CSE 100
Graph Traversals
Announcements

• Congratulations!! You are half way through 😊
• PA2 deadline tonight at 11 pm
• PA3 will be out tomorrow (Sat).
• Expect grades for midterm in a week

How was the midterm?
Kinds of Data Structures

Unstructured structures (sets)

Sequential, linear structures (arrays, linked lists)

Hierarchical structures (trees)

Graphs
Consist of:
• A collection of elements ("nodes" or "vertices")
• A set of connections ("edges" or "links" or "arcs") between pairs of nodes.
  • Edges may be directed or undirected
  • Edges may have weight associated with them

Graphs are not hierarchical or sequential, no requirements for a "root" or "parent/child" relationships between nodes
Kinds of Data Structures

- Unstructured structures (sets)
- Sequential, linear structures (arrays, linked lists)
- Hierarchical structures (trees)

**Graphs**

A. They consist of both vertices and edges
B. They do NOT have an inherent order
C. Edges may be weighed or unweighted
D. Edges may be directed or undirected
E. They may contain cycles
Kinds of Data Structures

Unstructured structures (sets)

Sequential, linear structures (arrays, linked lists)

Hierarchical structures (trees)

Graphs
Which of the following is true?
A. A graph can always be represented as a tree
B. A tree can always be represented as a graph
C. Both A and B
D. Neither A or B
Kinds of Data Structures

Unstructured structures (sets)

Sequential, linear structures (arrays, linked lists)

Hierarchical structures (trees)

Graphs
Which of the following is true?
A. A graph can always be represented as a tree
B. A tree can always be represented as a graph
C. Both A and B
D. Neither A or B

Note that trees are special cases of graphs; lists are special cases of trees.
Why Graphs?

Map well to real world systems:
- Computer networks
  - Road networks
  - Social networks
Why Graphs?

- Graphs provide a useful abstraction for many important problems
  - the set of machines on the internet, and network lines between them, form a graph
  - the set of statements in a program, and flow of control between them, form a graph
  - the set of web pages in the world, and HREF links between them, form a graph
  - the set of transistors on a chip, and wires between them, form a graph
  - the set of possible base sequences in a DNA gene, and mutations between them, form a graph
  - the set of possible situations that can arise in solving a problem or playing a game,
  - and moves that get you from one situation to another, form a graph
  - et cetera...
Remember: If your problem maps to a well-known graph problem, it usually means you can solve it blazingly fast!

By fast we usually mean $O(|V| + |E|)$ linear in the number of vertices and edges.
A directed graph

\[ G(V, E) \]

\[ V = \{ v_0, v_1, \ldots, v_5 \} \]

\[ |V| = 6 \]

\[ (v_0, v_2) \text{ is undirected} \]

\[ E = \{ (v_2, v_0), (v_0, v_1), (v_1, v_4), \ldots \} \]

\[ |E| = 6 \]

Path: from \( v_2 \) to \( v_4 \) is \( \{ v_2, v_0, v_1, v_4 \} \)
A graph $G = (V,E)$ consists of a set of vertices $V$ and a set of edges $E$

- Each edge in $E$ is a pair $(v,w)$ such that $v$ and $w$ are in $V$.
- If $G$ is an \textit{undirected} graph, $(v,w)$ in $E$ means vertices $v$ and $w$ are connected by an edge in $G$. This $(v,w)$ is an unordered pair.
- If $G$ is a \textit{directed} graph, $(v,w)$ in $E$ means there is an edge going from vertex $v$ to vertex $w$ in $G$. This $(v,w)$ is an ordered pair; there may or may not also be an edge $(w,v)$ in $E$.
- In a \textit{weighted} graph, each edge also has a “weight” or “cost” $c$, and an edge in $E$ is a triple $(v,w,c)$.
- When talking about the size of a problem involving a graph, the number of vertices $|V|$ and the number of edges $|E|$ will be relevant.
Connected, disconnected and fully connected graphs

- **Connected graphs:**
  - In undirected graphs it means there is a path between all pairs of vertices.
  - In directed graphs we have notions of:
    1. Strongly connected: Path exists between all vertex pairs.
    2. Weakly connected: Equivalent undirected graphs.

