In a chain of k records
- access i-th record takes i *rba*
- Retrieve all records in the chain independently = $k(k+1) / 2$
- $D_h(k)$ = number of chains whose length = k

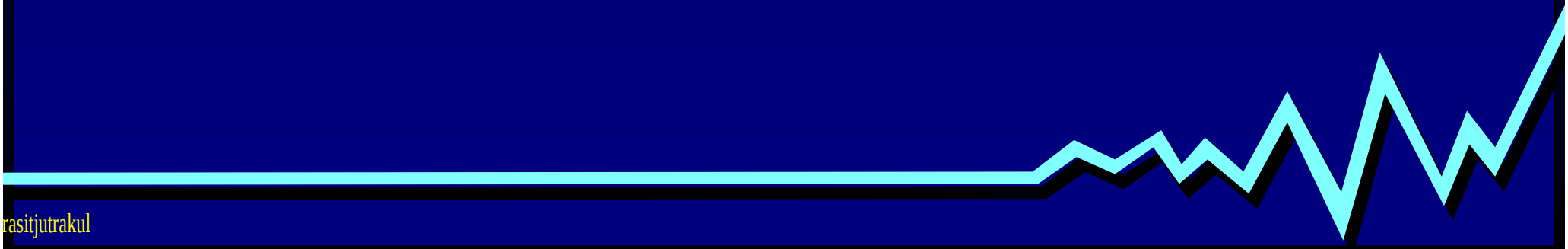
$$SL_{avg}[RetrieveOne, H] = \frac{1}{NR} \sum_{k=1}^{\infty} D_h(k) \cdot \frac{k(k+1)}{2}$$

