

IADS GUEST LECTURE: HASHINGTODAY:

- DICTIONARIES & APPS.
- DIRECT ADDRESS TABLES
- HASHING WITH CHAINING
- DYNAMIC TABLES (AMORT. ANAL.)
- CUCKOO HASHING
- OTHER APPS OF HASHING

DICTIONARIES

- STORE A SET  $K$  OF KEYS (AND ASSOCIATED INFORMATION).
- OPERATIONS: SEARCH, INSERT DELETE



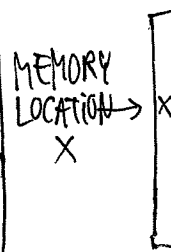
WHAT IMPL. DO YOU KNOW?

TODAY: SOMETHING FASTER!

APPLICATIONS OF DICTS

- LOOK UP FACT IN DATABASE
- COMPUTE COMMON ELEMENTS IN TWO SETS (JOIN IN DBMS)

## FIRST IDEA

DIRECT ADDRESS TABLES

ASSUMPTIONS:

- KEYS ARE INTEGERS
- ENOUGH MEMORY!

COMPARE WITH BUCKET SORTING.

## NEXT IDEA

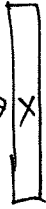
HASH FUNCTIONS

- ALL WE NEED IS A WAY TO ASSOC. EACH KEY WITH A (UNIQUE) MEMORY ADDRESS.  $x \mapsto$  ADDRESS
- WE THINK OF THIS AS A FUNCTION  $h$  APPLIED TO  $x$ , GIVING AN ADDRESS AS VALUE.
- FUNCTIONS THAT DO THIS IN A "GOOD" WAY ARE USUALLY CALLED HASH FUNCTIONS

## MODIFIED FIRST IDEA

### HASH TABLE

MEMORY LOCATION  $h(x)$  →



ASSUMPTION:  
NO COLLISIONS

IS IT REASONABLE TO ASSUME NO COLLISIONS?

### UNIFORM HASHING ASSUMPTION

- ASSUME THAT VALUES OF  $h$  ARE RANDOM AND INDEPENDENT
- UNTRUE! BUT GOOD APPROX. TO BEHAVIOR SEEN IN PRACTICE

LATER WE WILL SEE AN ALTERNATIVE THAT IS NOT BASED ON UNTRUE ASSUMPTIONS.

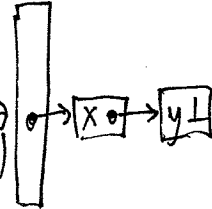
### WHAT IS A SUITABLE HASH TABLE SIZE?

- IF  $n = O(m)$ , THE EXPECTED TIME PER OPERATION IS CONSTANT!
- IF  $m = O(n)$  THE SPACE USAGE IS LINEAR!
- CONCLUSION: WANT TO KEEP  $m = \Theta(n)$ , ALSO  $k$  GROWS/SHRINKS

## MODIFIED HASH TABLE

### HASH TABLE WITH CHAINING

MEMORY LOCATION  $h(x) = h(y)$  →



IS THIS AN EFFICIENT DATA STRUCTURE? (DISCUSS WORST CASE VS AVERAGE CASE).

### ANALYSIS OF HASH TABLE W. CHAINING

- NOTATION:  $n = |K|$ ,  $m =$  SIZE OF HASH TABLE
- TIME TO SEARCH FOR  $x$  IS AT MOST THE LENGTH OF THE CHAIN AT MEMORY LOCATION  $h(x)$ .
- SINCE  $h(x)$  IS RANDOM, THE EXPECTED LENGTH IS THE AVERAGE CHAIN LENGTH
- AVG. CHAIN LENGTH  $\frac{n}{m} = \frac{1}{\alpha}$  WHERE  $\alpha = \frac{m}{n}$  IS THE LOAD FACTOR

WHERE IS INDEPENDENCE USED?

← TAKE A MOMENT TO DIGEST THIS...!

## PROBLEM SESSION

TO KEEP  $m = \Theta(n)$  WE NEED TO GROW AND SHRINK THE HASH TABLE ACCORDING TO THE SIZE OF  $K$ . HOW DO WE DO THIS EFFICIENTLY?

## WHAT MAKES A GOOD HASH FUNCTION?

ASSUME KEY IS INTEGER:

- MULTIPLY BY SOME (LARGE) NUMBER
- MAP THE RESULTING NUMBER TO  $\{0, \dots, m-1\}$  BY A MODULO OPERATION

POPULAR HEURISTIC: "MULTIPLICATION METHOD"

EXERCISE NEXT WEEK:  
WORST CASE STILL VERY BAD.

