

For Monday

- Read Weiss, chapter 9, section 3
- Homework:
 - Chapter 5, exercise 19
 - Chapter 8, exercises 1-2

Program 3

- Any questions?

Research Paper

Equivalence Relations

- A relation is an equivalence relation if it is:
 - Reflexive
 - Symmetric
 - Transitive
- Examples?

Dynamic Equivalence

- Suppose we have a group of objects
- Suppose we also have an equivalence relation
- We have partial information about individual equivalence relations among the objects
- We want to set up a data structure that allows us to quickly determine whether a is equivalent to b given the information

Disjoint Sets

- Basic idea:
 - We have a set of N items which fall into equivalence classes.
 - We partition the set into N sets of one item each.
 - We have two operations: **union** and **find**
 - Union combines two sets, indicating that their members are equivalent
 - Find returns the label for the set an item belongs to, allowing us to determine equivalence

Assumptions

- Items are numbered 0 to $N-1$
- Can determine number immediately

Simple Approaches

- Use array of set names
 - How does find work?
 - How does union work?
 - Time?
- Keep a linked list of each set
- Add knowledge of number in each set

A Better Structure

- Still have an array to represent the sets.
- Represent each subset as a tree
- Value in the array in the parent link of the item
- Root nodes have a parent link of -1
- Name of the set is the index of the root node

Find

- Go to the correct index in the array
- Follow the parent links to the root

Union

- Make the root of one tree point to the root of the other

Better Unions

- By size
- By height

Path Compression

- Try to improve the height of our trees while changing time cost by a constant factor

Maze Generation

Graphs

- A **graph** is a collection of **vertices** (singular **vertex**) and **edges**
- Vertices are sometimes called **nodes**
- Vertices are usually named, edges are identified by the vertices they connect: **AB** is an edge connecting vertex **A** to vertex **B**

Graphs (cont)

- We will denote the number of vertices in a graph by **V** and the number of edges by **E**
- A **path** from vertex A to vertex B is a listing of the vertices touched as we traverse edges from A to B.
- If there is a path from every node to every other node in a graph, the graph is **connected**.

More Graphs

- A **cycle** is a path which begins and ends at the same vertex.
- A **simple path** is a path with no cycles in it.
- A graph with no cycles is called a **tree**.
- A **spanning tree** of a graph is a tree subgraph that contains all the vertices in the original graph.

Even More Graphs

- A graph with fewer than $V-1$ edges cannot be connected.
- A graph with V or more edges must contain a cycle.
- **Complete** graphs have all possible edges present.
- **Dense** graphs have most possible edges present.
- **Sparse** graphs have few of the possible edges present.

Types of Graphs

- Graphs with edges that can go either way are **undirected**.
- Graphs with edges that have particular direction are **directed**.
- Graphs may also have weights associated with their edges. Those are **weighted** graphs.
- Weighted graphs may be either directed or undirected.

Graph Representation

- Adjacency matrix
 - Graph is represented using an 2-d array
 - Indices of array represent nodes in the graph
 - Cells of the matrix represent edges
- Adjacency lists

Graph Applications

- ???

Graph Applications

- Parallel computing
- Scheduling
- Games
- Problem solving
- Mapquest
- Networks (of wires, pipes, etc.)
- Knowledge representation

Topological Sorting