UCLA

CS143: Hash Index

Professor Junghoo “John” Cho

Hash Index

- Static hashing
- Extendable hashing

What is a Hash Table?

- Hash Table
  - Hash function
    - *h*(k): key → [0…n]
    - e.g., *h*(Susan) = 4
  - Array for keys: T[0…n]
  - Given a key *k*, store it in T[*h*(*k*)]

h(k): name → [0…4]

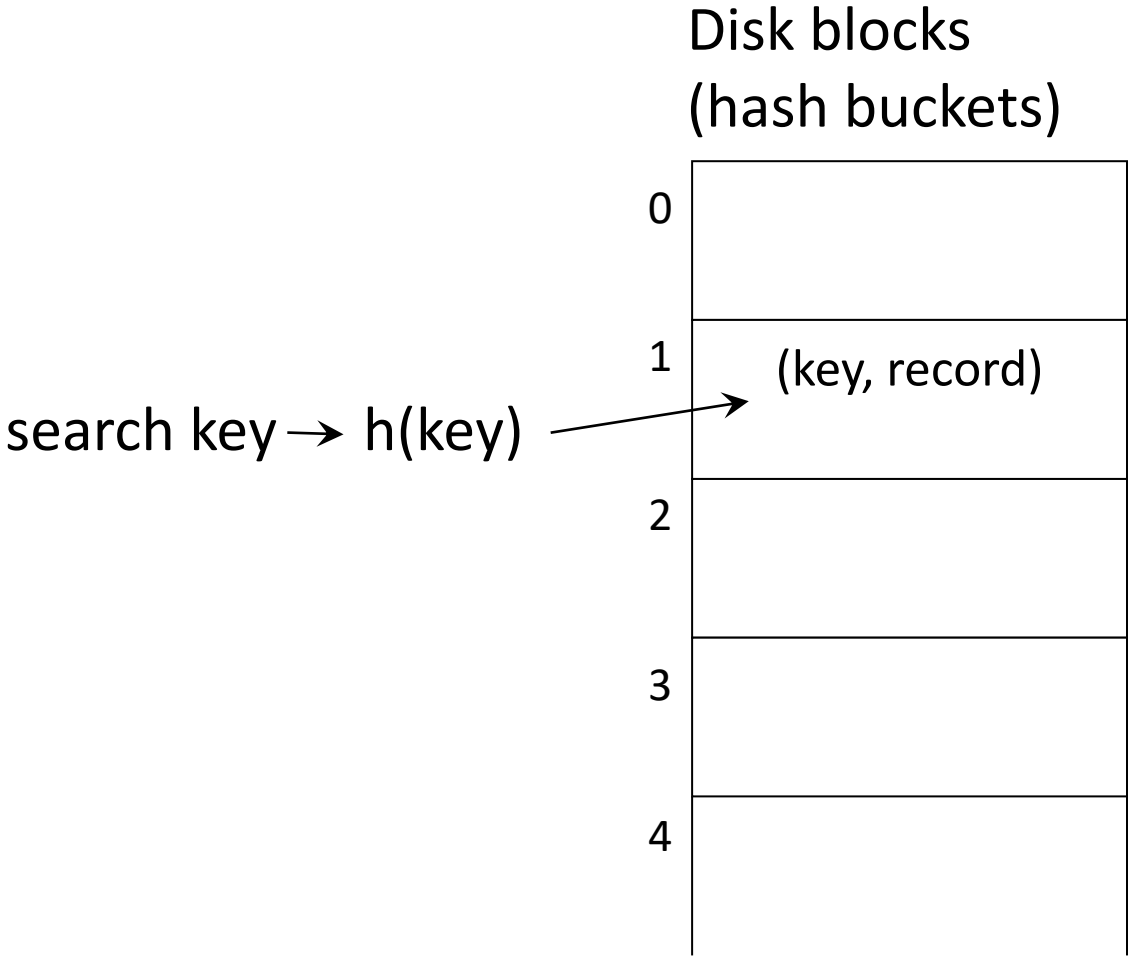
