Compsci 201
Graphs, APTS, and More
Part 1 of 5

Susan Rodger
April 17, 2020
W is for …

• World Wide Web
  • Invented in 1989 – Sir Tim Berners Lee


• Wifi
  • We need this everyday
Announcements

• APT-7 due Thursday, April 16
• APT-8 due Tuesday, April 21
• Assignment P6 Huffman due April 22
  • All late work turned in by April 22 (APTs and Asgnns)
  • Except Huffman grace through April 23
• Assignment P7 Optional out – Extra Credit!

• Exam 2 will focus on grading that the next few days
• APT Quiz 2 is April 12-18 – Your own work!
• Final Exam will be on April 30 – any time on this day

• Discussion 13, Monday, April 20, Last one!
Plan for the Day

• Logistics to Wrap up the Course
  • Extensions must end, Final Exam

• APTs and Algorithmic Concepts
  • Reminder about greedy
  • Memoizing

• Graphs and Graph Algorithms
  • Concepts, terminology, APTs
  • Toward Dijkstra's Algorithm
Last Chance

• Extensions on APTs and Assignments
  • You must fill out the extension form
  • We can’t take anything late after April 22!
    • Except P6 Huffman
      – only one grace day April 23!
      – NO LATE SUBMISSIONS!!!!!!
  • Except P7 Create
    – til Sunday April 26 with no penalty
Calculating your grade

• Discussion sections (6% of your grade)
  • Add up your total points, max points are 13
disc*4pt = 52
  • We drop 8 points
  • Divide your total points by 44.
• Examples:
  • 42pts/44 = 95%
  • 50pts/44 = 100% (can’t get > 100)
Calculating your grade (part 2)

• Programming and Analysis (23% of your grade)
  • Max points are: 178pts
  • Divide your total by 178 for your score
• WOTOS (3.75% of your grade)
  • If you have 40% of the points – you get 100%
• Reading Zybooks or 6 extra APTs (1.25% of grade)
• APTs (6%)
  • 38 APTs, 10 pts each 380+ points is 100%
Calculating your grade (part 3)

• 90 A-, 94 A

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Discussion Sections</td>
<td>6%</td>
</tr>
<tr>
<td>Programming and analysis</td>
<td>23%</td>
</tr>
<tr>
<td>assignments</td>
<td></td>
</tr>
<tr>
<td>WOTOs (75%)/Reading (25%)</td>
<td>5%</td>
</tr>
<tr>
<td>APTs</td>
<td>6%</td>
</tr>
<tr>
<td>APT Quizzes (2)</td>
<td>10%</td>
</tr>
<tr>
<td>Exam grade: Exam1, Exam2 and</td>
<td>50%</td>
</tr>
<tr>
<td>Final Exam</td>
<td></td>
</tr>
</tbody>
</table>
Final Exam

- Final exam – must take on Thursday, April 30
- Must take in a 24 hour window.
- Once you start - have 3 hours plus 1 hour
  - That is 4 hours total
- About a 2 hour exam.
- Similar format to Exam 2
- If you don’t take it get a 0.
- Exam grade is MAX of (exam1, exam2, final exam)
P7 Create assignments coming in ...
P7 Create assignments coming in ...
Be A UTA
https://www.cs.duke.edu/undergrad/uta

• SIGN UP NOW by April 29 or before…
• CompSci 201
• CompSci 101
  • Python
• CompSci 94
  • Programming with Alice, so easy to learn
CS Concepts Coming Alive

• What data structure is this?
YARN, in the shape of a binary tree. Subtrees made with molecule kit. What is it?
YARN, in the shape of a binary tree
Subtrees made with molecule kit
What is it?
2D-range tree

• Search in x-y plane
• Main tree organized by x-values
• Subtree organized by y values
2D-range tree

- Search in x-y plane
- Main tree organized by x-values
- Subtree organized by y values
Binary Search tree of points in the plane – sorted by X-value
Binary Search tree of points in the plane – sorted by X-value

In the x-range

Search each subtree by y-value

Each subtree organized by y-value
Compsci 201
Graphs, APTS, and More
Part 2 of 5

Susan Rodger
April 17, 2020
On to Graphs: From Bacon to Erdös

Bacon Number and Erdös Number

• Some actors are prolific: lots of movies
  • Chris in movie with Sam in movie with K. Bacon
    • Chris has a Bacon number of two

