

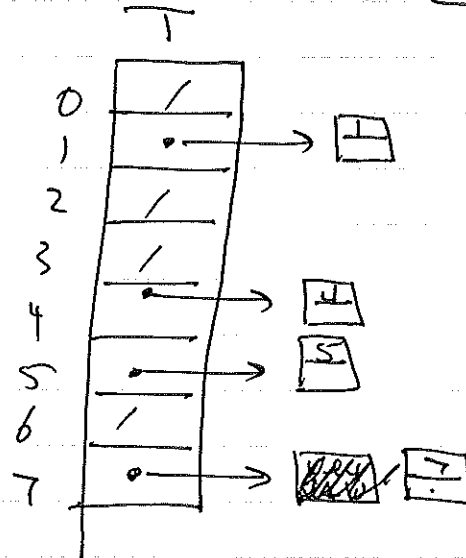
HASH TABLES (CHAP 13, SEC 2)

$U = \{ \text{all possible keys} \}$

$S = \{ \text{keys actually in use} \}$

$n = \text{length}[T]$

EX $n=7$ DIRECT ADDRESS TABLE



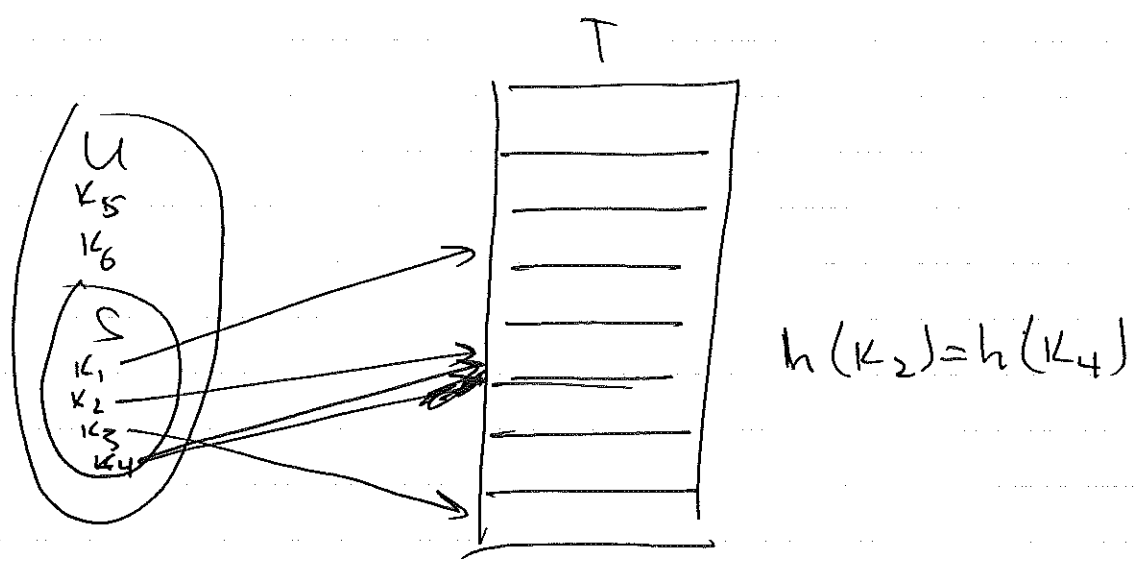
EX HASH TABLE: store (k, v) in $T[h(k)]$
where

$$h: U \rightarrow \{0, \dots, n-1\}$$

is a hash function.

NOTE $n < |U|$ so h is NOT ONE TO ONE. Thus there are collisions

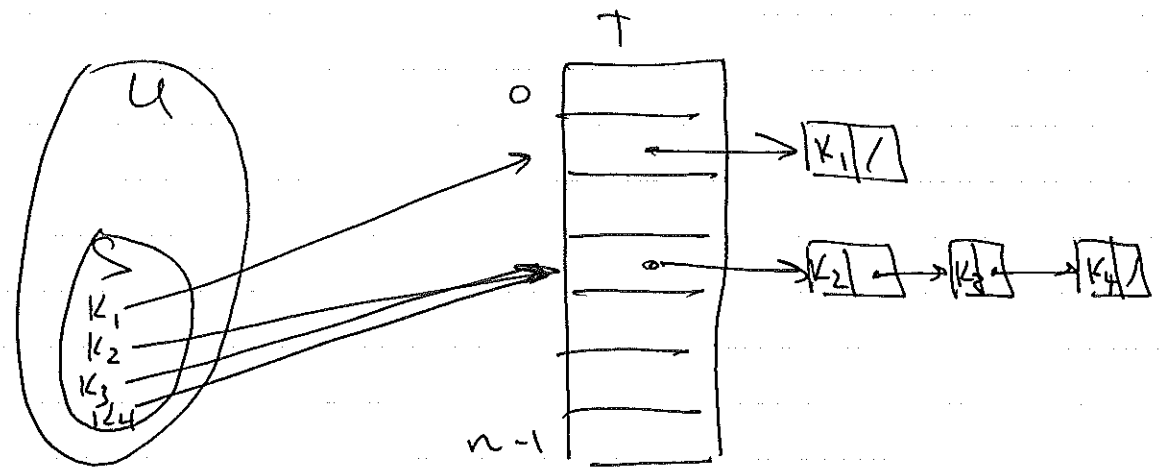
$$h(k_1) = h(k_2)$$



h should be chosen so as to minimize the probability of collisions, i.e. h 'distributes' the pairs of collisions evenly across all indices $\{0, \dots, n-1\}$.

