CMP448: Algorithms

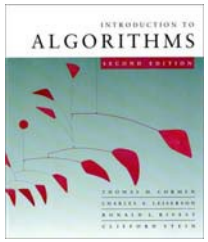
Lecture 06: Hashing II

Mohamed Alaa El-Dien Aly
Computer Engineering Department
Cairo University
Spring 2013

Agenda

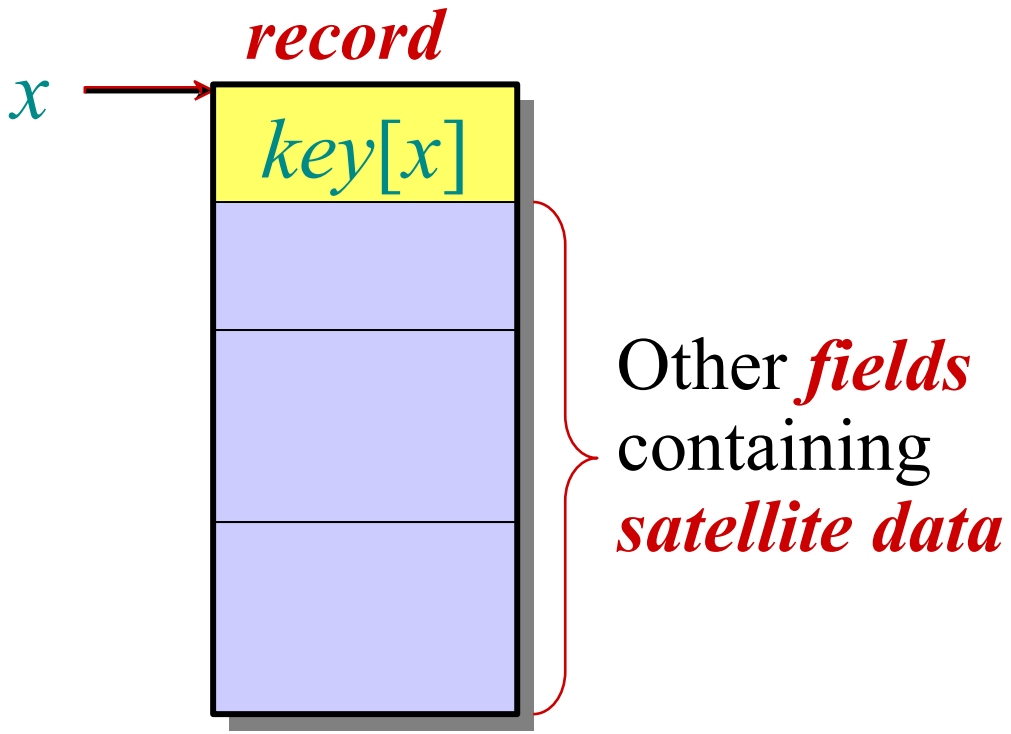
- Open Addressing
- Universal Hashing
- Perfect Hashing

Acknowledgment
A lot of slides adapted from the slides of Erik Demaine and Charles Leiserson



Symbol-table problem

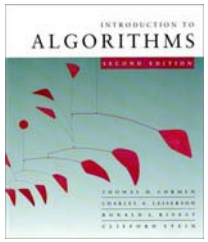
Symbol table S holding n ***records***:



Operations on S:

- INSERT(S, x)
- DELETE(S, x)
- SEARCH(S, k)

How should the data structure S be organized?



Direct-access table

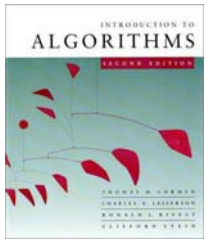
IDEA: Suppose that the keys are drawn from the set $U \subseteq \{0, 1, \ldots, m-1\}$, and keys are distinct. Set up an array $T[0 \,..\, m-1]$:

$$T[k] = \begin{cases} x & \text{if } x \in K \text{ and } key[x] = k, \\ \text{NIL} & \text{otherwise.} \end{cases}$$

Then, operations take $\Theta(1)$ time.

