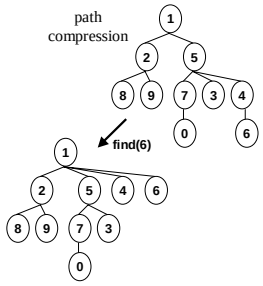
# Finish Union/Find, Finish Graph Algorithms, Quick Overview

CS211
Fall 2000

## Union/Find

- Operations
  - Union: Combine two sets
  - Find: Given an item, determine the "name" of the set that contains it
- Use reverse trees
  - Each item points at its parent
  - The root is the "name" of the set
- Union-by-Size
  - Always make the larger tree be the root
- Path Compression
  - Every time we "find" something, we update every item we touch so that it points at the root






## Union/Find Analysis

Theorem (Tarjan)
Using weighted union and path compression, a sequence of n union/find operations takes time $O(n\ \alpha(n))$

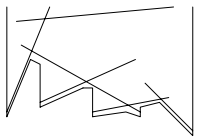
- Note that $\alpha(n) \leq 4$ for any integer n that we are *ever* likely to encounter
- Is the $\alpha(n)$ factor really necessary?
  - Yes: Tarjan showed a *lower* bound of $\Omega(n\ \alpha(n))$ for union/find
  - Claim: the inverse Ackerman's function is not just an artifact of this one problem



## Lower Envelope of Line Segments

- Given n line segments in the plane, what is the worst-case complexity of their lower envelope?

  $\Theta(n\ \alpha(n))$






## Two MST Algorithms (Both Greedy)

Kruskal's Algorithm

- Choose the shortest edge e such that
  - e is not yet processed
  - e does not make a cycle

Prim's Algorithm

- Choose the shortest edge e such that
  - e touches the tree
  - e touches a vertex not in the tree



## Kruskal's MST Algorithm

```
KruskalMST(G):
    E = edges of G;
    forest = empty;
    do
        <u,v> = least cost edge of E;
        E = E - <u,v>;
        if (u and v in different trees)
            forest = forest ∪ <u,v>;
    while (E is nonempty);
    return forest
```

- Can sort the edges initially (or can use a PQ)
- Use Union/Find to check for different trees and to combine trees
- Total worst-case time: $O(e \log e)$ when using adjacency lists
- Time is $O(v^2 + e \log e)$ for adjacency matrix

## Quick Review: Programming Topics

Object Oriented Programming
- Classes and Objects
  - Parameter Passing
  - Objects vs. References
  - Abstraction
  - Encapsulation
- Inheritance
  - Polymorphism
  - Dynamic Method Binding
  - Abstract Classes
  - Interfaces
  - Upcasting vs. Downcasting

Other Topics
- Access-Control Modifiers
- Packages
- Exceptions
- Program Design
  - UML Diagrams
  - Pseudo-code
  - Javadoc
- GUIs
  - Layout
  - Event Handling

7

## Quick Review: Data-Structure Topics

- Searching and Sorting in Arrays
  - Binary Search
  - Quick Sort
  - Merge Sort
  - Insertion Sort
  - Heap Sort
  - Sorting Lower Bound
- Data Structures for Searching (Dictionaries)
  - Hash Table
  - Binary Search Tree
  - Balanced Trees
- Data Structures for Sequencing
  - Stack
  - Queue
  - Priority Queue
- Graphs (Adj List, Adj Matrix)
  - Shortest Paths
    - Breadth First Search
    - Dijkstra's Algorithm
  - Minimum Spanning Trees
    - Prim's Alg (single tree)
    - Kruskal's Alg (forest)

8

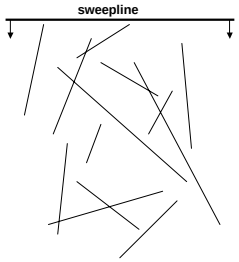
## Quick Review: Additional Topics

- Recursion
  - Recursive Descent Parsing
  - Stack Frames
  - Induction
- Algorithm Analysis
  - Big-O
  - Worst- vs. Expected-Case
- Algorithm Design Methods
  - Divide and Conquer
  - Greedy Method
- Union/Find
  - union-by-size
  - path compression
- The Java Collections Framework
  - Interfaces: Set, SortedSet, List, Map, SortedMap, Iterator
  - Classes: HashSet, TreeSet, ArrayList, LinkedList, HashMap, TreeMap
  - Utilities: java.util.Arrays, java.util.Collections
  - Comparator vs. Comparable

9

## What I Do: Computational Geometry

- Using a computer to solve geometric problems
  - Get to use lots of data structure ideas
  - Example
    - Given n line segments in the plane, report all intersections
    - Uses both a PQ and a Balanced Tree
- Areas I work in
  - Motion Planning
  - Meshing
  - Shape Matching
    - computer vision
    - protein matching
  - More theoretical questions




10