• Some authors are prolific: lots of papers/articles
  • Tina wrote paper with Tom wrote with P. Erdös
    • Tina has an Erdös number of two

• Graph terminology: connecting nodes with edges
  • In-movie-with or wrote-paper-with is edge
Erdös Numbers

- Authors connected by authorship/paper-writing

Erdös Numbers

• Authors connected by authorship/paper-writing
Erdös Numbers

• Authors connected by authorship/paper-writing

My Erdos number is 3!
Bacon Number

• Actors connected by acting/movie-roles

https://oracleofbacon.org/
Obama in Movie
Widely Used Mobile Apps

- Google Maps, Uber, Lyft
  - Why is UI important for drivers
  - Why shortest path algorithms important
  - Which use graphs?
Why Graphs are Important

• Google/Pagerank models webpages as a graph
  • Nodes are webpages
  • Links are hyperlinks between pages
  • Weights based on “importance” of link/page
• https://en.wikipedia.org/wiki/PageRank
Is the Internet a Graph? It depends …

- Internet as graph
  - Nodes are anything with an IP address (IP)
  - Nodes are Autonomous Systems (AS)
  - Edges connect Thermostat to Website or …
The Coronavirus graph you don’t see

• Who infected who
  • Nodes – people
  • Edges – person A infected person B
  • Need a lot of testing to make this graph

You can see the growth is exponential, HUGE!
Graphs

- Graphs are collections of vertices and edges
  - Vertices or nodes, edges or links
  - Undirected graph Tom-Kevin and Meg-Kevin
  - Sometimes edges have weights
Directed (weighted) Graph

- Edges can have direction: directed graph
  - Not Facebook. Yes Tinder?
Data Structures for Graphs

- Use number for vertex, index into array
  - Can use string and map as well
- Adjacency List Representation
  - Good for sparse graphs

http://lagodiuk.github.io/computer_science/2016/12/19/efficient_adjacency_lists_in_c.html
Adjacency Matrix

• Good for dense graphs, vertices still numbers
  • Symmetric matrix if undirected
  • Can have weights instead of 0,1

https://www.oreilly.com/library/view/php-7-data/9781788286463890/32fd15e8-423f-49aa-84c2-db1518023299.xhtml
Theory and Practice

- Code is often simpler with Adjacency "list"
  - `Map<String, Set<String>>` for "list"
  - Vertex identified by String
  - Connected-by-edge? set of vertices
  - Need something more for weighted graphs

- For APTs, this is a good approach as we'll see
  - Simple to make, simple to use, scaling? meh
Compsci 201
Graphs, APTS, and More
Part 3 of 5

Susan Rodger
April 17, 2020
Simple Graph Algorithms

- What vertices are reachable from starting vertex?
  - Can use DFS or BFS to find connected vertices
  - Must avoid visiting same vertex more than once
- Find connected components
  - Many applications

Breadth First Search

```java
public Set<String> bfs(String start){
    Set<String> visited = new TreeSet<>();
    Queue<String> qu = new LinkedList<>();
    visited.add(start);
    qu.add(start);

    while (qu.size() > 0){
        String v = qu.remove();
        for(String adj : myGraph.getAdjacent(v)){
            if (! visited.contains(adj)) {
                visited.add(adj);
                qu.add(adj);
            }
        }
    }
    return visited;
}
```
BFS Example

- BFS at A: B, C, D
  - from B: E, from C: …, from D: F
  - from E: …, from F, …

- BFS: all one-away
  - then all two, then all three, …
BFS Example

- BFS at A: B, C, D
  - from B: E, from C: ..., from D: F
  - from E: ..., from F