Separate Overflow Area

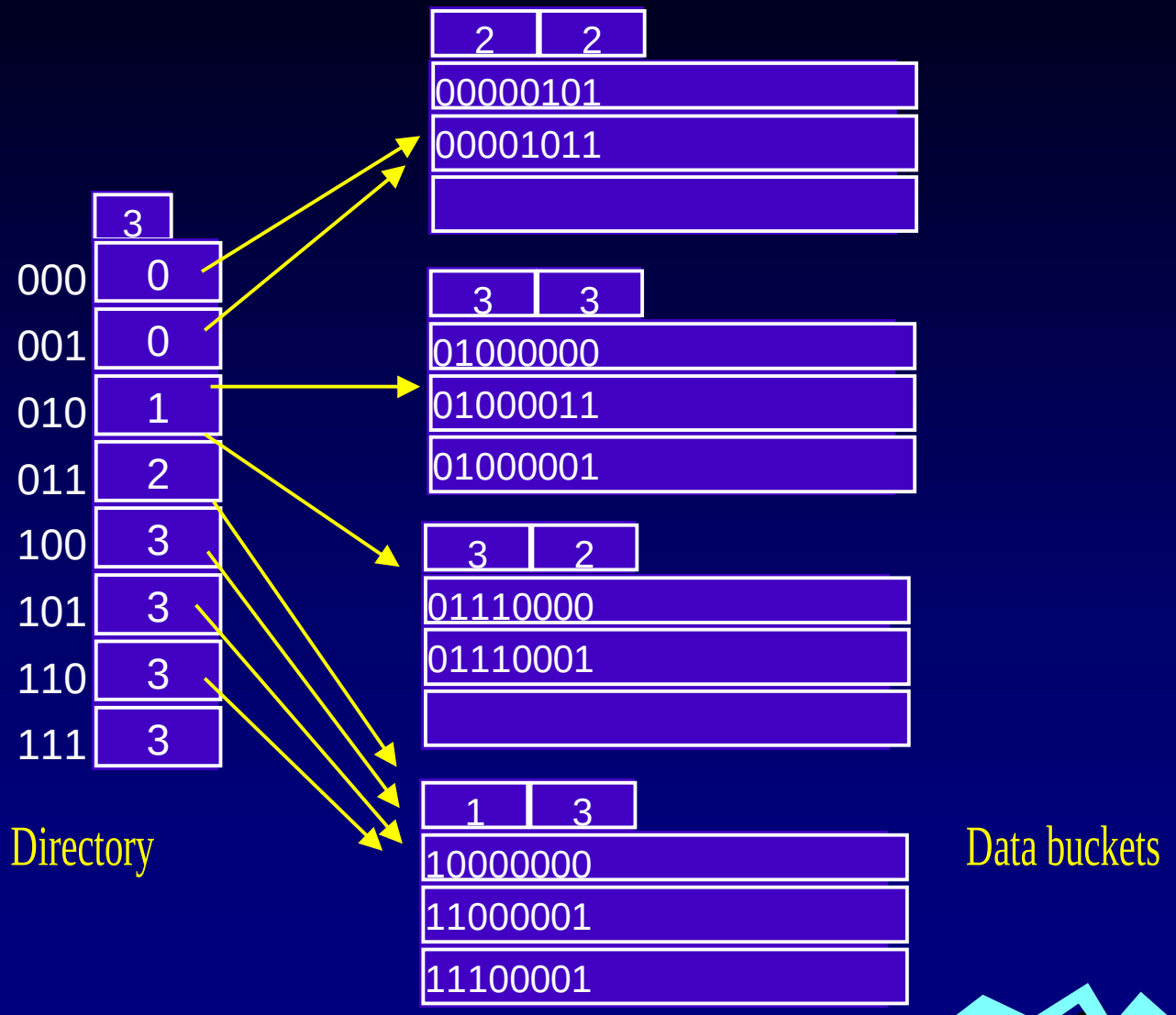


Extendible Hashing

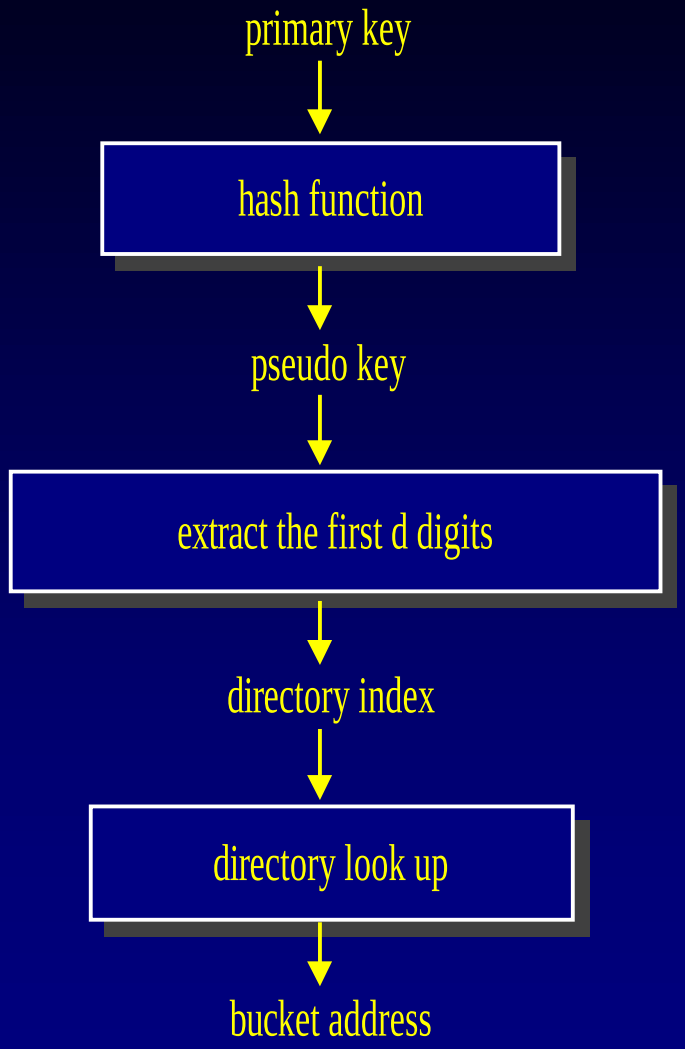
- Fixed-size hash table needs rehashing and reorganization if file size increases.
- Extendible hashing : dynamic file structure
  - two-level structure (directory & data blocks)
  - expand to accommodate new records
  - contract to keep load factor high
  - *RetrieveOne* needs only one *rba*



