CSE 373

MAY 5TH  – MORE GRAPHS
MINUTIAE

• HW4 is out
• Exam regrades until 4:30 after class today
• Also available through next week
• Exams not picked up are in my office
EXCEPTIONS

- HW4 requires exception throwing
  - [https://docs.oracle.com/javase/tutorial/essential/exceptions/throwing.html](https://docs.oracle.com/javase/tutorial/essential/exceptions/throwing.html)
  - Here’s a good tutorial
  - But, here are the basics
EXCEPTIONS

• What to do during unacceptable behavior?
  • Crashing isn’t ideal
  • Exiting doesn’t give the client much information on why the crash occurred
  • Throwing an exception allows the user to understand exactly what went wrong.
EXCEPTIONS

• You may use any exception that you want, throwing the default Exception() is fine, but you should get in a habit of throwing informative errors

  • DuplicateEdgeException on an edge insertion is much more useful than a crash or terminate

  • Null Pointer Exception, Array Index Out of Bounds Exception, Illegal Argument Exception (good for much of HW4)
GRAPHS

• Graphs are not an ADT
  • There is no “functions” that a graph supports
  • Rather, graphs are a theoretical framework for understanding certain types of problems.
  • Travelling salesman, path finding, resource allocating
GRAPHS

• A graph is composed of two things
  • A set of vertices
  • A set of edges (which are vertex tuples)

• Trees are types of graphs
  • Each of the nodes is a vertex
  • Each pointer from parent to child is an edge

• Represented as $G(V,E)$ to indicate that $V$ is the set of vertices and $E$ is the set of edges
• What this graphs vertices and edges?
• What this graphs vertices and edges?
  • \( V = \{A, B, C, D, E\} \)
  • \( E = \{(A,B) , (A,C), (A,D), (A,E)\} \)
• What this graphs vertices and edges?
  • \( V = \{A, B, C, D, E\} \)
  • \( E = \{(A,B), (A,C), (D,A), (E,A)\} \)
GRAPHS

• Graphs can be either directed or undirected
  • Undirected graph, if \((A,B)\) is in the set of edges, \((B,A)\) must be in the set of edges
  • Directed graphs, both can be in the set of edges, but those graphs have different connectivity

• We call a graph *connected* if there is a path between every pair of vertices
GRAPHS

• Paths and Cycles
  • A path: a set of edges connecting two vertices where all of the edges are connected and neither edges nor vertices are repeated
  • A cycle: a path that starts and ends on the same
• Is this graph connected?
  • