Today's agenda

- Class administration
- Recap
- Homework discussion
- Lecture
- Break (~2:30-2:45pm)
- Practice (~2:45pm-4:15pm)
- Discussion of problems
Class Administration
Class Homework

• Your homework grade will be
  \[(\# \text{ solved problems} + \# \text{ solved bonus problems}) / \# \text{ problems}\]

• Each class I will assign a reasonable number of problems for homework and skill level
  – Bonus questions meant to challenge the more advanced students
Recap
Competitive Programming

• Given *well-known* computer science problems, solve them *as fast as possible*
  – Find a solution that reduces down to a well-known problems, not research problems
  – Pass all the judge data correctly
  – Solution should run fast enough
  – Do not over-engineer the solution
Competitive Programming

• Tips:
  - Type fast and correctly
  - Quickly identify problem types
  - Perform algorithm analysis
  - Master a programming language
  - Test your code
  - Practice
  - Teamwork
Competitive Programming

• Tips:
  – Type fast and correctly
  – Quickly identify problem types
  – Perform algorithm analysis
  – Master a programming language
  – Test your code
  – Practice
  – Teamwork
Competitive Programming

• Rules of thumb:
  - \(2^{10} = 1024 \approx 1000 = \text{“one thousand”}\)
  - \(2^{20} \approx 1,000,000 = \text{“one million”}\)
  - Max 32-bit signed integer = \(2^{31} - 1 \approx 2,000,000,000 = \text{“two billion”}\)
  - Max 64-bit signed integer = \(2^{64} - 1\)
Competitive Programming

- Rules of thumb:
  - For a set of $n$ elements there are $n!$ permutations and $2^n$ combinations
  - $10! \approx 3.6$ million
Competitive Programming

- Know your language:
  - `System.out.printf("[%f]\n", 1.234567);`
  - `// prints [1.234567]`
  - `System.out.printf("[%.5f]\n", 1.234567);`
  - `// prints [1.23457]`
Competitive Programming

- Know your language:
  - System.out.printf("[%9.5f]\n", 1.234567);
  - // prints [ 1.23457]

  - System.out.printf("[%09.3f]\n", 1.234567);
  - // prints [00001.235]

  - Google “string format java”, or resources on website
Competitive Programming

• Know your language:
  – `int x = Integer.parseInt("FF", 16);`
  – `System.out.println(x);`
  – `// prints 255`
  – `System.out.println(Integer.toString(x, 2));`
  – `// prints 11111111`
Competitive Programming

- Know your language:
  - Arrays.sort(), Arrays.binarySearch()
    - For native Java arrays
  - Collections.sort(), Collections.binarySearch()
    - For complex data structures
  - Even Arrays.toString() for debugging
  - Arrays.fill() to set every element in an array
Data structures

• Native arrays
  – 1D: int[], long[], boolean[]
  – 2D: int[][][], long[][][], boolean[][][]
  – 3D: int[][][][][], long[][][][][], boolean[][][][][]
  – Used all the time to represent data (e.g., Minesweeper problem)
Data structures

- **ArrayList**
  - Frequently used for arrays of indeterminate sizes
  - I refer to this kind of data structure as a “List”, e.g., the Java supertype of ArrayList (which is a subtype of Collection)
  - Append time: $O(1)$
  - Insertion time: $O(N)$ where $N$ is length
  - Search time: $O(N)$
    - Except a sorted list, $O(\log N)$
Data structures

- LinkedList
  - Never used in programming contests*
    - *that I know of
Data structures

• Stack
  – **LIFO**, useful for reversing data
    • Bracket matching, postfix problems
  – Two operations: push and pop

• Queue
  – **FIFO**, useful for breadth first searching
    • (later…)
  – Two operations: push and pop
Data structures

• Heap
  - In Java, a PriorityQueue
  - Useful if you continually need the smallest (largest) item
  - Two operations: push and pop
  - Insertion time: $O(\log N)$
  - Removal time: $O(1)$
Data structures

- Binary tree
  - In Java, a TreeSet / TreeMap
  - Insertion time: $O(\log N)$
  - Search time: $O(\log N)$

- Hash table
  - In Java, a HashSet / HashMap
  - Insertion time: treat as $O(1)$
  - Search time: $O(1)$
Data structures

• Graphs
  – Adjacency matrix, adjacency list
    • Use 2D arrays or lists to represent these
  – Edge list

• Union-find disjoint sets
  – Fast data structure for determining which graph element is part of which set
  – Also a fast data structure for unioning sets
  – Treat $N$ operations performed as $O(N)$
Data structures

• Basic tools for tackling problems
  – Knowing the O-notation of the operations helps you perform algorithm analysis
  – So you know what to use before coding anything!
  – e.g., instead of sorting an array N times to keep it sorted, is there a data structure that will keep everything sorted for you?
  – Treated as black boxes in this class
Discussion of last week's homework
Simple skill testing question
List of Conquests

- How many women did Giovanni love in each country?
Event Planning

- Budget, hotel, and beds
Minesweeper

- Given locations of all mines, determine the minesweeper numberings
Magic Palindrome Square

- Can the given string be a magic palindrome square?
Automatic Answer Demo
Bitwise operations
Bit operations

- Another tool in your toolbox
  - Taking advantage of native data types
- OR, AND, XOR, NOT, left shift, right shift
Bit operations

• **OR**

\[
\begin{array}{c}
- \quad 11001110 \\
| \quad 10011000 \\
= \quad 11011110
\end{array}
\]

- In decimal: 206 | 152 = 222
Bit operations

- AND &

  \[
  \begin{array}{c}
  \text{AND} \\
  \hline
  0 & 0 & 0 \\
  1 & 0 & 1 \\
  \end{array}
  \]

  - In decimal: 206 & 152 = 136
Bit operations

- XOR ^

- 11001110
  ^ 10011000
  = 01010110

- In decimal: 206 ^ 152 = 86
Bit operations

- **NOT ~**

\[
\sim 11001110
\]

\[
= 00110001
\]

- In decimal: \( \sim 206 = 25 \)
Bit operations

- Right shift
  - In decimal: $218 \gg 1 = 109$
  - Same as integer division by two

- $11011010$
  \[ \gg \ \\
  \ 1 \]
  
  $= \ 01101101$
Bit operations

- Right shift
  - Arithmetic shift (>>)
    - Recall with signed integers, negative numbers have the highest bit set
    - Sign is retained with arithmetic shift
  - Logical shift (>>>)
    - Bits are treated as bits, no sign is retained
Bit operations

- Left shift
  - In decimal: $218 \ll 1 = 436$
  - Same as multiplication by two
  - No wrapping, so topmost bits are lost

- $11011010 \ll 1 = 110110100$
- $11011010 \ll 1 = 110110100$
Exercise

- Write a method `numberOfBits(x)` that returns the number of bits set in `x`
Exercise

• Write a method setBit(x, i) that returns x with the \(i\)th bit set
Exercise

- Write a method `clearBit(x, i)` that returns `x` with the `i`th bit cleared
Exercise

- Write a method `toggleBit(x, i)` that returns `x` with the `i`th bit toggled.
Exercise

- Write a method `testBit(x, i)` that returns true if and only if the $i$th bit in $x$ is set
Exercise

- Write a method firstBits\( (n) \) that returns an integer with the first \( n \) bits set.
Exercise

• Write a method printNBitsSet($n$, $s$) that prints all bit strings of size $s$ with $n$ bits set
Practice
For next class

- Readings:
  - Review chapter 2 data structures if it's hazy
  - New resources on website

- Exercises:
  - Two problems listed on website