Admin: $\emptyset$

6.006
Rivest
L5.1
9/18/08

Outline: **Hashing I**

- ☐ Intro
- ☐ motivation: docdist, DNA
- ☐ direct addressing
- ☐ hash functions - general idea
- ☐ chaining for collision resolution; analysis
- ☐ specific hash functions

Reading: CLRS Chaps 11.1-11.3

Intro: How are python "dictionaries" implemented?

Recall:
```
d = {}
d["ab"] = 5
d[7] = "the"
"ab" in d → True
d.items() → [("ab",5),(7,"the")]
del d[7]
```

operations: create empty dict
insert (key, value)
delete (key)
search (key)
→ enumerate

in time $\Theta(1)$ per operation (except enumerate)   **How is it done??**

(BST would have time $\Theta(\lg n)$ ...)
(BST effectively maintains a sorted list in a clever way; we can't afford sorting here...)

6.006
Rivest
L5.2
9/18/08

Motivation: Document Distance

```
def count_frequency(word_list):
    D = {}
    for word in word_list:
        if word in D: D[word] += 1
        else: D[word] = 1
```

[new "docdist7" uses dictionaries instead of sorting:]

```
def inner_product(D1, D2):
    sum = 0
    for word in D1:
        if word in D2:
            sum += D1[word]*D2[word]
```

⇒ optimal $\Theta(n)$ docdist program
assuming basic operations are $\Theta(1)$ time

Motivation: PS2

How close is chimp & human DNA?

⇒ What is longest common substring of two strings?

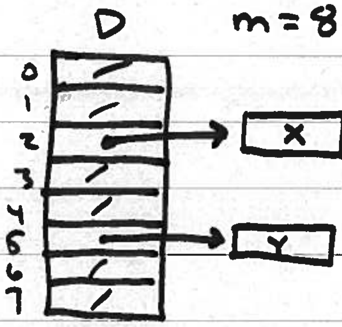
algo**rithm** vs a**rithm**etic   length 5

dictionaries can help speed this up
(put all substrings of length k into dict, look for duplicates from k-length substrings of other string search over k)

6.006
Rivest
L5.3
9/18/08

Ideal: array accessing
- Assume keys are in range $0..m-1$
- Use key as index into table

$D[2] = X$
$D[5] = Y$



array access takes $O(1)$ time: insert, delete, search
enumerate takes time $\Theta(m)$
(proportional to search of table, not # n of keys, but we'll try to keep $m \approx n$ later...)

What if keys are not small integers?

Suppose they are from some large set U?
e.g. U = set of length-20 strings over alphabet A,T,G,C
U = credit card #'s
U = english words

Soln: (idea) define a "random-looking" function
$$h: U \rightarrow \{0, 1, \ldots, m-1\}$$
store x in $T[h(x)]$

(e.g. in Python: hash(x) gives 32-bit integer, h(x)= hash(x)% m can be used.)

Note: x can't be "mutable", else its location would have to change. hash((2,3)) ok, hash([2,3]) not

Two issues: 6.006

Rivest

L5.4

9/18/08

① how to compute $h(x)$?

② what if we have a "collision" ?

$x \neq y$ but $h(x) = h(y)$

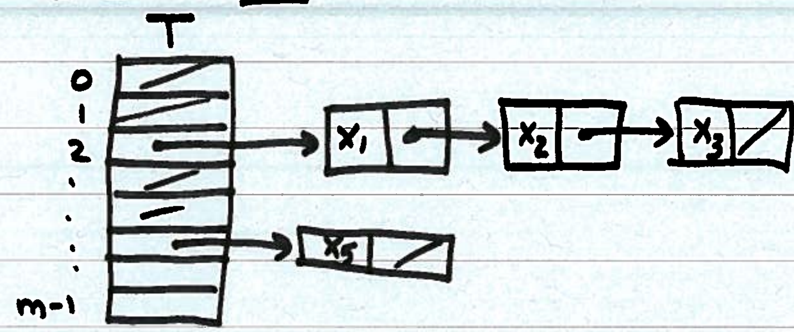
do ② first:

chaining: today

open addressing: next week

Chaining:

- table $T[0..m-1]$ as before
- $T[i]$ is list of elements that have $h(x) = i$



$h(x_1) = h(x_2)$
$= h(x_3)$
$= 2$

$T[i]$ could be linked list (as shown), or python list (array); python can take care of growing it as needed.

- Analysis: (hash n items into table of size m)
  - worst-case is bad: all hash to same position i
    list $T[i]$ has length n
    operation take time $\Theta(n)$:(
  - So, we'll look at expected time, not W.C. time, based on an assumption.

6.006
Rivest
L5.5
9/18/08

Assumption: ("Simple Uniform Hashing")
$\Rightarrow$ each key is equally likely to hash to any slot of table, independent of where other keys are hashed to.

Define: load factor $\alpha = n/m$
$= $ average # keys / slot

usually small constant ~~≈~~ 0.1 … 10

Expected performance given SUH:

- search / insert / delete
  time $\Theta(1 + \alpha)$
  - α: to search list $T[i]$
  - 1: to compute $i = h(x)$

*** $= \Theta(1)$ if $\alpha = \Theta(1)$ i.e. if $m = \Omega(n)$

[Note: analysis for successful search interesting… see CLRS]

- enumerate:

** time $\Theta(m+n)$ (search through T, & each list)
$= \Theta(n)$ if $m = \Theta(n)$

- We'd clearly like $m = \Omega(n)$ and $m = O(n) \Rightarrow m = \Theta(n)$
  e.g. $n/4 \leq m \leq 4n$ would be nice.
- discuss table resizing next time, show can keep table nicely sized at reasonable cost (without changing our basic results above)

6.006
Rivest
L5.6
9/18/08

How to compute $h(x)$?

Lots of ways: here's one that's good
assume x is an integer
let m be hash table size
let p be prime, $p \geq m$ (ok if $p=m$ if prime)
pick a: $0<a<p$
pick b: $0 \leq b<p$
let

$$h(x) = ((ax+b) \bmod p) \underbrace{\bmod m}_{\text{not needed if } p=m}$$

example:

$m = 1{,}000{,}000$
$p = 1{,}000{,}003$
$a = 314159$
$b = 271828$

If $x =$ "ATTGCATA" treat as base-4 integer
If $x =$ "weather" treat as base-26 integer
Note: can compute $x \bmod p$ as first step: $h(x) = h(x \bmod p)$
Note: if p reasonably large, can use same a, b, p
with tables of different size m

See text for other methods (division method, multiplication method).