

# CS61BL – Tutoring Section 9

## Hashing, Priority queues and Heaps

- Quick Review
- Quiz Review (Optional)
  - Worksheet

### Resources:

- [www.cs61bl.org/su20/resources](http://www.cs61bl.org/su20/resources)



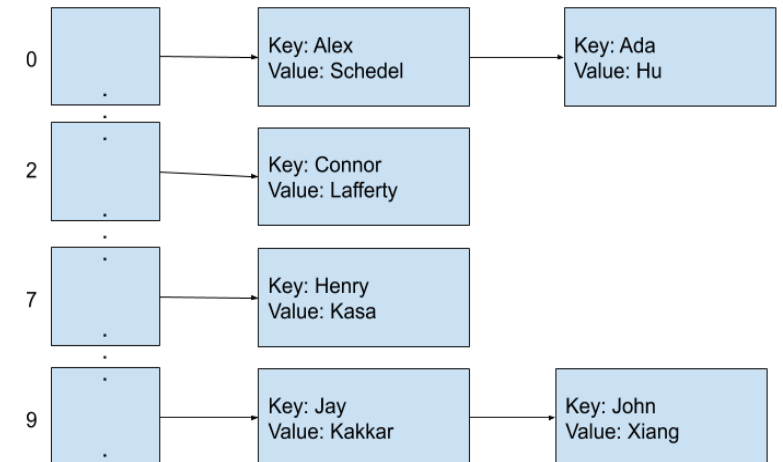


# YOU COMPLETED MT2



# Hashing

- **Objective:** Data structure that supports  $\Theta(1)$  runtime for adding and lookup.
- **Idea:** Combine the best of both worlds (Arrays + LinkedLists)
- **Hash Functions:**
  - **Valid:**
    - *Determinism:* Same items (.equals()), same code
    - *Consistency:* Every time you call hash function on same item it produces the same code
  - **Good**
    - *Uniform spreading and quick computation*
- **Memory Efficiency:**
  - Resizing when too crowded (Imagine: LinkedList)
  - *load factor* =  $array.length / size()$



# Priority queues

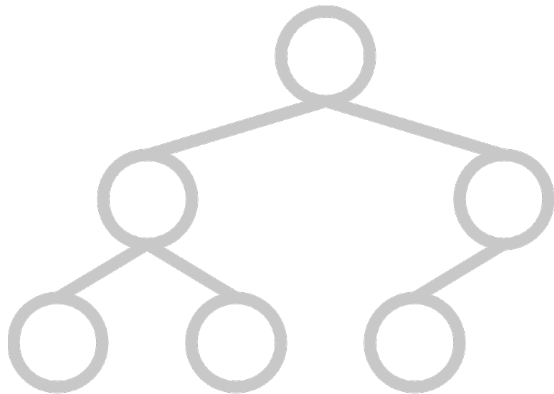
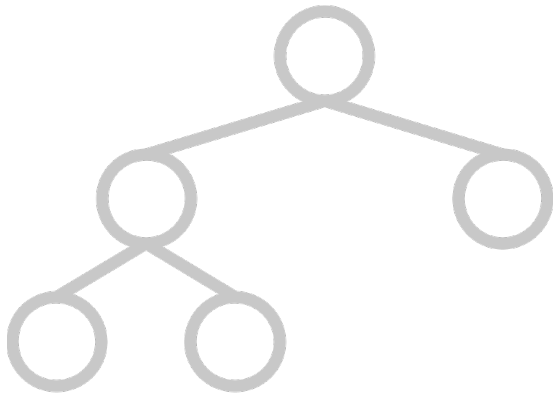
- **Objective:** Data Structure that processes based on priority
- **Variants:**
  - MaxPriorityQueue (Emergency Room)
  - MinPriorityQueue (Refrigerator)
- Each item in the PQ is in the form (Item, priority)
- **Functions:**
  - **Insert(item, priorityvalue)**
  - **Peek()** – Returns item to be popped off next
  - **Poll()** – Pops off item

