CSC 220 Data Structures

Graphs I – Introduction
Parkland College Fall 2016
20161116
Graphs

• nodes connected by arbitrary edges
• (list of terms in textbook/webnotes)

• Examples
  • social networks
  • OO class hierarchy
  • street maps
  • services networks
  • airline flight hubs
  • http://www.visualcomplexity.com/
Graph Theorems

1. Given a graph $G$ with $e$ edges and vertex set $V$, the sum of the degrees is $2e$.

2. If $G$ is a directed graph, the sum of in-degrees equals the sum of the out-degrees, which is $e$.

3. $G$ is a simple graph with $n$ vertices and $e$ edges
   • If $G$ is undirected: $e \leq n(n-1)/2$
   • If $G$ is directed: $e \leq n(n-1)$

4. $G$ is undirected, $n$ nodes, $e$ arcs
   • if $G$ is connected: $e \geq n - 1$
   • if $G$ is a tree: $e = n - 1$
   • if $G$ is a forest: $e \leq n - 1$
Graph Implementations

• We’ll look at 4 common, more are possible
• Which to use depends upon situation
  • Different tradeoffs and advantages to each
• (Graphs are the most common data structure which I see implemented from scratch in work situations. They’re just too general to have a one-size-fits-all solution.)
Edge Lists

- two separate lists, a vertex list and an edge list
Adjacency List

• a list of vertices, each vertex has an edge list
Adjacency Map

- list of vertices, each vertex has a map of edges and distal endpoints
Adjacency Matrix

- vertex-by-vertex array, entries are edges
Graph Implementation
Complexities

- Big table in textbook/webnotes
- Edge lists:
  - lots of $O(e)$ because of linear searches over edges
- Adjacency lists:
  - `get_edge(u,v)` depends on smaller degree of $u$, $v$
- Adjacency maps:
  - some expected $O(1)$ because of dictionary lookups
- Adjacency matrix:
  - $O(1)$ for lookups, changing edges
  - really bad for inserting/removing vertices!
Final Exam Study Prep

• There will be multiple questions that rely on being able to interpret that chart for different situations
  • how does changing number of vertices affect choice of implementation?
  • how does changing edges affect choice of implementation?
  • how much space is taken up by one implementation versus another based on number of verts/edges?
Graph ADT (Abstract Data Type)

- `graph.py`
- based on adjacency map
  - adj map is a decent default until you have more knowledge about the exact way the application is going to use the graph
- dictionary of vertices, each has a dictionary of incidences
- undirected by default
- directed use separate dicts for in-, out-bound edges