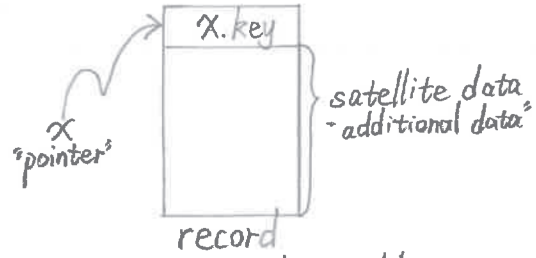
# Lec 07 哈希表

symbol-table problem (in compilers):

table S holding n records




operations on this table

- insert $(S, x)$: $S \leftarrow S \cup \{x\}$
  "insert a record into this table"
- delete $(S, x)$: $S \leftarrow S - \{x\}$

} dynamic set

- search $(S, k)$: return $x$ such that $x.key = k$ or nil if no such $x$
  "search for a given key"

direct access table

"it works when the keys are drawn from small distribution"

suppose keys are drawn from $U = \{0, 1, \cdots, m-1\}$.

assume the keys are distinct,

set up an array $T[0 \ldots m-1]$ to represent the dynamic set $S$.

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

相当于存放指针的数组

all operations take constant time in the worst case

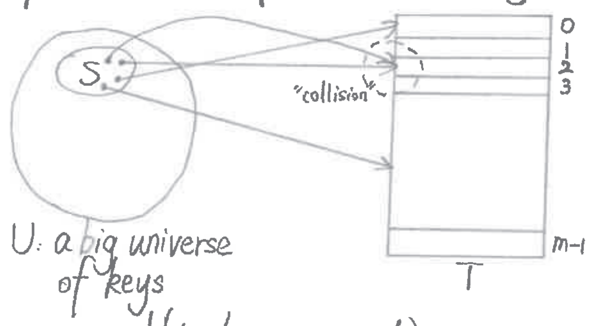
limitations: ① m should be small

② even worse, most of the table would be empty in some case

"我们希望在保存记录的同时，让表的规模尽可能的小、保留某些特性"

Hashing

a hash function h maps keys "randomly" into slots of table T
"数组的索引"



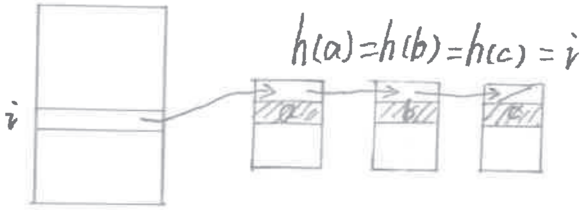

when a record (to be inserted) maps to an already occupied slot, a collision occurs

"对每个槽创建个链表，把所有映射到这个槽的元素都存放到这个槽的链表里面去"

resolving collisions by chaining!

the idea is to link records in the same slot into a list

Example:



## Analysis:

worst-case: every key hashes to the same slot (所有键都哈希映射到同一个槽)
access takes $\Theta(n)$ time if $|S| = n$

average-case: assumption of simple uniform hashing
"each key $k \in S$ is equally likely to be hashed to any slot in $T$, independent of where other keys are hashed"

Def. the load factor of a hash table with $n$ keys at $m$ slots is $\alpha = \frac{n}{m}$ = average number of keys per slot

expected unsuccessful search time $= \Theta(\underbrace{1}_{\text{hash \& access slot}} + \underbrace{\alpha}_{\text{search list}})$

expected search time $= \Theta(1)$ if $\alpha = O(1)$, i.e., if $n = O(m)$

expected successful search time $= \Theta(1+\alpha)$ too.