MENTION STOC 63 RESULT

INTUITIVELY, THE RANDOM CHOICE MEANS THAT WE NEED TO BE UNLUCKY TO SEE BAD BEHAVIOR.

## HALVING/DOUBLING - AMORTIZED ANALYSIS



PUT 1 CREDIT IN EACH CELL THAT IS UPDATED

- HALVE WHEN  $\frac{n}{2}$  CREDITS DUE TO DELETIONS  $\rightarrow$  SAVED CREDITS PAY FOR  $O(n)$  TIME HALVING ( $n = \Theta(m)$ )
- DOUBLING: SIMILAR

MORE GENERAL THING IN CORMEN 17.2: DYNAMIC ARRAYS. PART OF JAVA API!

IDEA: MAKE EVEN INTEGERS THAT ARE CLOSE TO EACH OTHER VERY DIFFERENT.

PROBABLY

## WHAT MAKES A GOOD HASH FUNCTION?

- AREA OF ACTIVE RESEARCH.
- FIRST SUCH RESULT (CARTER & WEGMAN '79) CALLED "UNIVERSAL HASHING":

$$h(x) = ((ax + b) \bmod p) \bmod m$$

WHERE  $a, b$  ARE CHOSEN AT RANDOM FROM  $\{1, \dots, p-1\}$  RESP  $\{0, 1, \dots, p-1\}$  AND  $p$  IS PRIME

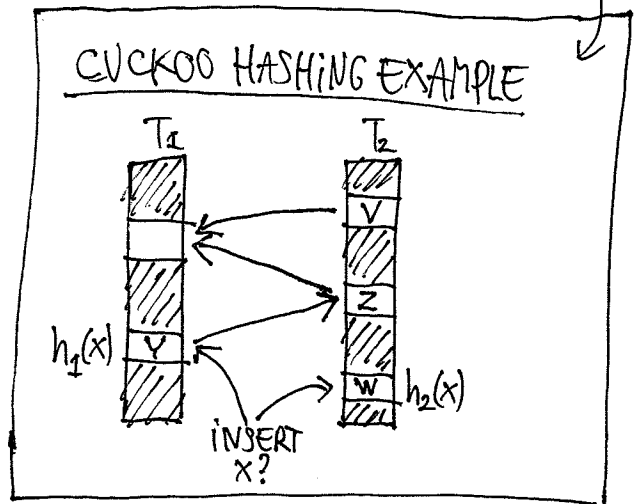
← RELATED TO "PERFECT HASHING" IN CORMEN - BUT SIMPLER!

## CUCKOO HASHING (PAGH AND ROPPER, 2001)

- IF ONE HASH FUNCTION IS GOOD, TWO ARE PROBABLY BETTER!
- PLACE  $x$  IN EITHER CELL  $h_1(x)$  OR  $h_2(x)$ .
  - POSSIBLE WITH HIGH PROB.
  - NO COLLISION RESOLUTION NEEDED! (ASSUMING SPACE  $3n$ )
  - FAST LOOKUP! FAST DELETE!

WE LOOK AT VARIANT WHERE  $h_1$  AND  $h_2$  GIVE ADDRESSES IN DIFFERENT TABLES.

ARROWS INDICATE ALTERNATIVE POSSIBLE POSITIONS



HOW DO WE INSERT HERE?  
IN GENERAL?

## CUCKOO HASHING PSEUDOCODE

```

INSERT(x):
  LOOP
     $x \leftrightarrow T_1[h_1(x)]$ 
    if  $x = \perp$  then return;
     $x \leftrightarrow T_2[h_2(x)]$ 
    if  $x = \perp$  then return;
  END LOOP
  
```

DISCUSS: DOES THIS ALWAYS WORK?  
(COME UP WITH EXAMPLE WHERE REHASH IS NECESSARY).

## CUCKOO HASHING: THEORY

- SPACE  $(2+e)n$ .
- LOOKUP, DELETE IN  $O(1)$  WORST CASE TIME
- INSERT IN  $O(1)$  EXPECTED AMORTIZED TIME
- ALGORITHM HAS BEEN GENERALIZED IN SEVERAL WAYS.

## OTHER APPS OF HASHING

- LOAD BALANCING.
- SIGNATURES (DIGESTS) OF DATA
- CRYPTOGRAPHY
- ...

DETAILS OUTSIDE SCOPE OF COURSE,  
BUT FEEL FREE TO LOOK AT THE PAPER.