Discussion 7

Hash Indexes
EECS 484

Logistics

- Midterm Grades released, regrade requests due **Today**
- **Project 3** due **Nov 1st** at 11:45 PM ET
- **HW 4** released, due **Nov 8th** at 11:45 PM ET

Hashing

Static Hashing

- Allocate a giant array that has one slot for every element you need
- Linear probe hashing, robin hood hashing, cuckoo hashing
  - Refer to the lecture slides for detailed walkthrough
- Static structure can be problematic for DBMS
  - Number of buckets is fixed ahead of time
  - Could periodically double buckets and rehash file, but…
    - Entire file has to be read and written
    - Index unavailable while rehashing

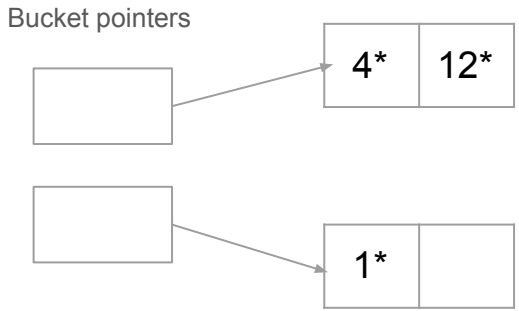
Dynamic Hashing

- Dynamically allocate “buckets” of storage on demand to store data
- 3 types
  - Chained Hashing
  - Extendible Hashing
  - Linear Hashing

Chained Hashing

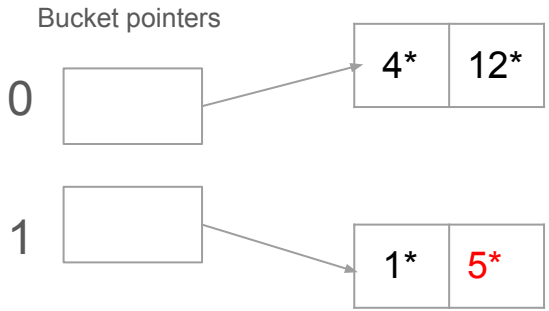
Chained Hashing

- Maintain a linked list of buckets for each slot in the hash table.
- Resolve collisions by placing all elements with the same hash key into the same bucket.



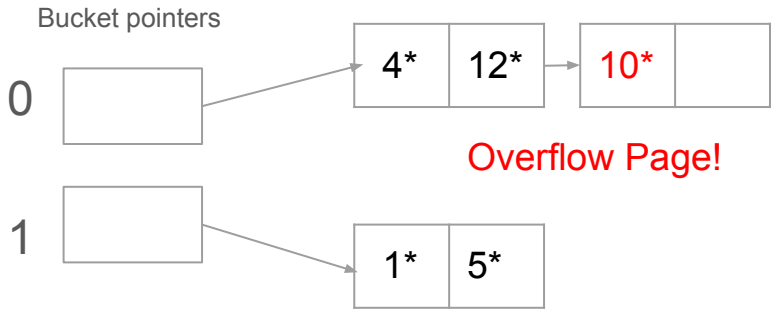
Chained Hashing

- Maintain a linked list of buckets for each slot in the hash table.
- Resolve collisions by placing all elements with the same hash key into the same bucket.
- Insert 5*
  - Bucket = key % 2



Chained Hashing

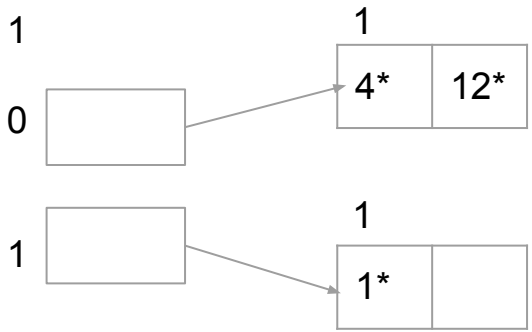
- Maintain a linked list of buckets for each slot in the hash table.
- Resolve collisions by placing all elements with the same hash key into the same bucket.
- Insert 10*
  - Bucket = key % 2
  - Consider many insertions
  - Long overflow chains can develop and degrade performance
  - O(N / # buckets) insertion time



Extendible Hashing

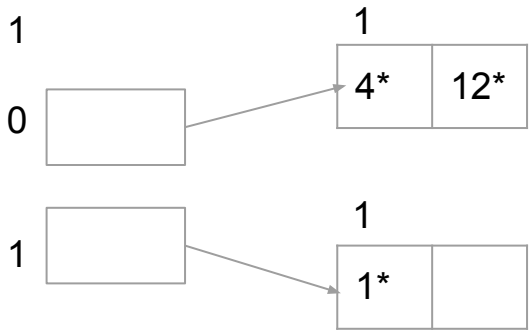
Extendible Hashing

- Use directory of pointers
  - Split on overflow
  - Once we're out of room in directory, double size
  - Global depth = number of bits considered globally
  - Local depth = number of bits considered locally



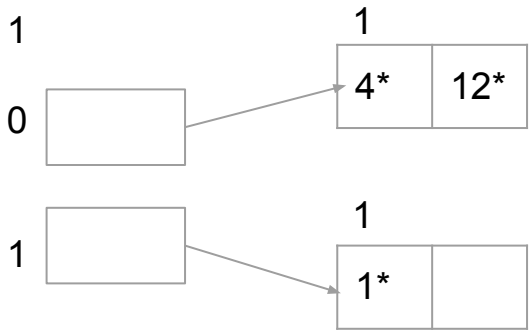
Extendible Hashing

- Use directory of pointers
  - Split on overflow
  - Once we're out of room in directory, double size
  - Global depth = number of bits considered globally
  - Local depth = number of bits considered locally
- Insert 5*



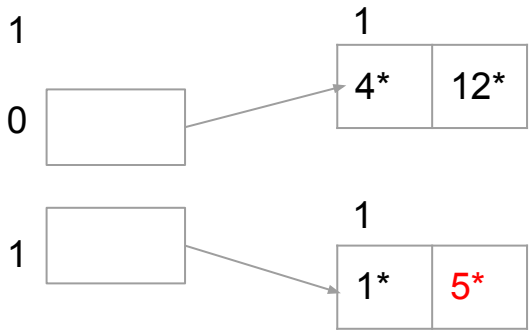
Extendible Hashing

- Use directory of pointers
  - Split on overflow
  - Once we're out of room in directory, double size
  - Global depth = number of bits considered globally
  - Local depth = number of bits considered locally
- Insert 5*
  - 5=0b101



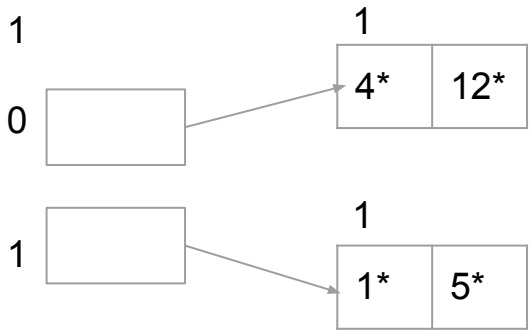
Extendible Hashing

- Use directory of pointers
  - Split on overflow
  - Once we're out of room in directory, double size
  - Global depth = number of bits considered globally
  - Local depth = number of bits considered locally
- Insert 5*
  - 5=0b101



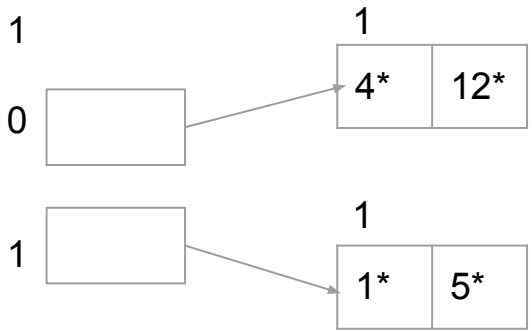
Extendible Hashing

- Use directory of pointers
  - Split on overflow
  - Once we're out of room in directory, double size
  - Global depth = number of bits considered globally
  - Local depth = number of bits considered locally
- Insert 5*
  - 5=0b101
  - Done :)



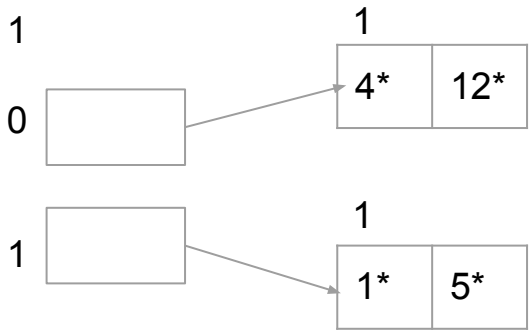
Extendible Hashing

- Use directory of pointers
  - Split on overflow
  - Once we're out of room in directory, double size
  - Global depth = number of bits considered globally
  - Local depth = number of bits considered locally
- Insert 10*



