Graphs

Introduction, Breadth First Search, Depth First Search, and Topological Sort

Cecily Heiner, PhD
Overview

• Motivating Toy Problem: Muddy City
• Motivations and Applications
  • Research/divergent thinking exercise
• Terminology
• Common Algorithms
• Where to Find Source Code
• Tips and Tricks for Tests and Interviews
• Example Problems
When I was an assistant professor at Harvard, Bill was a junior. My girlfriend back then said that I had told her: "There's this undergrad at school who is the smartest person I've ever met."

That semester, Bill was fascinated with a math problem called pancake sorting: How can you sort a list of numbers, say 3-4-2-1-5, by flipping prefixes of the list? You can flip the first two numbers to get 4-3-2-1-5, and the first four to finish it off: 1-2-3-4-5. Just two flips. But for a list of n numbers, nobody knew how to do it with fewer than 2n flips. Bill came to me with an idea for doing it with only 1.67n flips. We proved his algorithm correct, and we proved a lower bound—it cannot be done faster than 1.06n flips. We held the record in pancake sorting for decades. It was a silly problem back then, but it became important, because human chromosomes mutate this way.

Two years later, I called to tell him our paper had been accepted to a fine math journal. He sounded eminently disinterested. He had moved to Albuquerque, New Mexico to run a small company writing code for microprocessors, of all things. I remember thinking: "Such a brilliant kid. What a waste."

(http://awards.acm.org/info/papadimitriou_4558987.cfm)
Activity: Muddy City

Find the best route that connects all the houses, but uses as few counters (paving stones) as possible. (CS Unplugged)
Graphs

- Express relationships
  - Between pairs of items
- Have Nodes (circles)
- Have Edges (lines)
  - Edges may have costs (#'s)
  - Edges can be directed (->')
Research/Divergent Thinking Exercise

• Find as many applications where graphs are useful as possible
  • Theoretical applications count

• Must be able to identify
  • A problem people care about
  • What nodes represent
  • What edges represent

• Must write this down in chart format

• May work in groups

<table>
<thead>
<tr>
<th>Problem</th>
<th>Nodes</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Applications

- Transportation
- Communication
- Utility
- Social Networks
- Speech Processing
- Knowledge Representation
  - e.g. IBM Watson
- Document Link Graphs
  - e.g. Google Page Rank
- Protein Protein Interactions
- Scene Graphs
  - e.g. computer games
- Robot Planning
- Email Routing
- Brain Modelling
- Compilers (e.g. assigning variables to memory locations)
- Semantic networks (language processing)
- Processor scheduling
- Processor Design
- Theory
- Music (chord progressions)
- Course scheduling

- Many of these are from
  - Danny Sleator Lecture 9, 15-210, 2013
Graphs, Directed Graphs, Adjacency

• A *Graph*: consists of a set of vertices and a set of edges that connect the vertices

• A *Directed Graph*: is a graph with edges that are “ordered”

• *Adjacent* vertex w must have an edge leading from v to w
Paths, Unweighted Paths, Weighted Paths

• A *path* is a sequence of vertices connected by edges
• The *unweighted path length* is the number of edges on a path
• The *weighted path length* is the sum of the edge costs on a path
• The weight of the path could be
  • Distance
  • Time
  • Fuel cost
  • Etc.
Cycles, Directed Acyclic Graphs, Indegree

• A cycle in a directed graph is a path that begins and ends at the same vertex and contains at least one edge
• A directed acyclic graph has no cycles
• Indegree is the number of incoming edges
Dense Graphs, Sparse Graphs

• A complete graph has edges that connect every pair of vertices

• A dense graph has a large number of edges (usually quadratic or greater)

• A sparse graph has fewer edges (and is typically more common)
Representing Dense Graphs: Adjacency Matrix

- Great for dense graphs
- Quadratic space required ($O(|V|^2)$)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$\infty$</td>
<td>3</td>
<td>6</td>
<td>7</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>B</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>2</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>8</td>
</tr>
<tr>
<td>C</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>E</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>15</td>
<td>$\infty$</td>
<td>4</td>
</tr>
<tr>
<td>F</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>5</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
</tbody>
</table>
Representing Sparse Graphs: Adjacency List

• Great for sparse graphs
• Space required (O|E|)

