CSCI 232:
Data Structures and Algorithms

Midterm Review

Reese Pearsall

Spring 2024

Announcements

No lab this week

Midterm exam on Thursday (normal class time + location)

Class Registration

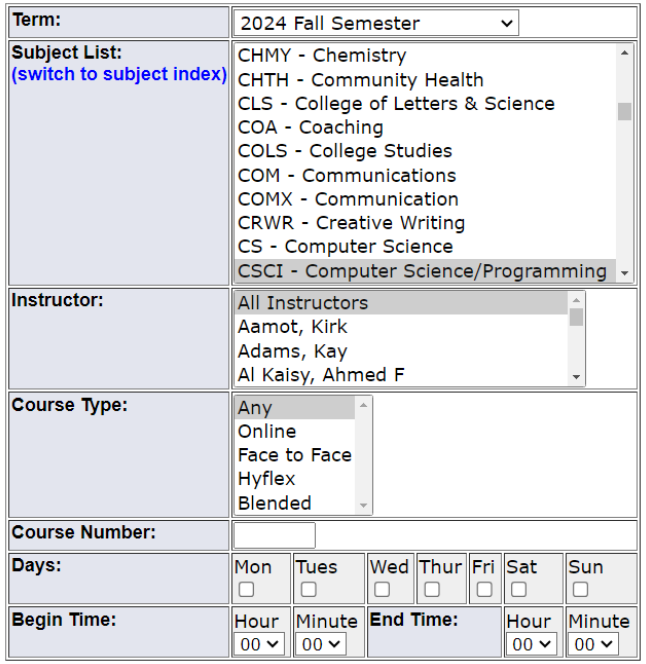
Next Classes (You can register for these anytime in the next couple years):

CSCI 366- Computer Systems
ESOF 322 – Software Engineering
CSCI 305 – Concepts of Programming Languages
CSCI 338 – Computer Science Theory

If you have not already:
CSCI 246 – Discrete Structures
CSCI 112- Programming in C
CS 145- Web Design

Other Classes that may be of interest

CSCI 291- Intro to Data Science
CSCI 331- Web Development
CSCI 351 – System Administration
CSCI 440 – Database Systems
CSCI 443 – User Interface Design
CSCI 447 – Machine Learning*
CSCI 460 – Operating Systems
CSCI 466 – Networks
CSCI 476 – Computer Security



Lab 6

Resources Available to you:

CS Tutoring Center

Smarty Cats Tutoring

Peter and Sultan’s Office Hours + email

My office hours + email

Exam Logistics

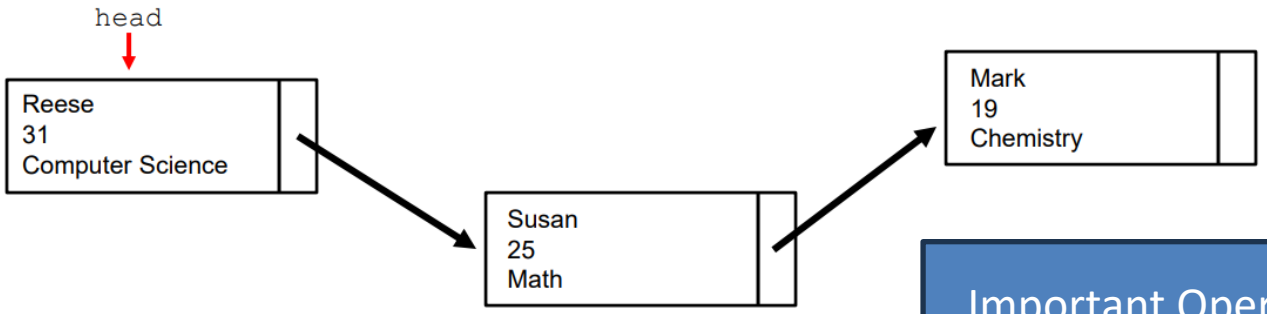
Thursday March 21st @ 10:50 in Barnard Hall 103
75 minutes

Open Notes: You can use laptop, notes, lecture slides, previous assignments, your IDE, and Java Documentation. This exam can be completed without a laptop