# Heaps (Max/Min)

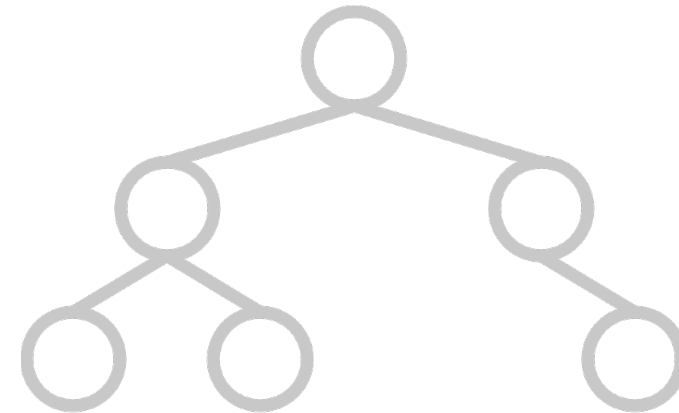
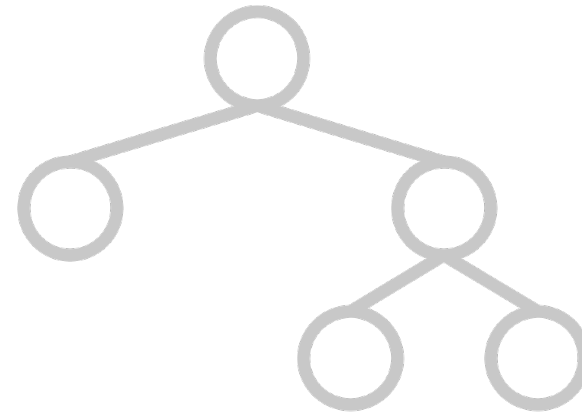
- **Objective:** Basically an implementation of a priority queue but more efficient in the form of a tree
- **NOTE: HEAPS ARE NOT BINARY SEARCH TREES**
- **Representation:** Complete Trees (i.e. completely filled, last row needs to be filled from left to right)
- **Implementation:** Array starting at  $i = 1$ . Left child =  $2N$ ; Right child =  $2N+1$
- **Insertion:** Add item to bottom; Recursively check if item is smaller/larger than parent. If so, swap all the way up to root.
- **Deletion:** Swap bottom item with root; Recursively check if item is smaller/larger than kid. If so, swap all the way up to bottom.

# Completeness

Complete



Incomplete



# Quiz Q1.1: Hashing

Which ones are valid hashing functions?

```
public class Course {
    public final int CCN;
    public final String instructor;
    public Student[] students;
    public int audited; //when the course was last audited
    public Course(int CCN, Student[] initial) {
        this.CCN = CCN;
        this.students = initial;
        this.instructor = "Matt";
    }
    //implementation
    public void audit() {
        this.audited = System.currentTimeMillis();
    }
    //implementation
    public void addStudent(Student s) {
    }
}
```

A)

```
@Override
public int hashCode() {
    return CCN; //Option A
}
```

B)

```
@Override
public int hashCode() {
    return this.students.length; //Option B
}
```

C)

```
@Override
public int hashCode() {
    return this.audited; //Option C
}
```

D)

```
@Override
public int hashCode() {
    return 5; //Option D
}
```

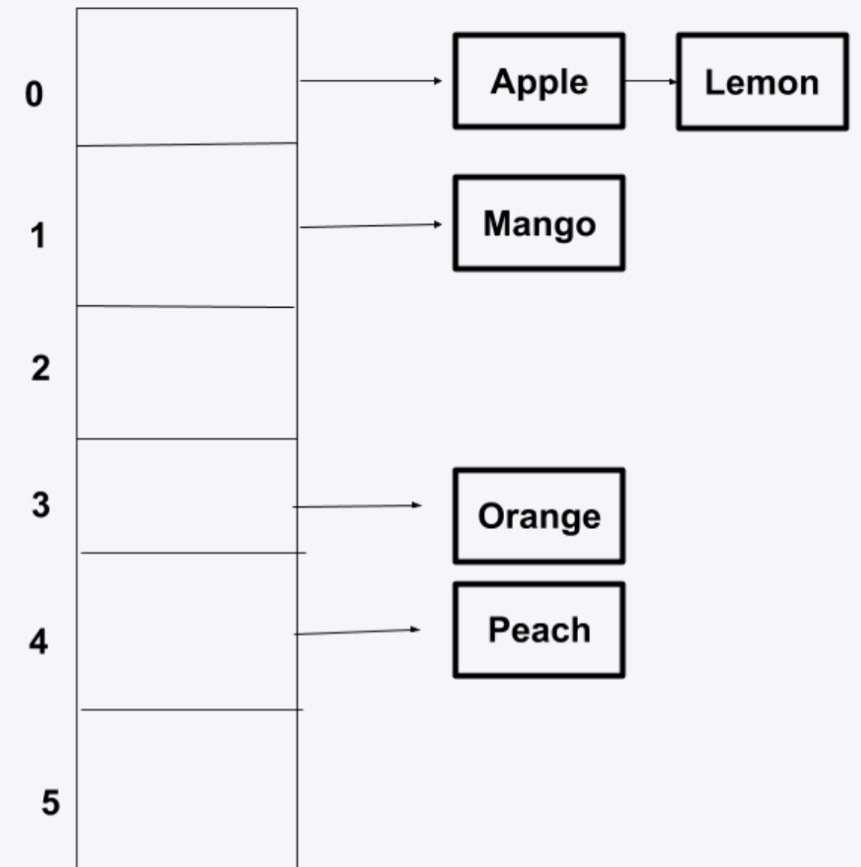
E)

```
@Override
public int hashCode() {
    return getNumericValue(this.instructor.charAt(0)); //Option E
}
```

# Quiz Q1.2: Hashing

If the load factor is 1.25, how many inserts can we make before resizing?

Monster Hashing Question is explained on video





# Quiz Q2: Heaps

**What is the left child of 4 and right child of 6?**

We have the following heap, representing a Min PQ:

[-, 1, 4, 6, 7, 10, 12, 15, 16, 22, 34, 56, 71]

Here, - represents null.

**MaxHeap: Peeking, polling and inserting. We only have access to a MinHeap. What do we do?**