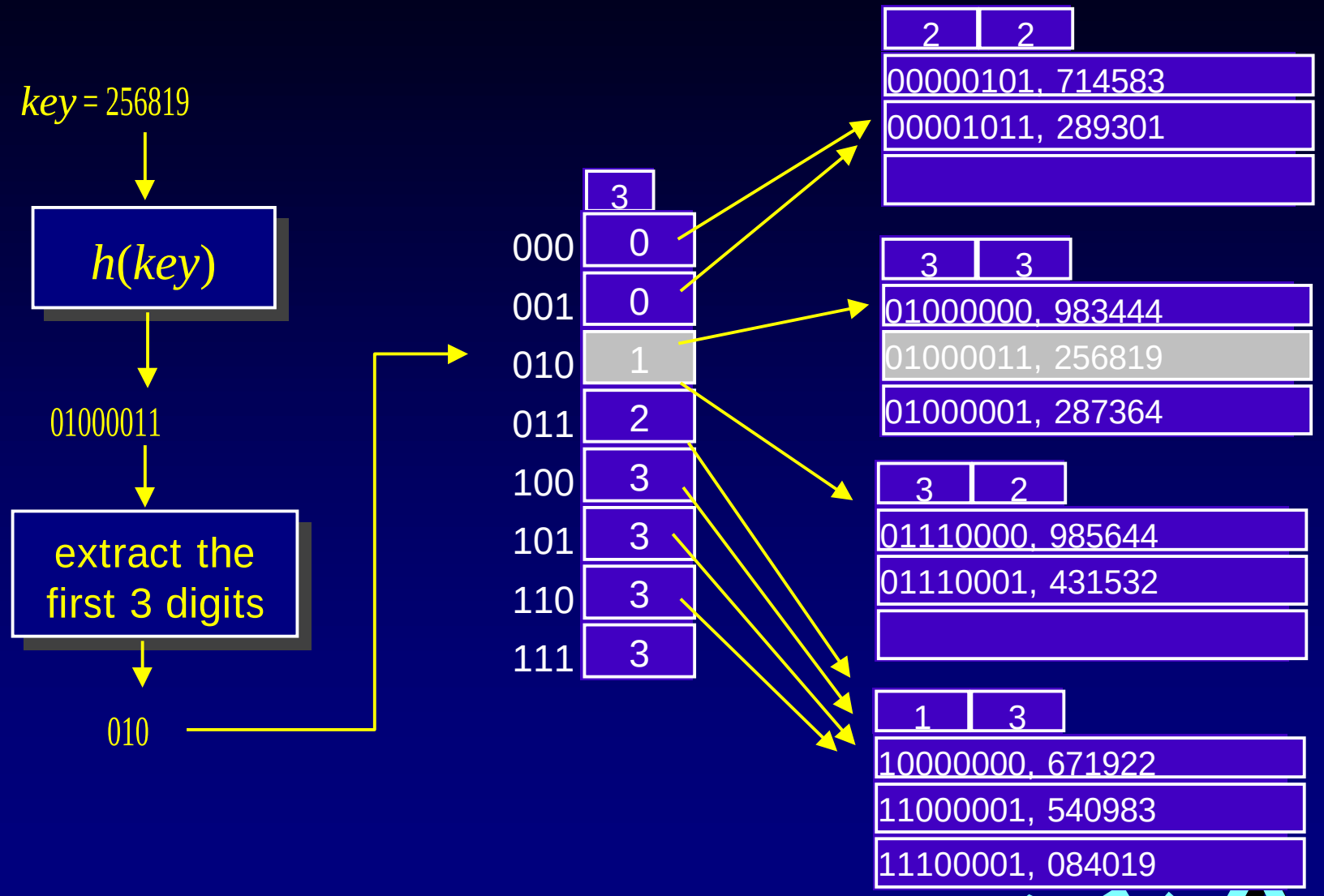
Extendible Hashing : Physical File



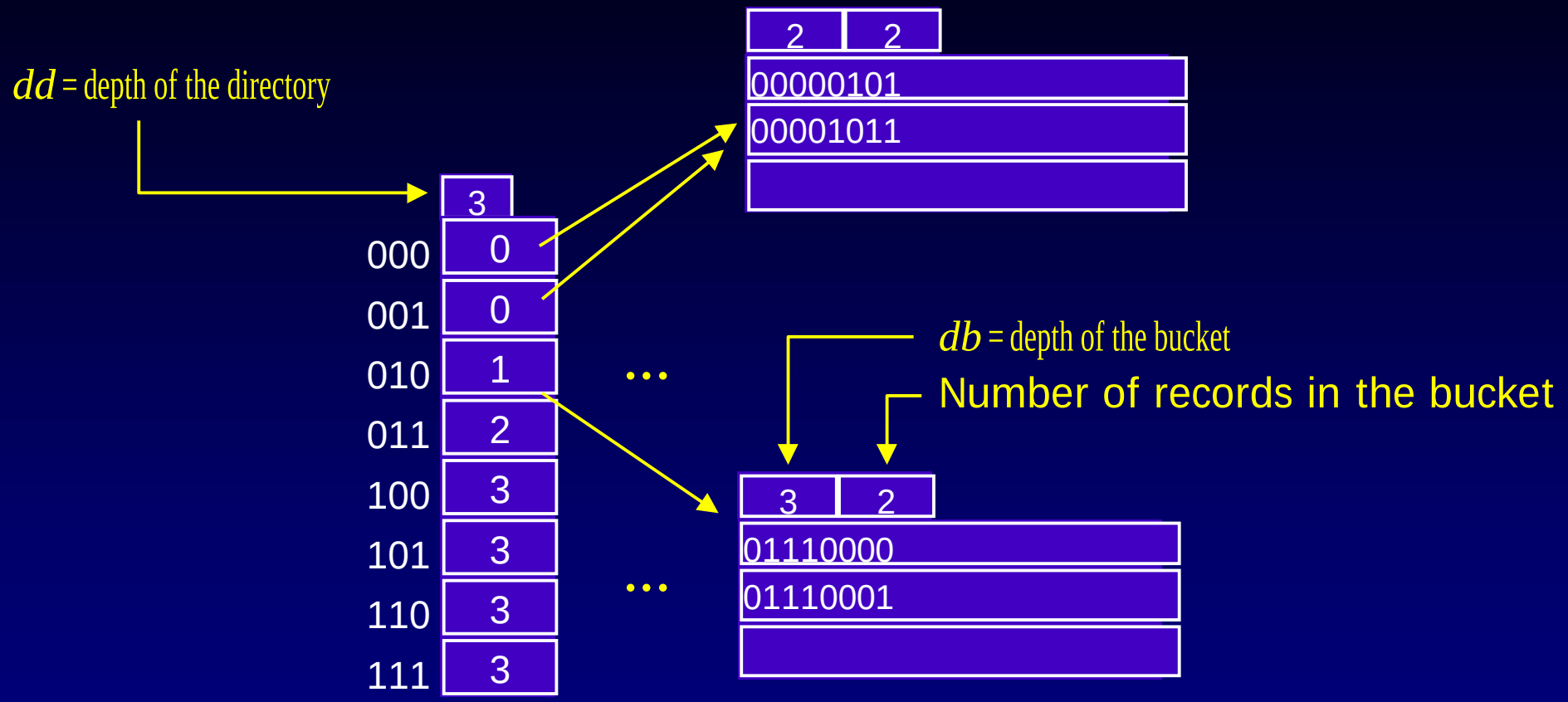
Key Transformation



Extendible Hashing : *RetrieveOne*

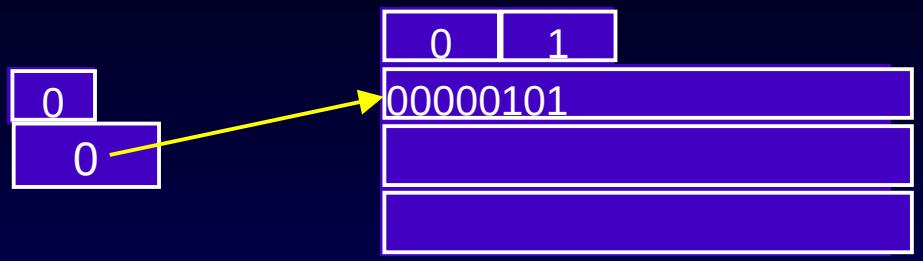


Extendible Hashing : Physical File

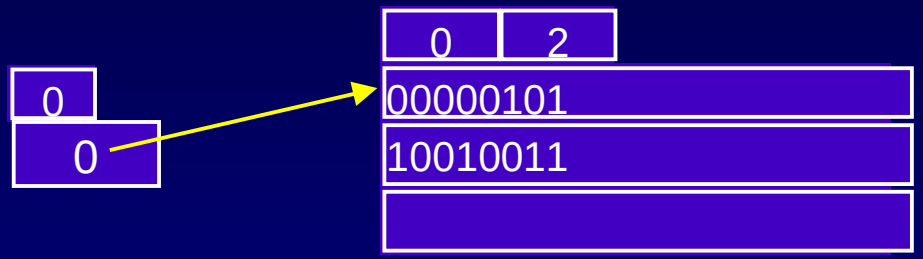


Directory has 2^{dd} entries. $db \leq dd$

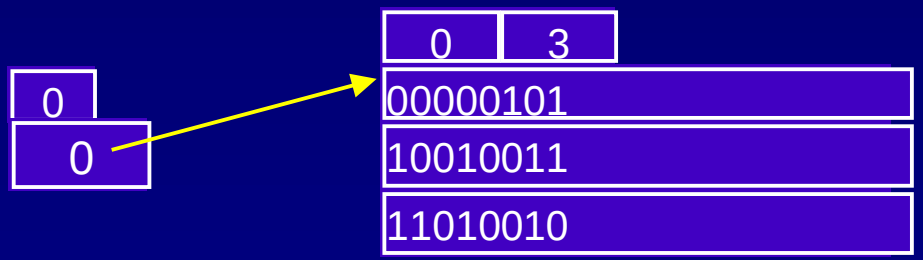
Extendible Hashing : *InsertOne*



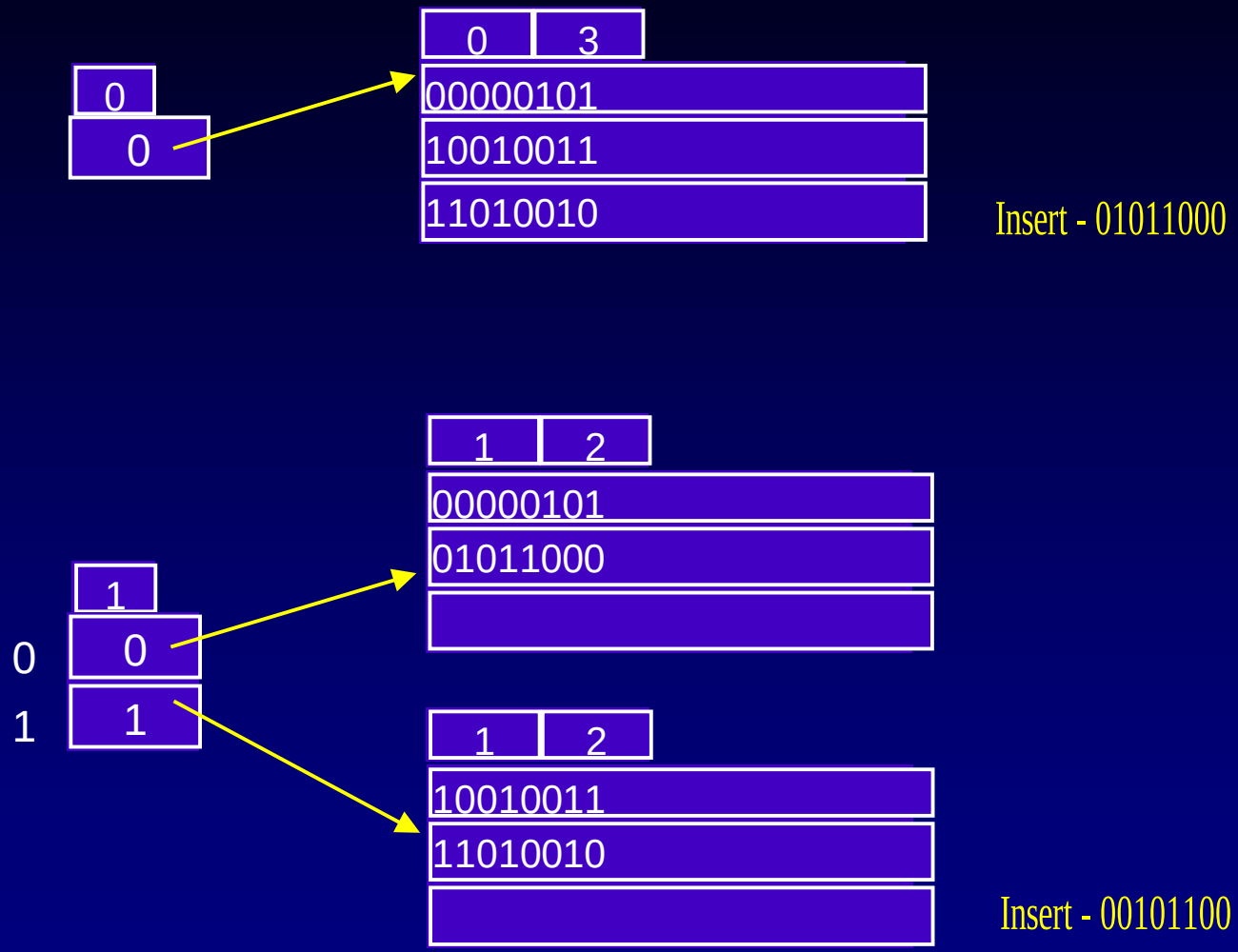
Insert - 10010011



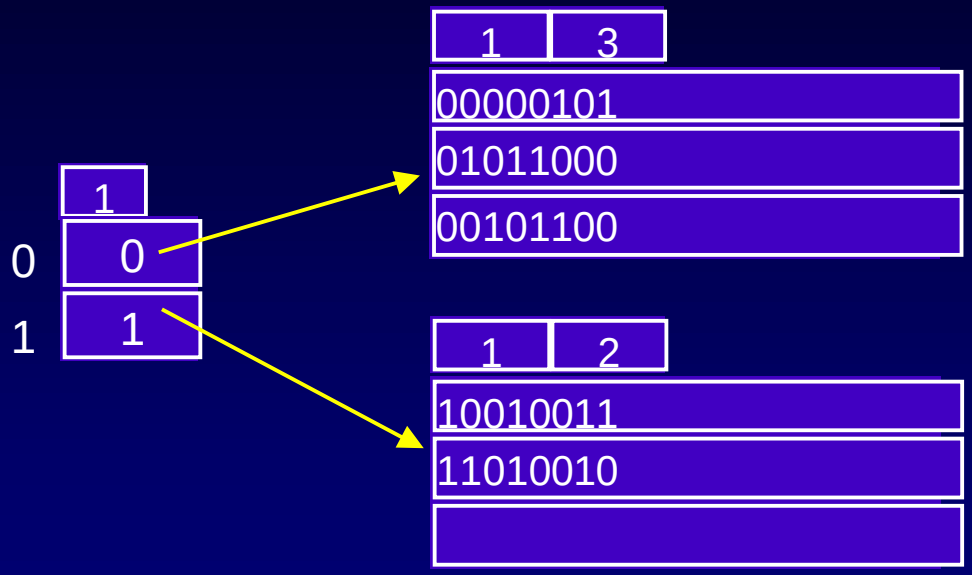