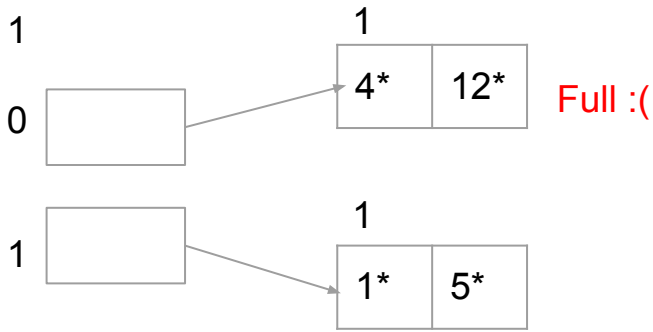
Extendible Hashing

- Use directory of pointers
  - Split on overflow
  - Once we're out of room in directory, double size
  - Global depth = number of bits considered globally
  - Local depth = number of bits considered locally
- Insert 10*
  - 10=0b1010



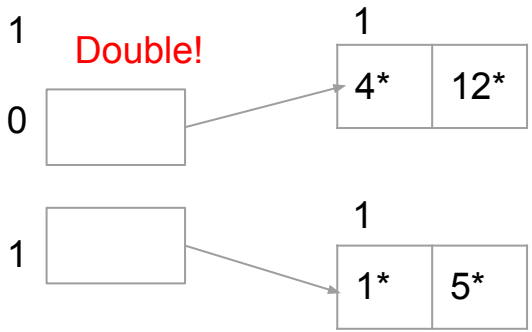
Extendible Hashing

- Use directory of pointers
  - Split on overflow
  - Once we're out of room in directory, double size
  - Global depth = number of bits considered globally
  - Local depth = number of bits considered locally
- Insert 10*
  - 10=0b1010



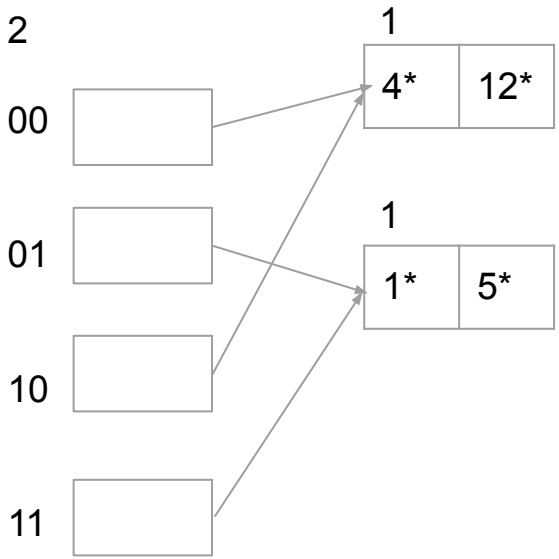
Extendible Hashing

- Use directory of pointers
  - Split on overflow
  - Once we're out of room in directory, double size
  - Global depth = number of bits considered globally
  - Local depth = number of bits considered locally
- Insert 10*
  - 10=0b1010



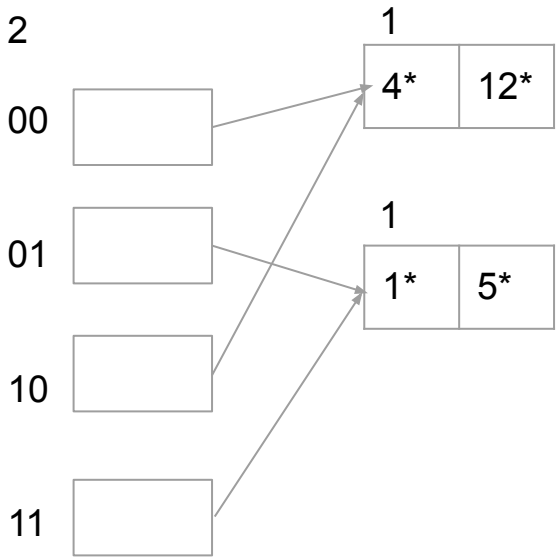
Extendible Hashing

- Use directory of pointers
  - Split on overflow
  - Once we're out of room in directory, double size
  - Global depth = number of bits considered globally
  - Local depth = number of bits considered locally
- Insert 10*
  - 10=0b1010
  - New directories point to "Split image"



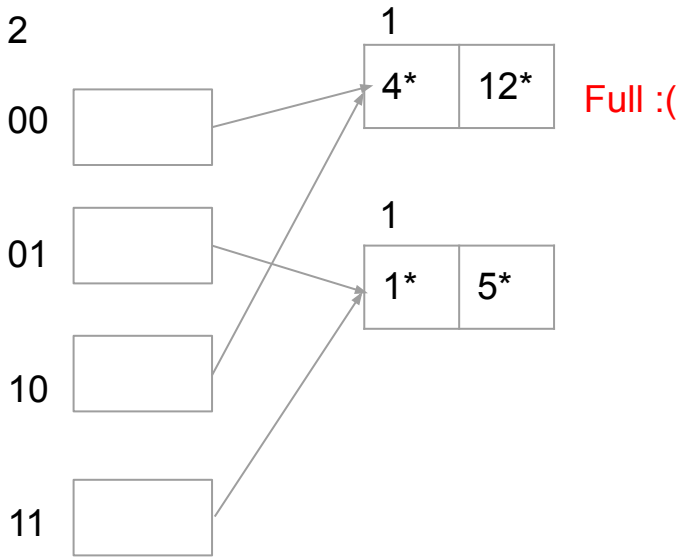
Extendible Hashing

- Use directory of pointers
  - Split on overflow
  - Once we're out of room in directory, double size
  - Global depth = number of bits considered globally
  - Local depth = number of bits considered locally
- Insert 10*
  - 10=0b1010
  - New directories point to "Split image"
  - Try to insert again



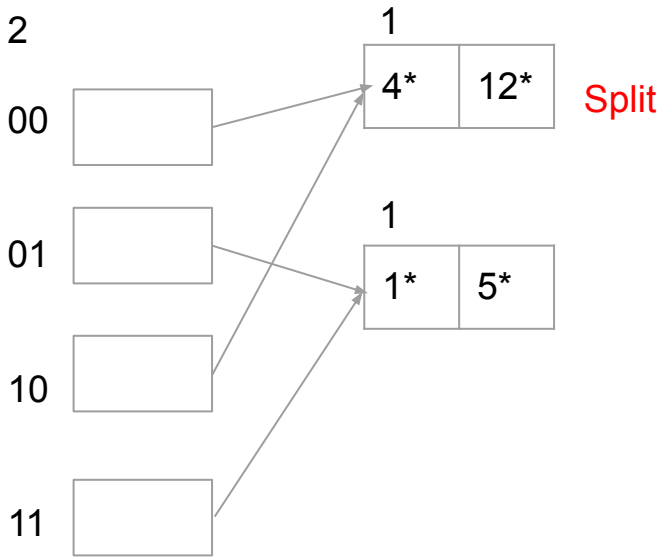
Extendible Hashing

- Use directory of pointers
  - Split on overflow
  - Once we're out of room in directory, double size
  - Global depth = number of bits considered globally
  - Local depth = number of bits considered locally
- Insert 10*
  - 10=0b1010
  - New directories point to "Split image"
  - Try to insert again



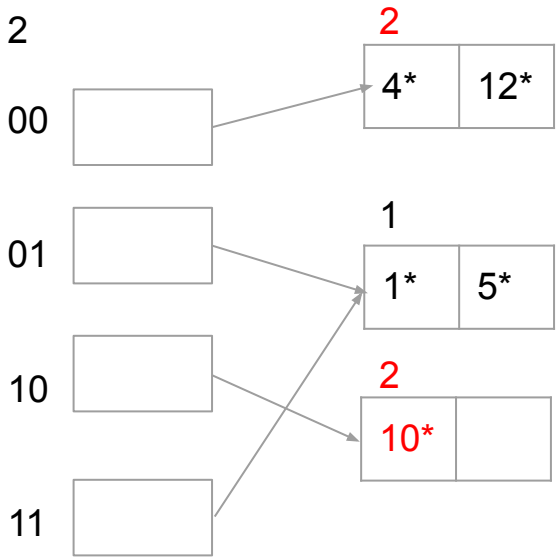
Extendible Hashing

- Use directory of pointers
  - Split on overflow
  - Once we're out of room in directory, double size
  - Global depth = number of bits considered globally
  - Local depth = number of bits considered locally
- Insert 10*
  - 10=0b1010
  - New directories point to "Split image"
  - Try to insert again



