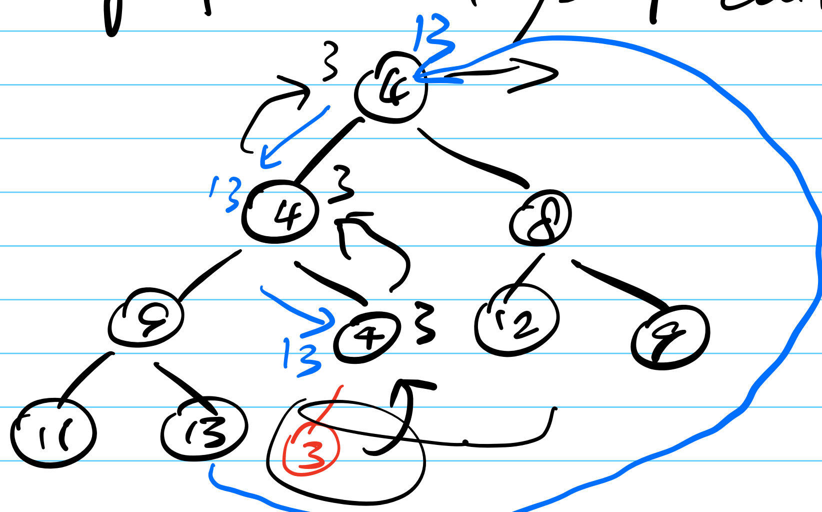
0. Warm-up.

Heap property: for every object x
key of x ≤ keys of children.



a. is it a valid heap? ✓

b. Insert(3) Bubble up! Time: O(log n)

c. Extract-min Bubble down
Swap w/ smaller child
Time: O(log n)

If your app.: requires fast min/max on an evolving set of obj's. Heap is usually the choice of D.S.

2

d. Given Heap w/ n obj's.
which can be solved in $O(1)$
insert & Extract-min ?

✓ a. Find obj w/ 5^{th} smallest key.

× b. obj. max key $\Theta(n)$

× c. obj. median key $\frac{n}{2} = \Theta(n)$

d. none of the above

1. Hash table.

a. app: 2-sum.

Input: unsorted array A of n integers.
& target sum t.

Goal: Determine $\exists ?$ $x, y \in A$
s.t. $x + y = t$

☹ check all pairs (x, y) $x+y \overset{?}{=} t$.
$\rightarrow \binom{n}{2} = O(n^2)$

😑 ① sort A $\Theta(n \log n)$ (Heap sort)
② for each $x \in A$
look for $(t-x) \in A$ $O(n \cdot \log n)$
binary search

$\Rightarrow \Theta(n \log n)$

① insert n elem's of A into Hash table. $O(n)$

② for each $x \in A$ look $t-x \in H$ $O(n)$

$\Rightarrow O(n)$

! A lot more apps.

b. Implementations.

- Setup: - U : universe (Big!)

