Prof. Alexandra Nilles

Western Washington University

13 November

Announcements

- Still have a few people taking mid-term, so mid-term exam review will be Friday
- No labs this week
- We will have labs Thanksgiving week: Thursday section can attend Monday morning or DIY

More Information on Graphs

- Graphs in day-to-day life
  - Human brain: 100 billion neurons, 100 trillion synaptic connections (approx 1k-5k connections per neuron)
  - ChatGPT: 176 billion neurons
    - 85,000 nodes in layer connected to all 85,000 nodes in next layer, quadrillion connections? $10 * *16$
  - C. Elegans: 302 neurons, 7000 connections
  - Internet:
    - 2.25 billion indexed pages
    - ~1 billion websites
    - 13 million web-facing computers
  - Cryptocurrency
    - Bitcoin network has 61k nodes
    - Ethereum has ~6k
    - Dogecoin has ~1k

Modifying graph data structures (Dynamic Graphs)

Adding / Removing Nodes:

- Adjacency list
  - Adjacency list is general concept, several ways to implement
  - hash table: node object -> array of neighbors
  - Adding node:
    - O(1) to create new entry in data structure
    - Rest of complexity depends on number of neighbors and directedness: O(1) to O(N)
    - Use O(k) such that k is degree of node
    - For each neighbor, add entry to its neighbor array (undirected case)
  - Case with edge data: Create list of node objects, each has a collection of incident edge objects
    - each edge object has two incident nodes
- Matrix Representation
  - Need to add *row* and *column*: complexity depends on representation
  - List of linked lists: O(1) + complexity of connectivity
  - Array of arrays: O($N * *2$) to copy into larger data structure
  - Deleting is similar; could also "hack" by setting entire row/column to zero and ignoring disconnected nodes

Adding / Removing Edges:

- Adjacency list
  - O(k) to delete two neighbors
- Adjacency Matrix

  - O(1) to set two elements to zero

Compressed Matrices as Graph Data Structures:

- Dictionary of Keys
  - DOK: map (row, column) -> value
  - each entry encodes an *edge* between two nodes
  - hard to store node data: can maintain separate O(N) list of nodes
  - For graph with no edge or node data, we can encode connectivity information as binary sparse matrix
  - O(1) addition of edge / node
  - O(1) deletion of edge / node
- Compressed Sparse Row Format
  - Maintain two (or three) arrays:
    - Offsets: Offsets[i] stores offset where vertex i's edges start in Edges
    - Edges: Node IDs of neighbors
    - Values: store data for nodes / edges

Example:

Offsets: [0, 2, 5, 5, 10 . . .] Edges: [2, 13, 0, 2, 3, 15, 16, 17, . . .]

Adding and deleting edges and nodes becomes O(N + M) because both arrays need to be re-made!

Project Information!!

Basic information:

- You will be modifying and adding to a terminal-based text adventure game
- Individual project
- Code base will be released Friday, due December 10 (final deadline, gives me one week to grade)
- Base game contains:
  - Code for interacting with user and getting next action (choice from discrete set)
  - Three "rooms"
  - A half-dozen "items"
  - Code for player inventory and item interactions
  - One "ending"
- "Rooms" contain:
  - Items
  - Doors
  - Actions (look around, drop item, etc)
  - Rooms are defined in YAML file; load from file when initializing game
- "Items":
  - There are sub-classes of items: Weapons, Healing, Keys, Books, etc
  - different items can be interacted with in different ways, all items can be inspected and picked up / dropped
  - Specific items are defined in YAML file

Your assignment:

- Add a new subclass of items with at least three instantiations
- Add at least two new rooms
- Add an ending related to at least one of your new rooms
- Create an "AI" game player that exhaustively searches all actions in your game to make sure it is complete-able no matter what actions the user chooses
- Extra credit: Visualize "game tree"

15 November 2024

Exam Review

1. Radix sort: how many outer iterations?
2. Stable sorts: radix, insertion, merge
3. Worst case merge sort: nlogn
4. Worst case quick sort: n^2
5. address book with large number of entries, looking up by last name
   - gave credit for either hash table or BST
6. Maximum balance factor in BST
   - Height in tree: number of edges on *longest path* from node to a leaf
   - Depth in tree: number of edges on path from root to node
   - Maximum balance factor for a node in a binary search tree with N nodes?
     - one subtree has height zero
     - other subtree has height (N-1)
     - will give 0.5 points for being off by one
7. asymptotic runtime of $f(N) = N(3N + lg(N))$
   - $N^2 > NlgN$
8. asymptotic runtime of $f(N) = 120(3N + lg(N + N^2))$
   - $f(N) = 360N + lg(N + N^2)$
   - as N becomes large this becomes $N + lg(N^2) \rightarrow N + 2lg(N) \rightarrow N$
   - half credit given for lgN because I also got this wrong ... but very important to remember that N > lg (N) as N becomes large!!
9. Consider the function. What is number of additions for myFunction(3)? What is big-O runtime?

```
public static int myFunction (int n) {
    if (n <= 1) return 1;
    int i = 0;
    while (i < n)
        i++;
    return myFunction(n/2) + myFunction(n/2);
}
```

Breadth-First Search (BFS)

- general technique for traversing entire graph
- returns nodes level by level, in order of distance from start
- search in general is defined by start node
- Example on board