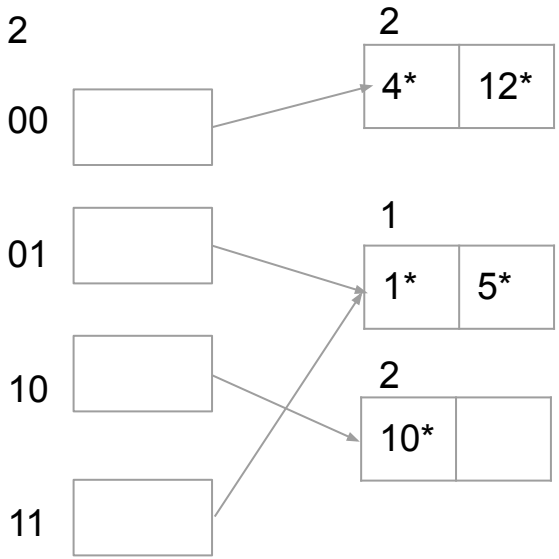
Extendible Hashing

- Use directory of pointers
  - Split on overflow
  - Once we're out of room in directory, double size
  - Global depth = number of bits considered globally
  - Local depth = number of bits considered locally
- Insert 10*
  - 10=0b1010
  - New directories point to "Split image"
  - Try to insert again



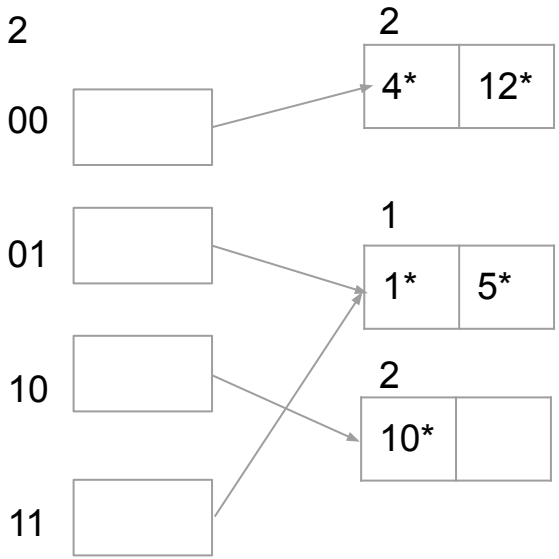
Extendible Hashing

- Use directory of pointers
  - Split on overflow
  - Once we're out of room in directory, double size
  - Global depth = number of bits considered globally
  - Local depth = number of bits considered locally
- Insert 10*
  - 10=0b1010
  - New directories point to "Split image"
  - Try to insert again
  - Done :)



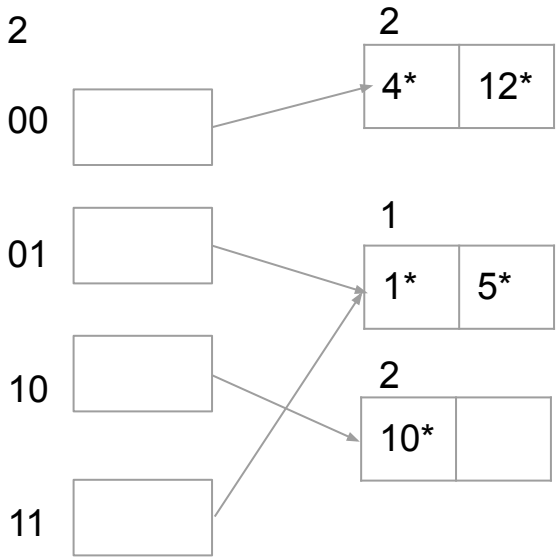
Extendible Hashing

- Use directory of pointers
  - Split on overflow
  - Once we're out of room in directory, double size
  - Global depth = number of bits considered globally
  - Local depth = number of bits considered locally
- Insert 7*



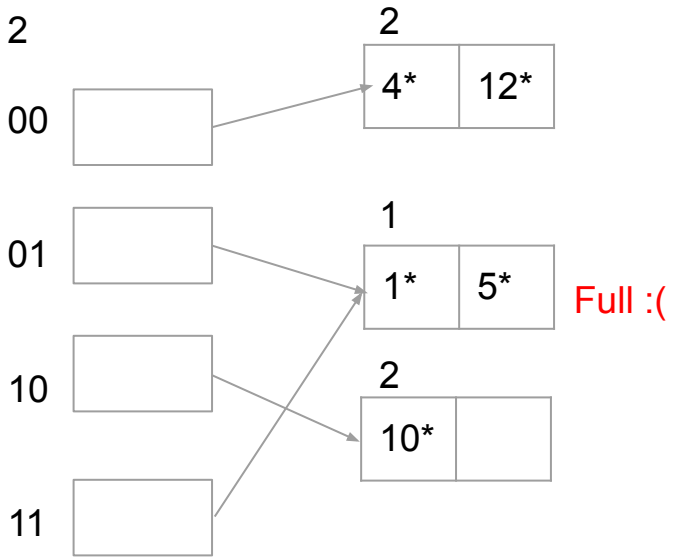
Extendible Hashing

- Use directory of pointers
  - Split on overflow
  - Once we're out of room in directory, double size
  - Global depth = number of bits considered globally
  - Local depth = number of bits considered locally
- Insert 7*
  - 7=0b111



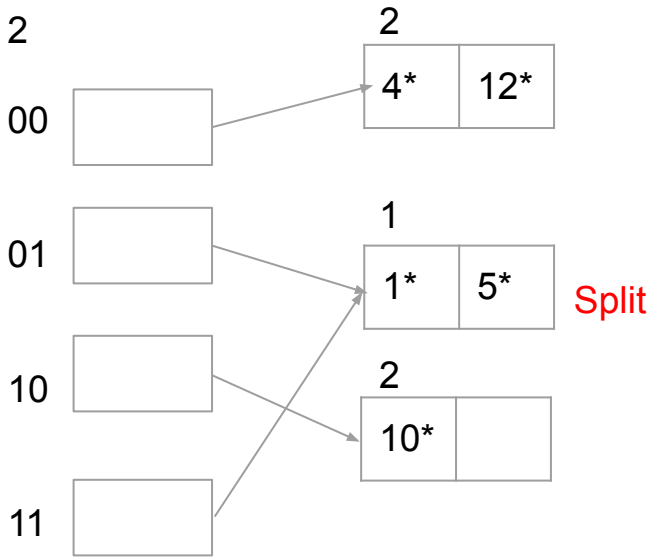
Extendible Hashing

- Use directory of pointers
  - Split on overflow
  - Once we're out of room in directory, double size
  - Global depth = number of bits considered globally
  - Local depth = number of bits considered locally
- Insert 7*
  - 7=0b111



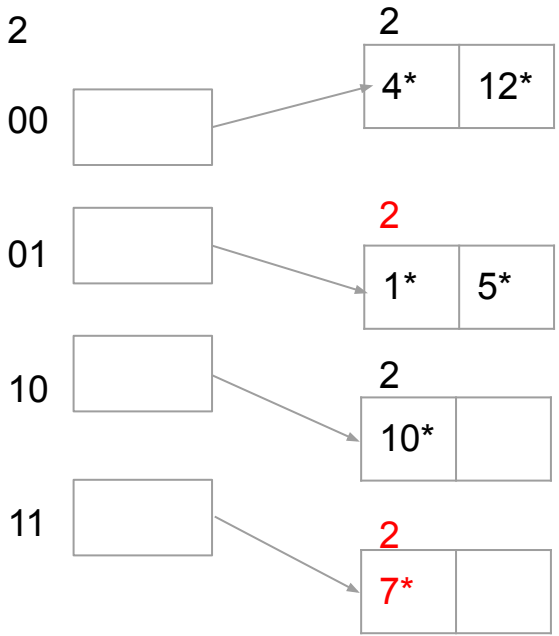
Extendible Hashing

- Use directory of pointers
  - Split on overflow
  - Once we're out of room in directory, double size
  - Global depth = number of bits considered globally
  - Local depth = number of bits considered locally
- Insert 7*
  - 7=0b111