The midterm will consist of different types of questions, such as

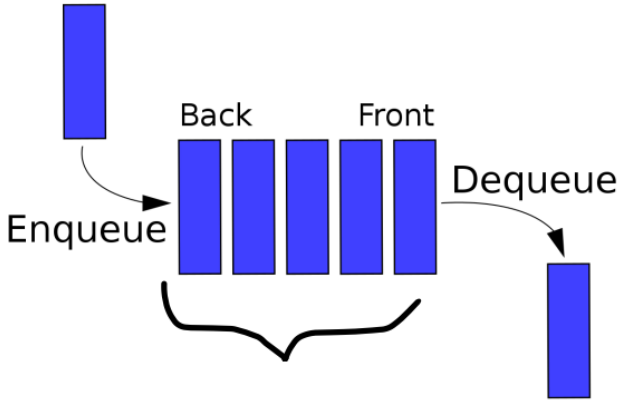
- Multiple Choice
- True/False
- Short Answer
- Describe an algorithm (no code) that does X
- Compare and Contrast different data structures
- Given some code, critique the code, or describe what it prints out
- Do some operation on a data structure

Linked Lists

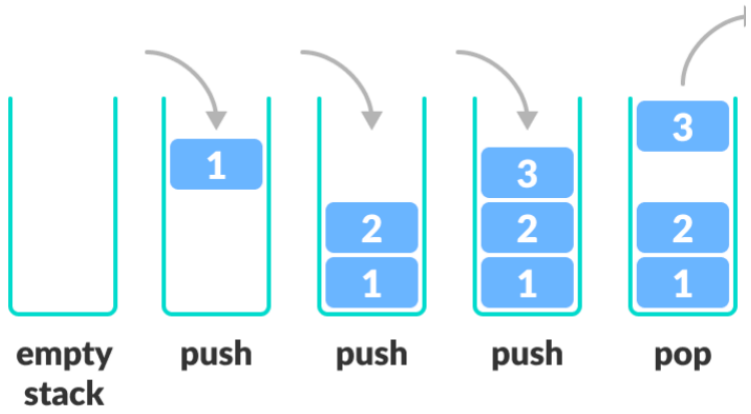


Important Operations:
Add()
Remove()
Contains()

