CS280 – Graphs
March 21, 2016

http://azrael.digipen.edu/~mmead/www/Courses/CS280/Graphs-1.html

Homework Questions

- Hash tableis doing one less expansion than it should be in the stress test
    - Make sure and use the example code *std::ciel* from the handout

Hash Tables - Wrapup

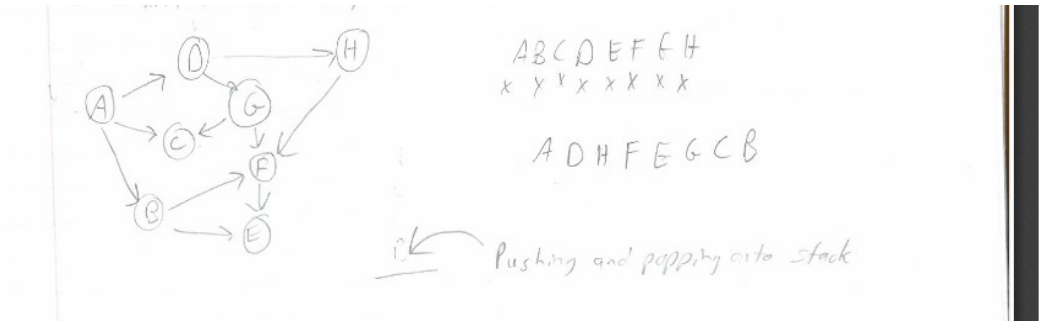
- Always keep in mind: **SPACE v. SPEED**
- Two methods of probing
    - Linear
    - Chaining
- Size of the lists in chaining are ideally small, so it is best to just use a singly linked list
    - You might think, why not use trees?!
        - May as well just grow the hash table instead of making lists any more complex than singly linked list
- In real life you usually copy from hash table into another data structure, such as a tree, when you need to read and manipulate the data where order is important
    - Drawback to hash tables is that you just have the key, which may or may not have significant real-world value to the user
    - E.g. Using SSN as key, but wanting to list users by last name
    - Point of hash tables is random, even distribution!
- Hash function generators don't necessarily have to use the entire key
- Hash table v. Array

- Hash table lets you avoid making a data structure that is the entire size of the largest element, which an array would force you to do

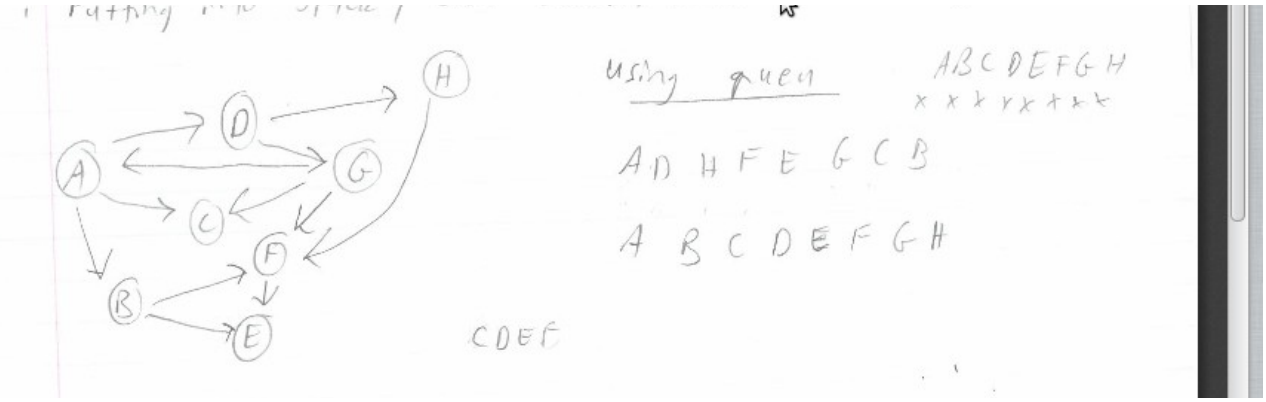
Graphs

- Trees are a simplified form of a graph
- Comprised of vertices and edges
- Arrowheads make it a 'directed graph'
  - No arrowhead means travel is bi-directional between vertices
- Can assign weights to each edge
  - Think cost to get from A to B – Money, time, distance, etc...
- No 'root' to a graph, start an an arbitrary vertice
- *{V1, V2}* - Bi-directional set
- *(V1, V2)* – Directed set
- You can have bidirectional edges with different weights
  - Upstream v. Downstream
- You start at one node and can only see your neighbors
  - Think fog of war in Civ
- We can use anything we want from the STL for the graph assignment
  - Priority queu for edges
- Strongly connected
  - Can visit every node in the graph by starting at any node
- Weakly connected
  - Picking an arbitrary node may cause you not to be able to visit every node in the graph
- Have to keep track of visited nodes

- Different 'degree' for incoming and outgoing edges
- In a directed graph, you have to walk the whole graph to determine the incoming degree
- We will be using stacks and queus for traversal
- Putting into stack, order doesn't matter if unweighted



Using a stack to traverse the graph



Using a queue to traverse the graph