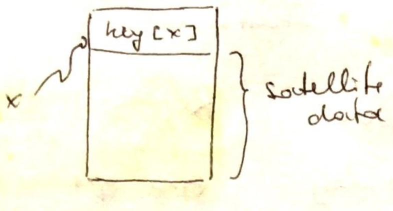
Lecture 7
6.046

Symbol-table problem (compilers)

static set is only lookup, etc...

Table S holding records



record

Operations:

- insert(S,x): $S \leftarrow S \cup \{x\}$
- delete(S,x): $S \leftarrow S - \{x\}$
- search(S,k)

dynamic set

return x [redacted] s.t. key[x] = k or nil if no such x

Direct access table

Suppose keys are drawn from $u = \{0, 1, \ldots, m-1\}$

Assume keys are distinct

Set up array $T[0 \ldots m-1]$ to represent dyn. set S

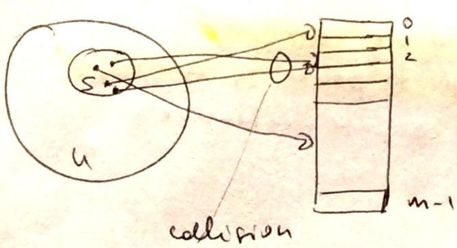
$$T[k] = \begin{cases} x & \text{if } x \in S \text{ and } key[x] = k \\ nil & \text{otherwise} \end{cases}$$

Ops take $\Theta(1)$ time

e.g. 64 bit #s drawn, need a table of size 2^{64}
↳ need hashing

Hashing

Hash function h maps keys "randomly" into slots of table T.



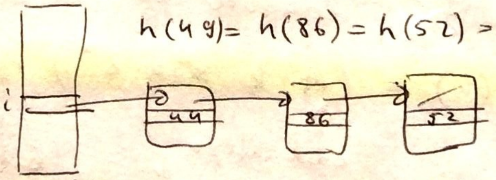
collision

when a record to be inserted maps to an already occupied slot, a collision occurs.

Resolving collisions by chaining

idea: link records in same slot into list.

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



hash table

Analysis

Worst case: every key hashes to the same slot. → long link list
access takes $\Theta(n)$ time if $|S| = n$

Average case

assumption of simple uniform hashing

- each key is equally likely to be hashed to any slot in T, independent where other keys are hashed

Def. the load factor of a hash table with n keys and m slots is $\alpha = n/m$ = [redacted] ave # keys per slot.

Expected unsuccessful search time: $\Theta(1 + \alpha)$

cost of hash & access to slot

cost of list search

Expected search time = $\Theta(1)$

if $\alpha = O(1)$, i.e. $n = O(m)$

if n grows m has to grow

Represent a dynamic set with each [redacted] time ops as long as:

- $n = O(m)$
- simple uniform hashing

Choosing a hash function:

- Should distribute keys uniformly into slots
- Regularity in key distribution should not affect uniformity

Division method

$h(k) = k \bmod m$ ← does not have to be eq. to size of table; can be around the size (len)

- don't pick m with small divisor d

Ex. $d = 2$ and all keys even

$\Rightarrow$ odd slots never used

Ex $m = 2^r \Rightarrow$ hash does not depend on all bits of k

$k = 1011\,000111\underbrace{011010}_{h(k)}$ $\quad r = 6 \quad m = 2^6$

h does not depend on other bits

Pick m = prime not too close to power of 2 or 10

However, the division method is compute-intensive.

Multiplication method

$m = 2^r$, computer has w-bit words

$h(k) = (A \cdot k \bmod 2^w)$ rsh $(w-r)$

odd integer $2^{w-1} < A < 2^w$

right shifted by bitwise

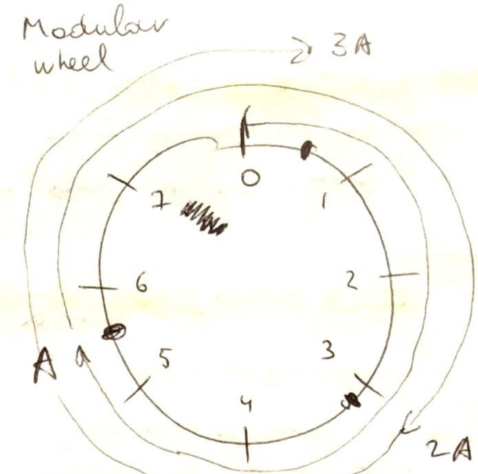
- Don't pick A too close to 2^{w-1} or 2^w
- Fast method: mult. mod 2^w faster than division; rsh is fast