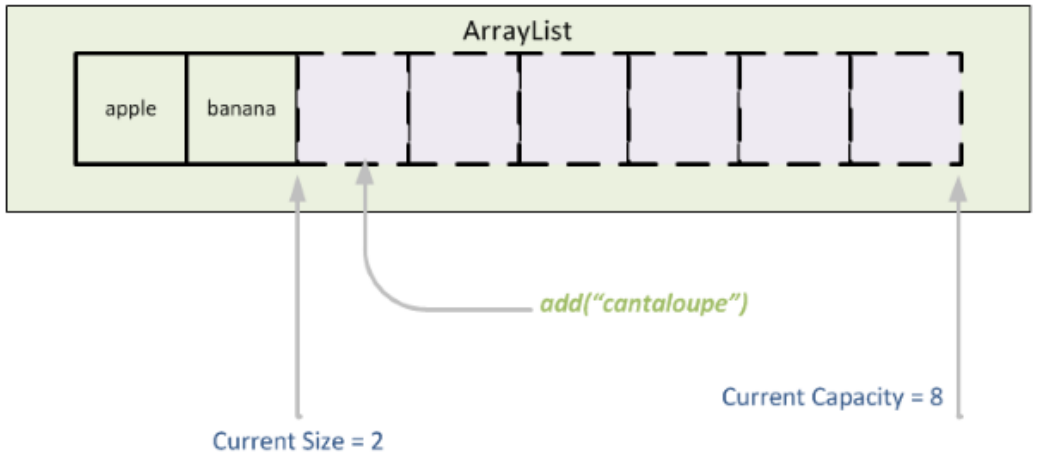
Queue



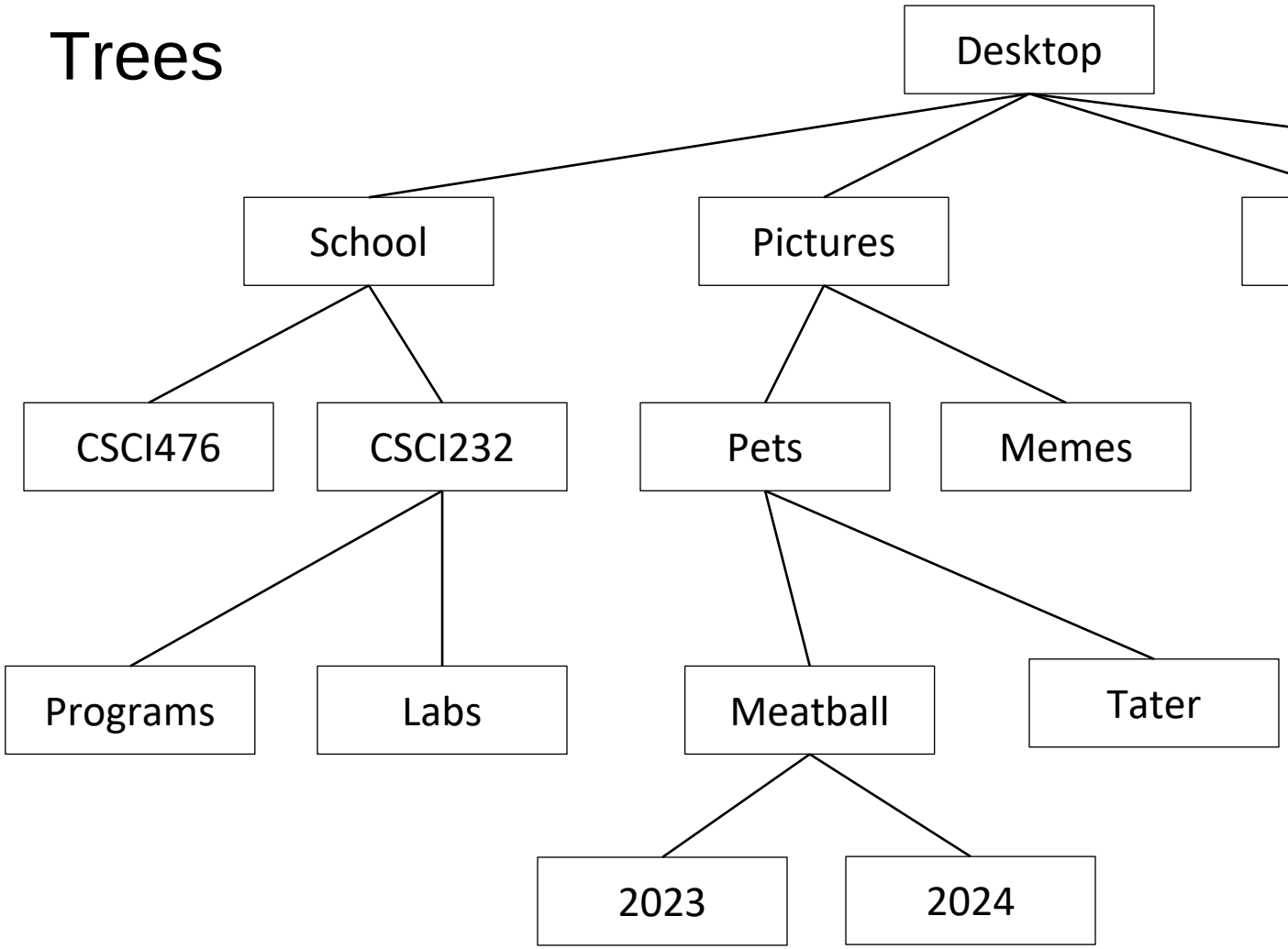
Stack



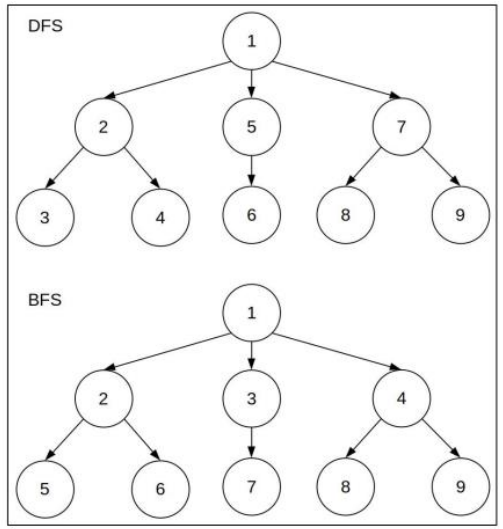