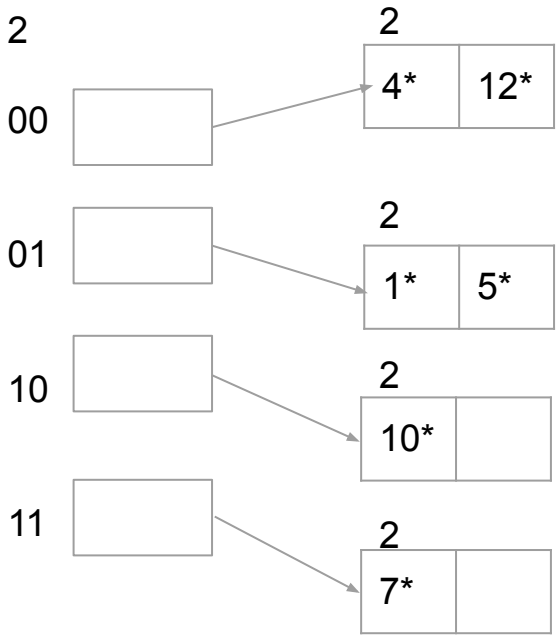
Extendible Hashing

- Use directory of pointers
  - Split on overflow
  - Once we're out of room in directory, double size
  - Global depth = number of bits considered globally
  - Local depth = number of bits considered locally
- Insert 7*
  - 7=0b111



Extendible Hashing

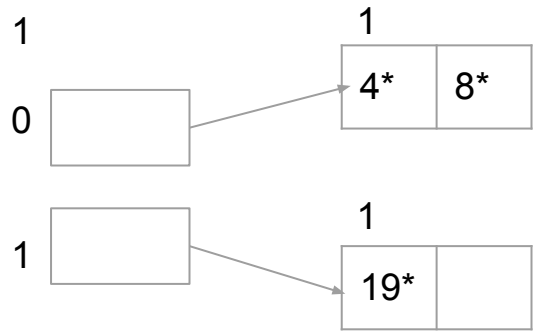
- Use directory of pointers
  - Split on overflow
  - Once we're out of room in directory, double size
  - Global depth = number of bits considered globally
  - Local depth = number of bits considered locally
- Insert 7*
  - 7=0b111
  - Done :)



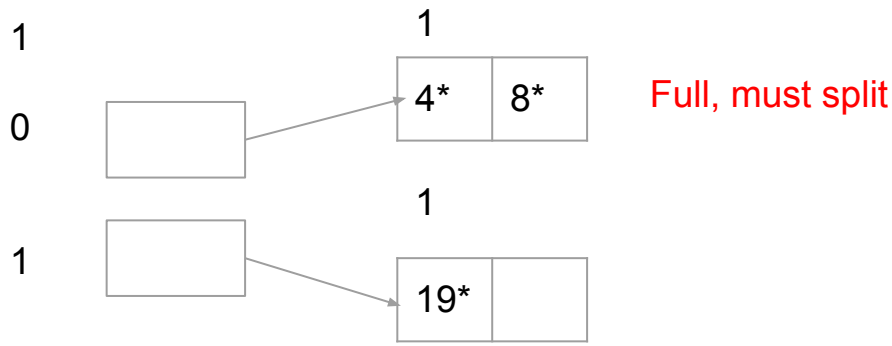
Example Problem

Given this extendible hashing index, what would be the number of pointers to the bucket containing 19* after performing the following operation: insert 12*