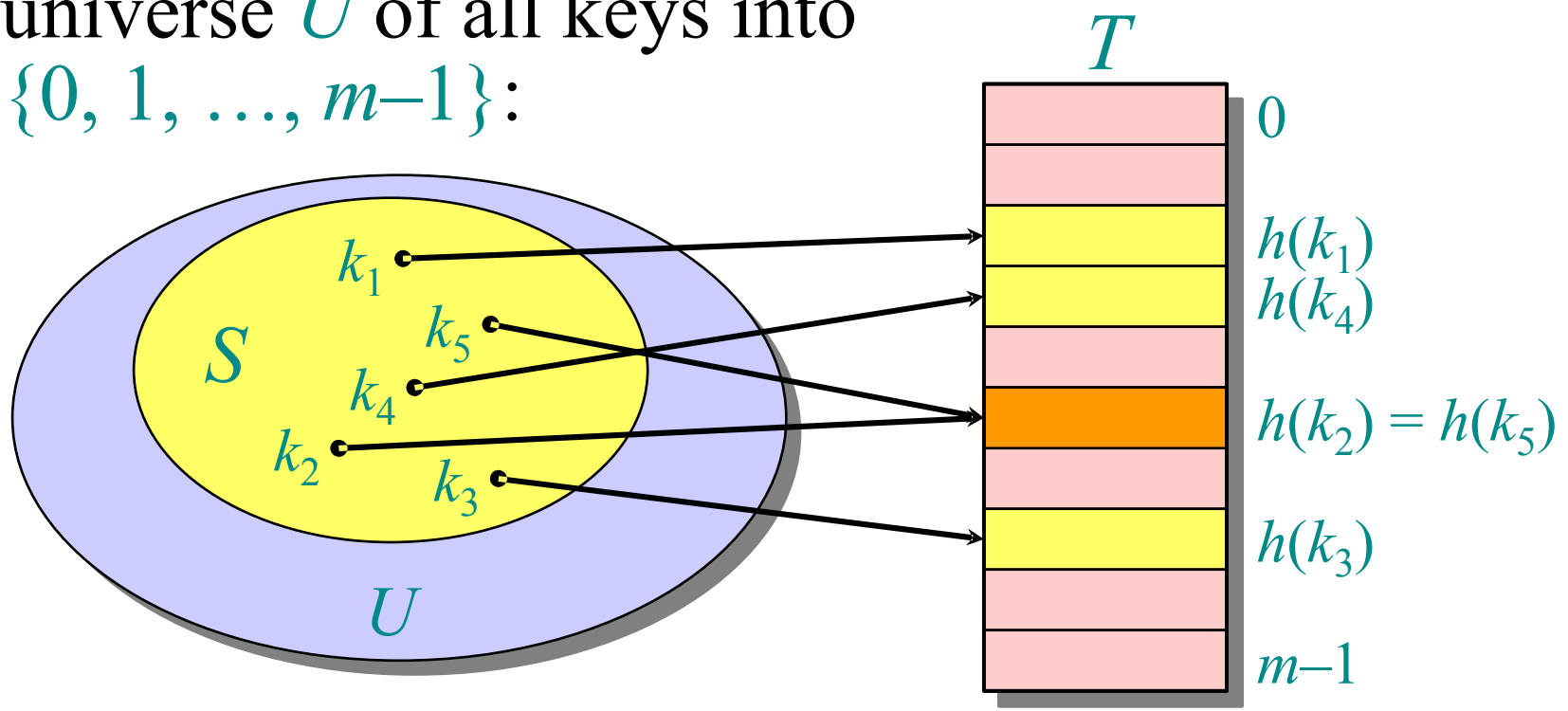
Problem: The range of keys can be large:

- 64-bit numbers (which represent 18,446,744,073,709,551,616 different keys),
- character strings (even larger!).



Hash functions

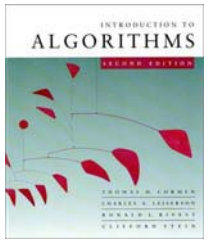
Solution: Use a ***hash function*** h to map the universe U of all keys into $\{0, 1, \ldots, m-1\}$:



When a record to be inserted maps to an already occupied slot in T, a ***collision*** occurs.

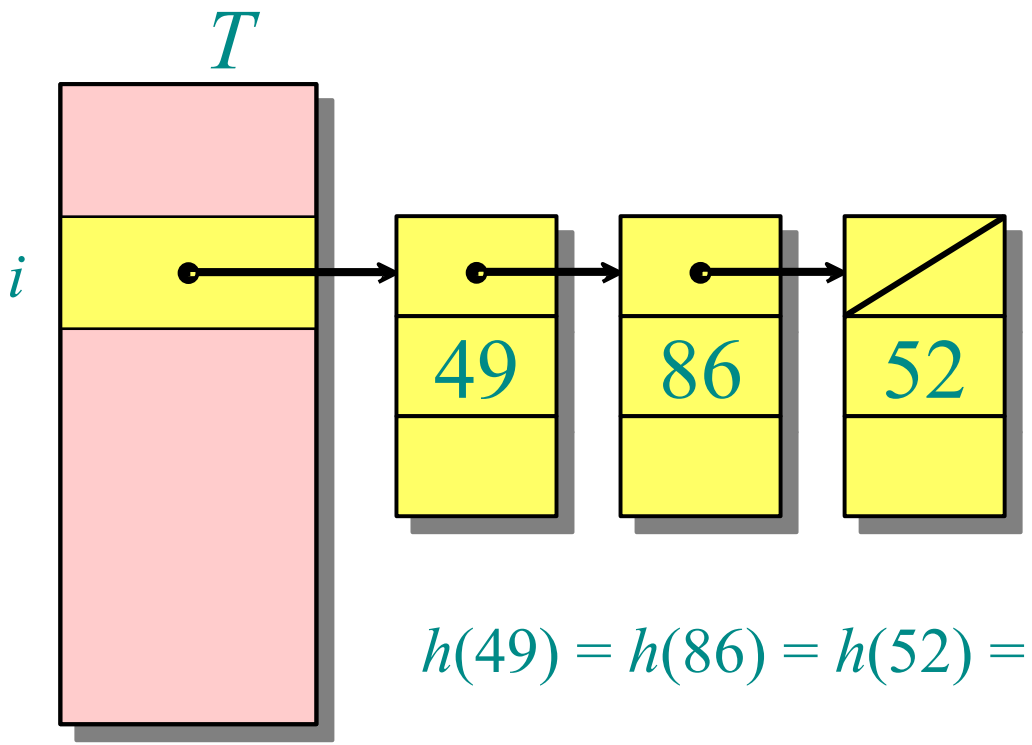
Collision Resolution

- We can resolve the collision by:
  - Chaining
  - Open Addressing

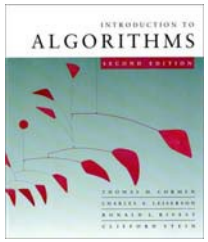


Resolving collisions by chaining

- Link records in the same slot into a list.



$h(49) = h(86) = h(52) = i$

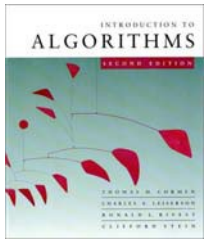


Resolving collisions by open addressing

No storage is used outside of the hash table itself.

