CSE 326: Data Structures

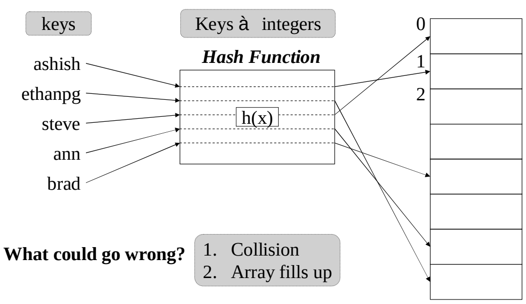
Topic #10: Hashing (2)

Ashish Sabharwal
Autumn, 2003

Today's Outline

- Admin
  - Project 2 due tonight!
  - Pick up Homework 2; due Friday
  - A word on collaboration and acknowledgement
- **Hashing**: collision resolution strategies
  - Separate chaining
  - Open addressing
    - Linear probing, quadratic probing, double hashing
  - Rehashing

Review: Hash Table Approach



What could go wrong?
1. Collision
2. Array fills up

Review: Hash Table Code

```
value find(Key k) {
  int index = hash(k) % tableSize;
  return Table[index];
}
```

Key Questions:

1. What should the **hash function** be?
2. How should we resolve **collisions**?
3. What should the **table size** be?

Review: A Good Hash Function…

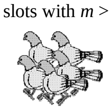
…is easy (fast) to compute
(O(1) *and* practically fast)

…distributes the data evenly $\Rightarrow$ few collisions
(ideally, hash(a) % size $\neq$ hash(b) % size $\Rightarrow$ no collision)

…uses the whole hash table
($\forall$ k, $0 \leq k <$ size, $\exists$ i such that hash(i) % size = k)

Collisions

- Pigeonhole principle says we can't avoid all collisions
  - try to hash without collision m keys into n slots with $m > n$
  - e.g., try to put 7 pigeons into 5 holes



- What do we do when two keys hash to the same entry?
  1. Separate chaining: put little dictionaries in each entry
     ↖ *shove extra pigeons in one hole!*
  2. Open addressing: pick a next entry to try

Load Factor

How often do collisions occur?

- Depends on the **load factor, λ**

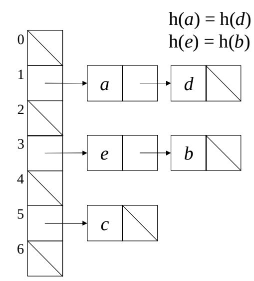
$$\lambda = \frac{\text{\# of entries in table}}{\text{tableSize}}$$

High λ ⇒ more collisions, bad performance

Low λ ⇒ less collisions, good performance

1. Separate Chaining

- Put a mini-Dictionary at each entry
  - Usually a linked list
  - Why not a search tree?
- Properties
  - Average list size =
  - Works even when λ > 1
  - performance degrades with length of chains



Remember Splay Trees?

- Where in the list would you put a new entry?
- What might you do when you perform find on a key?

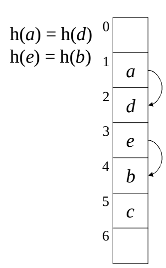
Load Factor in Separate Chaining

- Search cost
  - unsuccessful search:
  - successful search:
- Desired load factor:

2. Open Addressing

What if we only allow one Key at each entry?
- two objects that hash to the same spot can't both go there
- first one there gets the spot
- next one must **probe** for another spot

- Properties
  - Requires λ ≤ 1
  - performance degrades with difficulty of finding right spot



Salary-Boosting Obfuscation

"**Open** Hashing"	"Closed Hashing"
equals	equals
"Separate Chaining"	"**Open** Addressing"

Probing Function, *f*(*x*)

- The Probing process
  - First probe - given a key k, hash to $h(k)$
  - Second probe - if $h(k)$ is occupied, try $h(k) + f(1)$
  - Third probe - if $h(k) + f(1)$ is occupied, try $h(k) + f(2)$
  - And so on.
- Probing properties
  - force $f(0) = 0$
  - the i^{th} probe is to $(h(k) + f(i))$ mod size
- **When does the probe fail?**
- **Does that mean the table is full?**

13

2a. Linear Probing

$$f(i) = i$$

- Probe sequence is
  - $h(k)$ mod size
  - $(h(k) + 1)$ mod size
  - $(h(k) + 2)$ mod size
  - …

14

Linear Probing Example

insert(76) insert(93) insert(40) insert(47) insert(10) insert(55)

76%7 = 6 93%7 = 2 40%7 = 5 47%7 = 5 10%7 = 3 55%7 = 6

0	
1	
2	
3	
4	
5	
6	76

Problem?

