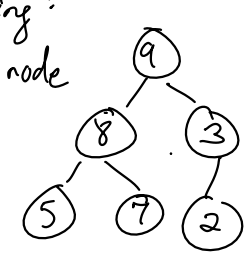
Heapify

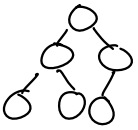
8 9 2 5 7 3

from end to beginning:

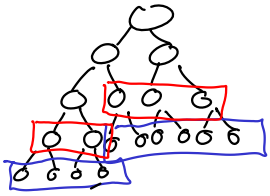
bubble down each node



Most bubble downs are small.



worst case for BD: $\log n$



heapify cost $< \frac{n}{2} \cdot 1 + \frac{n}{4} \cdot 2 + \frac{n}{8} \cdot 3 + \frac{n}{16} \cdot 4 + \dots$

$= n \cdot \left(\frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{4}{16} + \dots \right)$

$= n \cdot \left(\frac{1}{2} + \left(\frac{1}{4} + \frac{1}{4} \right) + \left(\frac{1}{8} + \frac{1}{8} + \frac{1}{8} \right) + \left(\frac{1}{16} + \frac{1}{16} + \frac{1}{16} + \frac{1}{16} \right) + \dots \right)$

$= n \cdot \left(\left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots \right) + \left(\frac{1}{4} + \frac{1}{8} + \dots \right) + \left(\frac{1}{8} + \frac{1}{16} + \dots \right) + \dots \right)$

$= n \cdot \left(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots \right)$

$= 2n$

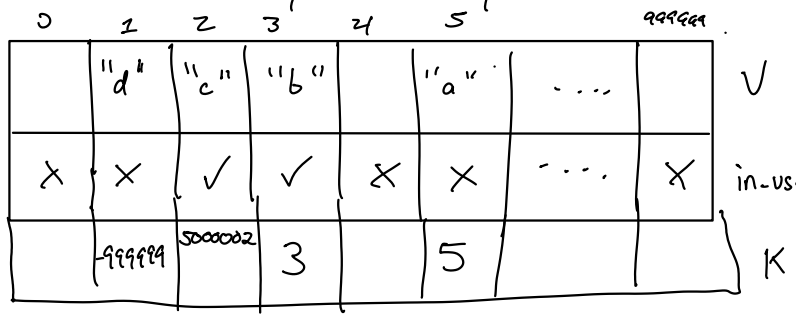
Dictionary

	Balanced BST	Hash table
get	$O(\log n)$	$O(1)^*$
insert	$O(\log n)$	$O(1)^*$
update	$O(\log n)$	$O(1)^*$
remove	$O(\log n)$	$O(1)^*$
contains		
getSize		

Guarantees

~~1. Keys are ints~~
~~2. All keys <= 1000000~~
3. ~~All keys are non-negative (>=0)~~
4. ~~Never duplicate key~~ no collisions

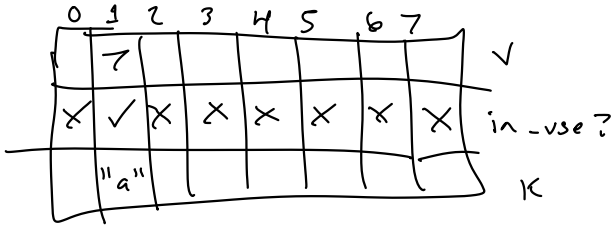
What is a O(1) dictionary? array.



insert(5, "a")
insert(3, "b")
remove(5)
insert(5000002, "c")
insert(-999999, "d")

if same slot is wanted by more than 1 mapping, this is called a "collision"

8



key --hash--> any int --modulus--> valid index

ex. hashing:

```
int hash(string s) {
    int acc = 0;
    for (int i=0; i<s.size(); i++) {
        acc += s[i];
    }
    return acc;
}
```

```
int hash(string s) {
    int acc = 1;
    for (int i=0; i<s.size(); i++) {
        acc *= 31;
        acc += s[i];
    }
    return acc;
}
```

insert("a", 7)

hash("a") ⇒ 97
⇓ mod 8
1

good hash b/c if distributes strings evenly across integers

```
int hash(string s) {
    return 0;
}
```

Collisions

- Forward chaining — Thursday
- Linear probing
  - keep moving until we find an empty spot.

0	1	2	3	4	5	6	7	8	9	
					"y"	"y"	"z"			V
x	x	x	x	x	✓	x	x	x	x	in_use?
					15	15	7			K

insert (5, "x")
insert (15, "y")
insert (7, "z")
get (25)
remove (7)
remove (5)
get (15)

	Want	average	worst case
get insert update remove	O(1)	O(1)	O(n)

invariant: array consist of blocks of values s.t. no empty spaces between K/V pair and its ideal slot

---

	0	1
V	"a"	"b"
in_use?	✓	✓
K	2	5

insert (2, "a")
insert (5, "b")
insert (3, "c")

out of space: grow hashtable just like in ArrayList

to avoid collisions, keep low load factor

$$LF = \frac{\text{size}}{\text{capacity}}$$