h(Susan) = 4

h(James) = 3

h(Neil) = 1

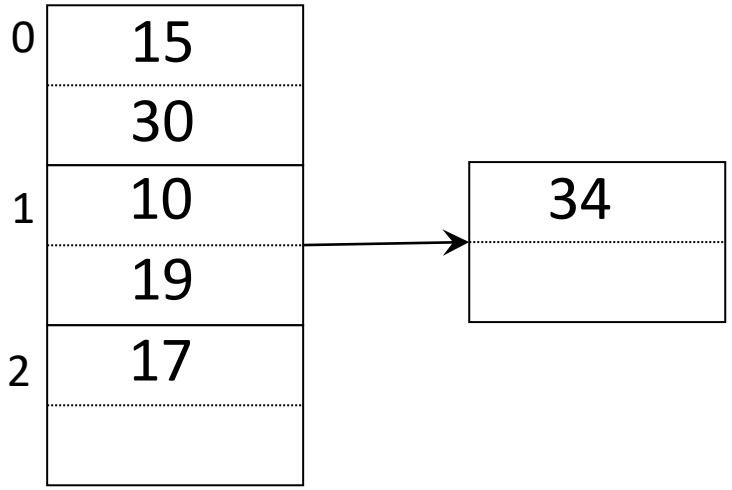
0	
1	Neil
2	
3	James
4	Susan

Hashing for DBMS (Static Hashing)



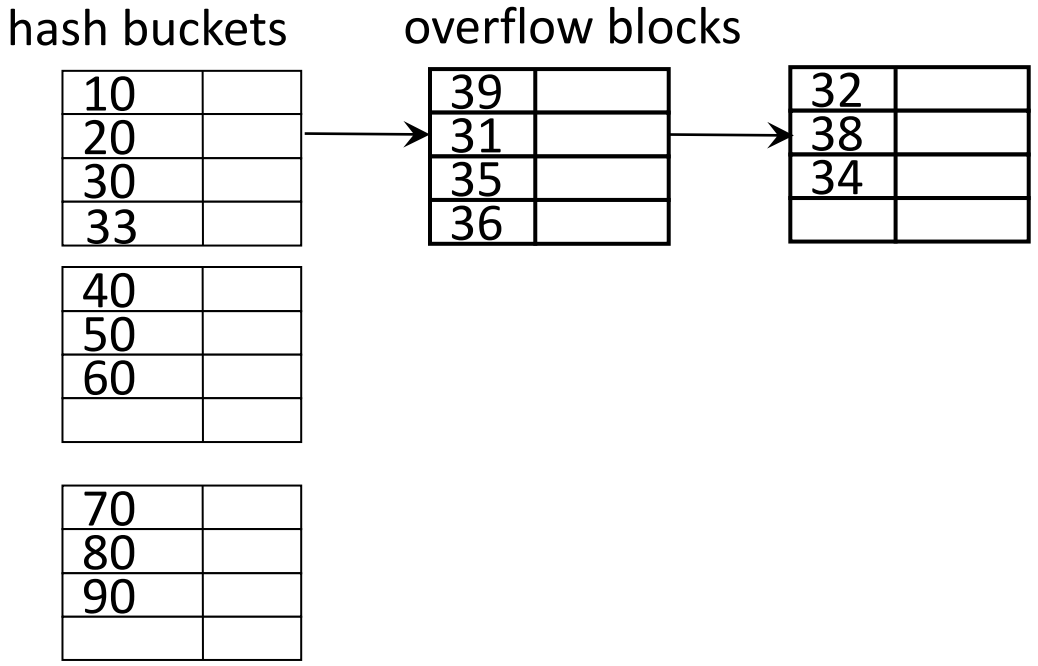
Overflow and Chaining

- h(n) = n mod 3
- Insert
  - h(10) = 1
  - h(15) = 0
  - h(17) = 2
  - h(19) = 1
  - h(30) = 0
  - h(34) = 1