15

Load Factor in Linear Probing

- Search cost
  - Unsuccessful search
  - Successful search

16

Load Factor in Linear Probing

- For *any* $\lambda < 1$, linear probing will find an empty slot
- Search cost (for large table sizes)
  - successful search: $\frac{1}{2}\left(1+\frac{1}{(1-\lambda)}\right)$
  - unsuccessful search: $\frac{1}{2}\left(1+\frac{1}{(1-\lambda)^2}\right)$
- Linear probing suffers from ***primary clustering***
- Performance quickly degrades for $\lambda > 1/2$

17

2b. Quadratic Probing

$$f(i) = i^2$$

- Probe sequence is
  - $h(k)$ mod size
  - $(h(k) + 1)$ mod size
  - $(h(k) + 4)$ mod size
  - $(h(k) + 9)$ mod size
  - …
- Implementation trick: $f(i+1) =$
  - No multiplication!

18

Quadratic Probing Example

insert(76) 76%7 = 6 | insert(40) 40%7 = 5 | insert(48) 48%7 = 6 | insert(5) 5%7 = 5 | insert(55) 55%7 = 6

But… insert(47) 47%7 = 5

0	
1	
2	
3	
4	
5	
6	76

19

Quadratic Probing: Success guarantee for $\lambda < ½$

- If size is prime and $\lambda < ½$, then quadratic probing will find an empty slot in size/2 probes or fewer.
  - show for all `0 ≤ i,j ≤ size/2` and `i ≠ j`
    `(h(x) + i²) mod size ≠ (h(x) + j²) mod size`
  - by contradiction: suppose that for some i ≠ j:
    `(h(x) + i²) mod size = (h(x) + j²) mod size`
    ⇒ `i² mod size = j² mod size`
    ⇒ `(i² - j²) mod size = 0`
    ⇒ `[(i + j)(i - j)] mod size = 0`
  - but how can `i + j = 0` or `i + j = size` when `i ≠ j` and `i,j ≤ size/2`?
  - same for `i - j mod size = 0`

20

Quadratic Probing: Properties

- For *any* $\lambda < ½$, quadratic probing will find an empty slot; for bigger λ, quadratic probing *may* find a slot
- Quadratic probing does not suffer from *primary* clustering: keys hashing to the same *area* are not bad
- But what about keys that hash to the same *spot*?
  - ***Secondary Clustering!***

21

2c. Double Hashing

$$f(i) = i \cdot \text{hash}_2(k)$$

hmm…. what was k?

- Probe sequence is
  - $h_1(k)$ mod size
  - $(h_1(k) + 1 \cdot h_2(k))$ mod size
  - $(h_1(k) + 2 \cdot h_2(k))$ mod size
  - …
- Goal?

22

A Good Double Hash Function…

…is quick to evaluate.

…differs from the original hash function – keys that h_1 hashes close by must hash far away using h_2

…never evaluates to 0 (mod size).

One good choice is to choose a *prime* R < size and:
$\text{hash}_2(k) = R - (k \bmod R)$

What could go wrong if table size S were not prime?

23

Double Hashing Example (R=5)

	insert(76)	insert(93)	insert(40)	insert(47)	insert(10)	insert(55)
	76%7 = 6	93%7 = 2	40%7 = 5	47%7 = 5 5 - (47%5) = 3	10%7 = 3	55%7 = 6 5 - (55%5) = 5
0						
1				47	47	47
2		93	93	93	93	93
3					10	10
4						55
5			40	40	40	40
6	76	76	76	76	76	76
probes:	1	1	1	2	1	2

24

Load Factor in Double Hashing

- For *any* $\lambda < 1$, double hashing will find an empty slot (given appropriate table size and $hash_2$)
- Search cost appears to approach optimal (random hash):
  - successful search: $\frac{1}{\lambda} \ln \frac{1}{1-\lambda}$
  - unsuccessful search: $\frac{1}{1-\lambda}$
- No primary clustering and no secondary clustering
- Cost?

25

Deletion with Open Addressing

insert(7) delete(2) find(7)

0	0
1	1
2	2
3	
4	
5	
6	

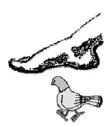
Solution?

(problem 4 on homework 2)

26

The Squished Pigeon Principle J

- An insert using open addressing *cannot* work with a load factor of 1 or more.
- An insert using open addressing with quadratic probing may not work with a load factor of ½ or more.
- Whether you use separate chaining or open addressing, large load factors lead to poor performance!



How can we relieve the pressure on the pigeons?

27

Rehashing

- When the load factor gets “too large” (over a constant threshold on λ), rehash all the elements into a new, larger table:
  - spreads keys back out, may drastically improve performance
  - avoids failure for open addressing techniques
  - allows arbitrarily large tables starting from a small table
  - clears out lazily deleted items
- Cost?
- Can we just copy over into a bigger array?

28

Rehashing Example

0	20
1	96
2	82
3	
4	89

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

29