A(0) → null
B(1) → 0(3) → null
C(2) → 0(6) → 1(2) → 5(5) → null
D(3) → 0(7) → 4(15) → null
E(4) → 2(1) → null
F(5) → 1(8) → 4(4) → null
Typical Graph Problem Input

B A 3
C A 6
D A 7
C B 2
F B 8
E C 1
D E 15
F E 4
C F 5
Representing Edges and Vertices in Java

• Edge class (e.g. 14.6)
  class Edge{
      public Vertex dest;
      public double cost;
      // costs are often probabilities
      // esp. in “real” problems
      public Edge(Vertex d, double c){
          dest=d;
          cost=c;
      }
  }

• Vertex (e.g. 14.7)
  class Vertex{
      public String name;
      public List<Edge> adj;
      public double dist;
      public Vertex prev;
      public int scratch;
      public Vertex (String n)
      {
          name=n; adj= new List<Edge>();reset();
          public reset()
          {
              dist=Graph.INFINITY; prev=null; scratch=0
          }
  }
Breadth First Search

• Pick a node (e.g. A)
  • Add it and all adjacent nodes to a QUEUE
• Round 1 (B D)
• Round 2 (D C E)
• Round 3 (C E G)
• Round 4 (E G F)
• Round 5 (G F H)
• Round 6+7 (H I)
Depth First Search

• Pick a node (e.g. A)
• Add adjacent vertices to the STACK
• Round 1 (B D)
• Round 2 (B E G)
• Round 3 (B E H)
• Round 4 (B E I)
Topological Sort

• Constraints
  • Acyclic graphs
  • Multiple correct solutions
• Find node with an indegree of 0
• Add that node to the topological sort, remove it and its edges from graph
• Repeat until graph is empty
Comparing Algorithms

• Breadth first search
  • Use a queue
  • Common in algorithms for shortest path
  • Easier to parallelize
    • Give each core/processor one branch

• Depth first search
  • Use a stack
  • Common in algorithms to find a single solution (e.g. maze), detecting cycles
  • Harder to parallelize
Comparing “Shortest Path” Problems

- **Unweighted graph**
  - Use a queue
  - Runtime $O(|E|)$

- **Positive weighted graph**
  - Dijkstra’s algorithm
  - Use a priority queue
  - Runtime $O(|E| \log |V|)$

- **Negative weighted graph**
  - Bellman-Ford Algorithm
  - Use a priority queue and scratch member
  - Must test for negative cycles
  - Runtime $O(|E| |V|)$

Very Restricted Graph
Low Runtime

Unrestricted Graph
High Runtime
Common Bugs

- Does the input graph satisfy conditions for the algorithm?
- Is the cost comparison function correct?
- For negative weighted graphs, is there a test for negative cycles?
Tips and Tricks for Interviews and Exams

Bloom’s Taxonomy

- Evaluation
- Synthesis
- Analysis
- Organization
- Recall
- Recognition

2 Great Commandments of CS

- CPU
- RAM

Time Complexity

Space Complexity
Scenario 1: Problem 14.22

A word can be changed to another work by a one-character substitution. Assume that a dictionary of five-letter words exists. Give an algorithm to determine whether a word A can be transformed to a word B by a series of one-character substitutions. If so, output the corresponding sequence of words. For example, bleed converts to blood by the sequence bleed, blend, blond, blood.
Scenario 2: Problem 14.25

A student needs to take a certain number of courses to graduate, and these courses have prerequisites that must be followed. Assume that all courses are offered every semester and that the student can take and unlimited number of courses. Given a list of courses and their prerequisites, compute a schedule that requires the minimum number of semesters.
Scenario 3: Problem 14.28

Suppose you have a graph in which vertex represents a computer and each edge represents a direct connection between 2 computers. Each edge \((v,w)\) has a weight \(p(v,w)\) representing the probability that a network transfer between \(v\) and \(w\) succeeds that is between 0 and 1. Write a program that finds the most reliable way to transfer data from a designated starting computer \(s\) to all other computers in the network. How would you modify the program to calculate the probability of the data successfully transferring?
Challenges

• Read the proofs in Chapter 14 (e.g. pg 548)
  • What assumptions/constraints?
  • What style of proof (contradiction/induction/etc.)?
  • How is Big O proven for various algorithms and data structures?
• Download the book’s code for graphs and trace algorithms for my toy graphs
  • How are nodes constructed?
  • Which paths are chosen?
• Write the code for a breadth-first search, depth-first search.
• Do your homework!!!