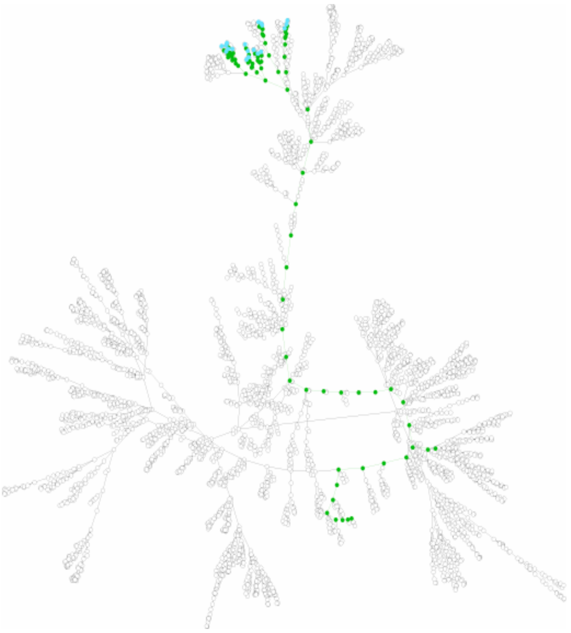
#32: Graphs: An Introduction

April 9, 2018 · *Wade Fagen-Ulmschneider*



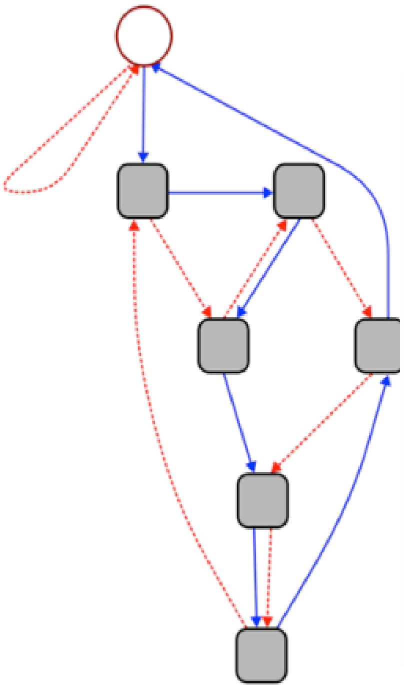
Disjoint Sets

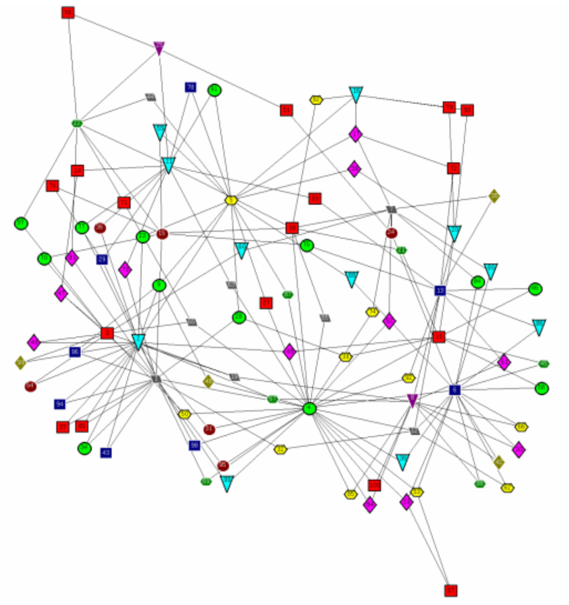
- Worst case running time of find(k)**:**
- Worst case running time of union(r1, r2), given roots:
- Iterated log: **log*(n)** = *number of times you can take a log*
- Overall running time:
    - A total of **m** union/find operation runs in:

A Review of Major Data Structures so Far

Array-based	List/Pointer-based
- Sorted Array	- Singly Linked List
- Unsorted Array	- Doubly Linked List
- Stacks	- Skip Lists
- Queues	- Trees
- Hashing	- BTree
- Heaps	- Binary Tree
- Priority Queues	- Huffman Encoding
- UpTrees	- kd-Tree
- Disjoint Sets	- AVL Tree

