CSM 61B Fall 2020

Heaps and Hashing

Mentoring 9: October 19-23, 2020

1 Min-Heapify This

1.1 In general, there are 4 ways to heapify. Which 2 ways actually work?

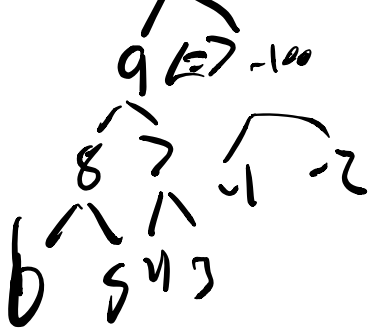
- Level order, bubbling up
- Level order, bubbling down
- Reverse level order, bubbling up
- Reverse level order, bubbling down

1.2 Are the values in an array-based min-heap sorted in ascending order?

1.3 Is an array that is sorted in descending order also a max-heap?

2 K Largest Items

2.1 The largest item in a heap must appear in position 1, and the second largest must appear in position 2 or 3. Give the list of positions in a heap where the kth largest can appear for $k \in \{1, 2, 3, 4\}$. Assume values are distinct.



3 Ls for LinkedLists

3.1 (a) In the worst case, how long does it take to index into a linked list?

$\Theta(N)$

(b) In the worst case, how long does it take to index into an array?

N = length $\Theta(1)$

(c) In the worst case, how long does it take to insert into a linked list?

$\Theta(N)$

(d) Assuming there's space, how long does it take to put a element in an array?

$\Theta(1)$

(e) What if we assume there is no more space in the array?

$\Theta(N)$

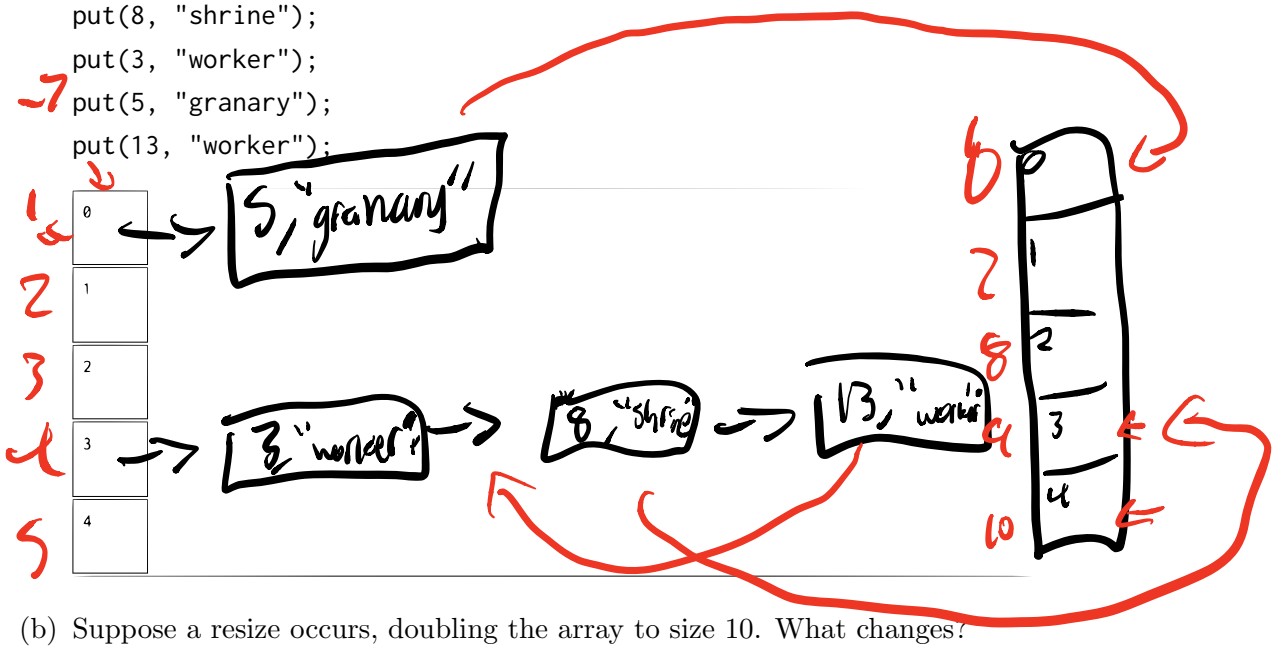
(f) Given what we know about linked lists and arrays, how could we build a data structure with efficient access and efficient insertion?

hashmap, table, set

4 Hashing Practice

4.1 (a) Draw the diagram that results from the following operations on a Java `HashMap`. `Integer::hashCode` returns the integer's value.

```
put(3, "monument");
put(8, "shrine");
put(3, "worker");
put(5, "granary");
put(13, "worker");
```



(b) Suppose a resize occurs, doubling the array to size 10. What changes?

5 Hash Codes

There is a problem with each `hashCode()` method below (correctness, distribution, efficiency). Assume there are no problems with the correctness of `equals()`.

5.1
```
class Person {
    Long id;
    String name;
    Integer age;
    public int hashCode() {
        return id.hashCode() + name.hashCode() + age.hashCode();
    }
    public boolean equals(Object o) {
        Person p = (Person) o;
        return p.id == id;
    }
}
```

5.2
```
class Phonebook {
    List<Human> humans;
    public int hashCode() {
        int h = 0;
        for (Human human : humans) {
            // Assume Human::hashcode is correct
            h = (h + human.hashCode());
        }
        return h;
    }
    public boolean equals(Object o) {
        Phonebook p = (Phonebook) o;
        return p.humans.equals(humans);
    }
}
```

5.3
```
class PokeTime {
    int startTime;
    int duration;
    public int getCurrentTime() {
        // Gets the current system clock time
    }
    public int hashCode() {
```

```
        return 1021 * (startTime + 1021 * duration + getCurrentTime());
    }
    public boolean equals(Object o) {
        PokeTime p = (PokeTime) o;
        return p.startTime == startTime && p.duration == duration;
    }
}
```

this.

incorrect, non-deterministic

P_0.equals(P_1)

P_1.startTime == P_0.startTime