Administrative Details

- Lab 10: Two Towers is online
  - Individual lab again this week
- Final Exam location: TBL 112
  - It's on Dec. 14\textsuperscript{th}, 9:30--noon
Last Time

• Binary Search Tree Implementation details

• Balanced Binary search trees
  • AVL Trees
    • Height is $O(\log n)$, so all operations are $O(\log n)$
  • Red-Black Trees
    • Different height-balancing idea: height is $O(\log n)$
    • All operations are $O(\log n)$

• Splay Trees
  • No guaranteed balance; good amortized performance
  • Any sequence of $m$ operations take $O(m \log n)$ time
Today’s Outline

Much less esoteric…

• Bit operations
  • Useful in general, but required for Lab 10

• Introduction To Graphs
  • Basic Definitions and Properties
  • Applications and Problems
Representing Numbers

• Humans usually think of numbers in base 10
• But even though we write `int x = 23;` the computer stores `x` as a sequence of 1s and 0s
• Recall Lab 3:
  ```java
  public static String printInBinary(int n) {
    if (n <= 1)
      return "" + n%2;
    return printInBinary(n/2) + n%2;
  }
  ```
• 00000000 00000000 00000000 00010111
Bitwise Operations

• We can use *bitwise* operations to manipulate the 1s and 0s in the binary representation
  • Bitwise ‘and’:  &
  • Bitwise ‘or’:  |

• Also useful: bit shifts
  • Bit shift left:  <<
  • Bit shift right:  >>
& and |

• Given two integers a and b, the bitwise or expression \( a \mid b \) returns an integer s.t.
  • At each bit position, the result has a 1 if that bit position had a 1 in EITHER a OR b
  • \( 3 \mid 6 = ? \)

• Given two integers a and b, the bitwise and expression \( a \& b \) returns an integer s.t.
  • At each bit position, the result has a 1 if that bit position had a 1 in BOTH a AND b
  • \( 3 \& 6 = ? \)


>> and <<

• Given two integers \(a\) and \(i\), the expression \((a << i)\) returns \((a \times 2^i)\)
  • Why? It shifts all bits left by \(i\) positions
  • \(1 << 4 = ?\)

• Given two integers \(a\) and \(i\), the expression \((a >> i)\) returns \((a / 2^i)\)
  • Why? It shifts all bits right by \(i\) positions
  • \(1 >> 4 = ?\)
  • \(97 >> 3 = ?\) \((97 = 1100001)\)

• Be careful about shifting left and “overflow”!!!
Revisiting `printInBinary(int n)`

- How would we rewrite a recursive `printInBinary` using bit shifts and bitwise operations?

```java
public static String printInBinary(int n) {
    if (n <= 1) {
        return "" + n;
    }
    return printInBinary(n >> 1) + (n & 1);
}
```
Revisiting `printInBinary(int n)`

- How would we write an iterative `printInBinary` using bit shifts and bitwise operations?

```java
public static String printInBinary(int n, int width) {
    String result = "";
    for(int i = 0; i < width; i++)
        if ((n & (1<<i)) == 0)
            result = 0 + result;
        else
            result = 1 + result;
    return result;
}
```
Lab 8: Two Towers

• **Goal:** given a set of blocks, iterate through all possible subsets to find the *best* set

```
1  2  3  4  ...  14  15
```

• “Best” set produces the most balanced towers

• **Strategy:** create an iterator that uses the bits in a binary number to represent subsets
Lab 8: Two Towers

- A block can either be in the set or out
  - If bit is a 1, in. If bit is a 0, out
Questions?

- We will write a “SubsetIterator” to enumerate all possible subsets of a Vector<E>
- We will use SubsetIterator to solve two problems
  - Two Towers
  - Identify all Subsequences of a String that are words
    - Use your LexiconTrie! (or an OrderedStructure)
Graphs Describe the World

- Transportation Networks
- Communication Networks
- Social Networks
- Molecular structures
- Dependency structures
- Scheduling
- Matching
- Graphics Modeling
- ....
Nodes = subway stops; Edges = subway lines
Nodes = cities; Edges = rail lines connecting cities
Note: Connections in graph matter, not precise locations of nodes
Internet (~1972)
Internet (~1998)
Word Game

Nodes = words; Edges = words that differ by exactly one letter
CS Pre-requisite Structure (subset)

Nodes = courses; Edges = prerequisites **
Basic Definitions & Concepts

Definition: An undirected graph $G = (V, E)$ consists of two sets

- $V$: the vertices of $G$, and $E$: the edges of $G$
- Each edge $e$ in $E$ is defined by a set of two vertices: its incident vertices.
- We write $e = \{u, v\}$ and say that $u$ and $v$ are adjacent.
Walking Along a Graph

• A walk from u to v in a graph G = (V,E) is an alternating sequence of vertices and edges:
  
u = v_0, e_1, v_1, e_2, v_2, \ldots, v_{k-1}, e_k, v_k = v
  
such that each e_i = \{v_i, v_{i+1}\} for i = 1, \ldots, k

• Note a walk starts and ends on a vertex

• If no edge appears more than once then the walk is called a path

• If no vertex appears more than once then the walk is a simple path
**Walking In Circles**

- A *closed walk* in a graph \( G = (V,E) \) is a walk
  \[ v_0, e_1, v_1, e_2, v_2, \ldots, v_{k-1}, e_k, v_k \]
  such that \( v_0 = v_k \) (it ends at the starting \( v \))

- A *circuit* is a *path* where \( v_0 = v_k \)
  - Circuit vs. closed walk? Circuit has no repeat edges

- A *cycle* is a *simple* path where \( v_0 = v_k \)
  - Circuit vs. cycle? Cycle has no repeated vertices.

- The *length* of any of these is the number of edges in the sequence