Graphs and Trees

Section 2.4

Dr. Mayfield and Dr. Lam

Department of Computer Science
James Madison University

Oct 09, 2015

acm International Collegiate Programming Contest

IBM event sponsor

Due today!

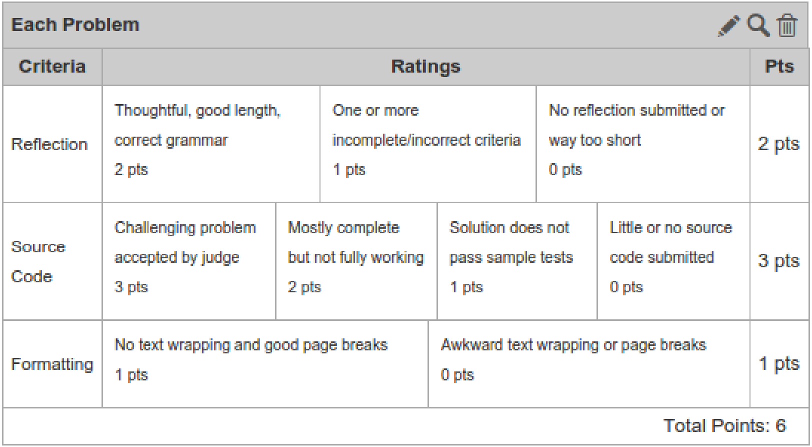
Your portfolio must be done in LaTeX

- See `portfolio.tex` in Template Files
  - In Texmaker, press F1 *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



Each Problem

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

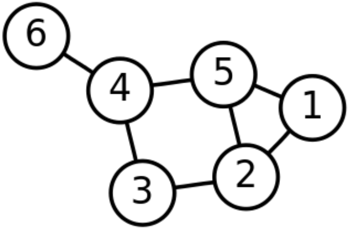
Check out http://visualgo.net/

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

visualising data structures and algorithms through animation

See Graph Data Structures

- Switch to "Tutorial Mode"



Adjacency matrix (easier)

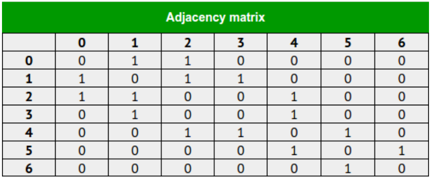
2D array of integers

- 0=no edge, 1=edge
- Or other values (if *weighted*)
- Symmetric if undirected

```
// V is the number of vertices
int AdjMat[V][V];                   // C++
int[][] AdjMat = new int[V][V];     // Java
```

Good for small, dense graphs

- small: $V < 1000$
- dense: not many zeros



Adjacency matrix

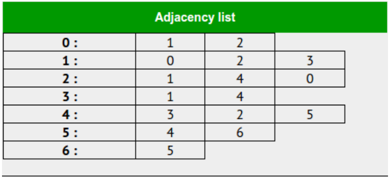
	0	1	2	3	4	5	6
0	0	1	1	0	0	0	0
1	1	0	1	1	0	0	0
2	1	1	0	0	1	0	0
3	0	1	0	0	1	0	0
4	0	0	1	1	0	1	0
5	0	0	0	0	1	0	1
6	0	0	0	0	0	1	0

Adjacency list (more useful)

Vector of vector of ints

- Stores a list of neighbors

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



Adjacency list

0 :	1	2	
1 :	0	2	3
2 :	1	4	0
3 :	1	4	
4 :	3	2	5
5 :	4	6	
6 :	5		

Or, vector of vector of pairs

- Stores neighbors and weights

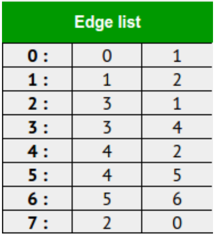
```
// weighted graph
typedef pair<int, int> ii;
typedef vector<ii> vii;
vector<vii> AdjList;
```

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;
```



Edge list		
0 :	0	1
1 :	1	2
2 :	3	1
3 :	3	4
4 :	4	2
5 :	4	5
6 :	5	6
7 :	2	0

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

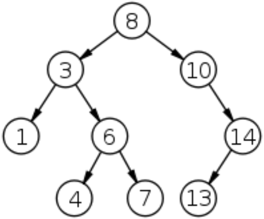
What is a tree?

Trees 101

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).



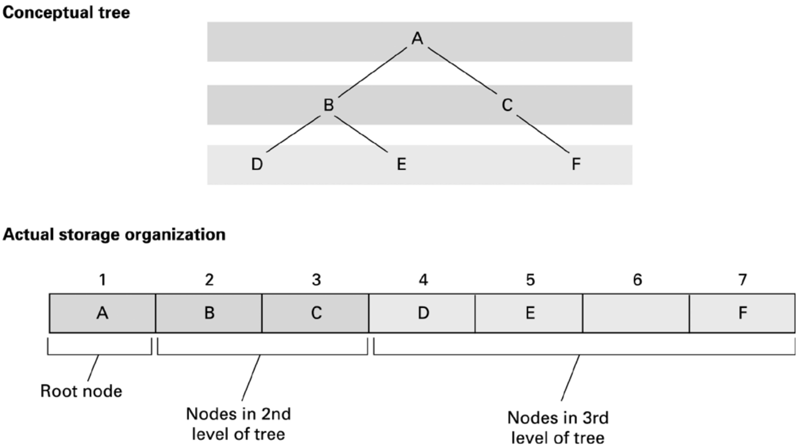
Implementation

Traditional binary tree implementation that we won't use:

```
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
A
B
C
D
E
F
Actual storage organization

1	2	3	4	5	6	7
A	B	C	D	E		F

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