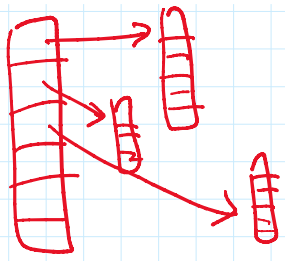


2-level structure

query $O(1)$ worst case

$$\text{space} = O\left(m + \sum_{i=0}^{m-1} |A[i]|^2\right)$$

$$= O(m + \#\text{ colliding pairs})$$

$$\text{expected space} = O\left(m + n^2 \cdot \frac{1}{m}\right)$$

choose $m = n$ $\quad = O(n)$ ← worst case by repeating until success

expected preproc time $O(n)$

Two issues:

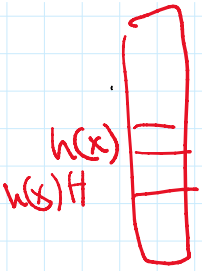
1. dynamic?

   Dietzfelbinger et al. '94: $O(1)$ worst-case query ←
   $O(1)$ expected update

2. Simpler? e.g. plain array

old idea - linear probing / "open addressing"

plain array $A[0 \ldots m-1]$ (no linked list)




```
insert(x):
  for i = 0, 1, ...
    if A[h(x)+i] == nil
      A[h(x)+i] ← x, return
```

or more generally, $h_i(x)$ for sequence of hash fns $h_0, h_1, h_2, \ldots$

```
query(y):
  check A[h(y)]
    then A[h(y)+1]
    then A[h(y)+2]
    ⋮
    etc.
```

not worst-case

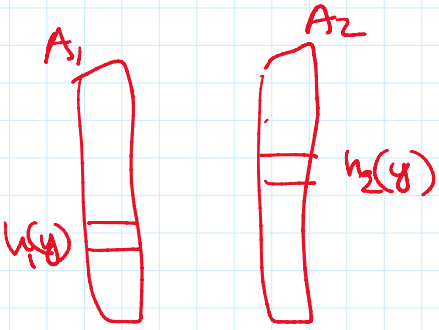
could get $O(1)$ expected query time, $M = cn$ for $c > 1$.

assuming 5-universality (Pagh, Pagh, Ruzic '09)

newer method - cuckoo hashing (Pagh, Rodler '04)

use 2 plain arrays

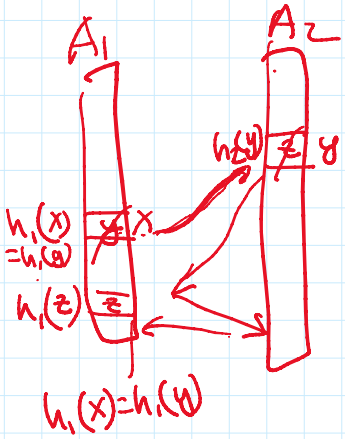
just use 2 hash fns h_1, h_2



$O(1)$ worst-case!

```
query(y):
  check A1[h1(y)] or A2[h2(y)]
```

```
insert(x):
  repeat {
    if A1[h1(x)] == nil
      A1[h1(x)] ← x, return
    x ↔ A1[h1(x)]
```



if $A_2[h_2(x)] = \text{nil}$

$A_2[h_2(x)] \leftarrow x$, return

$X \leftrightarrow A_2[h_2(x)]$

}

if stuck in cycle, restart

Claim $O(1)$ worst-case query

$O(1)$ expected amort. update

for $m \geq cn$ for const $c > 1$.

& h_1, h_2 are completely rand.

Pf Sketch:

Define graph $G = (V, E)$ where

$V = [m]$

$E = \{ (h_1(y), h_2(y)) : y \in S \}$.

↖ "random" graph

insert time = length of an alternating path in G from $h_1(x)$.

For fixed x,

$\Pr[\exists$ path of length ℓ from $h_1(x)$ in $G]$

$= \Pr[\exists$ (distinct) $y_1, \dots, y_{\ell-1} \in S$ s.t.

$h_1(x) = h_1(y_1) \wedge h_2(y_1) = h_2(y_2) \wedge h_1(y_2) = h_1(y_3)$

$\wedge \dots]$

$\leq n^{\ell-1} \cdot \left(\frac{1}{m}\right)^{\ell-1} = \frac{1}{c^{\ell-1}}$ $\quad m = cn$

expected insert time (ignoring restart) $\leq O\left(\sum_{\ell=1}^{\infty} \ell \cdot \frac{1}{c^{\ell+1}}\right) = O(1)$.

restart if $\exists$ alternating cycle containing $h_1(x)$

i.e. $\exists$ alt. path from $h_1(x)$ to $h_1(x)$

Prob $\sim \sum_{\ell} \frac{1}{c^{\ell+1}} \cdot \frac{1}{n} = O\left(\frac{1}{n}\right)$.

$\Rightarrow$ (roughly) expected amort time $O(1)$. □

Still holds if h_1, h_2 are from $O(\log n)$-universal family (Cohen-Kane '09)

Problem Pred search for integers in $[U]$

What we know: $O(\log n)$ query/update, $O(n)$ space

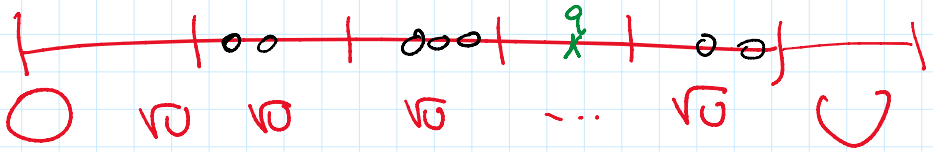
(or $O(1)$ query time, $O(U)$ space, $O(U)$ update time)

van Emde Boas '75: $O(\log \log U)$ query time & update time (expected)
$O(n)$ space

vEB Trees

idea - multi-$\sqrt{U}$-way D&C

divide $[U]$ into $\sqrt{U}$ chunks of length $\sqrt{U}$



recurse inside each chunk

let $D = \{$(indices of) all nonempty chunks$\}$

recurse in D.

⋮