- Insertion systematically probes the table until an empty slot is found.
- The hash function depends on both the key and probe number:

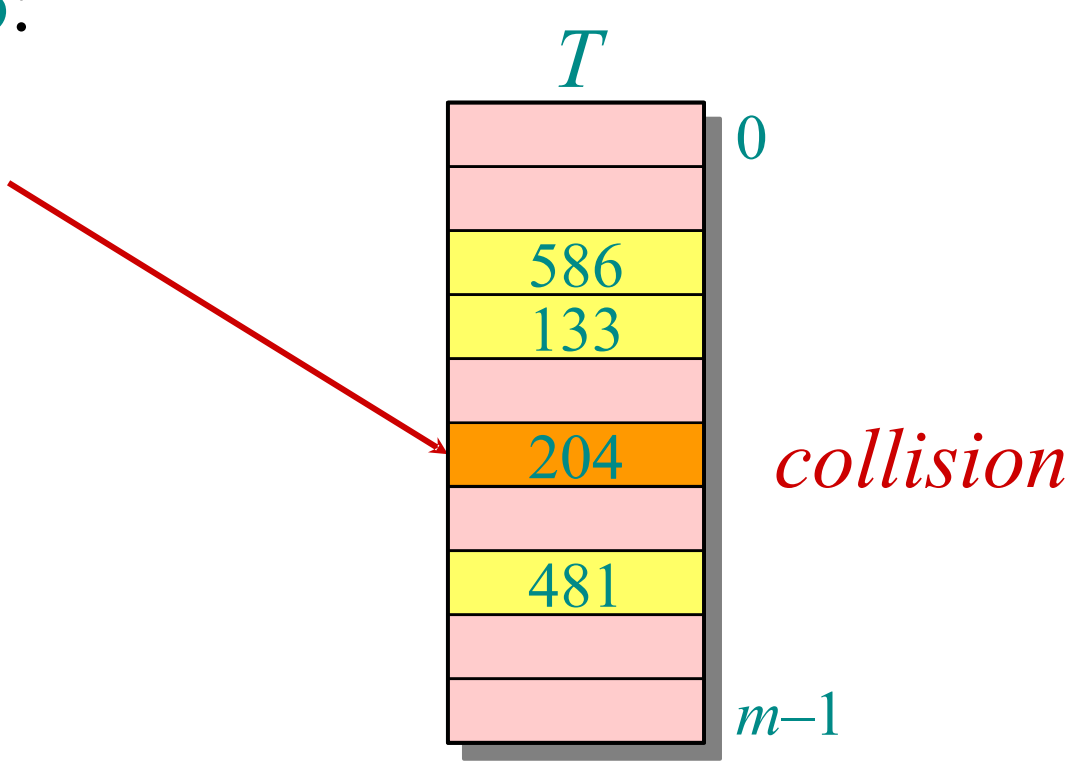
  $h : U \times \{0, 1, \ldots, m-1\} \to \{0, 1, \ldots, m-1\}$.
- The probe sequence $\langle h(k,0), h(k,1), \ldots, h(k,m-1)\rangle$ should be a permutation of $\{0, 1, \ldots, m-1\}$.
- The table may fill up, and deletion is difficult (but not impossible).

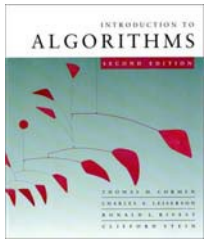


Example of open addressing

Insert key $k = 496$:

0. Probe $h(496,0)$

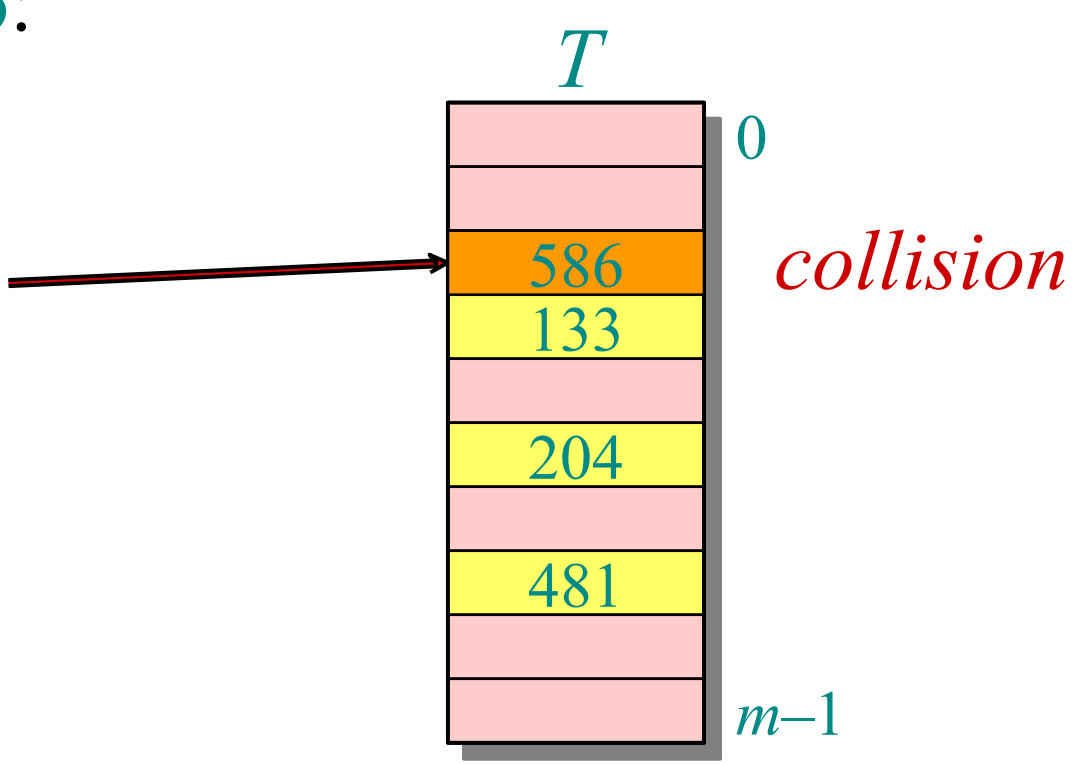


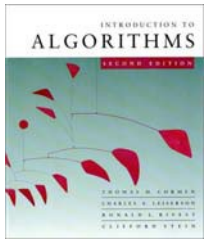


Example of open addressing

Insert key $k = 496$:

