CSE 326: Data Structures

Topic #10: Hashing (3)

Ashish Sabharwal
Autumn, 2003

Today's Outline

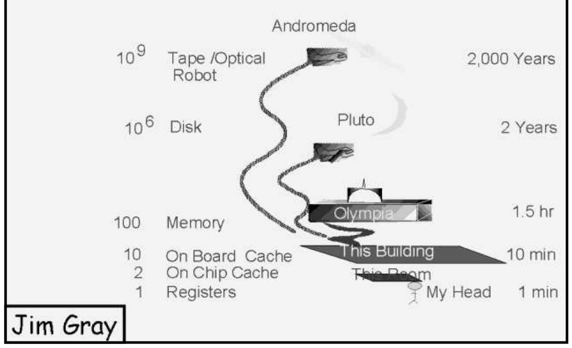
- Admin:
  - Hardcopy turnin for Project 2 – *now!*
  - Homework 2 due Friday
  - Start looking for a partner for Project 3
    (must be someone different from your Project 2 partner)
- Finish **Hashing**
  - Double hashing, rehashing
  - Extendible hashing
- Group Quiz #4

When to Rehash?

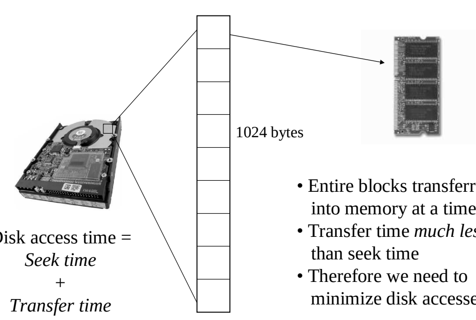
Many alternatives:

- Rehash when table is half full
- Rehash when insertion fails in open addressing
- Rehash when insertion becomes very slow in separate chaining
- Rehash when λ crosses a certain threshold

Something We Again Forgot: Disk Acesses



We Want To Minimize Disk Accesses!



Disk access time =
Seek time
+
Transfer time

- Entire blocks transferred into memory at a time
- Transfer time *much less* than seek time
- Therefore we need to minimize disk accesses!

Solution: Extendible Hashing

Hashing technique for huge data sets
- Optimizes to reduce disk accesses

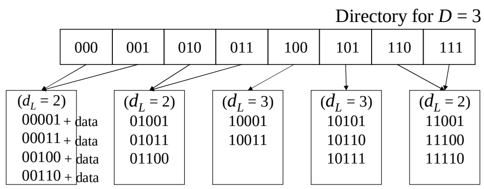
Hash "table" contains
1. Directory
   2^D entries, D bits per entry, pointers to leaf buckets
2. Leaf Buckets
   Keys in leaf L have $d_L \leq D$ bits in common with parent key, leaves store all data

Properties
- Only 2 levels in the table – only 2 disk accesses for find!
- Each leaf bucket fits on one disk block – caching
- Better than B-Trees if order is not important – *why?*

Extendible Hash Table

Directory entry : *key prefix* (first *D* bits) and a pointer to the bucket with all keys starting with that prefix

Bucket entry : keys matching on first $d_L \leq D$ bits, plus the data associated with those keys



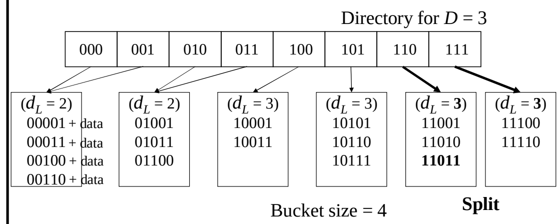
insert(**11010**)?

insert(**11011**)?

Bucket size = 4

7

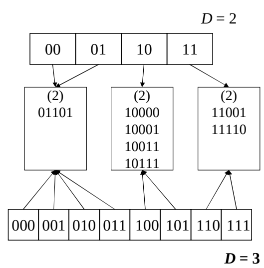
Inserting Using Bucket-Split



8

Insertion Using Directory-Expansion

1. insert(10010)
   But, no room to insert,
   only one parent,
   and *no adoption!*
2. Solution:
   Expand directory
   Now do a bucket-split



More expensive!

How to ensure this is uncommon?

9

What if Extendible Hashing Doesn't Cut It?

Option 1: Store only pointers/references to the items: (key, value) pairs separately on disk

Option 2: Improve hash function; Rehash

10

The One-Slide Hash

Hash function: maps keys to integers

Collision resolution

1. Separate Chaining
   – Expand beyond hashtable via secondary Dictionaries
   – Allows $\lambda > 1$
2. Open Addressing
   – Expand within hashtable
   – Secondary probing: {linear, quadratic, double hash}
   – $\lambda \leq 1$ (by definition!)
   – $\lambda \leq ½$ (by preference!)

Choosing a Hash Function

- Make sure table size is prime
- Careful choice for strings
- **"Perfect hashing"**
  – If keys known in advance, tune hash function for them!

Rehashing

- Tunes up hashtable when, e.g., λ crosses a threshold

Extendible hashing

- For disk-based data

11

Search ADT Implementations

	insert	find	delete
• Unsorted list	$\Theta(1)$	$\Theta(n)$	$\Theta(n)$
• Sorted list	$\Theta(n)$	$\Theta(\log n)$?	$\Theta(n)$
• Trees	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(\log n)$
• Hash Table	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$

(average case)

Is there anything a hash table *cannot* do efficiently?

You'll answer this in quiz #4!

12