Insert - 11010010



Extendible Hashing : *InsertOne*

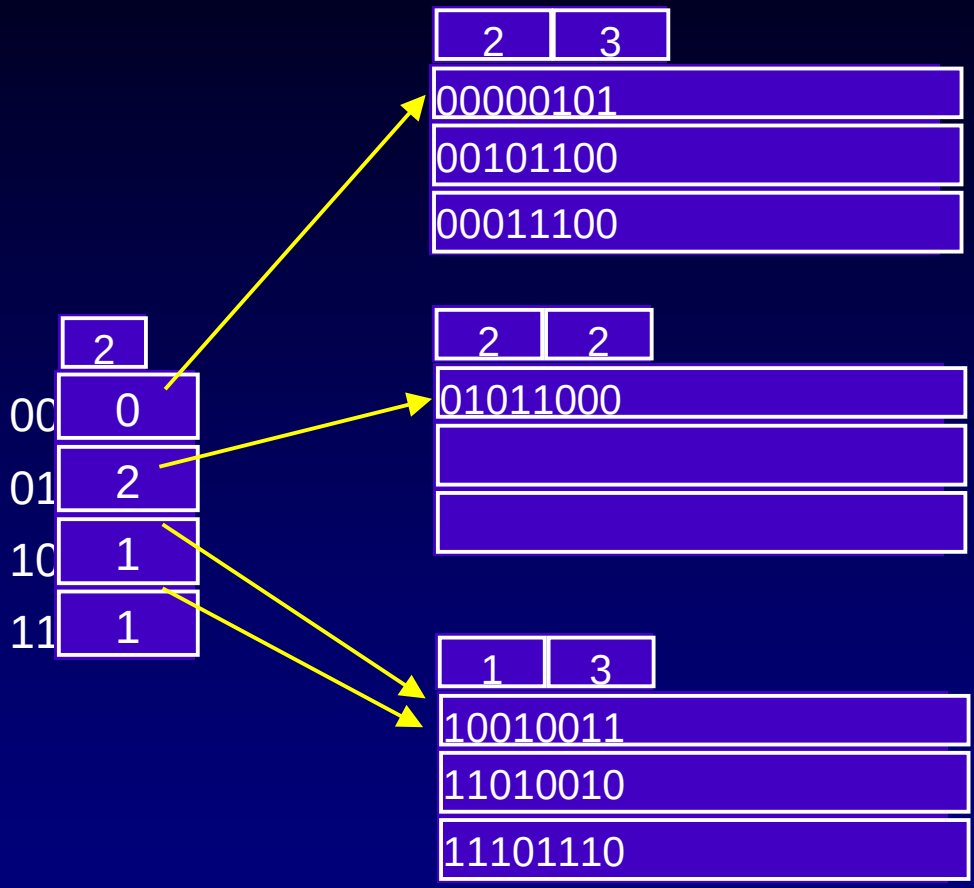


Extendible Hashing : *InsertOne*

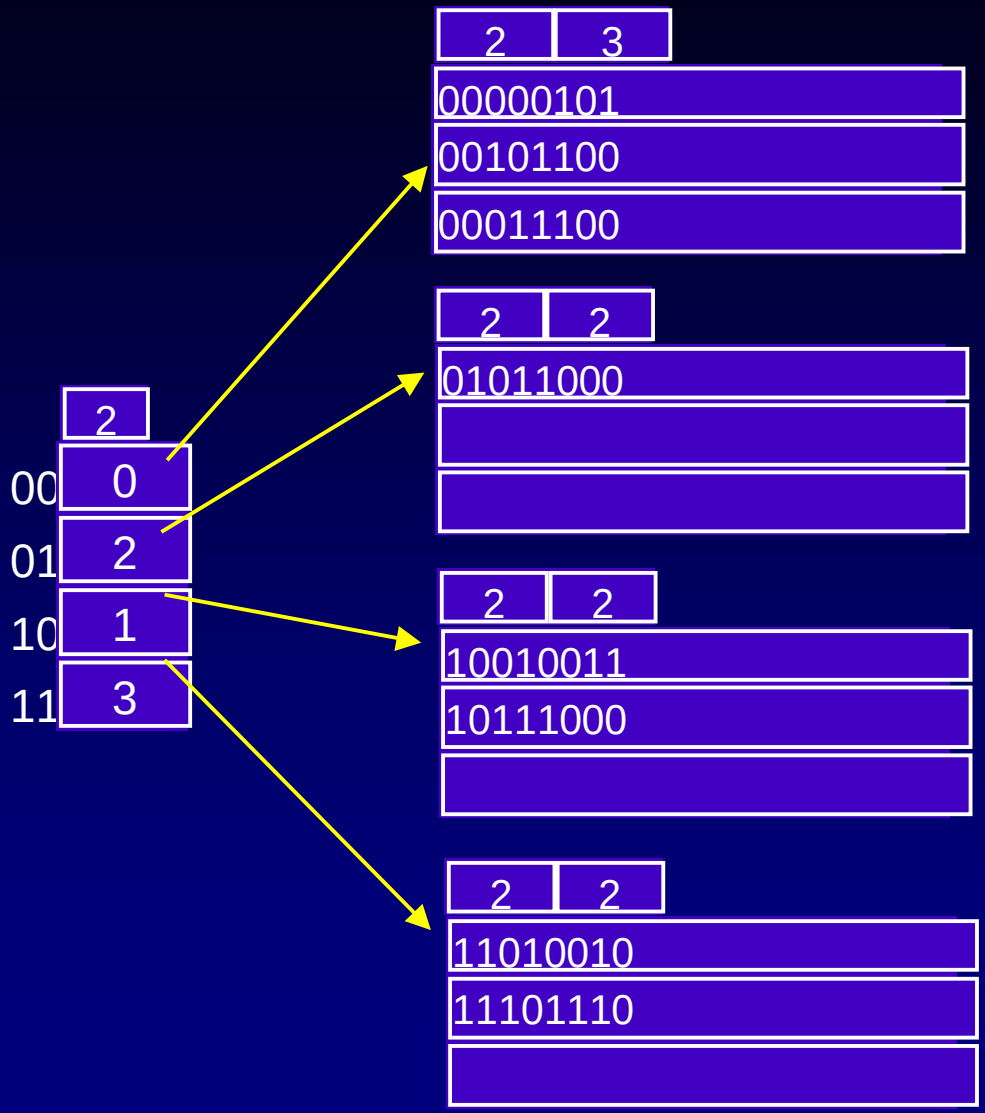


Insert - 00011100

Extendible Hashing : *InsertOne*



Extendible Hashing : *InsertOne*



Extendible Hashing : *DeleteOne*

- Delete a record
  - buckets may be combined
  - directory may be collapsed
- Which buckets can be combined ?
  - buddy bucket - bucket formed from a split

Space Utilization

- Space utilization for buckets