Ex $m = 8 = 2^3$, $w = 7$ ← full word

$$
\begin{array}{rl}
1011001 & = A \\
1101011 & = k \\
\hline
\underbrace{1001010}_{\text{high-order bits ignored with mod } 2^w}\underbrace{0110011}_{\text{two-word product}} &
\end{array}
$$

high-order bits ignored with mod 2^w ↑ two-word product

$\underbrace{011}_{h(k)}\ \underrightarrow{0011}$ rsh removes 4 bits



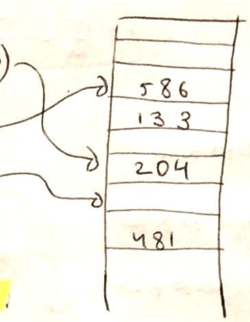
k spins the wheel...
↳ heuristic

Resolving collisions by open addressing

- No storage for links
- Probe table systematically until an empty slot is found (hash function after hash function)
- $h: \underbrace{U}_{\text{universe of keys}} \times \underbrace{\{0, 1, \ldots, m-1\}}_{\text{probe \# ← index of a hash function}} \to \{0, 1, \ldots, m-1\}$
- Probe seq. should be permutation [redacted] ← given a key, the sequence of slots hit, no repeating
- Table may fill up, $n \leq m$
- Deletion is difficult

Ex: Insert $k = 496$

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

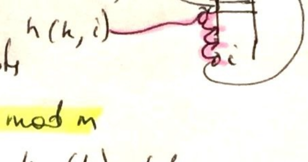


Search - same probe sequence
- successful - find record
- unsuccessful - find nil

My comment:
- need to save the ID of the hash function together with k

Probing strategies

- Linear - $h(k, i) = (h(k, 0) + i) \bmod m$
  "primary clustering" - long runs of filled slots



- Double hashing - $h(k, i) = (h_1(k) + i \cdot h_2(k)) \bmod m$
  excellent method, usually pick $m = 2^r$ and $h_2(k)$ odd

Analysis of open addressing

stronger than simple uniform hashing

Assumption of uniform hashing:

each key is equally likely to have any one of the $m!$ perms as its probe seq., indep of other keys

Theorem

$$E[\#\text{ probes}] \le \frac{1}{1-\alpha} \quad \text{if } \alpha < 1 \ (\text{i.e. } n < m)$$

Pf. (unsuccessful search):

1. probe always necessary

with prob. n/m collision $\Rightarrow$ 2. probe is necessary

2. probe prob. of collision $\frac{n-1}{m-1}$ $\Rightarrow$ 3. probe is necessary

3. probe prob. of collision $\frac{n-2}{m-2}$ ...

Note: $\frac{n-i}{m-i} < \frac{n}{m} = \alpha$ for $i = 1, 2, \ldots, n-1$, assumption $n < m$ (necessary for open addressing)

$$E[\#\text{ probes}] = 1 + \frac{n}{m}\left(1 + \left(\frac{n-1}{m-1}\right)\left(1 + \frac{n-2}{m-2}\left(\cdots\left(1 + \frac{1}{m-n}\right)\right)\cdots\right)\right)$$

$$\le 1 + \alpha(1 + \alpha(1 + \alpha(\ldots 1 + \alpha)\ldots))$$

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

$$= \sum_{i=0}^{\infty} \alpha^i = \frac{1}{1-\alpha}$$

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

$\alpha < 1$ const $\Rightarrow$ $O(1)$ probes in expectation

- Table 50% full $\Rightarrow$ 2 probes in expectation
- Table 90% full $\Rightarrow$ 10 probes in expectation

const goes up

need to keep α low