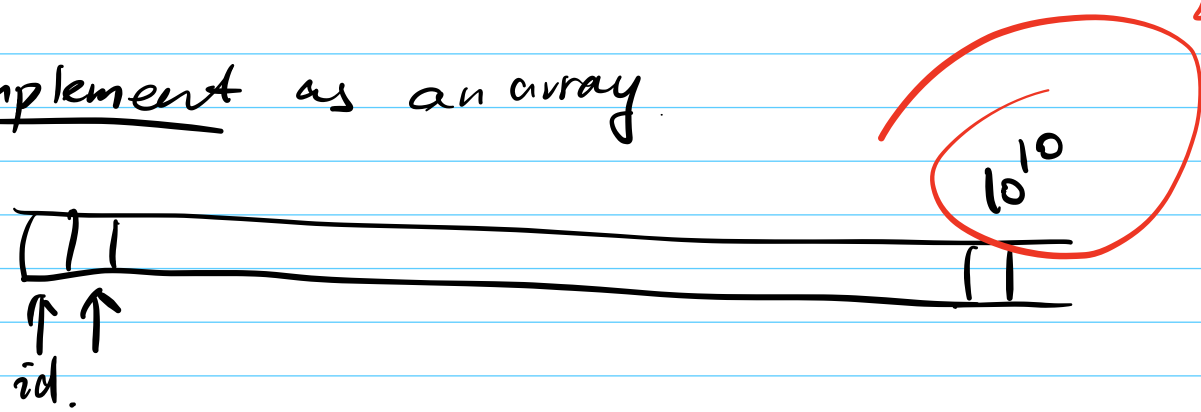
Ex: key

ID	Name	...
$\underbrace{01\cdots9}_{\text{(10 digits)}}$	Alice	

- maintain evolving set $S \subseteq U$

(e.g. CS students)

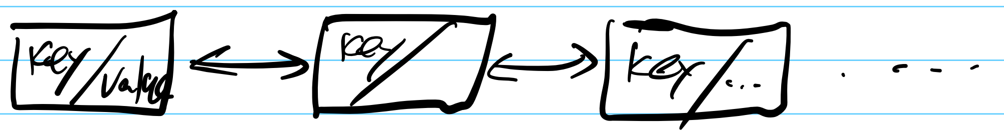
- Implement as an array



constant lookup. $\Theta(1)$
insert / del.

space-costly $\sim 10^{10}$ $\Theta(|U|)$

- Implement as linked list. S, $|S|$

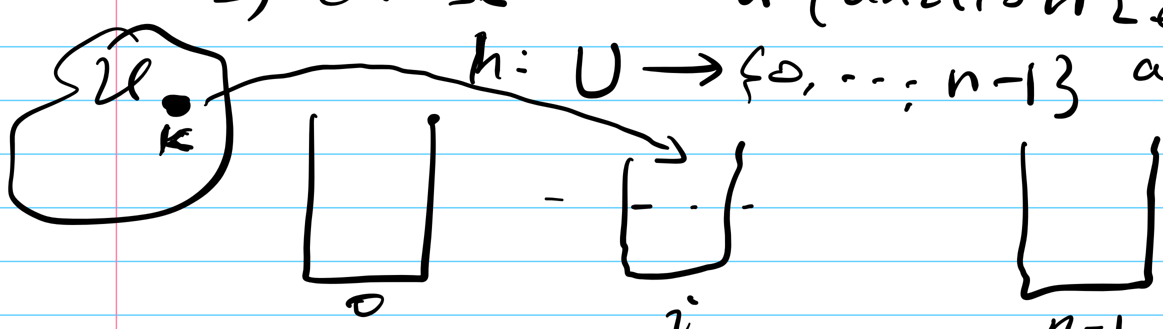


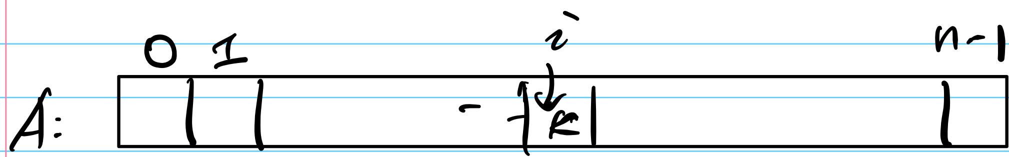
lookup: $\Theta(|S|)$
space: $\Theta(|S|)$

★ solution: (H.T: buckets + hash func.)

1) pick n = # of "buckets".
2) choose a hash function [assign every elem in U] a bucket

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





$h(k) = i$ Array of size n.

$A[h(k)] = k$.