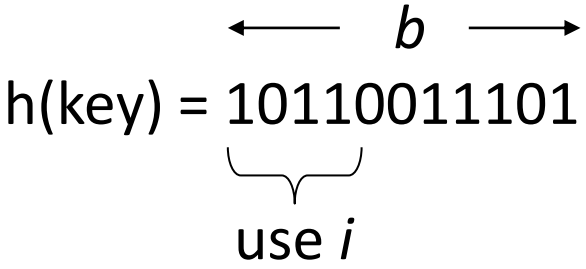
Major Problem of Static Hashing

- As data grows in size, overflow blocks unavoidable

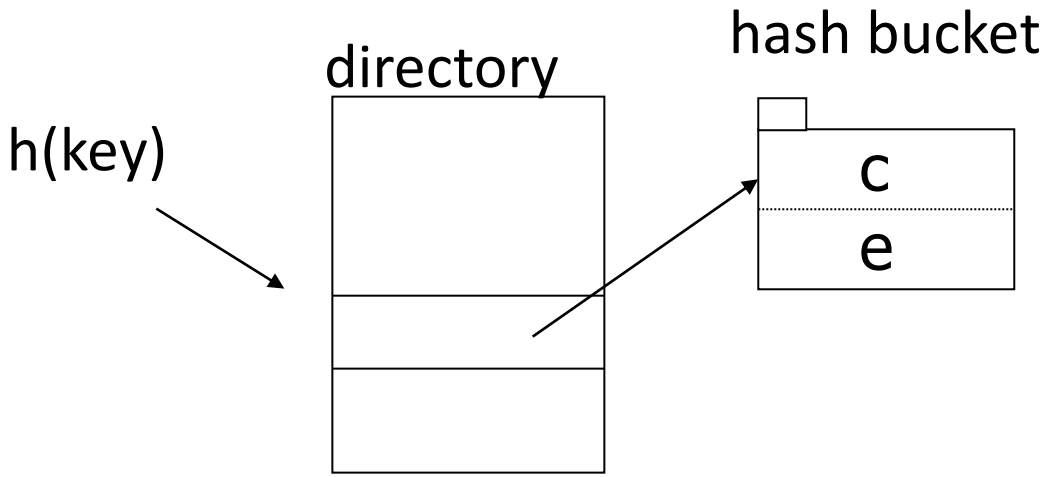


Extendable Hashing

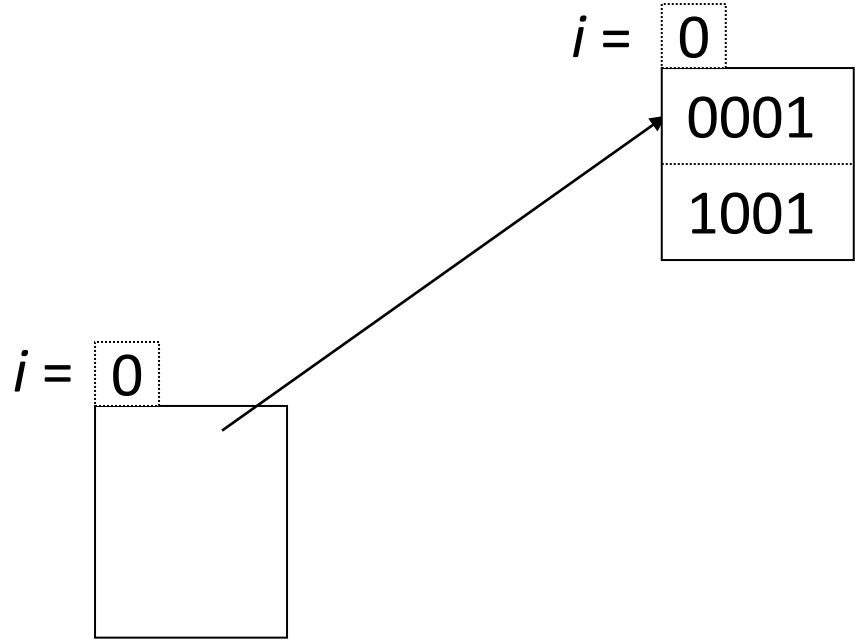
1. Use *i* of *b* bits from hash output, increasing *i* as needed