  - Simulation shows that space utilization is periodic and fluctuates between 53% to 94%.
  - Average utilization is 69%.
- Space utilization for directory
  - the directory grows by doubling its size
  - number of records, bucket size

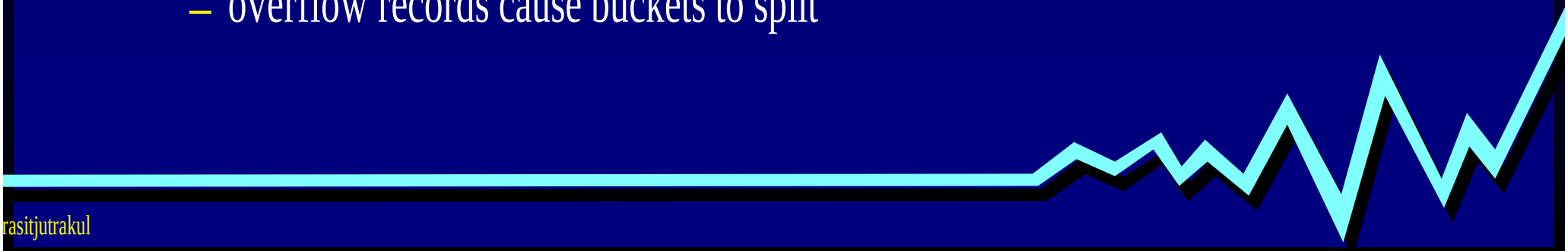
Directory Size

r	b=5	b=10	b=20	b=50	b=100	b=200
1K	1.5K	0.30K	0.10K	0.00K	0.00K	0.00K
10K	25.60K	4.80K	1.70K	0.50K	0.20K	0.00K
100K	424.10K	68.20K	16.80K	4.10K	2.00K	1.00K
1M	6.90M	1.02M	0.26M	62.50K	16.8K	8.10K
10M	111.11M	11.64M	2.25M	0.52M	0.26M	0.13M