A) 1
B) 2
C) 3
D) 4
E) None of the above

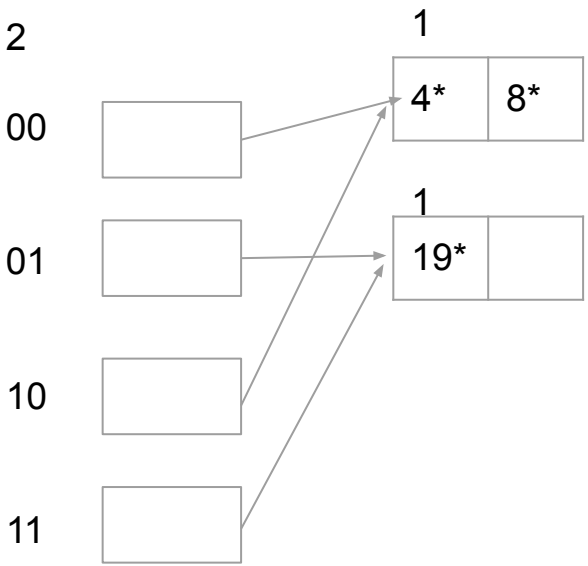


Insert 12*

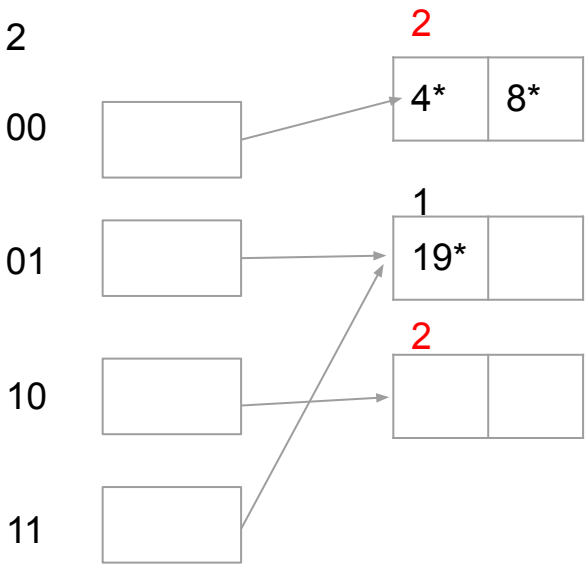


Full, must split

Insert 12*

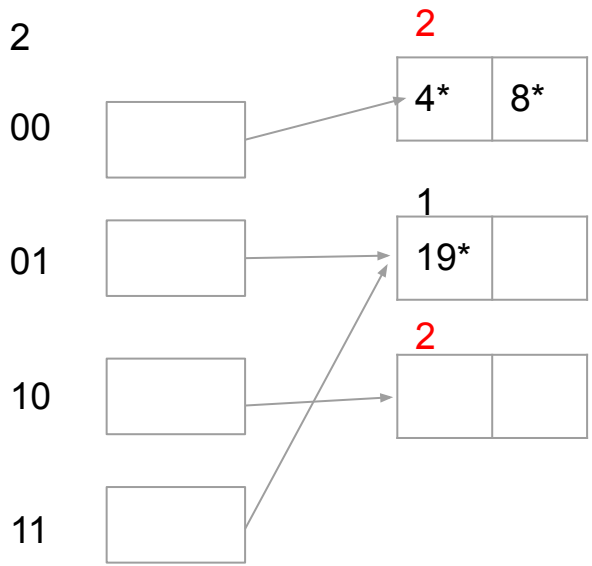


Insert 12*



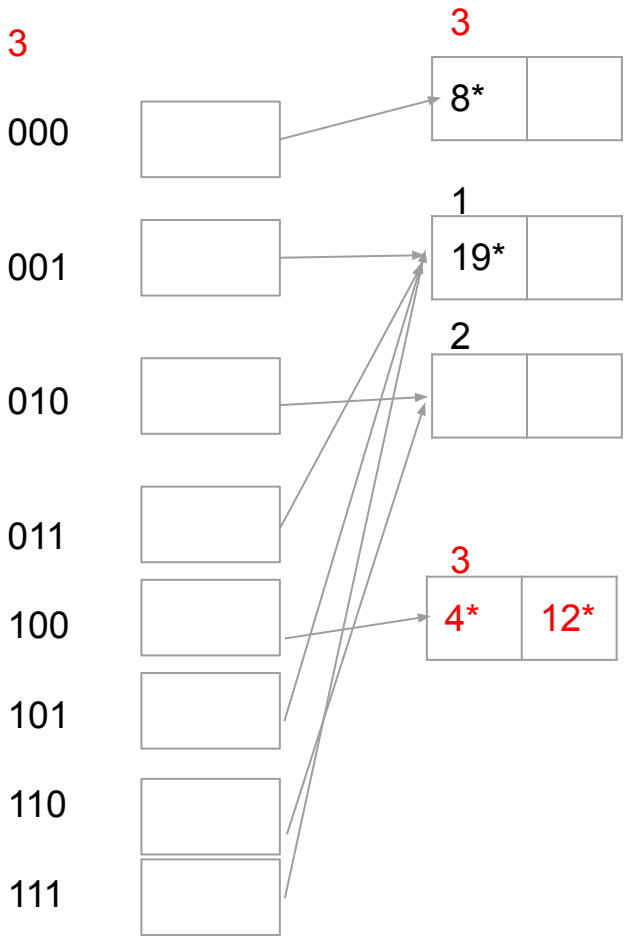
Redistribute if necessary

Insert 12*

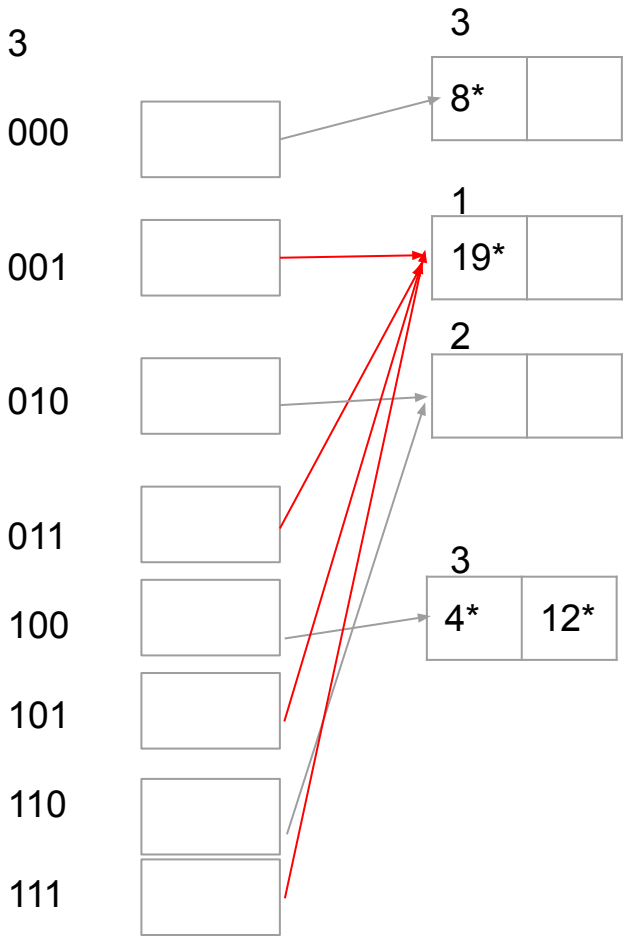


Still can't insert 12* since 4*, 8*, and 12* all map to 00_2

Insert 12*



Insert 12*



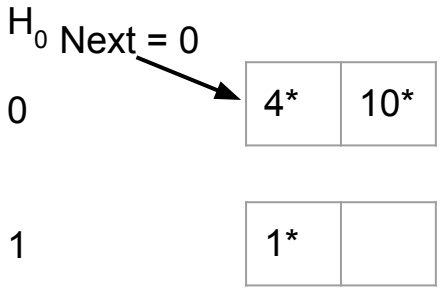
The number of pointers to the bucket with 19* is 4.

Linear Hashing

Linear Hashing

N = 2
$H_i(x) = x \pmod{N * 2^i}$
Level = 0

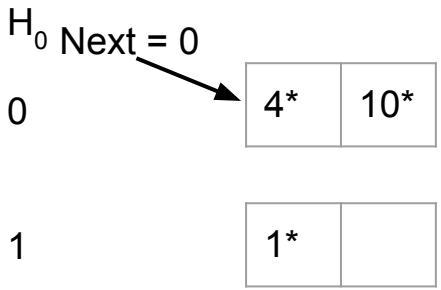
- Family of hash functions
- Split one bucket at a time upon an overflow
- N = **fixed** base number of buckets
- Level = current level in hash family
- Next = pointer to next bucket to be split
- Split policy: split on insertion into overflow page



Linear Hashing

- Family of hash functions
- Split one bucket at a time upon an overflow
- N = **fixed** base number of buckets
- Level = current level in hash family
- Next = pointer to next bucket to be split
- Split policy: split on insertion into overflow page

N = 2
$H_i(x) = x \pmod{N * 2^i}$
Level = 0
$H_0(x) = x \pmod{N * 2^0} = x \pmod 2$

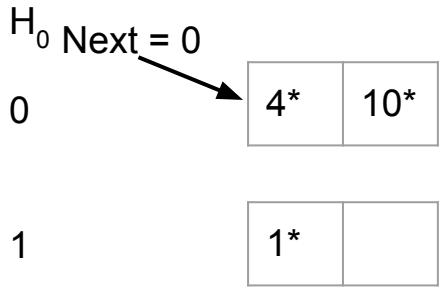


Linear Hashing

- Family of hash functions
- Split one bucket at a time upon an overflow
- N = **fixed** base number of buckets
- Level = current level in hash family
- Next = pointer to next bucket to be split
- Split policy: split on insertion into overflow page

- Insert 23*

N = 2
$H_i(x) = x \pmod{N * 2^i}$
Level = 0
$H_0(x) = x \pmod{N * 2^0} = x \pmod{2}$

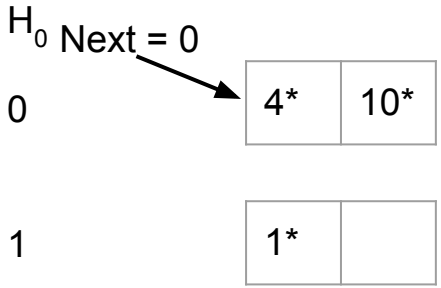


Linear Hashing

- Family of hash functions
- Split one bucket at a time upon an overflow
- N = **fixed** base number of buckets
- Level = current level in hash family
- Next = pointer to next bucket to be split
- Split policy: split on insertion into overflow page

- Insert 23*
  - $H_0(23) = 23 \pmod 2 = 1 \bmod 2 = 0b1$

N = 2
$H_i(x) = x \pmod{N * 2^i}$
Level = 0
$H_0(x) = x \pmod{N * 2^0} = x \pmod 2$

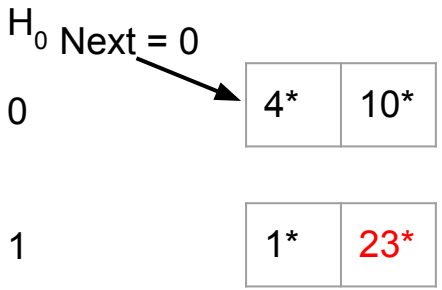


Linear Hashing

N = 2
$H_i(x) = x \pmod{N * 2^i}$
Level = 0
$H_0(x) = x \pmod{N * 2^0} = x \pmod 2$

- Family of hash functions
- Split one bucket at a time upon an overflow
- N = **fixed** base number of buckets
- Level = current level in hash family
- Next = pointer to next bucket to be split
- Split policy: split on insertion into overflow page

- Insert 23*
  - $H_0(23) = 23 \pmod 2 = 1 \bmod 2 = 0b1$

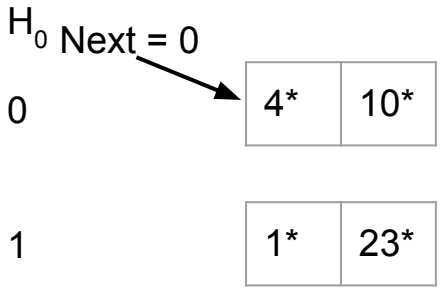


Linear Hashing

- Family of hash functions
- Split one bucket at a time upon an overflow
- N = **fixed** base number of buckets
- Level = current level in hash family
- Next = pointer to next bucket to be split
- Split policy: split on insertion into overflow page

- Insert 23*
  - $H_0(23) = 23 \pmod 2 = 1 \bmod 2 = 0b1$
- Done :)

N = 2
$H_i(x) = x \pmod{N * 2^i}$
Level = 0
$H_0(x) = x \pmod{N * 2^0} = x \pmod 2$

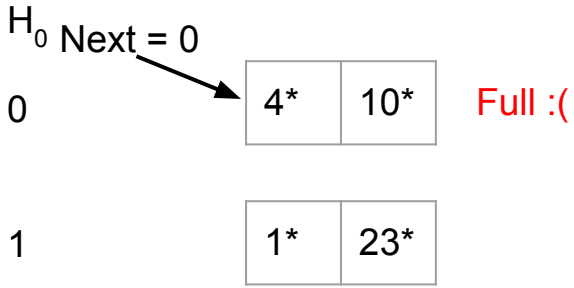


Linear Hashing

- Family of hash functions
- Split one bucket at a time upon an overflow
- N = **fixed** base number of buckets
- Level = current level in hash family
- Next = pointer to next bucket to be split
- Split policy: split on insertion into overflow page