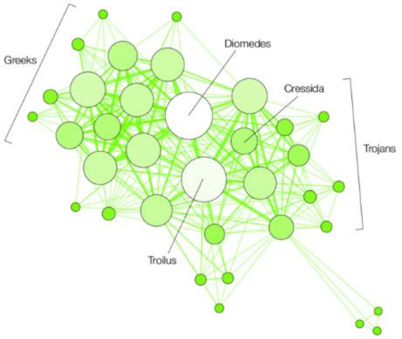
An Introduction to Graphs



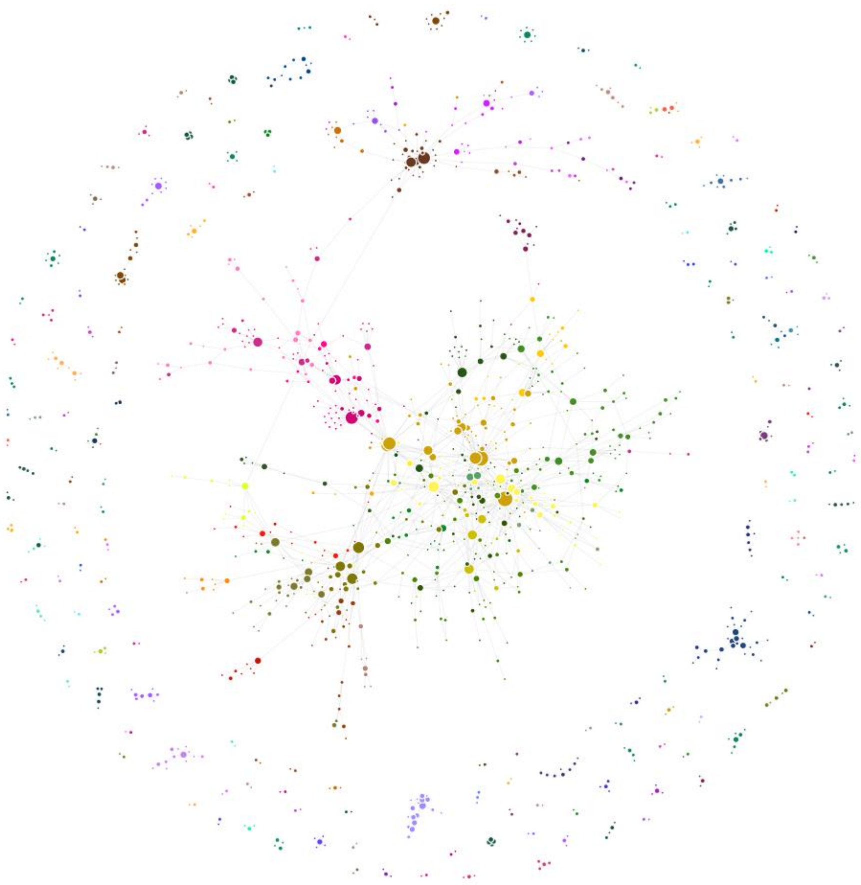


Queen Gertrude
King Claudius
Hamlet

HAMLET



TROILUS AND CRESSIDA

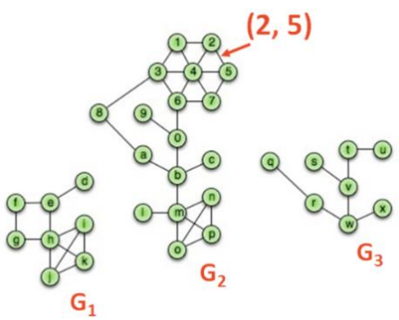


Motivation:
Graphs are awesome data structures that allow us to represent an enormous range of problems. To study these problems, we need:

1. A common vocabulary to talk about graphs
2. Implementation(s) of a graph
3. Traversals on graphs
4. Algorithms on graphs

Graph Vocabulary

Consider a graph **G** with vertices **V** and edges **E**, **G=(V,E)**.



Incident Edges:
I(v) = { (x, v) in E }

Degree(v): |**I**|

Adjacent Vertices:
A(v) = { x : (x, v) in E }

Path(G_2): Sequence of vertices connected by edges

Cycle(G_1): Path with a common begin and end vertex.

Simple Graph(G): A graph with no self loops or multi-edges.

Subgraph(G): **G' = (V', E')**:
$V' \in V$, $E' \in E$, and $(u, v) \in E \rightarrow u \in V', v \in V'$

Graphs that we will study this semester include:
Complete subgraph(G)
Connected subgraph(G)
Connected component(G)
Acyclic subgraph(G)
Spanning tree(G)

CS 225 – Things To Be Doing:

1. Theory Exam 3 final day is **today**
2. lab_heaps due Sunday, April 8th
3. MP6 released; Extra Credit deadline on Monday, April 9th
4. Daily POTDs are ongoing!