QU: A

- BFS: all one-away
  - then all two, then all three, ...
BFS Example

- BFS at A: B, C, D
  - from B: E, from C: …, from D: F
  - from E: …, from F

QU: B, C, D

- BFS: all one-away
  - then all two, then all three, …
BFS Example

- BFS at A: B, C, D
  - from B: E, from C: …, from D: F
  - from E: …, from F

QU: C, D, E

- BFS: all one-away
  - then all two, then all three, …
BFS Example

• BFS at A: B, C, D
  • from B: E, from C: …, from D: F
  • from E: …, from F

QU: D, E

• BFS: all one-away
  • then all two, then all three, …
BFS Example

- BFS at A: B, C, D
  - from B: E, from C: …, from D: F
  - from E: …, from F

QU: E, F

- BFS: all one-away
  - then all two, then all three, …
BFS Example

• BFS at A: B, C, D
  • from B: E, from C: …, from D: F
  • from E: …, from F

QU: F

• BFS: all one-away
  • then all two, then all three, …
BFS Example

- BFS at A: B, C, D
  - from B: E, from C: …, from D: F
  - from E: …, from F

QU:

- BFS: all one-away
  - then all two, then all three, …
BFS Example

• BFS at H:

• BFS: all one-away
  • then all two, then all three, …

Starting with H, Find this connected component!
public Set<String> dfs(String start){
    Set<String> visited = new TreeSet<>();
    Queue<String> qu = new LinkedList<>();
    visited.add(start);
    qu.add(start);

    while (qu.size() > 0){
        String v = qu.remove();
        for(String adj : myGraph.getAdjacent(v)){
            if (! visited.contains(adj)) {
                visited.add(adj);
                qu.add(adj);
            }
        }
    }
    return visited;
}
public Set<String> dfs(String start) {
    Set<String> visited = new TreeSet<>();
    Stack<String> qu = new Stack<>();
    visited.add(start);
    qu.push(start);

    while (qu.size() > 0) {
        String v = qu.pop();
        for (String adj : myGraph.getAdjacent(v)) {
            if (! visited.contains(adj)) {
                visited.add(adj);
                qu.push(adj);
            }
        }
    }
    return visited;
}
DFS Example

• DFS at A: B, C, D
  • then F, E

• DFS: goes deep one at a time
DFS Example

- DFS at A: B, C, D
  - then F, E

- DFS: goes deep one at a time
DFS Example

- DFS at A: B, C, D
  - then F, E

**Stack**
**QU: B**

- DFS: goes deep one at a time
DFS Example

- DFS at A: B, C, D
  - then F, E

Stack
QU: C, B

- DFS: goes deep one at a time
DFS Example

- DFS at A: B, C, D
  - then F, E

Stack
QU: D, C, B

- DFS: goes deep one at a time
DFS Example

- DFS at A: B, C, D
  - then F, E

Stack
QU: F, C, B

- DFS: goes deep one at a time
DFS Example

- DFS at A: B, C, D
  - then F, E

Stack
QU: C, B

- DFS: goes deep one at a time
DFS Example

- DFS at A: B, C, D
  - then F, E

Stack
QU: E, B

- DFS: goes deep one at a time
DFS Example

• DFS at A: B, C, D
  • then F, E

Stack
QU: B

• DFS: goes deep one at a time
DFS Example

- DFS at A: B, C, D
  - then F, E

Stack
QU:

- DFS: goes deep one at a time

Starting with A, Find this connected component!
DFS Example

• DFS at G:

• DFS: goes deep one at a time

Starting with G, Find this connected component!
Example: Word Ladder Problem

• Change a word into another word
  • Change one letter at a time
  • Change COLD to WARM

COLD -> CORD -> **WORD** -> WORM -> WARM

![Word Ladder Diagram]
Word Ladder Problem

• Change a word into another word
  • Change one letter at a time
  • Change COLD to WARM

COLD -> CORD -> WORD -> WORM -> WARM
• **BFS + Graphs = Word Ladder or Bacon Number**
  • Getting from “above” to “zeros” in 17 steps!
  • above abode anode anole anile anise arise prise prime prims prams prats peats heats heads herds heros zeros

• These edge weights are 1, so BFS works

• **We can find the shortest path efficiently**
  • Dijkstra's algorithm used in Internet today
  • Heuristics augment, absolute shortest needed?
Shortest Path and Longest Path

- We use breadth first search to find shortest path
  - Same code we saw in word-ladder problem
    - White, While, Whale, Shale, … House
    - Efficient and polynomial time: edge-weight == 1
    - Need Dijkstra for positive edge-weight, still good

- No efficient algorithm for longest path, it's hard
  - If one found, every hard problem becomes easy
    - Most computer scientists don't think we'll find one
Connected Components: APT

• [https://www2.cs.duke.edu/csed/newapt/internet.html](https://www2.cs.duke.edu/csed/newapt/internet.html)

• Given a graph, a set of connected vertices
  • Which are important aka articulation points
  • Remove one? disconnect graph

• In example, removing 2 means …
  • Disconnect 3 from 0 and 1
Connected Components: APT

- [https://www2.cs.duke.edu/cseds/newapt/internet.html](https://www2.cs.duke.edu/cseds/newapt/internet.html)

- What is this problem asking you to do?
  - What router, if removed, disconnects others?