0. Probe $h(496,0)$
1. Probe $h(496,1)$

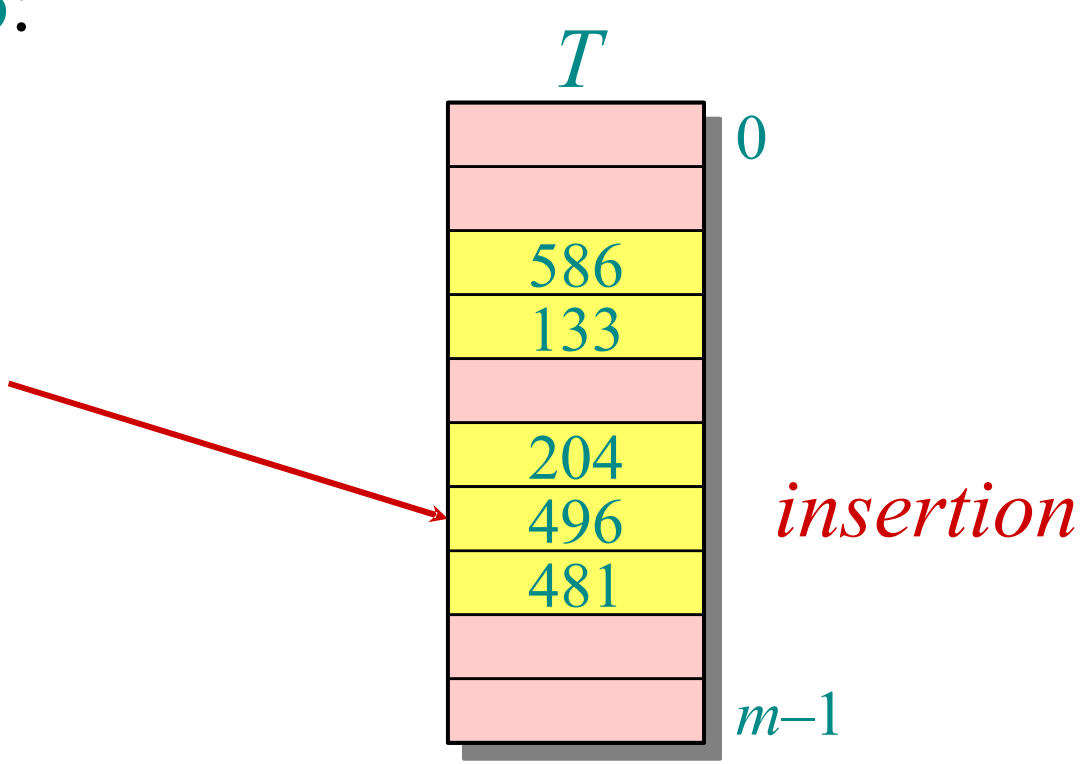


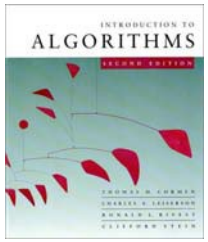


Example of open addressing

Insert key $k = 496$:

0. Probe $h(496,0)$
1. Probe $h(496,1)$
2. Probe $h(496,2)$

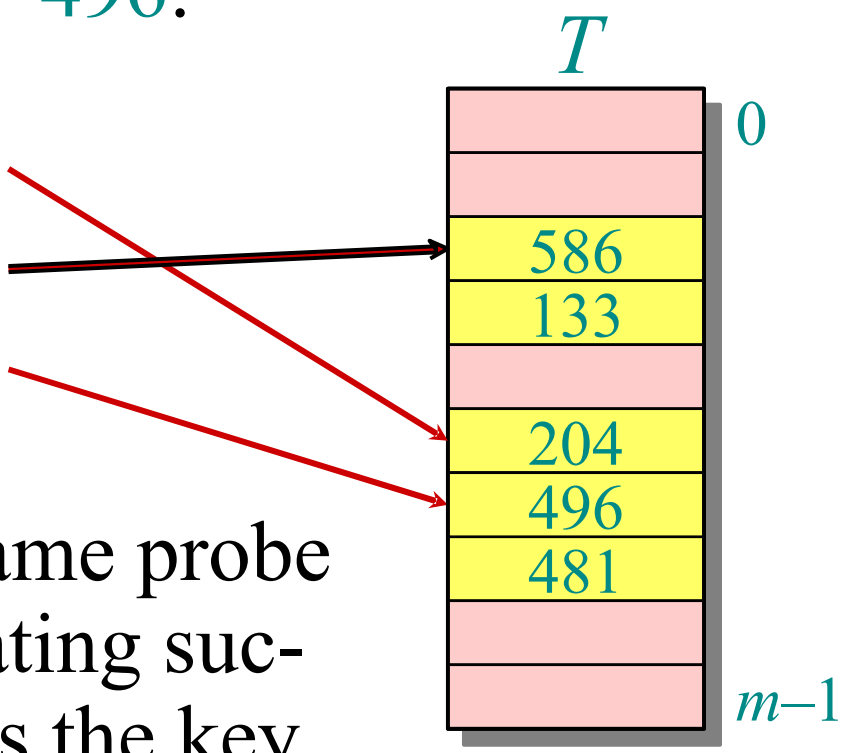




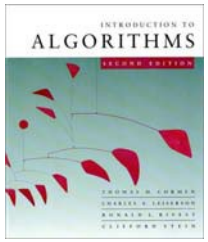
Example of open addressing

Search for key $k = 496$:

0. Probe $h(496,0)$
1. Probe $h(496,1)$
2. Probe $h(496,2)$



Search uses the same probe sequence, terminating successfully if it finds the key and unsuccessfully if it encounters an empty slot.



