Module 3: Hashing

Lecture 9: Static and Dynamic Hashing

Prev topic Next topic Prev page
Next page

The Lecture Contains:

- Static hashing
- Hashing
- Dynamic hashing
- Extendible hashing
  - Insertion

Static hashing

- Single hash function $h(k)$ on key k
- Desirable properties of a hash function
  - Uniform:

Static hashing

- Single hash function $h(k)$ on key k
- Desirable properties of a hash function
  - Uniform: Total domain of keys is distributed uniformly over the range
  - Random:

Static hashing

- Single hash function $h(k)$ on key k
- Desirable properties of a hash function
  - Uniform: Total domain of keys is distributed uniformly over the range
  - Random: Hash values should be distributed uniformly irrespective of distribution of keys
- $O(1)$ search
- Example of hash functions: $h(k) = k \mod m$

Static hashing

- Single hash function $h(k)$ on key k
- Desirable properties of a hash function
  - Uniform: Total domain of keys is distributed uniformly over the range
  - Random: Hash values should be distributed uniformly irrespective of distribution of keys
- $O(1)$ search
- Example of hash functions: $h(k) = k \mod m$
- Collision resolution
- Chaining
  - Load factor
  - Primary pages and overflow pages (or buckets)
  - Search time more for overflow buckets

Static hashing

- Single hash function $h(k)$ on key k
- Desirable properties of a hash function
  - Uniform: Total domain of keys is distributed uniformly over the range
  - Random: Hash values should be distributed uniformly irrespective of distribution of keys
- $O(1)$ search
- Example of hash functions: $h(k) = k \mod m$
- Collision resolution
- Chaining
  - Load factor
  - Primary pages and overflow pages (or buckets)
  - Search time more for overflow buckets
- Open addressing
  - Linear probing
  - Quadratic probing
  - Double hashing
- Cannot support range or kNN queries

Module 3: Hashing

Lecture 9: Static and Dynamic Hashing

Hashing

- Problems of static hashing
  - Fixed size of hash table due to fixed hash function
  - May require rehashing of all keys when chains or overflow buckets are full
- Dynamic hashing
  - Hash function modified dynamically as number of records grow
  - Needs to maintain determinism
  - Extendible hashing
  - Linear hashing

Dynamic hashing

- Organize overflow buckets as binary trees
- m binary trees for m primary pages
- $h_0(k)$ produces index of primary page
- Particular access structure for binary trees
- Family of functions $g(k) = \{h_1(k), \ldots, h_i(k), \ldots\}$
- Each $h_i(k)$ produces a bit
- At level i, $h_i(k) = 0$, take left branch, otherwise right branch
- Example: bit representation

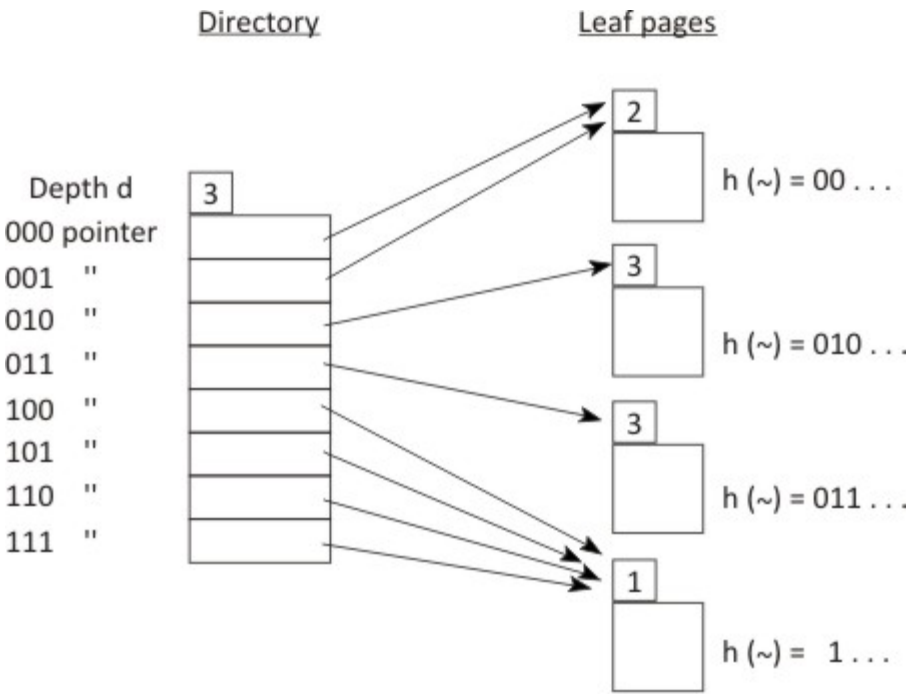
Module 3: Hashing

Lecture 9: Static and Dynamic Hashing

Prev topic Next topic Prev page
Next page

Extendible hashing

- Directory of pointers to buckets (leaf pages)
- Directory has global depth d
- 2^d pointers to leaf pages
- Pointer i contains keys starting with bit string i
- Leaf page has local depth $l \leq d$
- Leaf page j contains keys starting with bit string j



Module 3: Hashing

Lecture 9: Static and Dynamic Hashing

Insertion

- When leaf page overflows
  - If $l < d$, leaf page split into two and l is incremented for both new leaf pages
  - If $l = d$, directory doubles in size, d is incremented and leaf page splits