ArrayList



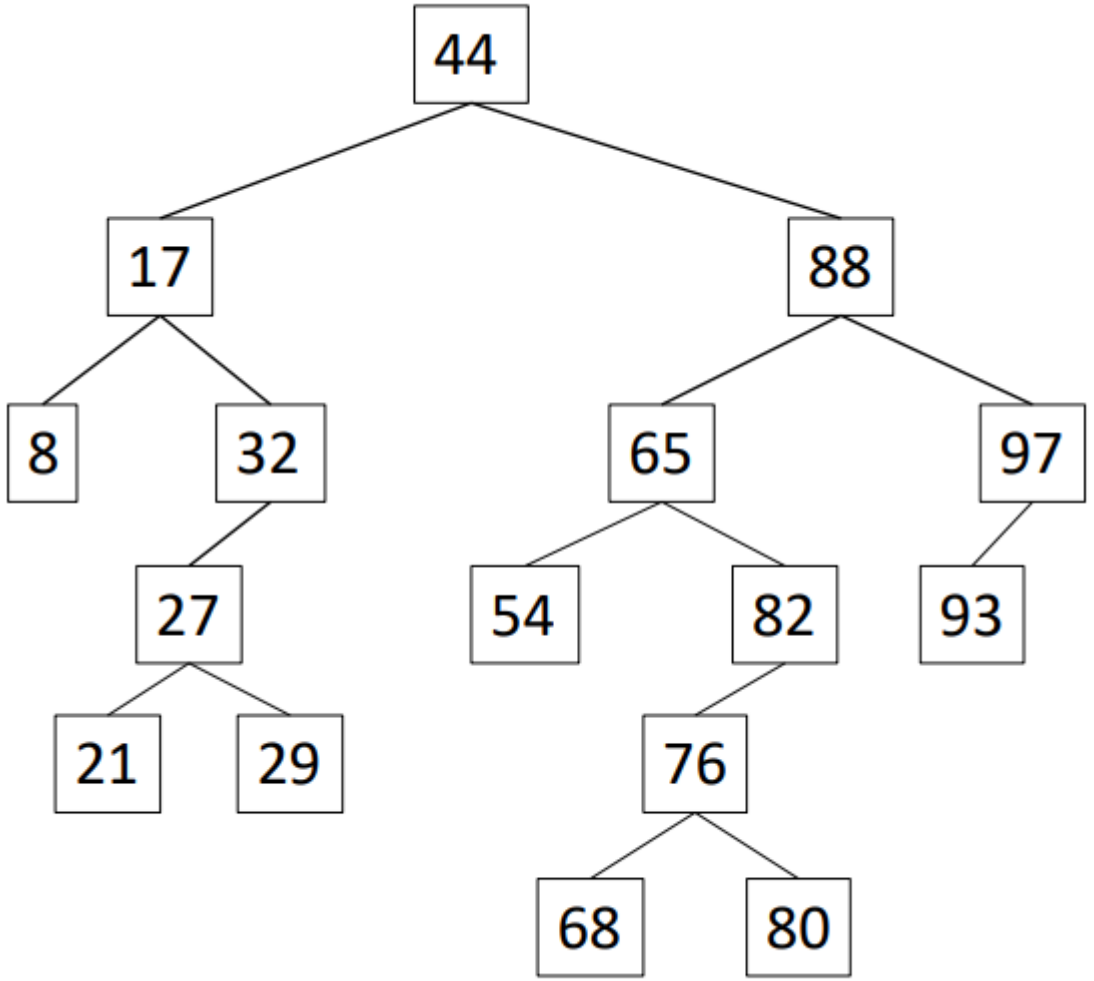
Trees



- Tree Vocabulary
- Tree Traversal (Depth First, Breadth First)