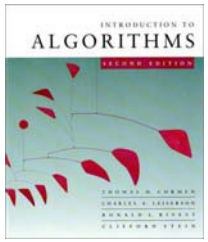
Probing strategies

Linear probing:

Given an ordinary hash function $h'(k)$, linear probing uses the hash function

$$h(k,i) = (h'(k) + i) \bmod m.$$

This method, though simple, suffers from ***primary clustering***, where long runs of occupied slots build up, increasing the average search time. Moreover, the long runs of occupied slots tend to get longer.



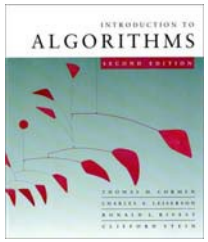
Probing strategies

Double hashing

Given two ordinary hash functions $h_1(k)$ and $h_2(k)$, double hashing uses the hash function

$$h(k,i) = (h_1(k) + i\ \ h_2(k)) \bmod m.$$

This method generally produces excellent results, but $h_2(k)$ must be relatively prime to m. One way is to make m a power of 2 and design $h_2(k)$ to produce only odd numbers.

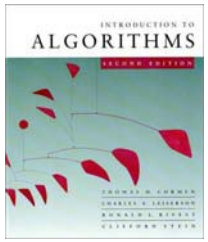


Analysis of open addressing

We make the assumption of ***uniform hashing:***

- Each key is equally likely to have any one of the $m!$ permutations as its probe sequence.

Theorem. Given an open-addressed hash table with load factor $\alpha = n/m < 1$, the expected number of probes in an unsuccessful search is at most $1/(1-\alpha)$.

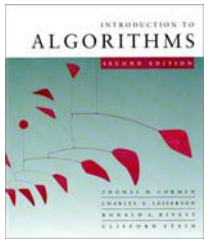


Proof of the theorem

Proof.

- At least one probe is always necessary.
- With probability n/m, the first probe hits an occupied slot, and a second probe is necessary.
- With probability $(n-1)/(m-1)$, the second probe hits an occupied slot, and a third probe is necessary.
- With probability $(n-2)/(m-2)$, the third probe hits an occupied slot, etc.

Observe that $\frac{n-i}{m-i} < \frac{n}{m} = \alpha$ for $i = 1, 2, \ldots, n$.



Proof (continued)

Therefore, the expected number of probes is

$$1+\frac{n}{m}\left(1+\frac{n-1}{m-1}\left(1+\frac{n-2}{m-2}\left(\cdots\left(1+\frac{1}{m-n+1}\right)\cdots\right)\right)\right)$$

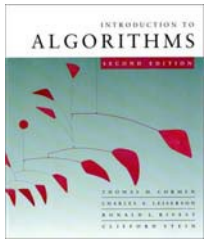
$$\le 1+\alpha\left(1+\alpha\left(1+\alpha\left(\cdots\left(1+\alpha\right)\cdots\right)\right)\right)$$

$$\le 1+\alpha+\alpha^2+\alpha^3+\cdots$$

$$=\sum_{i=0}^{\infty}\alpha^i$$

$$=\frac{1}{1-\alpha}\,.$$ □

The textbook has a more rigorous proof and an analysis of successful searches.

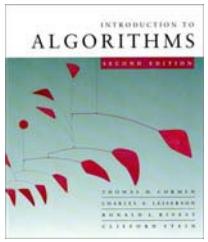


Implications of the theorem

- If α is constant, then accessing an open-addressed hash table takes constant time.
- If the table is half full, then the expected number of probes is $1/(1–0.5) = 2$.
- If the table is 90% full, then the expected number of probes is $1/(1–0.9) = 10$.

So far

- Direct Access Tables
  - If number of keys is not too large
- Hash Tables, resolving collisions with:
  - Chaining
  - Open Addressing



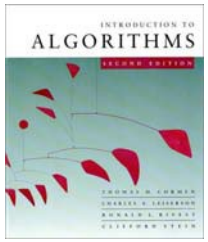
A weakness of hashing

Problem: For any hash function h, a set of keys exists that can cause the average access time of a hash table to skyrocket.

- An adversary can pick all keys from $\{k \in U : h(k) = i\}$ for some slot i.

IDEA: Choose the hash function at random, independently of the keys.

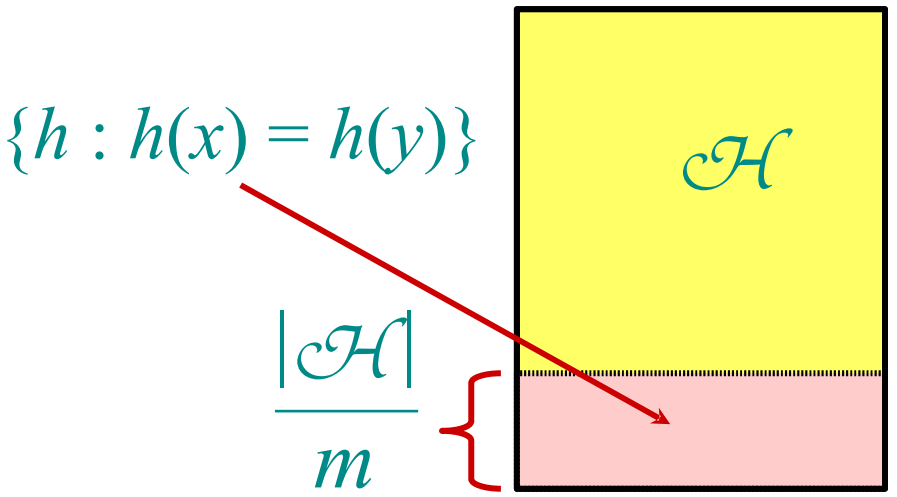
- Even if an adversary can see your code, he or she cannot find a bad set of keys, since he or she doesn't know exactly which hash function will be chosen.