h is designed to look 'random' on some sense, hence name 'hash'.

CHAINING: Multiple keys into linked list.



Insert: Place new node at head of list $T[h(k)]$.

Delete: Delete node from $T[h(k)]$.

lookup: Do linear search of list $T[h(k)]$.

DEFN: THE LOAD FACTOR OF A HASH TABLE IS

$$\alpha = \frac{\# \text{Keys in use}}{\text{Table Size}} = \frac{|S|}{n}$$

(i.e. α is the average # of keys stored in any list, for chaining.)

THE EFFICIENCY OF A COLLISION RES. METHOD IS ANALYZED IN TERMS OF α .

in collision res. by chaining we have Full

① UNSUCCESSFUL SEARCH:

$$\# \text{Comparisons} = \alpha$$

② SUCCESSFUL SEARCH

$$\# \text{Comparisons} = 1 + \frac{\alpha}{2}$$

↑
check list NOT
EMPTY

↑ ONE AVG. TARGET IS
HALFWAY DOWN LIST.

SINGLE VALUE HASHING (SUH)

A Random key is equally likely to hash into ANY SLOT: $\text{Pr}(h(k)=i) = \frac{1}{n}$ for $0 \leq i \leq n-1$.

137

IN THE LANGUAGE OF PROB. THY U IS THE SAMPLE SPACE AND

$$h: U \rightarrow \{0, \dots, n-1\}$$

IS A DISCRETE RANDOM VARIABLE WHICH IS UNIFORMLY DISTRIBUTED, I.E. THE PROB. DENSITY FUNCTION FOR h IS THE CONSTANT $\frac{1}{n}$.

NOTE WE DO NOT ASSUME THAT ANY TWO KEYS ARE EQUALLY LIKELY TO BE IN USE, SOME KEYS MAY BE MORE LIKELY THAN OTHERS. THUS TO CONFORM TO SUH, h MUST BE CHOSEN WITH THE DISTRIBUTION OF KEYS IN MIND.

EX IF KEYS ARE UNIF. DIST. OVER $U = \mathbb{Z}$, THEN $h(k) = k \bmod n$ SATISFIES SUH.

EX IF KEYS ARE UNIF. DIST. OVER $0 \leq k < 1$ I.E. $U = [0, 1)$, THEN $h(k) = \lfloor kn \rfloor$ SATISFIES SUH.

$$\text{OBSERVE } h: \left[\frac{i}{n}, \frac{i+1}{n} \right) \rightarrow \{i\} \quad (0 \leq i \leq n-1)$$

IF THE PROB. DIST. OF KEYS IS NOT KNOWN WE MUST TAKE A HEURISTIC APPROACH TO CHOOSING h .

ASSUME FROM NOW ON: $U = \mathbb{N} = \{0, 1, 2, \dots\}$
GO TO P. 96.

Ex Insert Keys

17 33 29 13 12, 44
 45 57 9 3 81, 90
 19 97 35

into hash table of length = 11

• Resolve coll by chaining $h(k) = k \bmod 11$

• determine α , # collisions

• Resolve coll by open addressing.

$$h(k, i) = (k \bmod 11 + 3i) \bmod 11$$

P. 100 in C

ASSUME FOR SAKE OF SIMPLICITY
THAT THE BINARY SEARCH KEYS WITH
NO SATELLITE DATA (i.e. NO 'value').

TO INSERT KEY K INTO T DO

```
fail = 1;
i = 0;
while (i < n AND fail) {
    j = h(k, i);
    if (T[j] = "EMPTY") {
        T[j] = k;
        fail = 0;
    } else {
        i++;
    }
}
if (fail) return error;
else return j;
```

sky #define ... -1

TO LOOKUP ~~SEARCH~~ FOR KEY K IN T DO:

```
i = 0;
do {
    j = h(k, i);
    if (T[j] = k) return j;
    i++;
} while (i < n);
```

101 in C

```
found = 0;
```

```
i = 0;
```

```
do {
```

```
    j = h(k, i);
```

```
    if (T[j] = k);
```

```
        found = 1;
```

```
    else
```

```
        i++;
```

```
while (i < n && !found);
```

```
if (!found) return Empty;
```

```
else return j;
```