CSE 373
Data Structures & Algorithms

Lecture 12
Hashing (III)
Summary

• Extensible Hash Tables
  – Weiss chapter 5.7

• Linear Hash Tables
  – Not covered in the textbook
  – Hence, not required for the midterm
Hash Tables on Disk

- Idea: 1 block = 1 bucket
- M = # keys per block

```
<table>
<thead>
<tr>
<th>0</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>f</td>
</tr>
<tr>
<td>2</td>
<td>g</td>
</tr>
<tr>
<td>3</td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>c</td>
</tr>
</tbody>
</table>
```
Example

- Assume $M = 2$
- $h(e) = 0$
- $h(b) = h(f) = 1$
- $h(g) = 2$
- $h(a) = h(c) = 3$

Here: $h(x) = x \mod 4$
Searching in a Hash Table

- Search for a:
- Compute $h(a)=3$
- Read block 3
- 1 disk access
Insertion in Hash Table

- Place in right bucket, if space
- E.g. h(d)=2

```
 0 e
 1 b
 2 f
g
 2 d
 3 a
c
```
Extensible Hash Table

• Allows hash table to grow, to avoid performance degradation
• Assume a hash function \( h \) that returns numbers in \( \{0, \ldots, 2^D - 1\} \)
• Start with \( n = 2^d \ll 2^D \), only look at first \( d \) most significant bits
Extensible Hash Table

- D=4=number of bits computed by hash function
- d=1=number of bits currently used
- n=2^d=2=directory size

Note: we only look at the first bit (0 or 1)
Insertion in Extensible Hash Table

• Insert 1110
Insertion in Extensible Hash Table

• Now insert 1010

• Need to extend directory, split buckets
• \( d \) becomes 2
Insertion in Extensible Hash Table

- $d=2$

- 00
- 01
- 10
- 11

- 0(010)
- 10(11)
- 10(10)
- 11(10)
Insertion in Extensible Hash Table

• Now insert 0000, then 0101

• Need to split bucket
Insertion in Extensible Hash Table

• After splitting the block
Extensible Hash Table

• How many buckets do we need to inspect during an insertion?

• How many entries in the directory do we need to touch after an insertion?
Performance Extensible Hash Table

• No chaining needed: access always O(1)

• BUT:
  – Extensions can be costly and disruptive
  – After an extension directory may no longer fit in memory
Linear Hash Table

• Idea: extend directory size n only by one
• Problem: n = no longer a power of 2
• Let d be such that $2^d \leq n < 2^{d+1}$
• After computing $h(k)$, use last d bits:
  – If last d bits represent a number $\geq n$, change Most Significant Bit (MSB) from 1 to 0
  – This way we are guaranteed to get a number $< n$
  – This is called BIT FLIP
• Note:
  – Extensible hash tables use the first d bits
  – Linear hash table use the last d bits
  – What are the tradeoffs? Think about this during the next few slides...
Linear Hash Table Example

• n=3
Linear Hash Table Example

- Insert 1000: overflow blocks...
Linear Hash Tables

- Extension: independent on overflow blocks
- Extend $n := n + 1$ when average number of records per block exceeds (say) 80%
Linear Hash Table Extension

- From \( n=3 \) to \( n=4 \)

- Only need to split one bucket (which one?)
Linear Hash Table Extension

• From $n=3$ to $n=4$ finished
Linear Hash Table Extension

From n=4 to n=5: need new bit

Example: Insert 1000