2. Add a level of indirection: directory of pointers to hash buckets

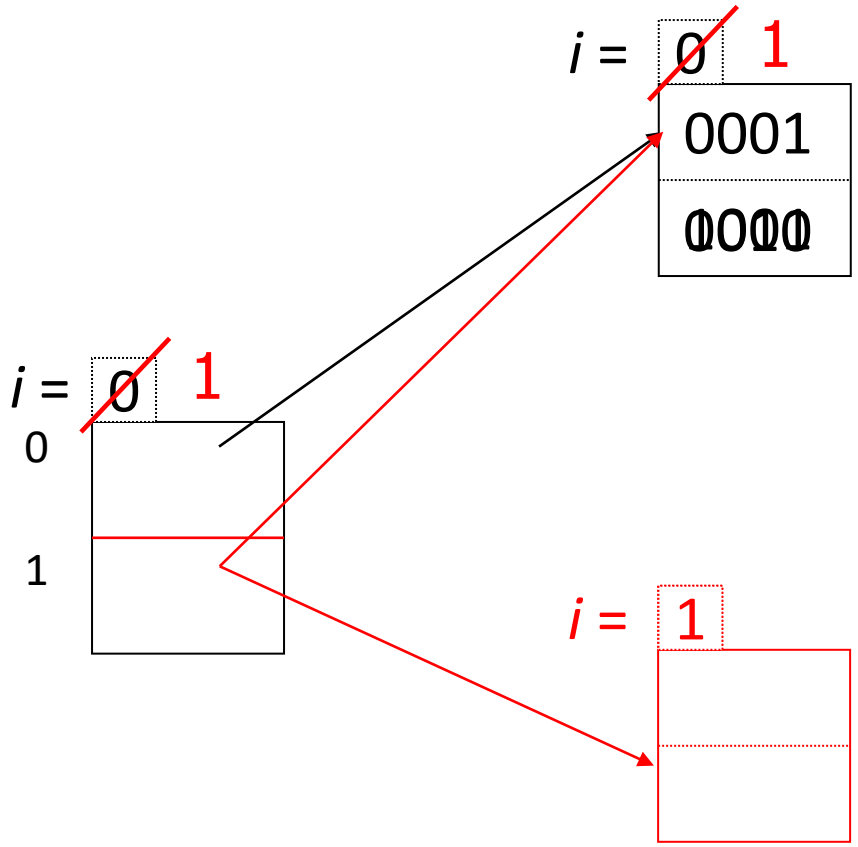


Extendable Hash: Insertion



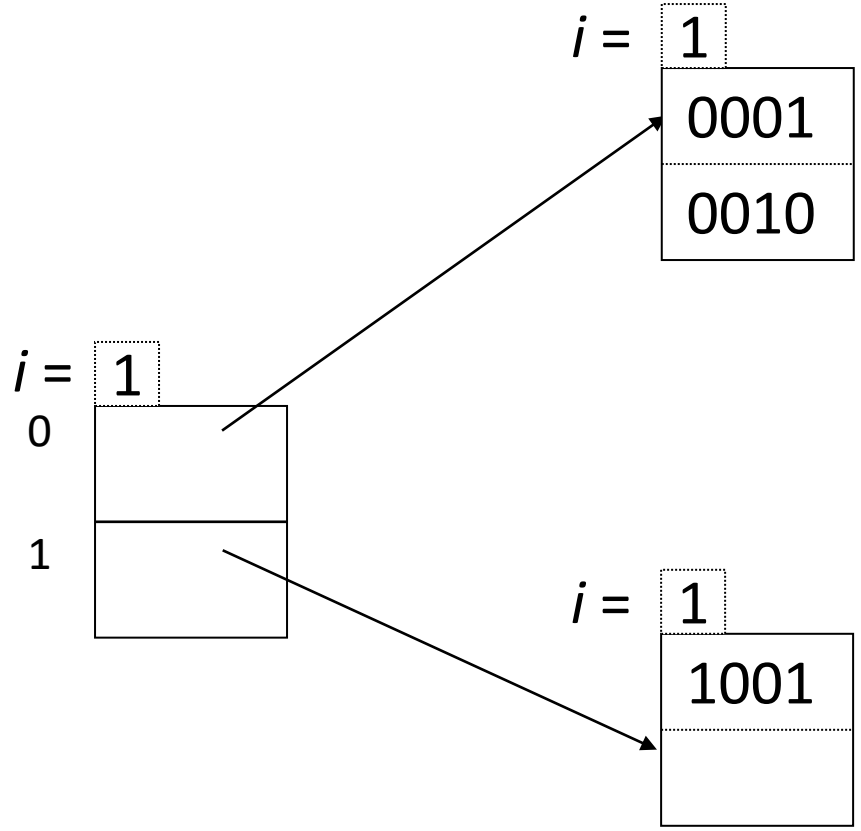
- Insert 0001, 1001
- Insert 0010

Extendable Hash: Insertion



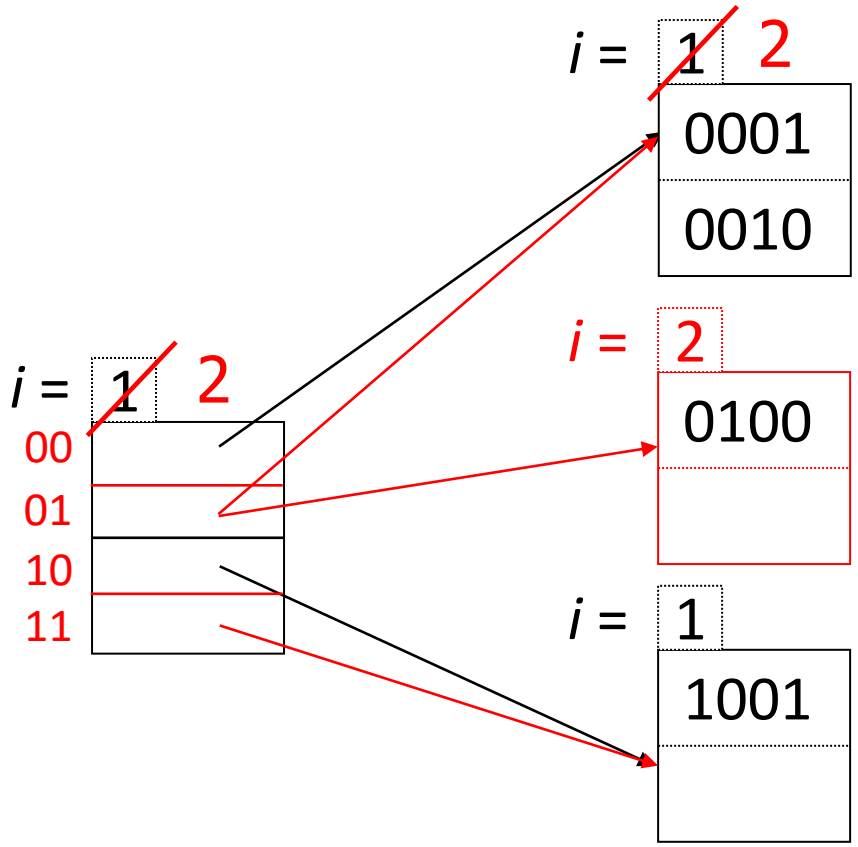
- Insert 0010