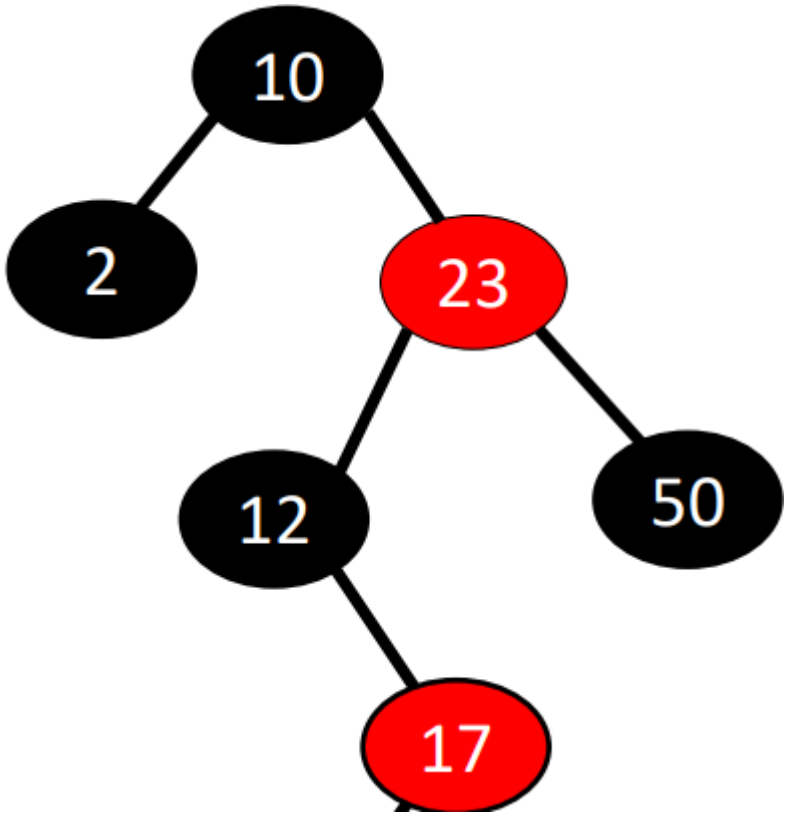


Binary Search Trees



- Rules of BST
- How to add and remove in a BST
- Running time of add(), remove(), contains()
- BST Navigation (preorder, post order, in order)
- “Good tree” and “Bad tree”
- Why are BST helpful

Red Black Trees



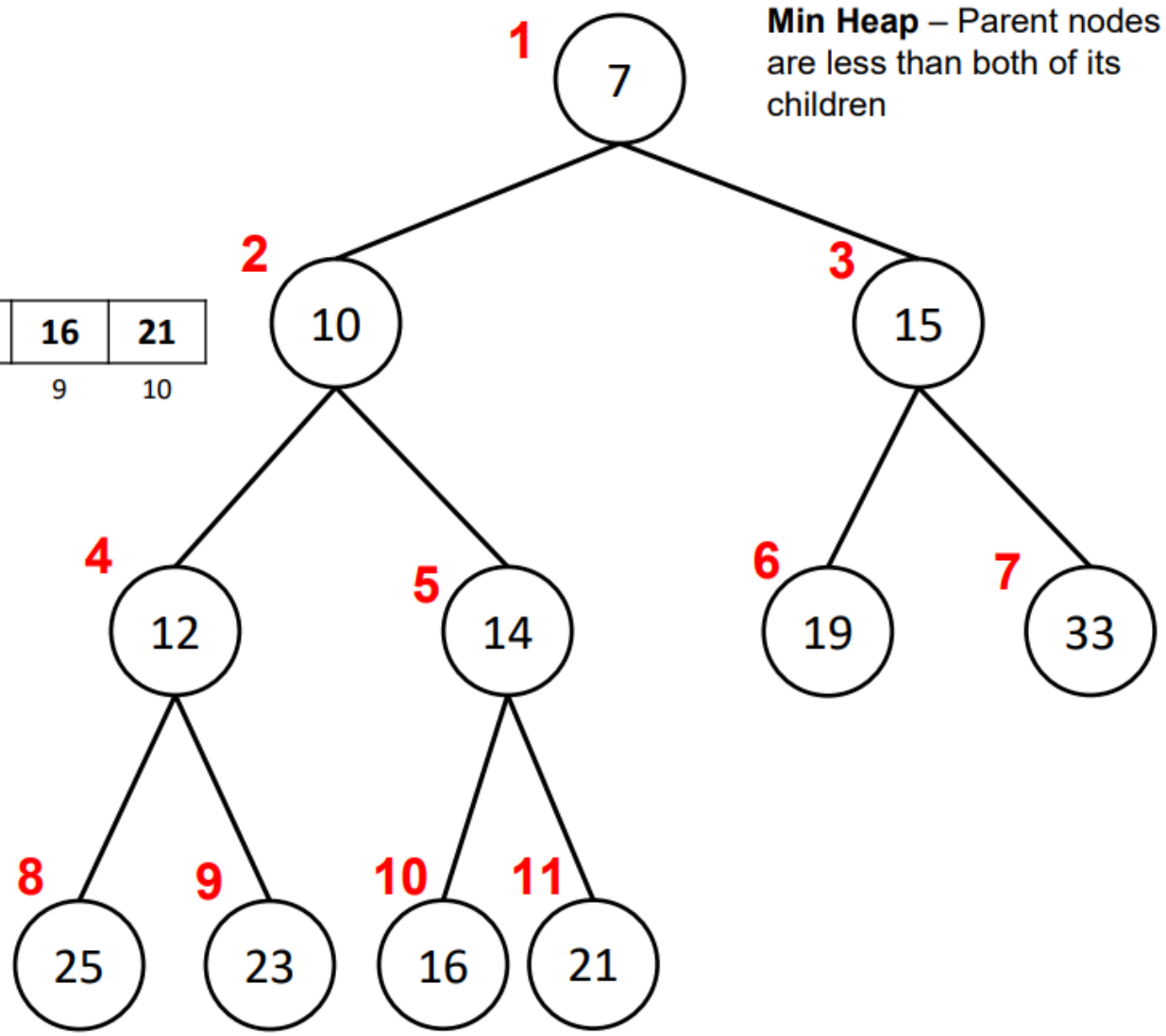
-Rules of RB tree
-Purpose of RB tree
-Verify a RB tree
-Explain why something is not an RB tree and how to fix it

