CMPT307: Hash Tables

Week 2-1

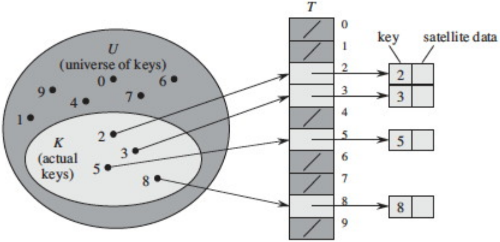
Xian Qiu

Simon Fraser University

xianq@sfu.ca

Direct Address Tables

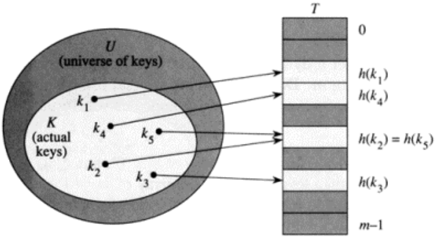
- dynamic set with key universe $U = \{0, 1, \ldots, N-1\}$
- assume unique key values



- direct address table (array) $T[0.., N-1]$
- Search, Insert and Delete take $O(1)$
- downside: wasting memory

Hash Table

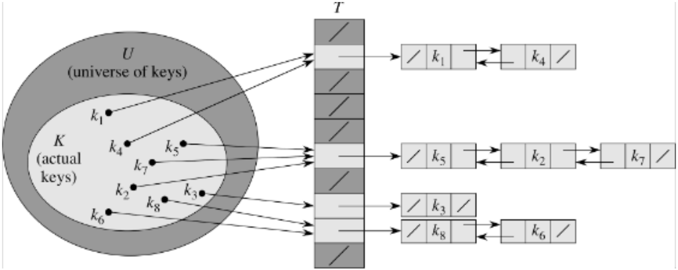
hush function: $h : U \to \{0, 1, \ldots, m-1\}$



collision: $h(k_2) = h(k_5)$

- ▷ hashing with chaining
- ▷ open addressing

Hashing with Chaining



- SEARCH
- INSERT
- DELETE

Analysis

load factor: $\alpha =$ average $\#$ elements stored in a chain

$= \frac{n}{m}$, where $n = \#$ element

simple uniform hashing: the probability of hashing any key to any of the m slots is identical

- $E[n_j] = \frac{n}{m} = \alpha$, where $n_j =$ length of $T[j]$
- assume $h(k)$ can be computed in $O(1)$

Theorem

SEARCH takes average-case time $\Theta(1 + \alpha)$, under simple uniform hashing.

- consider unsuccessful search (easy) and successful search

Analysis

let $x_i = i$th element inserted to the table and $x_i.\text{key} = k_i$

$$X_{ij} = \text{I}\{h(k_i) = h(k_j)\} = \begin{cases} 1, & \text{if } h(k_i) = h(k_j) \\ 0, & \text{else} \end{cases}$$

under uniform hashing, $\Pr\{h(k_i) = h(k_j)\} = \frac{1}{m}$

$$\mathbb{E}[X_{ij}] = \frac{1}{m} \cdot 1 + \left(1 - \frac{1}{m}\right) \cdot 0 = \frac{1}{m}$$

- if k_j is in front of k_i in linked list, then $j \geq i + 1$
- search all keys takes $T_n := \mathbb{E}\left[\sum_{i=1}^{n}(1 + \sum_{j=i+1}^{n} X_{ij})\right]$
- one (successful) search takes $\frac{T_n}{n} = 1 + \frac{\alpha}{2} - \frac{\alpha}{2n}$

Hash Functions

what makes a good hash function?

- satisfies (approximately) simple uniform hashing
- computable in constant time

Example

If k is uniformly distributed in $[0, 1)$, then $h(k) = \lfloor km \rfloor$ is good.

- division method: $h(k) = k \mod m$, where m is prime
- avoid "patterns": want $h(k)$ to depend on all bits of k
- multiplication method: $h(k) = \lfloor m(kA \mod 1) \rfloor$, where $0 < A < 1$ and m is not critical
- universal hashing: randomly choose h from its family $\mathcal{H}$
- character keys can be transformed into integer keys

Interpreting Keys as Natural Numbers

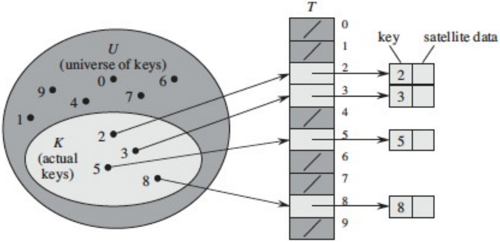
decimal: $1234 = 1 \times 10^3 + 2 \times 10^2 + 3 \times 10 + 4$

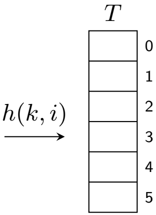
- characters: `pt`
- `p`=112 and `t`=116 in ASCII character set
- transform (112,116) into radix-128 integer

$$\texttt{pt} = 112 \times 128 + 116 = 14452$$

Open Addressing

recall direct addressing





- ▷ open addressing: all elements occupy hash table itself $(\alpha \leq 1)$
- ▷ no linked lists
- ▷ may get overflow of memories

Open Addressing

hash function: $h : U \times \{0, 1, \ldots, m-1\} \to \{0, 1, \ldots, m-1\}$ s.t. the probe sequence

$$\pi(k) := \langle h(k, 0),\ h(k, 1),\ \ldots,\ h(k, m-1)\rangle$$

is a permutation of $\langle 0, 1, \ldots, m-1\rangle$

- $\#$ permutations $= m!$
- uniform hashing: $\forall k$, $\pi(k)$ has the same chance to be any of the $m!$ permutations

HASH-INSERT(T, k)

```
for i = 0 to m − 1 do
    j = h(k, i);
    if T[j] == nil then
        T[j] = k;
        return j;
error "hash table overflow";
```

Search

HASH-SEARCH(T, k)

```
for i = 0 to m − 1 do
    j = h(k, i);
    if T[j] == k then
        return j;
    if T[j] == nil then
        return nil;
return nil;
```

Deletion

to delete a key stored in $T[i]$, can we simply set $T[i] =$nil?

$h(k_1, 0) = h(k_2, 0)$

k_1	k_2		

insert k_1, k_2

	k_2		

delete k_1 searching k_2 returns nil

solutions

- search the whole table in $O(m)$ bad
- use special value "deleted" and modify insertion properly
- with searching time $O(n)$ also bad