- This is a graph problem! Vertices and edges?
  - Parse input, build graph, traverse graph

- Adjacency List: `Map<String, Set<String>>`
  - `map.get("2")` -- set of connected vertices
Toward All Green

• What part of this haven't you seen?
• How is DFS or BFS used? Modify based on ...

```java
public int articulationPoints(String[] routers) {
    makeGraph(routers);
    int total = 0;
    for(int k=0; k < routers.length; k++) {
        String vertex = ""+k;
        String start = "0";
        if (k == 0) start = "1";
        Set<String> set = reachFromSkip(start,vertex);
        if (set.size() != routers.length-1) {
            total += 1;
        }
    }
    return total;
}
```
What is reachFromSkip method?

• Use BFS or DFS as provided, but …
  • Do not push or enqueue skippable vertex/node
  • Can we reach everything from start? good!
    • Start from "0" unless skipping "0", …

• Must create graph from input

```java
for(String s : adj) {
    myGraph.putIfAbsent(s, new TreeSet<>());
    myGraph.get(vertex).add(s);
    myGraph.get(s).add(vertex);
}
```
WOTO (4 minutes)

Jon Kleinberg

• Developed HITS, same time-frame as PageRank
• Professor at Cornell University
• MacArthur Genius award, Nevanlinna Prize, more

"It's much easier to make progress on a problem when you are enjoying what you are doing. In addition to finding work that is important, find work that has some personal interest for you....

[ACM Infosys Interview]
Compsci 201
Graphs, APTS, and More
Part 4 of 5

Susan Rodger
April 17, 2020
Greedy Algorithms

• Which candles to burn?
  • The tallest ones: leads to more burning days

• Which votes to steal?
  • Opponent with the most: fewer "steals" to win

• Which weighted nodes to join in Huffman coding?
  • Smallest weights first: save bits, optimal!
A friend of a friend: APT

- [https://www2.cs.duke.edu/csed/newapt/friendscore.html](https://www2.cs.duke.edu/csed/newapt/friendscore.html)
- Model as a graph? Vertex: number, Edge? == 'Y'
  - 0: has one friend: 1
  - 1: 0 and 2
  - 2: 1 and 3
  - 3: 2 and 4
  - 4: 3

- So 2 has four two-friends
  - 1 has three two-friends
  - 0 has two two-friends

\[ \{ "NYNNN" , "YNYNN" , "NYNYN" , "NNYNY" , "NNNYN" \} \]

Returns: 4
A friend of a friend: APT

- [link]: https://www2.cs.duke.edu/csed/newapt/friendscore.html
- Model as a graph? Vertex: number, Edge? == 'Y'
  - 0: has one friend: 1
  - 1: 0 and 2
  - 2: 1 and 3
  - 3: 2 and 4
  - 4: 3

So 2 has four two-friends
  - 1 has three two-friends
  - 0 has two two-friends

Returns: 4
General Framework to Solve

• How to write `twoFriends`?
  • Make graph, find two-friends via …
  • Find 1 friends? index of each 'Y'. Repeat

```java
Map<Integer, Set<Integer>> myGraph;

public int highestScore(String[] friends) {
    makeGraph(friends);
    int max = 0;
    for (int k = 0; k < friends.length; k++) {
        Set<Integer> set = twoFriends(k);
        max = Math.max(set.size(), max);
    }
    return max;
}
```
Set.addAll --- all my friends

- Model data using graph: parse via makeGraph
  ```java
  Map<Integer, Set<Integer>> myGraph;
  ```

- My friends: `myGraph.get(my_number)`
  - Friend of a friend? for each of my friends ...
  ```java
  for (int friend : myGraph.get(my_index)) {
    set.addAll(myGraph.get(friend));
  }
  ```
Mathematics and Computer Science

