1 Introduction

Some questions to ask before starting on a problem

- Extract out important keywords (what DS to use?)
- Edge cases? *e.g.* if `size==0` or `size==1`,
- Trivial cases? can just hardcode

Code styling

- CS2030 Code Styling Guide
- Google Java Styling Guide
- **Modularity**: use method to print answers inside main method

```
\\ print answer
ans = simulate(n,k,m);
printAns();
```

- **No global variables**

2 Java

How to throw exception?

```
public class MyException extends
    Exception {
  private int var;
  public MyException(int var) {
    this.var = var
  }
  public int getVar() {
    return this.var;
  }
}

public class Main {
  public static void main(String[] args
    ) {
    try {
      ...
      throw new MyException(errorVar);
    } catch (MyException e) {
      System.out.println(e.getVar());
    }
  }
}
```

3 Data Structures

$$O(1) < O(\log(n)) < O(n^c) \text{ where } c < 1$$

$$O(n) < O(\log(n!)) = O(n\log(n)) < O(n^2)$$

$$O(n^k)[\text{ where } k > 2] < O(k^n)[\text{ where } k \geq 1] < O(n!)$$

How to implement Data Structures?

- Composition: use well-known DS as an attribute of the implemented DS
- Inheritance: extends well-known DS

3.1 Linked List

- Motivation: implementation of list using array needs to occupy contiguous memory space (can result in memory error)
- Variants of linked list:
  - Tailed (need to maintain `head` and `tail`)
  - Circular
  - Doubly linked (`prev` and `next` attributes for `ListNode`)
- How to find cycle?
  Answer: use fast and slow pointers

```
    slow = slow.next;
    fast = fast.next.next;
```

- **[IMPT]** Drawing pictures is very important to visualize the program!

Java API: `ArrayList` or `LinkedList`

```
\\ constructor
ArrayList<Integer> list = new
 ArrayList<Integer>();
```

3.2 Stack

```
// to construct an array of generics
E[] arr = (E[]) new Object[size];
/*
// does not work
E[] arr = new E[size]
*/
```

Uses:

- **[IMPT]** Converting infix to postfix expression (Lecture 4 Slide 28)
- **[IMPT]** Evaluating postfix expression

3.3 Queue

Uses:

- **[IMPT]** Breadth-first traversal of trees

4 Recursion

[IMPT] Recipe for recursion (3 fingers)

1. General recursive case: identify simpler instances of the same problem
2. Base case: cases that we can solve without recursion
3. Be sure that we are able to reach the simplest instance so that we won't end up in infinite loop

Uses

- Insert item into sorted LinkedList
- Tower of Hanoi
- **[IMPT]** Combination (n choose k)
- Binary search
- Finding k-th smalles element (use pivot element `p`)
  move elements < `p` to the left of `p`
  move elements > `p` to the right of `p`
- Printing all permutations of a String

Overloading: same function name but with different parameters (useful in Java)

5 Algorithms

5.1 Sorting

6 Java Tricks

- OOP is important (CardGame)
  - If it involves an array, OOP is useful, methods can just modify properties/attributes of the object class (*e.g.* `reversed=true; increment=4`)
    - Especially true if only need to print statement at the end
- Use `StringBuilder` for return statements
  - Java StringBuilder API
  - Zigzag conversion