CS 209
Data Structures and Mathematical Foundations

04 / 24 / 2024

Instructor: Michael Eckmann

Today's Topics

- Questions?/Comments?
- Balanced Binary Search Trees
- Start Graphs

Balanced BSTs

- Recall the definition of an AVL tree (which is one type of Balanced BST)

- An AVL tree is a BST with the added restriction that
  - for all nodes, the height of its left subtree is at most 1 different than the height of its right subtree. The height of an empty subtree is -1.
  - A node in an AVL tree contains
    - the data item,
    - reference to left child node
    - reference to right child node
    - reference to parent node
    - height of the node

Efficiency

- Let's look at the tallest AVL tree for some number of nodes.
- AVL trees cause the height of the tree to be no more than 1.44*log(n)
- So, search is O(log(n))
- Insert initially then also takes at worst log(n) and then we need to go up the tree setting heights and verifying heights of LST vs. RST. If we go all the way up the tree that is another log(n) amount of work. Any rebalancing that is necessary takes a constant amount of time.
- So the work is at most: log n + log n + c
- So, insert is also O(log(n))

Other Balanced Trees

- Red/Black trees are another form of balanced binary search tree, that we will not discuss.
- There are others too.

Graphs

- Graphs consist of a set of vertices and a set of edges.
- An *edge* connects two *vertices*.
- Edges can be *directed* or *undirected*.
- Directed graphs' edges are all directed. Undirected graphs' edges are all undirected.
- Directed graphs are sometimes called *digraphs*.
- Two vertices are *adjacent* if an edge connects them.
- The *degree* of a vertex is the number of edges starting at the vertex.
- Two vertices v1 and v2 are on a *path* if there are a list of vertices starting at v1 and ending at v2 where each consecutive pair of vertices is adjacent.

Graphs

- The *length of a path* is the number of edges in the path.
- A *simple path* is one whose edges are all unique.
- A *cycle* is a simple path, starting and ending at the same vertex.
- A vertex is *reachable* from another vertex if there is a path between them.
- A graph is *connected* if all pairs of vertices in the graph have a path between them. Example on the board of a connected and an unconnected graph.
- A *complete graph* (aka *fully connected* graph) is a connected graph where all pairs of vertices in the graph are adjacent. Example on the board.

Graphs

- Degree of a vertex in a digraph
  - *In-degree* of a vertex is the number of edges entering the vertex
  - *Out-degree* of a vertex is the number of edges leaving the vertex

Graphs

- Edges often have a *weight* associated with them.
- An edge's weight is some numeric value.

Graphs

- Trees (assuming the nodes are vertices and the edges are undirected) are graphs with restrictions.

Graphs

- Trees (assuming the nodes are vertices and the edges are undirected) are graphs with restrictions.
- They are connected graphs without cycles.

Graphs

- Examples of using graphs to represent real world entities
  - A weighted graph that connect cities (vertices) and the weights of the edges might be length of time to travel between the two cities by car.
  - A graph whose vertices represent airports and directed edges exist if there is an at least daily flight between those airports.
  - A graph whose vertices represent airports and directed, weighted edges exist for flights and their costs in dollars.
  - History of computer programming languages (directed edge from a language that influenced a later language)

Graphs

- Examples of using graphs to represent real world entities
  - How about graphs of social networks?

Graphs

- Graph's Edges can be represented in programs in several ways. Two common ways to represent them are:
  - adjacency matrix --- each row number r represents a vertex and the value at column c is true if there's an edge from vertex r to vertex c, false otherwise
    - Notice that the type stored in the adjacency matrix is boolean
    - Could we use this for weighted graphs?
    - Could we use this for directed graphs?
  - edge lists
    which is a list of linked lists
    - store a linked list for each vertex, v_i
    - items in the list are those vertices v_j for which there's an edge from v_i to v_j

Graphs

- If say we have an unweighted graph and directed edges represent flights. We would typically want to find paths that are shortest (in terms of length) rather than longest.
- If cost or distance or time or other similar quantities are stored as edge weights, we would typically want to find shortest paths (with the weights summed up) rather than highest cost paths.
- Hence, people have developed various shortest path algorithms for graphs