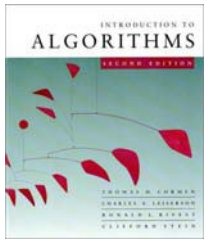


Universal hashing

Definition. Let U be a universe of keys, and let $\mathcal{H}$ be a finite collection of hash functions, each mapping U to $\{0, 1, \ldots, m-1\}$. We say $\mathcal{H}$ is ***universal*** if for all $x, y \in U$, where $x \neq y$, we have $|\{h \in \mathcal{H} : h(x) = h(y)\}| = |\mathcal{H}|/m$.

That is, the chance of a collision between x and y is $1/m$ if we choose h randomly from $\mathcal{H}$.

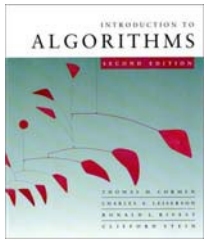




Universality is good

Theorem. Let h be a hash function chosen (uniformly) at random from a universal set $\mathcal{H}$ of hash functions. Suppose h is used to hash n arbitrary keys into the m slots of a table T. Then, for a given key x, we have

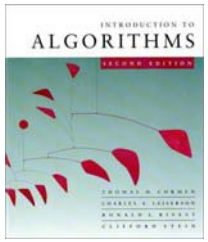
$$E[\#\text{collisions with } x] < n/m.$$



Proof of theorem

Proof. Let C_x be the random variable denoting the total number of collisions of keys in T with x, and let

$$c_{xy} = \begin{cases} 1 & \text{if } h(x) = h(y), \\ 0 & \text{otherwise.} \end{cases}$$



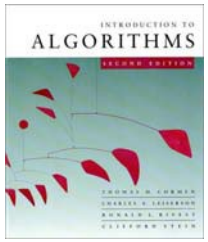
Proof of theorem

Proof. Let C_x be the random variable denoting the total number of collisions of keys in T with x, and let

$$c_{xy} = \begin{cases} 1 & \text{if } h(x) = h(y), \\ 0 & \text{otherwise.} \end{cases}$$

Note: $E[c_{xy}] = 1/m$ and $C_x = \sum_{y \in T - \{x\}} c_{xy}$.

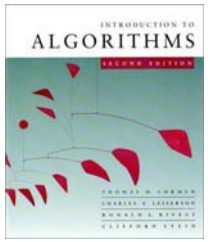
Why? Universal Set



Proof (continued)

$$E[C_x] = E\left[\sum_{y \in T - \{x\}} c_{xy}\right]$$

- Take expectation of both sides.

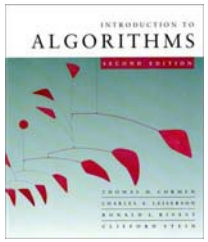


Proof (continued)

$$E[C_x] = E\left[\sum_{y \in T - \{x\}} c_{xy}\right]$$

$$= \sum_{y \in T - \{x\}} E[c_{xy}]$$

- Take expectation of both sides.
- Linearity of expectation.



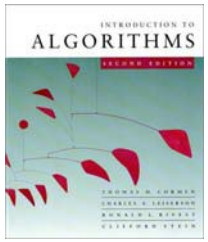
Proof (continued)

$$E[C_x] = E\left[\sum_{y \in T - \{x\}} c_{xy}\right]$$

$$= \sum_{y \in T - \{x\}} E[c_{xy}]$$

$$= \sum_{y \in T - \{x\}} 1/m$$

- Take expectation of both sides.
- Linearity of expectation.
- $E[c_{xy}] = 1/m$.



Proof (continued)

$$E[C_x] = E\left[\sum_{y \in T - \{x\}} c_{xy}\right]$$

- Take expectation of both sides.

$$= \sum_{y \in T - \{x\}} E[c_{xy}]$$

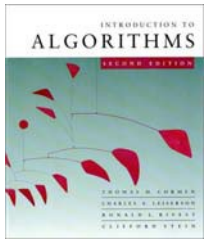
- Linearity of expectation.

$$= \sum_{y \in T - \{x\}} 1/m$$

- $E[c_{xy}] = 1/m$.

$$= \frac{n-1}{m} . \quad \square$$

- Algebra.



Constructing a set of universal hash functions

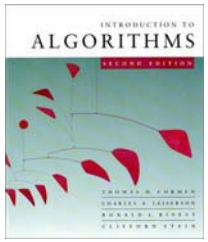
Let m be prime. Decompose key k into $r + 1$ digits, each with value in the set $\{0, 1, \ldots, m-1\}$. That is, let $k = \langle k_0, k_1, \ldots, k_r \rangle$, where $0 \le k_i < m$.

Randomized strategy:

Pick $a = \langle a_0, a_1, \ldots, a_r \rangle$ where each a_i is chosen randomly from $\{0, 1, \ldots, m-1\}$.

Define $h_a(k) = \sum_{i=0}^{r} a_i k_i \bmod m$.