Estimated directory size $\approx \frac{3.92}{b} \cdot r^{(1+1/b)}$

Linear Hashing

- Extendible Hashing (1979)
  - expand directory w/o expanding # of buckets.
  - adds an additional layer of indirection.
  - if directory can be in RAM, additional seek(s).
  - directory must be loaded and rewritten back.
- Linear Hashing (1980)
  - no directory
  - buckets expand linearly
  - overflow records cause buckets to split



Key-to-Address Transformation

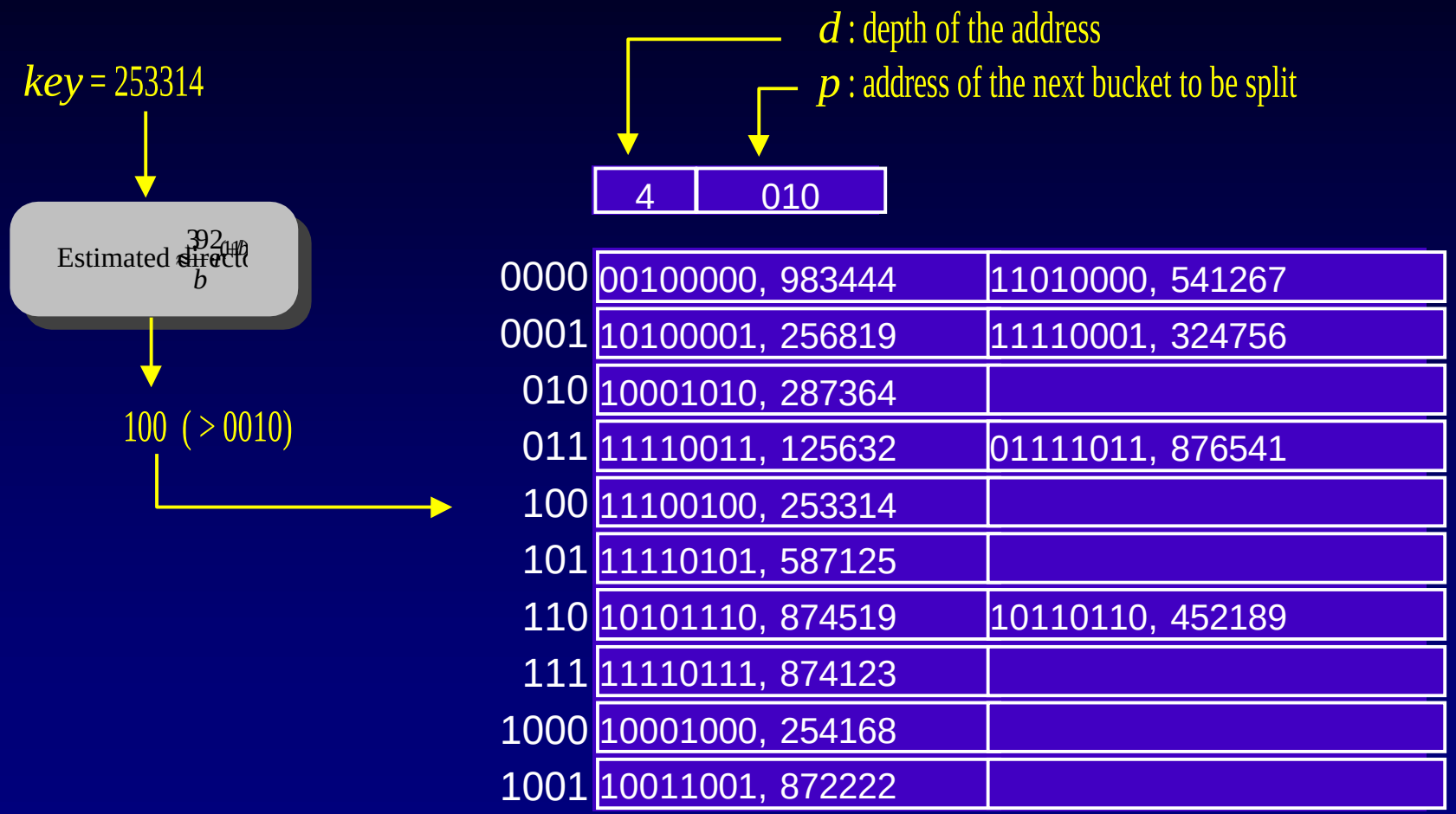
Let d be the depth of the address

p be the address of the next bucket to be split

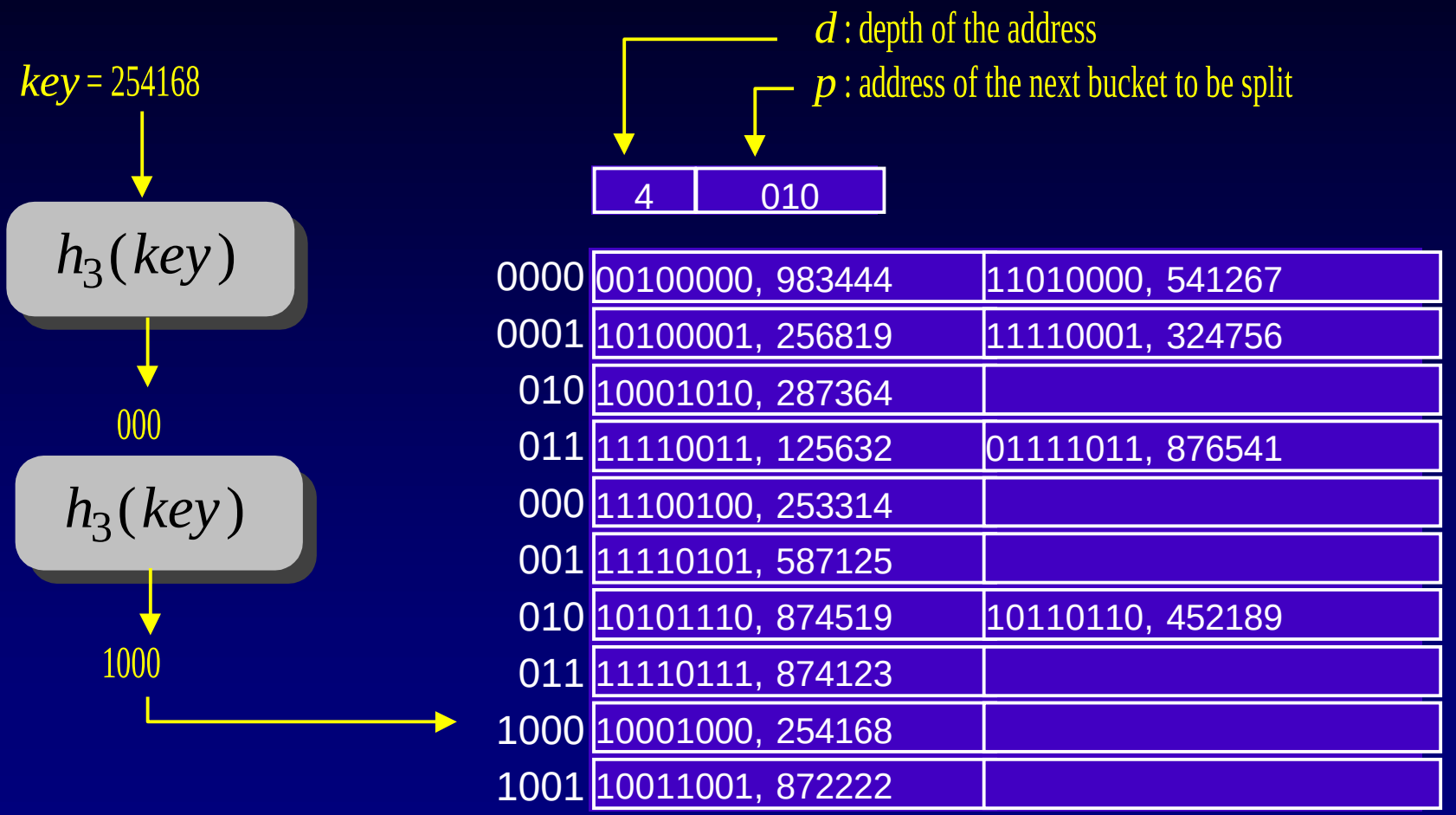
Estimated directory size $\approx \frac{3.92}{b} \cdot r^{(1+1/b)}$