1. Every node is either **red** or **black**
2. The `null` children are **black**
3. The root node is **black**
4. If a node is **red**, both children must be **black**
5. For each node, all paths from the node to descendant leaves contain the same number of **black** nodes

Heaps (priority queue)

Array

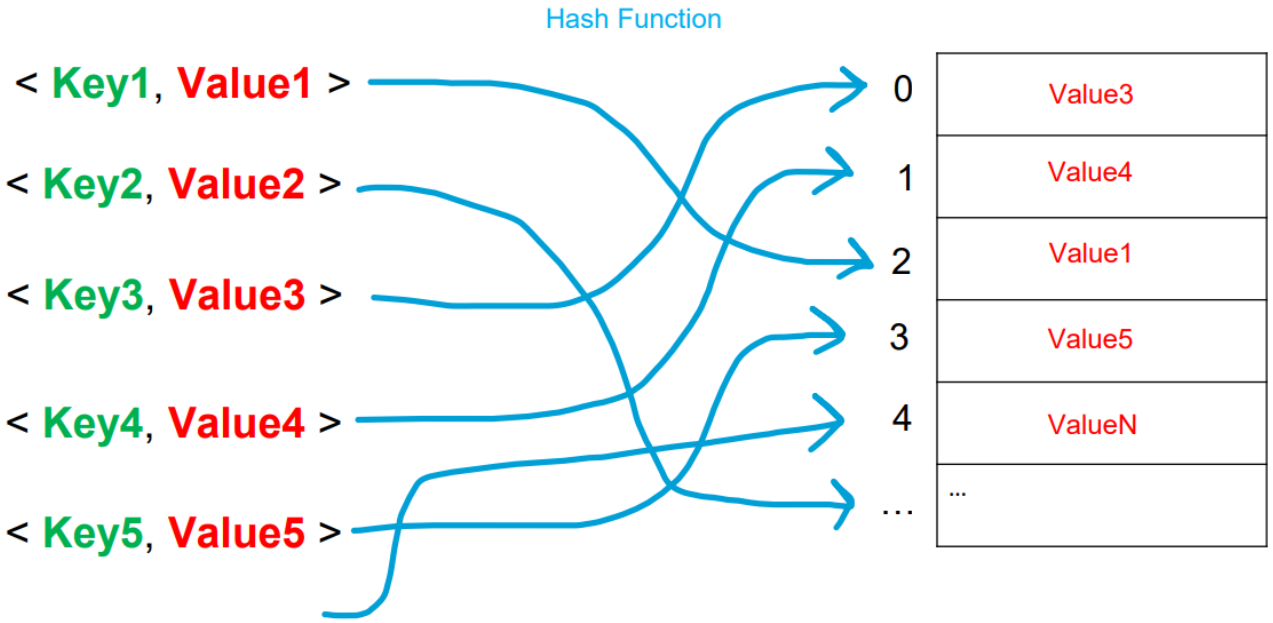
7	10	15	12	14	19	33	25	23	16	21
0	1	2	3	4	5	6	7	8	9	10



Min Heap – Parent nodes are less than both of its children

- What a heap is
- How to add or remove something from a heap
- Heapify
- Representation via an array (how to get parent and children)
- What is the purpose of heap? What does it do well?
- Running time of maintaining, adding, removing, and searching a heap

Hash Tables / HashMaps / HashSets/ Dictionaries




```
import java.util.HashMap;
import java.util.HashSet;
```

- Rules for a Hash Table
- How to insert, remove, lookup, and search in a Hash Table
- Running time of adding, removing, and contains
- Why are hash tables helpful?
- How to reduce and deal with collisions (separate chaining and linear probing)
- Java HashMaps and HashSets
- Hashing Functions