Dot product, modulo m



Constructing a set of universal hash functions

Let m be prime. Decompose key k into $r + 1$ digits, each with value in the set $\{0, 1, \ldots, m-1\}$. That is, let $k = \langle k_0, k_1, \ldots, k_r \rangle$, where $0 \le k_i < m$.

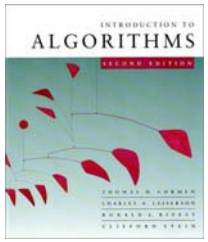
Randomized strategy:

Pick $a = \langle a_0, a_1, \ldots, a_r \rangle$ where each a_i is chosen randomly from $\{0, 1, \ldots, m-1\}$.

Define $h_a(k) = \sum_{i=0}^{r} a_i k_i \bmod m$.

Dot product, modulo m

How big is $\mathcal{H} = \{h_a\}$?



Constructing a set of universal hash functions

Let m be prime. Decompose key k into $r + 1$ digits, each with value in the set $\{0, 1, \ldots, m-1\}$. That is, let $k = \langle k_0, k_1, \ldots, k_r \rangle$, where $0 \le k_i < m$.

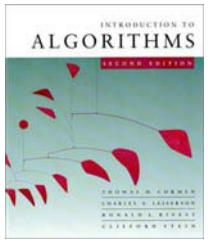
Randomized strategy:

Pick $a = \langle a_0, a_1, \ldots, a_r \rangle$ where each a_i is chosen randomly from $\{0, 1, \ldots, m-1\}$.

Define $h_a(k) = \sum_{i=0}^{r} a_i k_i \bmod m$.

Dot product, modulo m

How big is $\mathcal{H} = \{h_a\}$? $|\mathcal{H}| = m^{r+1}$. ← **REMEMBER THIS!**



Universality of dot-product hash functions

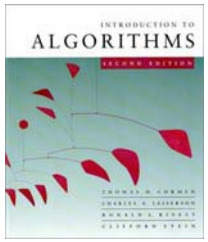
Theorem. The set $\mathcal{H} = \{h_a\}$ is universal.

Proof. Suppose that $x = \langle x_0, x_1, \ldots, x_r \rangle$ and $y = \langle y_0, y_1, \ldots, y_r \rangle$ be distinct keys. Thus, they differ in at least one digit position, wlog position 0.

For how many $h_a \in \mathcal{H}$ do x and y collide?

We must have $h_a(x) = h_a(y)$, which implies that

$$\sum_{i=0}^{r} a_i x_i \equiv \sum_{i=0}^{r} a_i y_i \pmod{m} .$$



Proof (continued)

Equivalently, we have

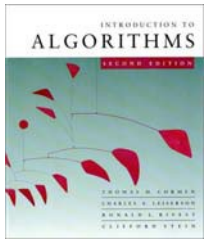
$$\sum_{i=0}^{r} a_i (x_i - y_i) \equiv 0 \pmod{m}$$

or

$$a_0 (x_0 - y_0) + \sum_{i=1}^{r} a_i (x_i - y_i) \equiv 0 \pmod{m} ,$$

which implies that

$$a_0 (x_0 - y_0) \equiv -\sum_{i=1}^{r} a_i (x_i - y_i) \pmod{m} .$$



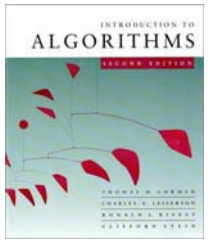
Fact from number theory

Theorem. Let m be prime. For any $z \in \mathbb{Z}_m$ such that $z \neq 0$, there exists a unique $z^{-1} \in \mathbb{Z}_m$ such that

$$z \cdot z^{-1} \equiv 1 \pmod{m}.$$

Example: $m = 7$.

z	1	2	3	4	5	6
z^{-1}	1	4	5	2	3	6



Back to the proof

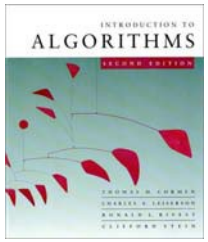
We have

$$a_0(x_0 - y_0) \equiv -\sum_{i=1}^{r} a_i(x_i - y_i) \pmod{m},$$

and since $x_0 \neq y_0$, an inverse $(x_0 - y_0)^{-1}$ must exist, which implies that

$$a_0 \equiv \left(-\sum_{i=1}^{r} a_i(x_i - y_i) \right) \cdot (x_0 - y_0)^{-1} \pmod{m}.$$

Thus, for any choices of $a_1, a_2, \ldots, a_r$, exactly one choice of a_0 causes x and y to collide.



Proof (completed)

Q. How many h_a's cause x and y to collide?

A. There are m choices for each of $a_1, a_2, \ldots, a_r$, but once these are chosen, exactly one choice for a_0 causes x and y to collide, namely

$$a_0 = \left(\left(-\sum_{i=1}^{r} a_i (x_i - y_i) \right) \cdot (x_0 - y_0)^{-1} \right) \bmod m .$$

Thus, the number of h's that cause x and y to collide is $m^r = m^{r+1}/m = |\mathcal{H}| / m$. □

Another Example

Define $\mathbb{Z}_p = \{0, 1, \ldots, p\text{-}1\}$ and $\mathbb{Z}_p^* = \{1, \ldots, p\text{-}1\}$ for p prime