Estimated $\approx \frac{3.92}{b} \cdot r^{(1+1/b)}$

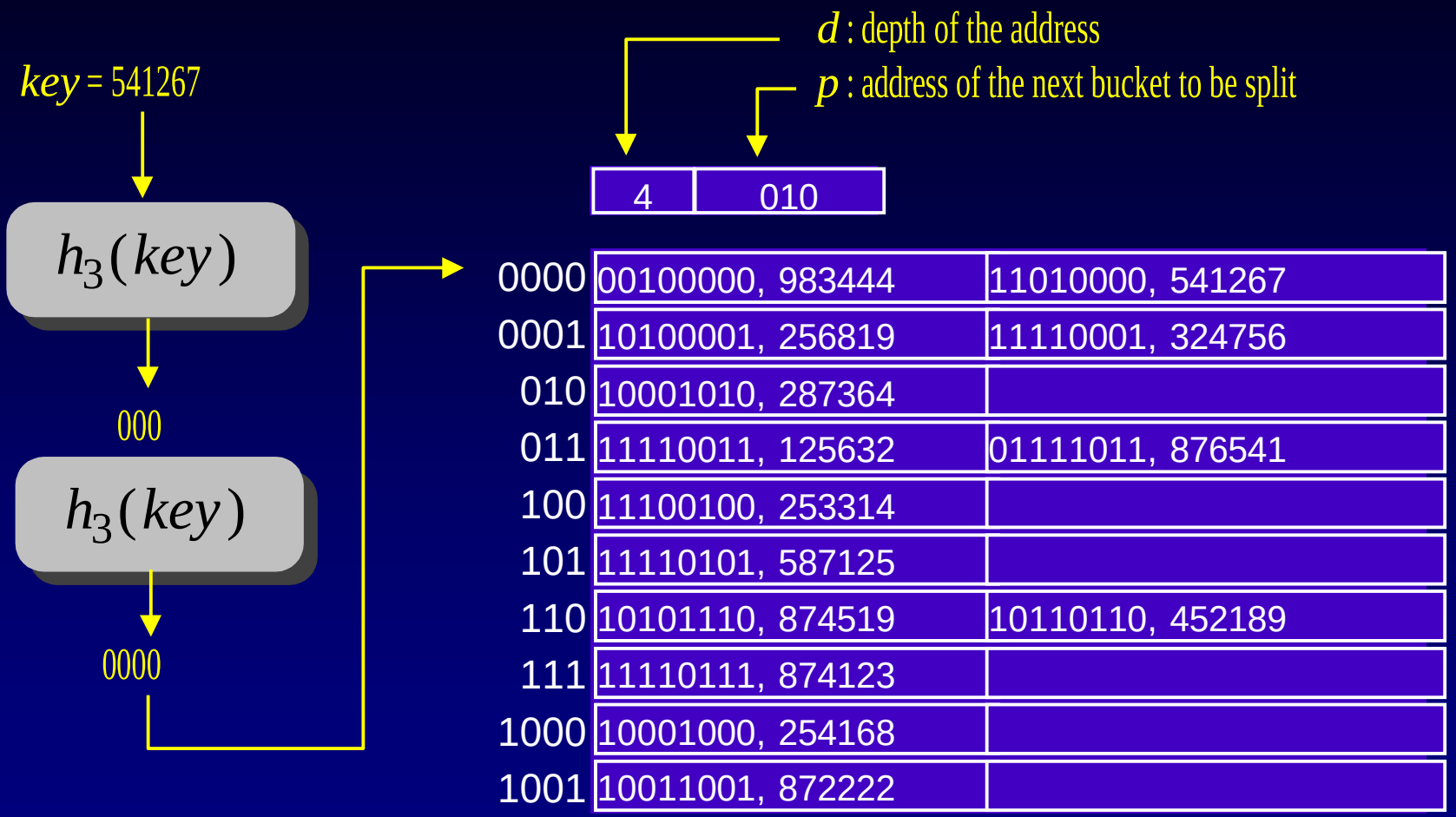
Linear Hashing : *RetrieveOne*



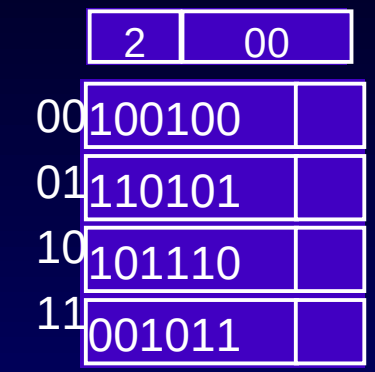
Linear Hashing : *RetrieveOne*

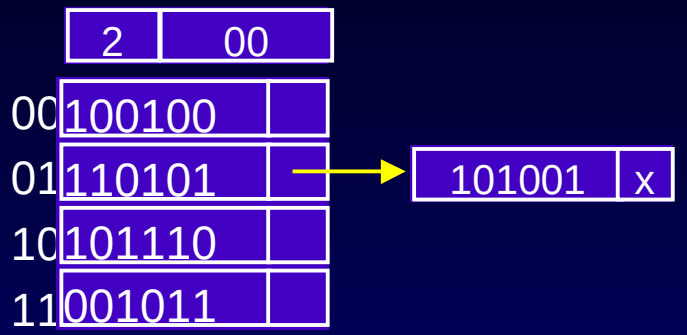


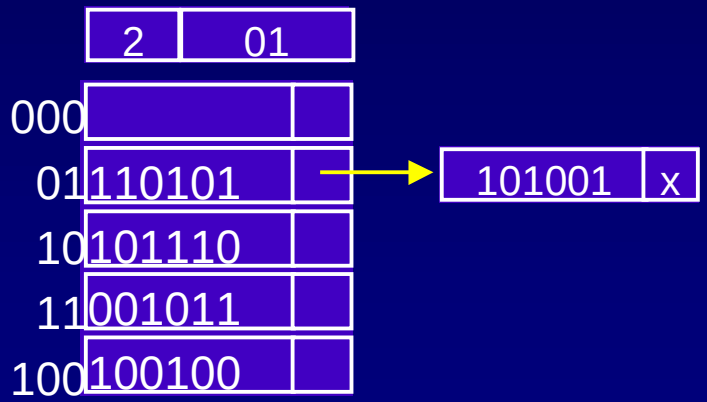
Linear Hashing : *RetrieveOne*

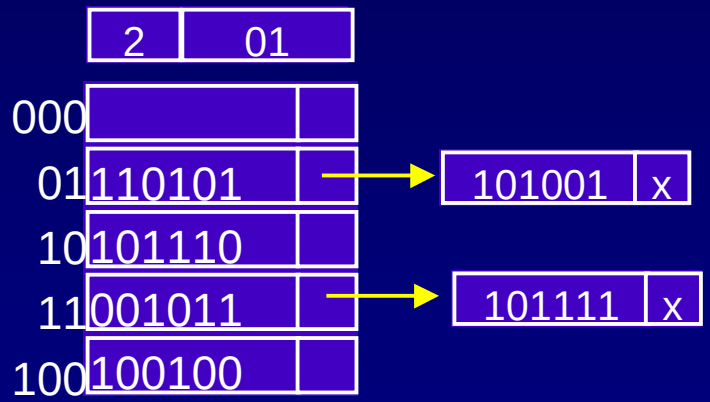


Linear Hashing : *InsertOne*