Extendable Hash: Insertion



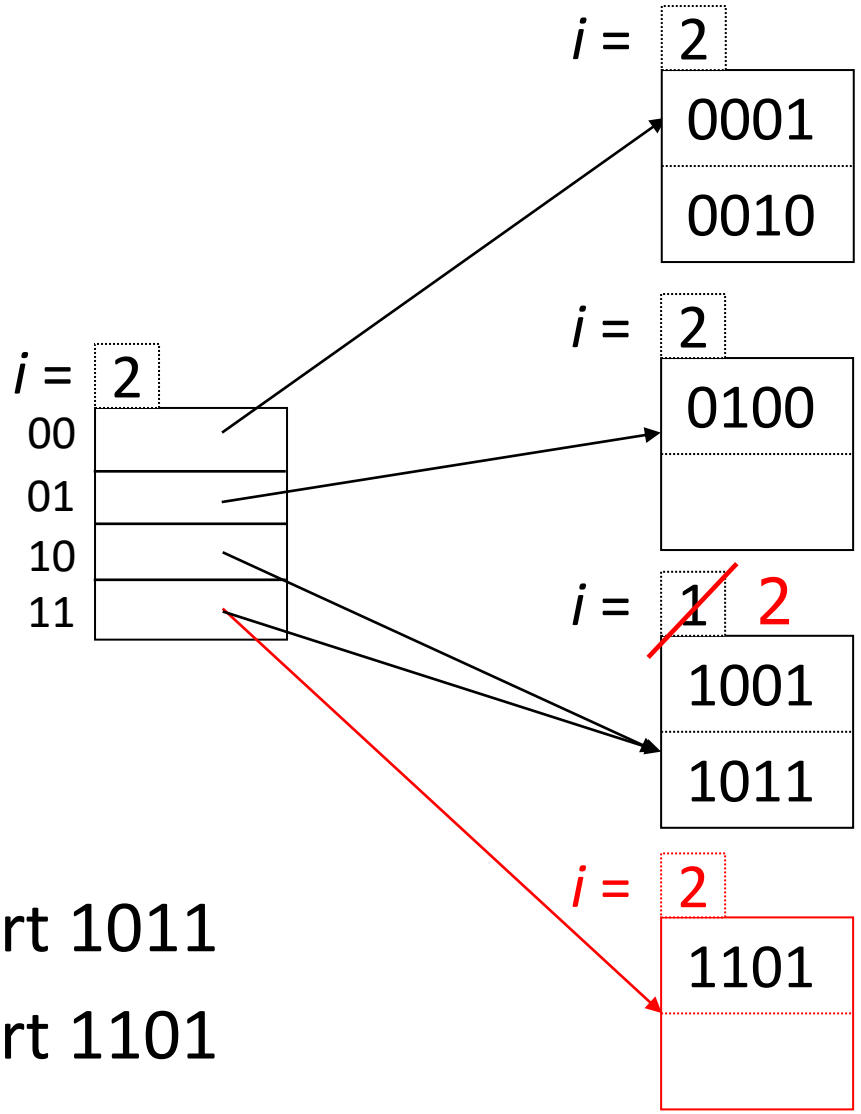
- Insert 0100

Extendable Hash: Insertion



- Insert 0100

Extendable Hash: Insertion

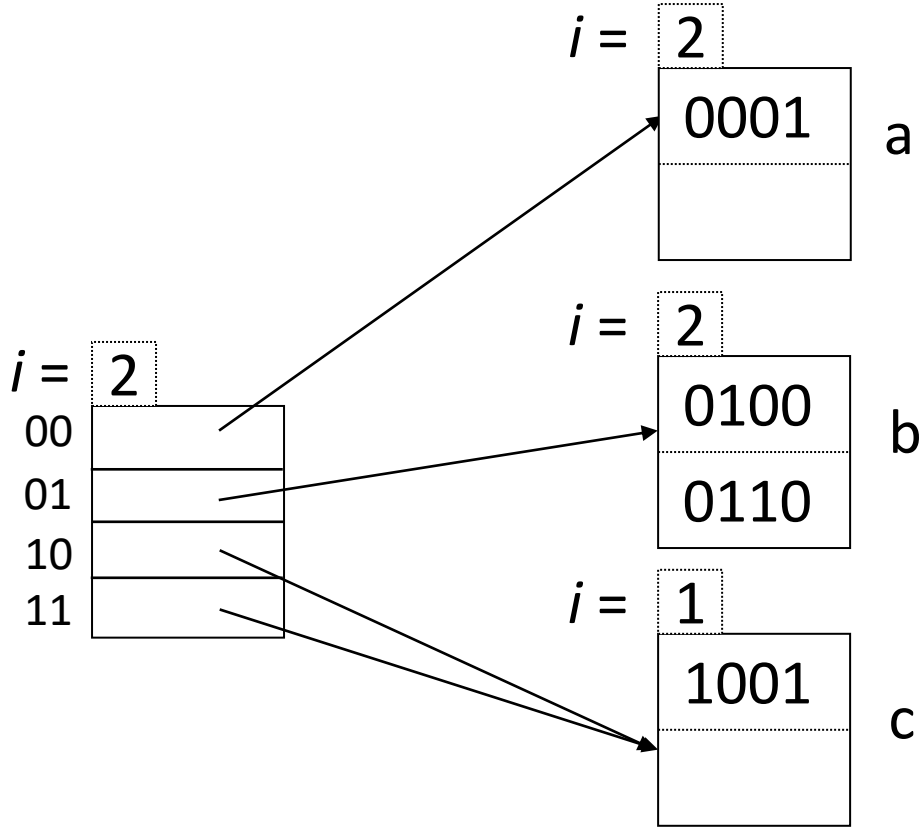


- Insert 1011
- Insert 1101

Extendable Hash: Insertion

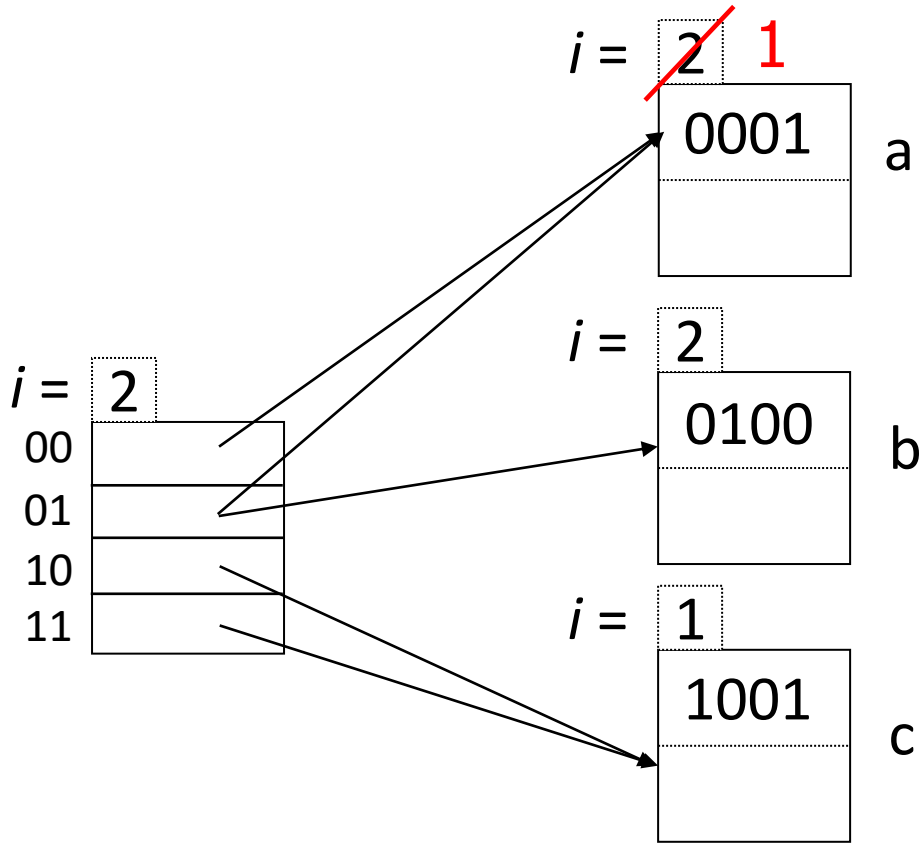
- If no hash bucket overflow:
