Open addressing: Analysis

- Consider load factor α
  - for n keys
  - in m cells
  - $\alpha = n/m$

Open Addressing: Analysis

Average case, under some simplifying assumptions, expected time for insertion is:

- Double hashing: $1/(1-\alpha)$
- Linear probing: $1/(1-\alpha)^2$
- Example: $\alpha = 0.75$
  - Double hash insertion: 4 probes
  - Linear probing insertion: 16 probes

A nice explanation of the assumptions, by Tim Roughgarden:

- https://www.youtube.com/watch?v=nWQv4BCEhjM&list=PLXFMmIk03Dt7Q0xr1PIAriY5623cKiH7V&index=73

Open Addressing: Analysis

- Average case lookup:
  - Double hash ~ ½(1 + 1/(1-α))
  - Linear probing ~ ½(1 + 1/(1-α)^2)

	Double hash	Linear probe
α	$\frac{1}{2}(1+\frac{1}{1-\alpha})$	$\frac{1}{2}(1+\frac{1}{(1-\alpha)^2})$
50%	1.5	2.5
75%	2.5	8.5
90%	5.5	50.5

Open Addressing: Analysis

Degraded performance as table nears full.

α	$\frac{1}{2}(1+\frac{1}{1-\alpha})$	$\frac{1}{2}(1+\frac{1}{(1-\alpha)^2})$
50%	1.5	2.5
75%	2.5	8.5
90%	5.5	50.5

Catastrophic failure when table full.

- Performance depends on $\alpha = (n/m)$, so choice of table size must be appropriate

Open Addressing: Analysis

Degraded performance as table nears full.

α	$\frac{1}{2}(1+\frac{1}{1-\alpha})$	$\frac{1}{2}(1+\frac{1}{(1-\alpha)^2})$
50%	1.5	2.5
75%	2.5	8.5
90%	5.5	50.5

Catastrophic failure when table full.

- How and why do people use open addressing?

Open Addressing: Analysis

Degraded performance as table nears full.

α	$\frac{1}{2}(1+\frac{1}{1-\alpha})$	$\frac{1}{2}(1+\frac{1}{(1-\alpha)^2})$
50%	1.5	2.5
75%	2.5	8.5
90%	5.5	50.5

Catastrophic failure when table full.

- How might you prevent degraded performance?

Hash tables: Summary

O(1) lookup!!

- But only on average
- And only for small α

Some bad worst cases:

- Table full (open addressing)
- Table near full (open addressing)
- Everything hashes to same/similar slot (all)

Hash tables: Summary

Performance degrades:

- For linear chaining, degrades gracefully
- For open address chaining, degrades, then can fail catastrophically.

Cannot retrieve items in sorted order

A nice review of hashing, including some advanced topics:

- http://courses.csail.mit.edu/6.006/fall10/lectures/lecture5.pdf
- http://courses.csail.mit.edu/6.006/fall10/lectures/lecture6.pdf
- http://courses.csail.mit.edu/6.006/fall10/lectures/lecture7.pdf

Some notes about hash tables

Hash tables show fast lookup

- O(1) lookup
- Better than log n

But a good hash function may be very computationally expensive (more operations than searching a small linked list or tree).

Other uses of hashing

Duplicate detection, *e.g.* for documents:

- If hash signatures are different, documents can't be duplicates
- Only have to thoroughly check a few documents

Plagiarism detection

Other uses of hashing

Cryptography

- Hash function can be used to encode data (example: website passwords)
- Some hash functions are "one way" – easy to compute hash of x, but hard to compute x from hash