☺ <u>look up</u>: $O(1)$

☹ <u>Space</u>: $O(n)$ $n = \Theta(|S|)$

$\neq$

$\Theta(|S|)$

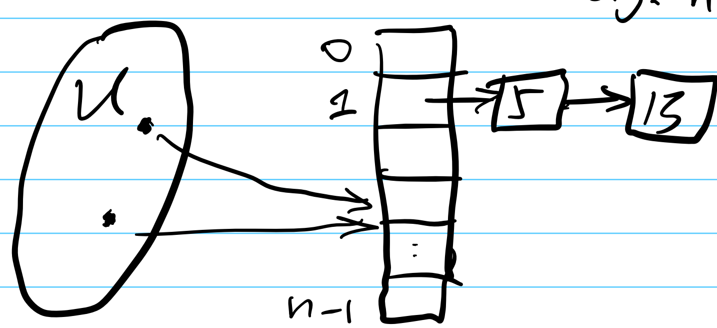
! Devils come w/ the pigeons

c. <u>collisions</u>:

distinct $x, y \in U$ s.t. $h(x) = h(y)$

<u>Two soln's</u>:
- chaining
- open addressing.

· <u>chaining</u>: e.g. $h(5) = h(13) = 1$



- keep linked list in each bucket.
- given a key k: ins/del/lookup $A[h(k)]$ in the list.

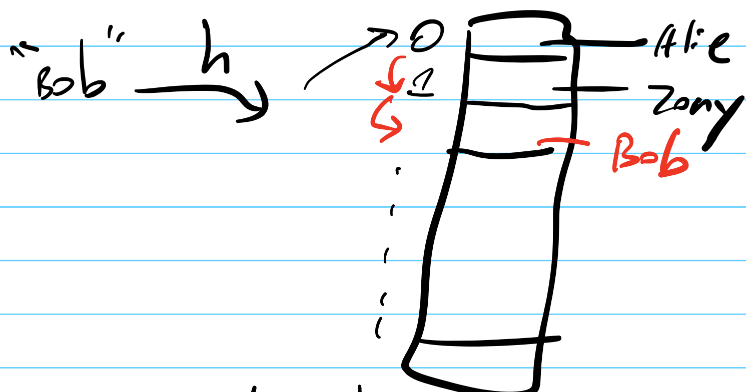
• open addressing

Idea: try multiple bucket until available

associate each k w/ a probe sequence

How to choose probe sequence?

. linear probing. $h(k) \to h(k)+1 \to h(k)+2$

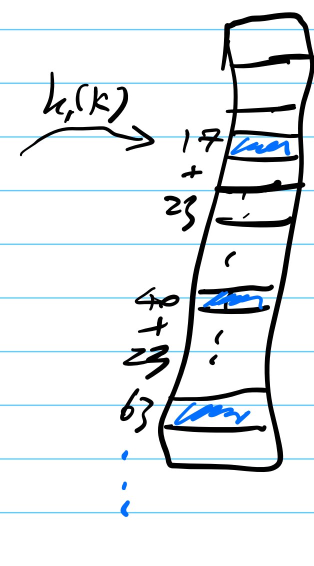


(look consequtively).

• Double hashing.

- $h_1(k)$ starting point
- $h_2(k)$ offset

$h_1(k)=17$
$h_2(k)=23$



☆ Take-away:

- Regardless of resolving strategy:
  H.T. performance downgrades w/ collisions
- Choice of hash function matters!
  e.g. $h(x) = 0 \ \forall x$ Terrible!

Ex:
- A hash table length $n \geq 1$
- Hash function: $h: x \mapsto 0 \ \forall x \in U$.
- Set S inserted in hash table $|S| \leq n$.

What's the typical running time of subsequent Lookup op's?

	chaining	& open addressing (linear)
A.	$\Theta(1)$	$\Theta(1)$
B.	$\Theta(1)$	$\Theta(\|S\|)$
C.	$\Theta(\|S\|)$	$\Theta(1)$
✓ D.	$\Theta(\|S\|)$	$\Theta(\|S\|)$.

⇓ what is a "good" hash function?

Random $h: U \to [n]$:

for $k \in U$, $h(k)$ chosen indep. and uniformly at random from $\{0, \dots n-1\}$.