  - Insert the tuple into the hash bucket
- If a hash bucket overflows:
  - Is the hash bucket i == directory i, then
    - Double directory size by copying existing pointers
    - Increase directory i value by 1
  - Split the overflowing hash bucket
    - Move tuples in the bucket to the new bucket based on their hash values
    - Update directory pointer
    - Increase the hash bucket i value by 1

Extendable Hash: Deletion

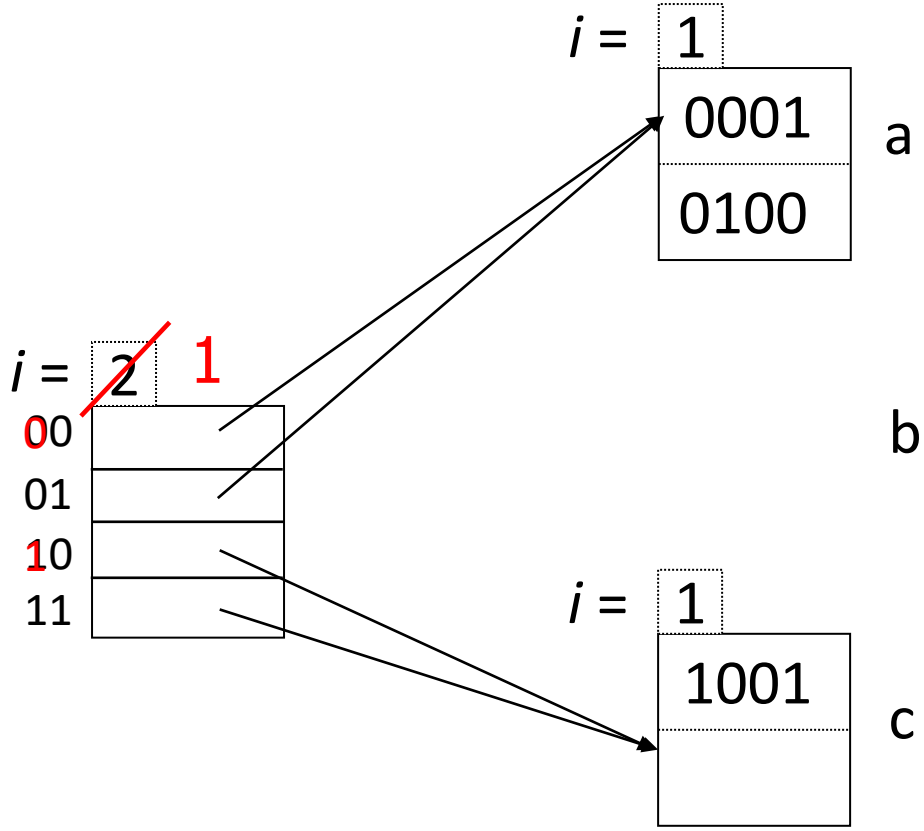


- Delete 0110
- Can we merge a and b? b and c?

Extendable Hash: Deletion



Extendable Hash: Deletion



Q: Can we shrink directory?