## Choosing a Hash Function

- should distribute keys uniformly into slots
- regularity in key distribution should not affect uniformity
"键值分布的特点"

**Example:** division method, $h(k) = k \bmod m$

don't pick $m$ (with small divisor $d$)!

if $d = 2$ (i.e. $m$ is even) and all keys are even, then odd slots never used
"regularity in key distribution"

if $m = 2^r$, then hash doesn't depend on all bits of $k$

$k = 1011000111\underbrace{011010}_{h(k)}$ $\quad r = 6$, $m = 2^6$

pick $m$ = prime (质数) not too close to a power of 2 or 10 (有很多关于质数的好定理)

**Example:** multiplication method

槽的数量 $m = 2^r$, and computer has $w$ bit words

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

an odd integer in the range $2^{w-1} < A < 2^w$

"right shifted"

fast method! (faster than division)

if $m = 8 = 2^3$ ($r=3$), $w = 7$, $A = 1011001$, $k = 1101011$

then $A \cdot k = 10010100110011$

$A \cdot k \bmod 2^w$ = (忽略前几位，只取后w位) $0110011$

$(A \cdot k \bmod 2^w) \text{ rsh } (w - r) = 011 = h(k)$

$$
\begin{array}{r}
1011001 = A \\
\times \; 1101011 = k \\
\hline
\underbrace{1001010}_{\text{high-order ignored}}\;|\;\underbrace{011}_{h(k)}\underbrace{0011}_{\text{rsh}}
\end{array}
$$

"times"

modular wheel for intuition:

$$\begin{array}{r} 1\,0\,1\,1\,0\,0\,1 = A \\ \times\ 1\,1\,0\,1\,0\,1\,1 = k \end{array} \longrightarrow \begin{array}{r} .1\,0\,1\,1\,0\,0\,1 = A' \\ \times\ 1\,1\,0\,1\,0\,1\,1 = k \end{array}$$

小数点


resolving collisions by open addressing - no storage for links

回到一开始那个问题

- the idea is that to probe the table systematically, until an empty slot is found
- $h : U \times \{0, 1, \cdots, m-1\} \longrightarrow \{0, 1, \cdots, m-1\}$

universe of keys　　probe number　　slot

- the probe sequence should be permutation of 0 to $m-1$
- the table may actually fill up in the end ($n \leq m$, $n$ is #elements, $m$ is #slots)
- deletion is difficult, yet not impossible

"有人按照探查序列来查找另一个键，他本应先发现这里不是他要的键，继而再向下查找，然而现在却发现这个槽是空的"

Example: insert $k=496$ into table as below

| |
|---|
| |
| |
| 586 |
| 133 |
| |
| 204 |
| |
| 481 |
| |

0-step: probe $h(496, 0)$
假设哈希映射到204这个槽，发现已经被占了。

1-step: 则再探查一次，$h(496, 1)$
假设哈希映射到586这个槽，发现已经被占了，

2-step: $h(496, 2)$
假设哈希映射到一个空槽，则把键放入这个槽。

search is the same probe sequence.
if successful, it finds the record.
if unsuccessful, it finds nil

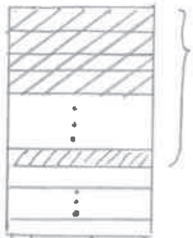
probing strategies for open addressing

- linear probing

$h(k,i) = (h(k,0) + i) \bmod m$

"一个个地查找"

"primary clustering": long runs of filled slots



如果一连块区域都被占用了，那接下来都得先遍历到这个区域的尾部

- double hashing probing

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

excellent!

usually pick $m = 2^r$ and $h_2(k)$ to be odd

Analysis of open addressing

assumption of uniform hashing: each key is equally likely to have any one of the $m!$ permutations as its probe sequence is independent of other keys

Theorem: the expected number of probes is at most $\frac{1}{1-\alpha}$ if $\alpha < 1$ 键数小于槽数 这是开放寻址法的前提条件

proof: (unsuccessful search)

1st probe always necessary,

with $\frac{n}{m}$ probability, we have a collision $\Rightarrow$ 2nd probe necessary

(you are not going to hit the same slot) with probability $\frac{n-1}{m-1}$, collision $\Rightarrow$ 3rd probe nec.

$\cdots\cdots \frac{n-2}{m-2} \cdots\cdots$

note $\frac{n-i}{m-i} < \frac{n}{m} = \alpha$ for $i = 1, 2, \cdots, n-1$

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

"每步都有连锁的影响"

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

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

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

$$= \frac{1}{1-\alpha}$$ geometric series

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

if $\alpha = 0.5$ (i.e. 50% full), then 2 probes, if 90% full, then 10 probes (急剧上升)