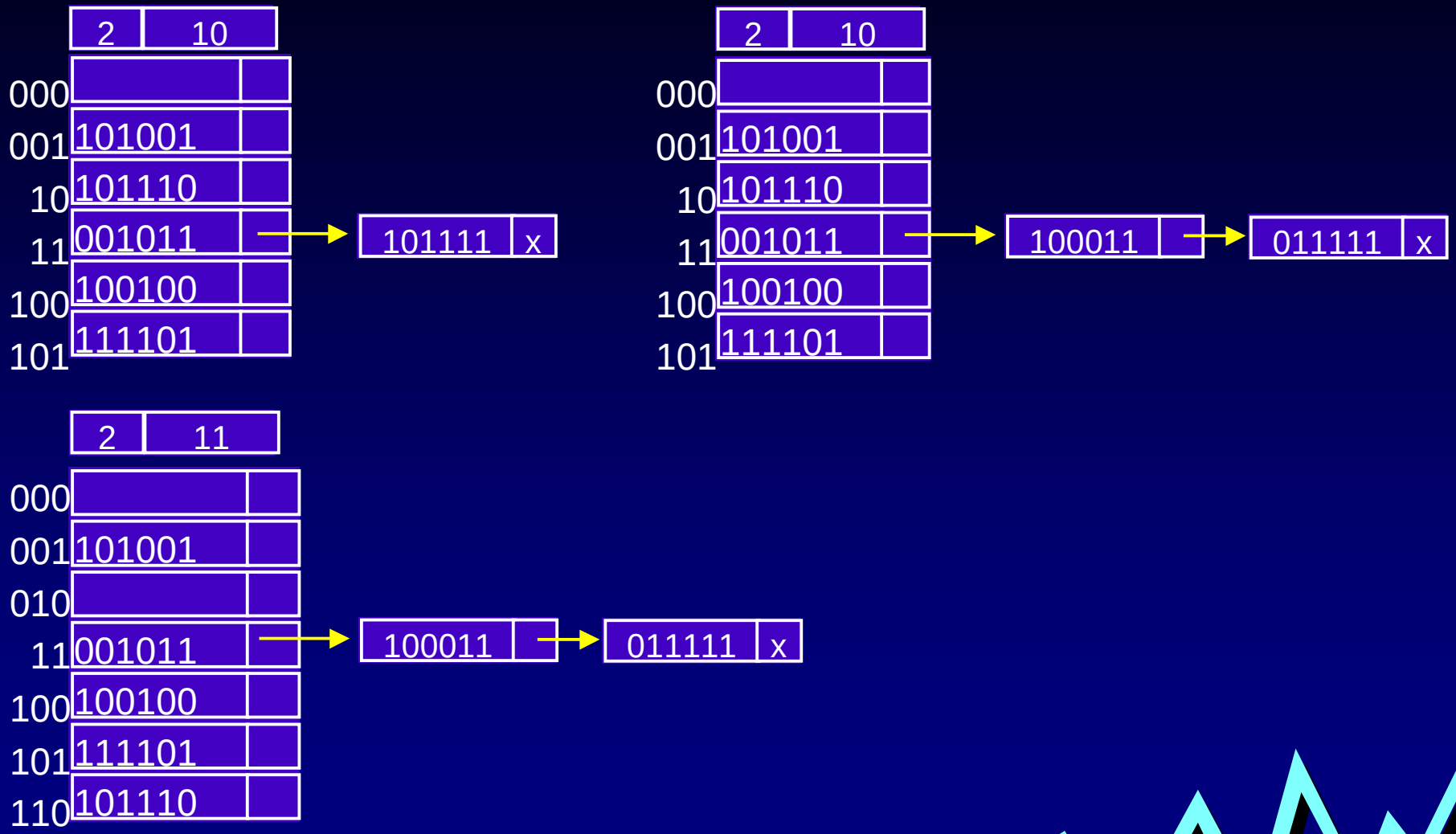








Linear Hashing : *InsertOne*



Linear Hashing :Splitting

- split a bucket every time any bucket overflows.
- split buckets until the bucket that overflows is split.
- use overflow buckets to defer splitting, then split when load factor > threshold
  - experiment : set threshold = 75%, average search length of *RetrieveOne* < 2.