Graphs and Trees

Section 2.4

Dr. Mayfield and Dr. Lam

Department of Computer Science
James Madison University

Oct 09, 2015
Due today!

Your portfolio must be done in \LaTeX

- See \texttt{portfolio.tex} in Template Files
  - In Texmaker, press F1 \textit{twice} to build
  - The rest should be self-explanatory :)  

- Write your name, email, today's date
- Fill in the table on Page 2 (rows 1–3)

For each problem you submit:

- Write 1–2 paragraphs of reflection
- Format your code neatly for printing
  - There should be no text wrapping
  - Double check the page boundaries
### Portfolio rubric

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Ratings</th>
<th>Pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reflection</td>
<td>Thoughtful, good length, correct grammar</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 pts</td>
<td>2 pts</td>
</tr>
<tr>
<td></td>
<td>One or more incomplete/incorrect criteria</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 pts</td>
<td>1 pts</td>
</tr>
<tr>
<td></td>
<td>No reflection submitted or way too short</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 pts</td>
<td>0 pts</td>
</tr>
<tr>
<td>Source Code</td>
<td>Challenging problem accepted by judge</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 pts</td>
<td>3 pts</td>
</tr>
<tr>
<td></td>
<td>Mostly complete but not fully working</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 pts</td>
<td>2 pts</td>
</tr>
<tr>
<td></td>
<td>Solution does not pass sample tests</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 pts</td>
<td>1 pts</td>
</tr>
<tr>
<td></td>
<td>Little or no source code submitted</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 pts</td>
<td>0 pts</td>
</tr>
<tr>
<td>Formatting</td>
<td>No text wrapping and good page breaks</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 pts</td>
<td>1 pts</td>
</tr>
<tr>
<td></td>
<td>Awkward text wrapping or page breaks</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 pts</td>
<td>0 pts</td>
</tr>
</tbody>
</table>

**Total Points: 6**

- Note: 3 problems = 18 points possible
Today’s Problems

What is a graph?
Graphs 101

Collection of nodes (vertices) and edges (links)

http://en.wikipedia.org/wiki/Graph_(mathematics)

- Can be directed or undirected
- Can be weighted or unweighted

Some examples

- Driving directions (cities and roads)
- Social networks (people and friendships)
- The Internet (hosts and connections)
- World Wide Web (URLs and hyperlinks)
Representing graphs

Check out http://visualgo.net/

- Started in 2011 by Dr. Steven Halim
- “VisuAlgo is like a 24/7 copy of himself.”

See Graph Data Structures

- Switch to “Tutorial Mode”
Adjacency matrix (easier)

2D array of integers
- 0=no edge, 1=edge
- Or other values (if weighted)
- Symmetric if undirected

```c++
int AdjMat[V][V];  // C++
```
```java
int[][] AdjMat = new int[V][V];  // Java
```

Good for small, dense graphs
- Small: \( V < 1000 \)
- Dense: not many zeros

![Adjacency matrix](image)
Adjacency list (more useful)

Vector of vector of ints
  ▶ Stores a list of neighbors

```cpp
// unweighted graph
typedef vector<int> vi;
vector<vi> AdjList;
```

Or, vector of vector of pairs
  ▶ Stores neighbors and weights

```cpp
// weighted graph
typedef pair<int, int> ii;
typedef vector<ii> vii;
vector<vii> AdjList;
```
Edge list (uncommon)

Sorted list of edges (useful for some algorithms)

// unweighted
typedef pair<int, int> ii;
vector<ii> EdgeList;

// weighted
vector< pair<int, ii> > EdgeList;

C++ tip

- nested templates can't use << or >>
- you need a space for it to compile
Adjacency list (more useful)

In Java, you probably should just write some simple wrapper classes and be prepared to use them.

- Examples at Open Data Structures

One issue: Java doesn’t have built-in pairs. Possible work-around:

```java
public class Pair<F, S> {
    public final F first;
    public final S second;

    public Pair(F first, S second) {
        this.first = first;
        this.second = second;
    }
}
```
Today’s Problems

What is a tree?
Graph with **no cycles** / one path between any two nodes

http://en.wikipedia.org/wiki/Tree_(graph_theory)

A **binary** tree contains nodes with a maximum of two children (left and right).
Implementation

Traditional binary tree implementation that we \textit{won't} use:

```cpp
class Node {
public:
    int value;
    Node *left;
    Node *right;
};

int main() {
    Node *root = new Node();
    root->value = 8;
    root->left = NULL;
    root->right = NULL;
}
```
Figure 8.17  A tree stored without pointers

Conceptual tree

Actual storage organization

Root node

Nodes in 2nd level of tree

Nodes in 3rd level of tree
Binary tree

Store the root at index 1

// n is index of a node
#define left(n) ((n)*2)   // left subchild
#define right(n) ((n)*2+1) // right subchild
#define parent(n) ((n)/2)  // parent

// array of max nodes
int tree[MAX] = {0};

Ideas for today’s contest See the book for more details!

- Union-Find Disjoint Sets
  - UVa 11503 Virtual Friends
- Segment Tree
  - UVa 11235 Frequent Values
Implicit structures

Some graphs don’t need to be generated
- Navigating a 2D grid (e.g., chessboard)
- Determine edges with simple rules, e.g.:
  - (1,2), (2,3), ..., (n-1, n), (n, 1)
  - All \((u, v)\) such that \(u + v\) is prime

Some trees don’t need to be generated
- UVa 11350 Stern-Brocot Tree
- Construct new nodes as you go