- How do we solve differential equations?
  - It depends
- How do we estimate percolation threshold?
  - It depends
- How do we model cardiac behavior? …

- Use simulation when no analytic solutions
  - Monte Carlo simulation for many problems
  - [https://en.wikipedia.org/wiki/Monte_Carlo_method](https://en.wikipedia.org/wiki/Monte_Carlo_method)
Thinking about math+compsci

[Link: https://www2.cs.duke.edu/csed/newapt/bstcount.html]

• How many different binary search trees are there?
  • Size = 4, Size = 5 … Size = N?
  • What about N = 6?

<table>
<thead>
<tr>
<th>N</th>
<th># trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>42</td>
</tr>
</tbody>
</table>
Thinking about math+compsci

https://www2.cs.duke.edu/csed/newapt/bstcount.html

- How many different binary search trees are there?
  - Size = 4, Size = 5 … Size = N?
  - What about N = 6?

<table>
<thead>
<tr>
<th>N</th>
<th># trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>42</td>
</tr>
</tbody>
</table>
Thinking about math+compsci

https://www2.cs.duke.edu/csed/newapt/bstcount.html

• How many different binary search trees are there?
  • Size = 4, Size = 5 … Size = N?
  • What about N = 6?

<table>
<thead>
<tr>
<th>N</th>
<th># trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>42</td>
</tr>
</tbody>
</table>
Combinatorics and Catalan

- Binary search trees with 6 nodes
  - Left subtree has: 0, 1, 2, 3, 4, 5 nodes
    - What will right subtree have?
  - For each left, there is a right...
    - Count how many ways this happens

\[(1 \times 42) + (1 \times 14) + (2 \times 5) + (5 \times 2) + (14 \times 1) + (42 \times 1) = 132\]

Aside: From Catalan to Fibonacci

• Read about the Golden Ratio and Fibonacci #'s
  • 1, 1, 2, 3, 5, 8, 13, 21, … it's about rabbits?
  • Inevitable we discuss this, factorial, Bubble sort

• Do not do this at home, see classwork on Git
  https://coursework.cs.duke.edu/201spring20/classcode/

```java
15 public static long rfibo(int n) {
16     if (n <= 2) return 1;
17     return rfibo(n-1) + rfibo(n-2);
18 }
```
Exponential number of calls

• Since fib(8) calls fib(7) and fib(6)
  • And fib(6) calls … which calls … which …
  • What is the recurrence? \( T(n) = 2T(n-1) + O(1) \)
  • Solution to this is \( O(2^n) \)

• Actual fib isn't \( 2^n \), is exponential
  • Golden ratio: \( \varphi^n \)

\[
\lim_{n \to \infty} \frac{F_{n+1}}{F_n} = \varphi
\]
Memoize aka Caching

- Caching in computer science is … store to re-use
  - Similar to dynamic programming, but top-down
- If already seen? use that result, no recursion
  - Otherwise, recurse, store, return

```
static long[] memo = new long[5000];
public static long rfib(int n) {
    if (n <= 2) return 1;
    if (memo[n] != 0) {
        return memo[n];
    }
    memo[n] = rfib(n-1) + rfib(n-2);
    return memo[n];
}
```
Look at this tree again

• Instead of doing this.....
Look at this tree again

• We do this…
Avoid repeated recursion ...

- Store calculated values in a map
  - Look up first, re-use what's already done
  - Use `Map<Integer, Long>` or `long[]`
- An array is a map of index to value
All Green? Which one ...

- This solution will time out, too many helper calls
- Use memoization to get all green
- Add array or map, store, re-use

```java
public long helper(int n) {
    if (n == 0 || n == 1) return 1;
    long total = 0;
    for(int leftCount = 0; leftCount < n; leftCount++) {
        total += helper(leftCount)*helper(n-leftCount-1);
    }
    return total;
}
```
All Green? Do NOT turn this in

• Catalan via Wikipedia: this should NOT be used.
  • Notice 6564120420L, long constant

```java
    public long howMany(int[] values) {
        long[] catalan = {
            1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420L,
            24466267020L, 91482563640L, 343059613650L, 12899041473
        };
        return catalan[values.length];
    }
```
WOTO