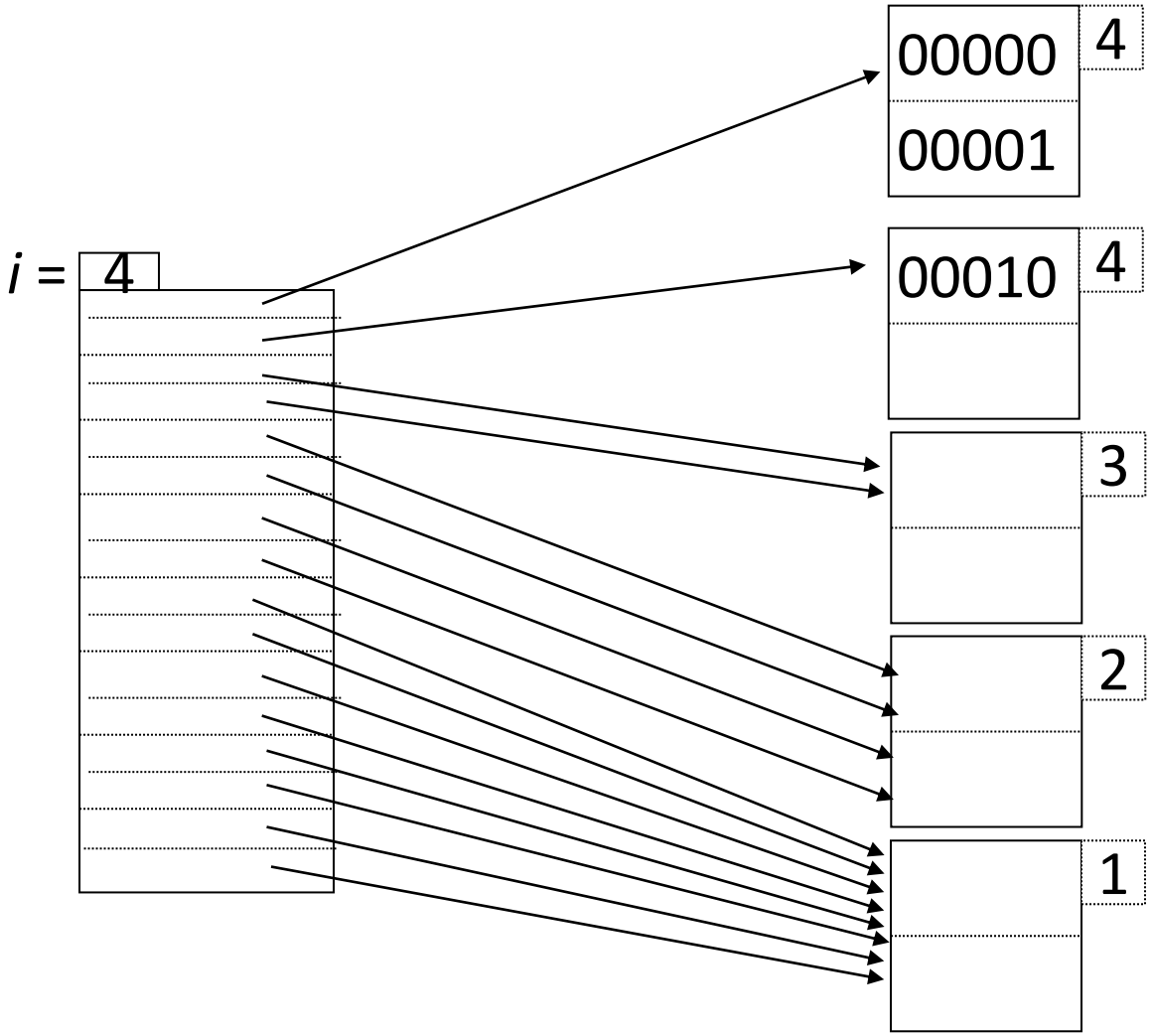
Merge Condition

- Hash bucket merge condition
  - Bucket i’s are the same
  - First (i-1) bits of the hash key values are the same
- Directory shrink condition
  - All bucket i’s are smaller than the directory i

Questions on Extendable Hashing

- Can we provide minimum space guarantee?

Space Waste



Hash index summary

- Static hashing
  - Overflow and chaining
- Extendable hashing
  - Can handle growing files
    - No periodic reorganizations
  - Indirection
    - Up to 2 disk accesses to access a key
  - Directory doubles in size
    - Not too bad if the data is not too large

Hashing vs. Tree

- Can an extendable-hash index support?

```
SELECT *
FROM R
WHERE R.A > 5
```

- Which one is better, B+tree or extendable hashing?

```
SELECT *
FROM R
WHERE R.A = 5
```