- Insert 12*
  - $H_0(12) = 12 \pmod 2 = 0 \bmod 2 = 0b0$

N = 2
$H_i(x) = x \pmod{N * 2^i}$
Level = 0
$H_0(x) = x \pmod{N * 2^0} = x \pmod 2$

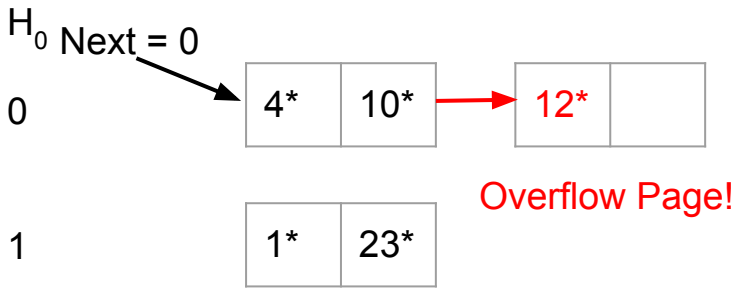


Linear Hashing

- Family of hash functions
- Split one bucket at a time upon an overflow
- N = **fixed** base number of buckets
- Level = current level in hash family
- Next = pointer to next bucket to be split
- Split policy: split on insertion into overflow page

- Insert 12*
  - $H_0(12) = 12 \pmod 2 = 0 \bmod 2 = 0b0$

N = 2
$H_i(x) = x \pmod{N * 2^i}$
Level = 0
$H_0(x) = x \pmod{N * 2^0} = x \pmod 2$

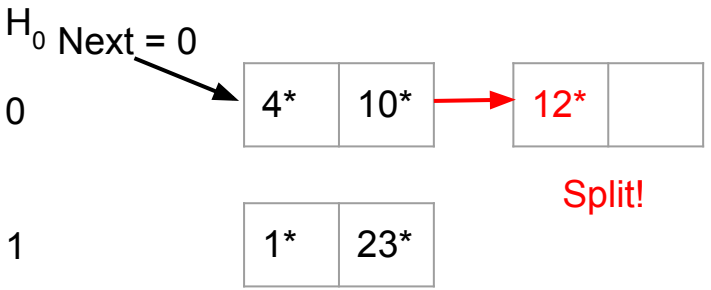


Linear Hashing

- Family of hash functions
- Split one bucket at a time upon an overflow
- N = **fixed** base number of buckets
- Level = current level in hash family
- Next = pointer to next bucket to be split
- Split policy: split on insertion into overflow page

- Insert 12*
  - $H_0(12) = 12 \pmod 2 = 0 \bmod 2 = 0b0$

N = 2
$H_i(x) = x \pmod{N * 2^i}$
Level = 0
$H_0(x) = x \pmod{N * 2^0} = x \pmod 2$

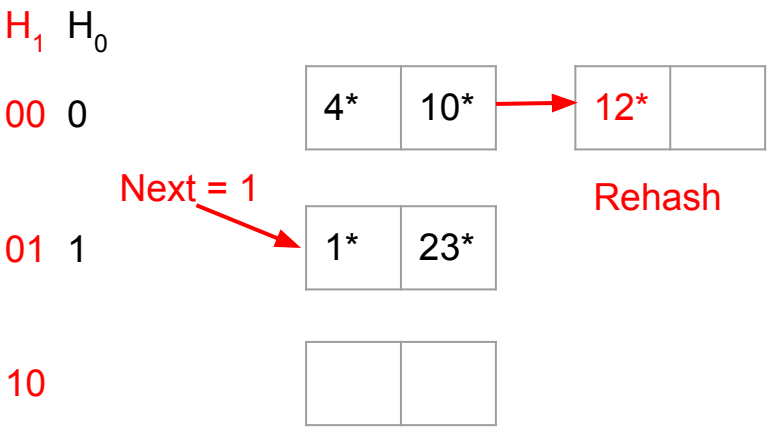


Linear Hashing

- Family of hash functions
- Split one bucket at a time upon an overflow
- N = **fixed** base number of buckets
- Level = current level in hash family
- Next = pointer to next bucket to be split
- Split policy: split on insertion into overflow page

- Insert 12*
  - $H_0(12) = 12 \pmod 2 = 0 \bmod 2 = 0b0$

N = 2
$H_i(x) = x \pmod{N * 2^i}$
Level = 0
$H_0(x) = x \pmod{N * 2^0} = x \pmod 2$
$H_1(x) = x \pmod{N * 2^1} = x \pmod 4$

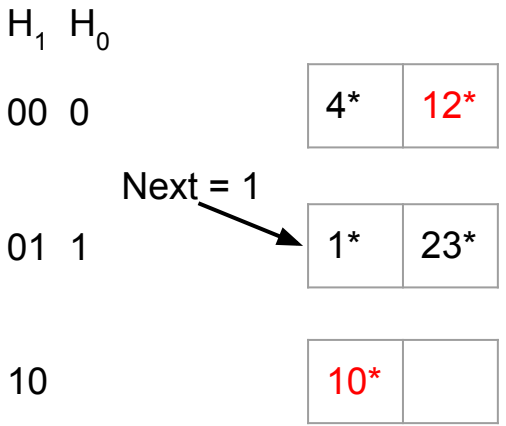


Linear Hashing

- Family of hash functions
- Split one bucket at a time upon an overflow
- N = **fixed** base number of buckets
- Level = current level in hash family
- Next = pointer to next bucket to be split
- Split policy: split on insertion into overflow page

- Insert 12*
  - $H_0(12) = 12 \pmod 2 = 0 \bmod 2 = 0b0$
- Rehash 4*, 10*, 12*
  - $H_1(4) = 4 \pmod 4 = 0 \bmod 4 = 0b00$
  - $H_1(10) = 10 \pmod 4 = 2 \bmod 4 = 0b10$
  - $H_1(12) = 12 \pmod 4 = 0 \bmod 4 = 0b00$

N = 2
$H_i(x) = x \pmod{N * 2^i}$
Level = 0
$H_0(x) = x \pmod{N * 2^0} = x \pmod 2$
$H_1(x) = x \pmod{N * 2^1} = x \pmod 4$

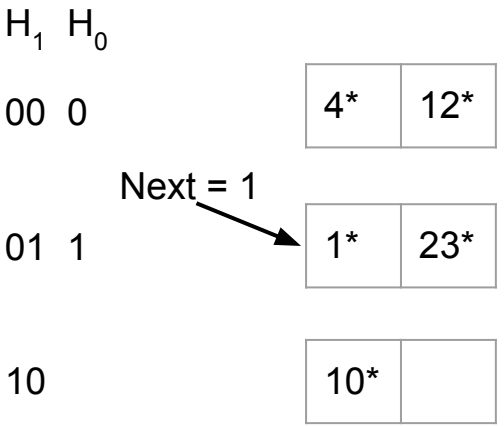


Linear Hashing

- Family of hash functions
- Split one bucket at a time upon an overflow
- N = **fixed** base number of buckets
- Level = current level in hash family
- Next = pointer to next bucket to be split
- Split policy: split on insertion into overflow page

- Insert 12*
  - $H_0(12) = 12 \pmod 2 = 0 \bmod 2 = 0b0$
- Done :)

N = 2
$H_i(x) = x \pmod{N * 2^i}$
Level = 0
$H_0(x) = x \pmod{N * 2^0} = x \pmod 2$
$H_1(x) = x \pmod{N * 2^1} = x \pmod 4$

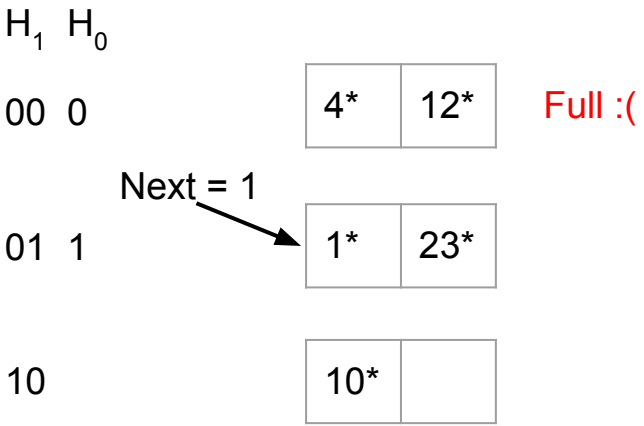


Linear Hashing

- Family of hash functions
- Split one bucket at a time upon an overflow
- N = **fixed** base number of buckets
- Level = current level in hash family
- Next = pointer to next bucket to be split
- Split policy: split on insertion into overflow page

