HASH-BASED DISK INDEXING
(BASED ON THE COW BOOK: 11.1 AND 11.2)
Introduction

• Hash-based indexes are best for equality selections. Cannot support range searches.
  – Static hashing
  – Extendible hashing (dynamic)
  – Linear hashing (dynamic) – not covered in the course, see 11.3 in the cow book
Static Hashing

- # primary bucket pages fixed, allocated sequentially, never de-allocated; overflow pages if needed.
- \( h(k) \mod M = \) bucket to which data entry with key \( k \) belongs. (\( M = \# \) of buckets)
Static Hashing (Contd.)

• Buckets contain data entries.
• Hash function works on search key field of record r. Must distribute values over range 0 ... M-1.
  – What is a good hash function?
  – \( h(key) = (a \times key + b) \) usually works well.
  – a and b are constants; lots known about how to tune h.
• Long overflow chains can develop and degrade performance.
  – Reorganization is expensive and may block queries
  – Extendible and Linear Hashing: Dynamic techniques to fix this problem.
Extendible Hashing

• Why not re-organize file by doubling the number of buckets?
  – Note that reading and writing all pages is expensive!

• Idea:
  – Use directory of pointers to buckets
  – On overflow, double the directory (not the # of buckets)
  – Why does this help?
    • Directory is much smaller than the entire index file
    • Only one page of data entries is split.
      • No overflow page! (caveat: duplicates w.r.t. the hash function)
  – Trick lies in how the hash function is adjusted!
Example

• Directory an array

• Search for k:
  – Apply hash function \( h(k) \)
  – Take last \textit{global depth} \\
    \# bits of \( h(k) \)

• Insert:
  – If the bucket has space, insert, done
  – If the bucket is full, \textit{split} it, re-distribute
  – If necessary, double the directory

\textbf{Can you identify this hash function?}
Example

Can you identify this hash function?
Insert 20

LOCAL DEPTH
GLOBAL DEPTH

DATA PAGES
Insert 20

Does splitting a bucket always require doubling the directory?

What happens if we had used most-significant bits?
Comments on Extendible Hashing

• How many disk accesses for equality search?
  – One if directory fits in memory, else two.

• Directory grows in spurts, and, if the distribution of hash values is skewed, directory can grow large.

• Do we ever need overflow pages?
  – Multiple entries with same hash value cause problems!

• **Delete:** Reverse of inserts
  – Can merge with split image.
  – Can shrink the directory by half. When?
    • Each directory element points to same bucket as its split image
  – Is shrinking/merging a good idea?