- **Disconnected graphs:**
  - If a graph is not connected, then it is disconnected.

- **Fully connected (complete graphs):**
  - Edges exist between all vertex pairs.
Q: What are the minimum and maximum number of edges in an undirected connected graph G(V,E) with no self loops, where N=|V|?

A. 0, N^2
B. N, N^2
C. N-1, N(N-1)/2
Sparse vs. Dense Graphs

A dense graph is one where $|E|$ is “close to” $|V|^2$.

A sparse graph is one where $|E|$ is “closer to” $|V|$.

$|E| = O(|V|)$
A 2D array where each entry \([i][j]\) encodes connectivity information between \(i\) and \(j\)

- For an unweighted graph, the entry is 1 if there is an edge from \(i\) to \(j\), 0 otherwise.
- For a weighted graph, the entry is the weight of the edge from \(i\) to \(j\), or “infinity” if there is no edge.
- Note an undirected graph’s adjacency matrix will be symmetrical.
Representing Graphs: Adjacency Matrix

How big is an adjacency matrix in terms of the number of nodes and edges (BigO, tightest bound)?

A. $|V|$
B. $|V| + |E|$
C. $|V|^2$
D. $|E|^2$
E. Other

When is that OK? When is it a problem?
A dense graph is one where $|E|$ is “close to” $|V|^2$.
A sparse graph is one where $|E|$ is “closer to” $|V|$.

Adjacency matrices are space inefficient for sparse graphs.
Representing Graphs: Adjacency Lists

- Vertices and edges stored as lists
- Each vertex points to all its edges
- Each edge points to the two vertices that it connects
- If the graph is directed: edge nodes differentiate between the head and tail of the connection
- If the graph is weighted edge nodes also contain weights

Space requirement: 
\[2 \cdot (V + E)\]

Total no. of pointers in all vertex objects is \(|E|\), use this in your analysis rather than bounding the no. of pointers in each vertex object.
Representing Graphs: Adjacency Lists

Each vertex has a list with the vertices adjacent to it. In a weighted graph this list will include weights.

How much storage does this representation need? (BigO, tightest bound)

A. $|V|$
B. $|E|$
C. $|V| + |E|$
D. $|V|^2$
E. $|E|^2$

$|E| = O(|V|)$

Sparse

Adjacency list: $O(|V| + |E|) = O(|V|)$

Dense

Adjacency matrix: $O(|V|^2)$

$O(|V|^2)$ space for adjacency matrix
Searching a graph

• Find if a path exists between any two nodes
• Find the shortest path between any two nodes
• Find all nodes reachable from a given node

Generic Goals:
• Find everything that can be explored
• Don’t explore anything twice
Generic approach to graph search

\[ V_0 \rightarrow V_1 \rightarrow V_3 \rightarrow V_2 \rightarrow V_4 \]

- **X**: all explored vertices
  - Initially: \( X = \{ V_0 \} \) (source)

- Explore via edges that cross the frontier.

- These are edges that connect vertices in \( X \) to vertices in \( V \times X \)
Depth First Search for Graph Traversal

- Search as far down a single path as possible before backtracking
Depth First Search for Graph Traversal

• Search as far down a single path as possible before backtracking

Assuming DFS chooses the lower number node to explore first, in what order does DFS visit the nodes in this graph?

A. V0, V1, V2, V3, V4, V5
B. V0, V1, V3, V4, V2, V5
C. V0, V1, V3, V2, V4, V5
D. V0, V1, V2, V4, V5, V3
Depth First Search for Graph Traversal

- Search as far down a single path as possible before backtracking

Does DFS always find the shortest path between nodes?
A. Yes
B. No