- Insert 8*
  - $H_0(8) = 8 \pmod 2 = 0 \bmod 2 = 0b0$
  - $H_1(8) = 8 \pmod 4 = 0 \bmod 4 = 0b00$

$N = 2$
$H_i(x) = x \pmod{N * 2^i}$
Level = 0
$H_0(x) = x \pmod{N * 2^0} = x \pmod 2$
$H_1(x) = x \pmod{N * 2^1} = x \pmod 4$

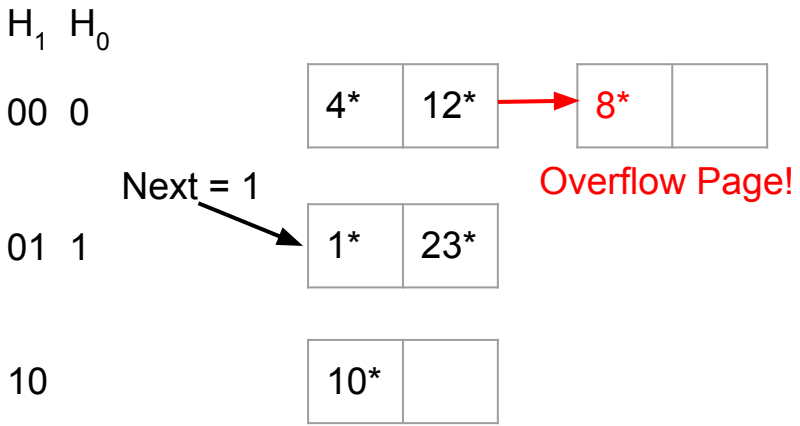


Linear Hashing

- Family of hash functions
- Split one bucket at a time upon an overflow
- N = **fixed** base number of buckets
- Level = current level in hash family
- Next = pointer to next bucket to be split
- Split policy: split on insertion into overflow page

- Insert 8*
  - $H_0(8) = 8 \pmod 2 = 0 \bmod 2 = 0b0$
  - $H_1(8) = 8 \pmod 4 = 0 \bmod 4 = 0b00$

N = 2
$H_i(x) = x \pmod{N * 2^i}$
Level = 0
$H_0(x) = x \pmod{N * 2^0} = x \pmod 2$
$H_1(x) = x \pmod{N * 2^1} = x \pmod 4$

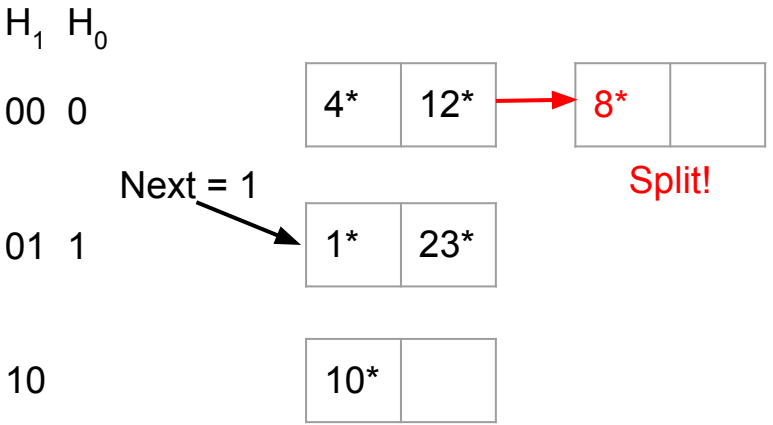


Linear Hashing

- Family of hash functions
- Split one bucket at a time upon an overflow
- N = **fixed** base number of buckets
- Level = current level in hash family
- Next = pointer to next bucket to be split
- Split policy: split on insertion into overflow page

- Insert 8*
  - $H_0(8) = 8 \pmod 2 = 0 \bmod 2 = 0b0$
  - $H_1(8) = 8 \pmod 4 = 0 \bmod 4 = 0b00$

N = 2
$H_i(x) = x \pmod{N * 2^i}$
Level = 0
$H_0(x) = x \pmod{N * 2^0} = x \pmod 2$
$H_1(x) = x \pmod{N * 2^1} = x \pmod 4$

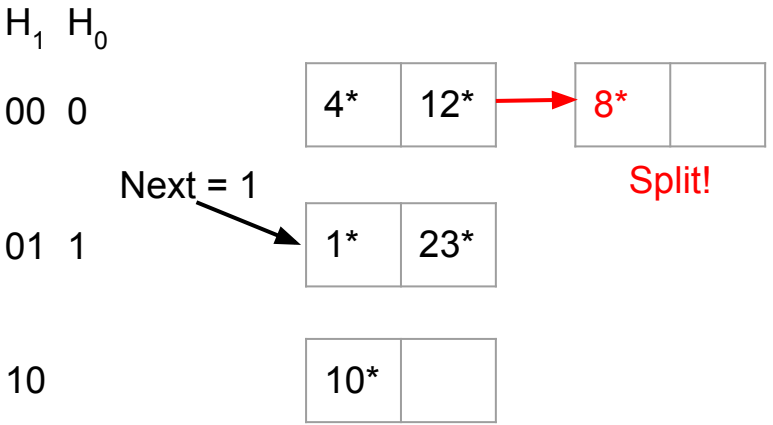


Linear Hashing

- Family of hash functions
- Split one bucket at a time upon an overflow
- N = **fixed** base number of buckets
- Level = current level in hash family
- Next = pointer to next bucket to be split
- Split policy: split on insertion into overflow page

- Insert 8*
  - $H_0(8) = 8 \pmod 2 = 0 \bmod 2 = 0b0$
  - $H_1(8) = 8 \pmod 4 = 0 \bmod 4 = 0b00$
  - Remember we split the next node always even though it might not be overflow

N = 2
$H_i(x) = x \pmod{N * 2^i}$
Level = 0
$H_0(x) = x \pmod{N * 2^0} = x \pmod 2$
$H_1(x) = x \pmod{N * 2^1} = x \pmod 4$



Linear Hashing

- Family of hash functions
- Split one bucket at a time upon an overflow
- N = **fixed** base number of buckets
- Level = current level in hash family
- Next = pointer to next bucket to be split
- Split policy: split on insertion into overflow page

- Insert 8*
  - $H_0(8) = 8 \pmod 2 = 0 \bmod 2 = 0b0$
  - $H_1(8) = 8 \pmod 4 = 0 \bmod 4 = 0b00$
  - Remember we split the next node always even though it might not be overflow

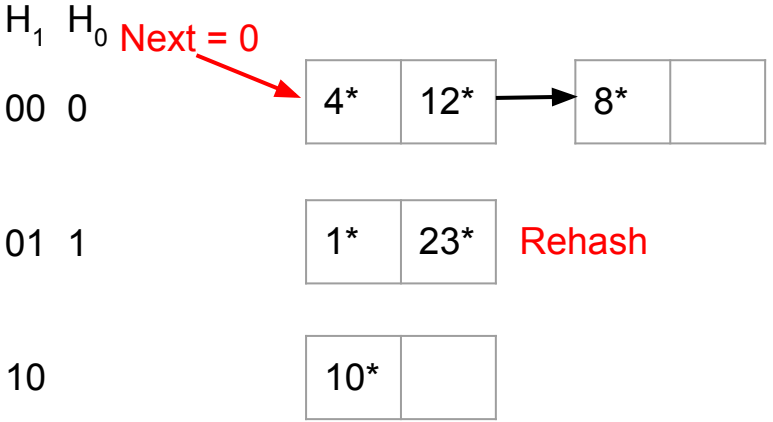
$N = 2$

$H_i(x) = x \pmod{N * 2^i}$

Level = 1

$H_0(x) = x \pmod{N * 2^0} = x \pmod 2$

$H_1(x) = x \pmod{N * 2^1} = x \pmod 4$

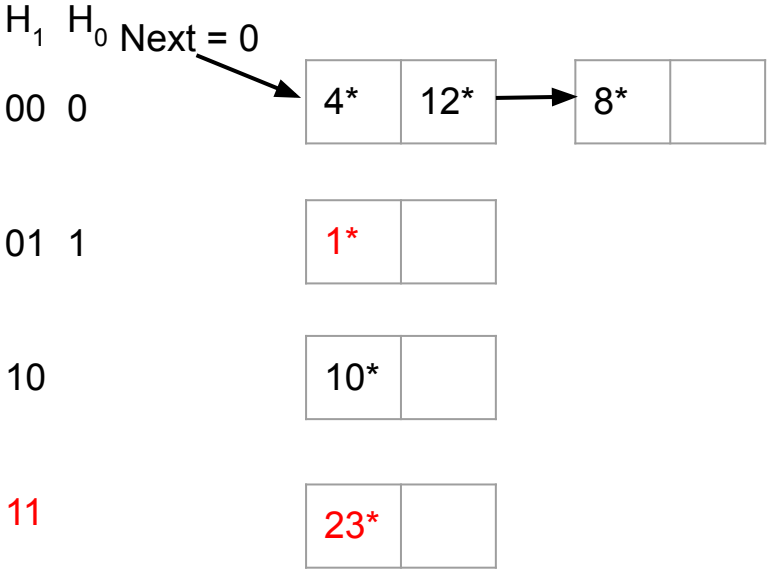


Linear Hashing

- Family of hash functions
- Split one bucket at a time upon an overflow
- N = **fixed** base number of buckets
- Level = current level in hash family
- Next = pointer to next bucket to be split
- Split policy: split on insertion into overflow page