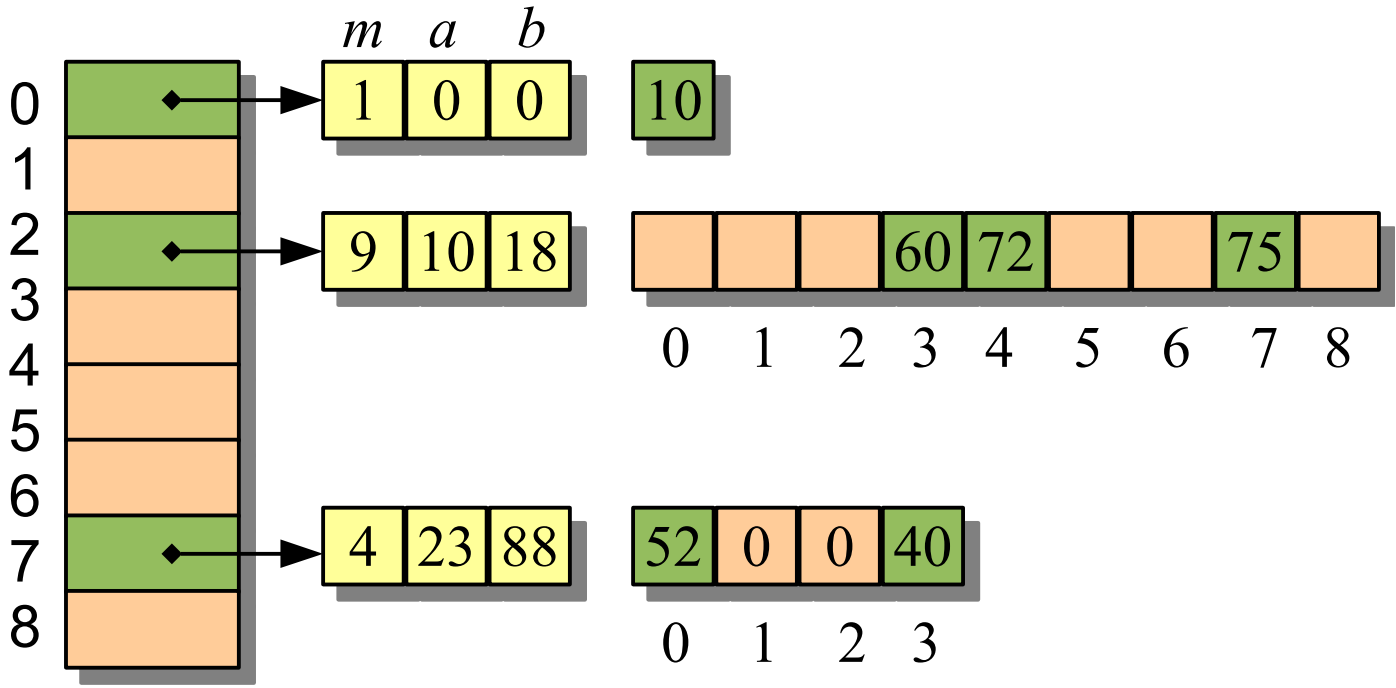
Define $h_{ab}(k) = ((a\,k + b) \bmod p) \bmod m$
for $a \in \mathbb{Z}_p^*$ and $b \in \mathbb{Z}_p$

The family $\mathcal{H}_{pm} = \{h_{ab} : a \in \mathbb{Z}_p^* \text{ and } b \in \mathbb{Z}_p\}$ is a universal family of hash functions.

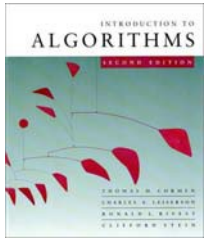
Perfect Hashing

Given a static set of n keys, construct a hash table of total size $m = O(n)$ such that the SEARCH operation takes $\Theta(1)$ in the worst case.

Idea: Use a two stage hash table such that there are no collisions at the second stage.



$$m = 9,\ a = 3,\ b = 42,\ p = 101$$

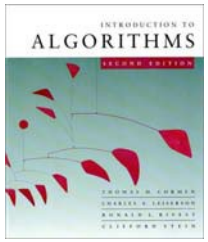


Collisions at level 2

Theorem. Let $\mathcal{H}$ be a class of universal hash functions for a table of size $m = n^2$. Then, if we use a random $h \in \mathcal{H}$ to hash n keys into the table, the expected number of collisions is at most $1/2$.

Proof. By the definition of universality, the probability that 2 given keys in the table collide under h is $1/m = 1/n^2$. Since there are $\binom{n}{2}$ pairs of keys that can possibly collide, the expected number of collisions is

$$\binom{n}{2} \cdot \frac{1}{n^2} = \frac{n(n-1)}{2} \cdot \frac{1}{n^2} < \frac{1}{2} . \quad \square$$



No collisions at level 2

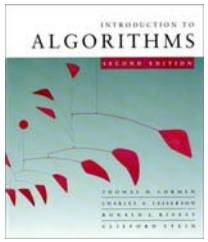
Corollary. The probability of no collisions is at least 1/2.

Proof. ***Markov's inequality*** says that for any nonnegative random variable X, we have

$$\Pr\{X \geq t\} \leq E[X]/t.$$

Applying this inequality with $t = 1$, we find that the probability of 1 or more collisions is at most 1/2. □

Thus, just by testing random hash functions in $\mathcal{H}$, we'll quickly find one that works.



Analysis of storage

For the level-1 hash table T, choose $m = n$, and let n_i be random variable for the number of keys that hash to slot i in T. By using n_i^2 slots for the level-2 hash table S_i, the expected total storage required for the two-level scheme is therefore

$$E\left[\sum_{i=0}^{m-1}\Theta\left(n_i^2\right)\right] = \Theta(n),$$

See the textbook for the proof.

Perfect Hashing Summary

- Use two levels of hashing
  - First level $m = n$
  - Second level $m_i = n_i^2$
- Search time $\Theta(1)$
- Storage $O(n)$

Recap

- Open Addressing
- Universal Hashing
- Perfect Hashing
- Next:
  - Dictionaries using Binary Search Trees