- Insert 8*
  - $H_0(8) = 8 \pmod 2 = 0 \bmod 2 = 0b0$
  - $H_1(8) = 8 \pmod 4 = 0 \bmod 4 = 0b00$
  - $H_1(1) = 1 \pmod 4 = 1 \bmod 4 = 0b01$
  - $H_1(23) = 23 \pmod 4 = 3 \bmod 4 = 0b11$

N = 2
$H_i(x) = x \pmod{N * 2^i}$
Level = 1
$H_0(x) = x \pmod{N * 2^0} = x \pmod 2$
$H_1(x) = x \pmod{N * 2^1} = x \pmod 4$

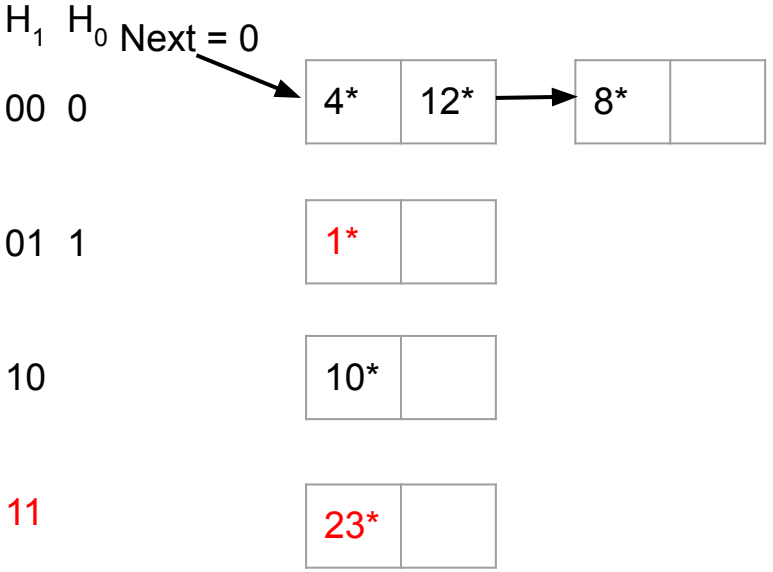


Linear Hashing

- Family of hash functions
- Split one bucket at a time upon an overflow
- N = **fixed** base number of buckets
- Level = current level in hash family
- Next = pointer to next bucket to be split
- Split policy: split on insertion into overflow page

- Insert 8*
  - $H_0(8) = 8 \pmod 2 = 0 \bmod 2 = 0b0$
  - $H_1(8) = 8 \pmod 4 = 0 \bmod 4 = 0b00$
  - $H_1(1) = 1 \pmod 4 = 1 \bmod 4 = 0b01$
  - $H_1(23) = 23 \pmod 4 = 3 \bmod 4 = 0b11$
  - Still have overflow page; that's ok

N = 2
$H_i(x) = x \pmod{N * 2^i}$
Level = 1
$H_0(x) = x \pmod{N * 2^0} = x \pmod 2$
$H_1(x) = x \pmod{N * 2^1} = x \pmod 4$

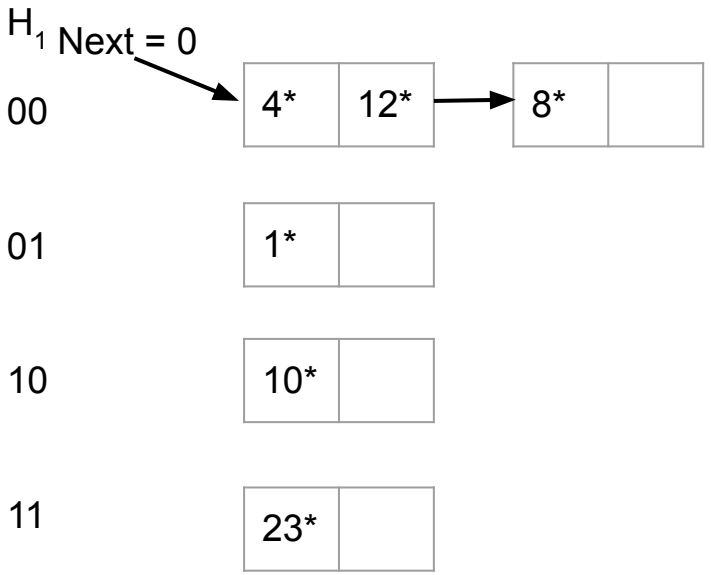


Linear Hashing

- Family of hash functions
- Split one bucket at a time upon an overflow
- N = **fixed** base number of buckets
- Level = current level in hash family
- Next = pointer to next bucket to be split
- Split policy: split on insertion into overflow page

- Insert 8*
  - $H_0(8) = 8 \pmod 2 = 0 \bmod 2 = 0b0$
  - $H_1(8) = 8 \pmod 4 = 0 \bmod 4 = 0b00$
  - $H_1(1) = 1 \pmod 4 = 1 \bmod 4 = 0b01$
  - $H_1(23) = 23 \pmod 4 = 3 \bmod 4 = 0b11$
  - Still have overflow page; that's ok
  - Done :)

N = 2
$H_i(x) = x \pmod{N * 2^i}$
Level = 1
$H_0(x) = x \pmod{N * 2^0} = x \pmod 2$
$H_1(x) = x \pmod{N * 2^1} = x \pmod 4$



Example Problem

If N = 2, how many buckets are there (excluding overflow buckets) when Level = 2 and Next = 0?

A) 2
B) 3
C) 4
D) 8
E) None of the Above

Example Problem

If N = 2, how many buckets are there (excluding overflow buckets) when the Level is 2 and Next = 0?

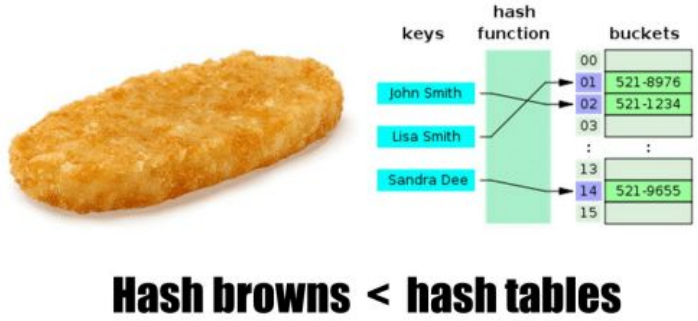
A) 2
B) 3
C) 4
D) 8
E) None of the Above

If bucket being inserted into is full:

- Add overflow page and insert data entry
  - If bucket is Next, split first and then see if you still need overflow
- Split *Next* bucket and increment Next
  - If Next = $N_{Level} - 1$ and a split is triggered
    1. We split the bucket # N_{Level}-1 (i.e. the last bucket before the split image buckets)
    2. Next = 0 (reset the pointer)
    3. Level = Level + 1 (the size has doubled at this point)

$$\# \text{ buckets in the file} = N_{Level} = N * 2^{Level}$$

Hashing Summary



- Extendible Hashing
  - Directory size can double
  - Can have recursive splitting
  - Global depth >= local depth always
  - # pointers to any specific bucket = 2^{GD-LD}
  - Overflow pages only in rare cases (duplicate keys)
- Linear Hashing
  - Only splits one bucket at a time
  - No directory doubling
  - Often has overflow pages, but over time these are minimized
  - Usually better memory usage, since directory typically takes up more space than overflow pages

Get started on HW4 and Project 3!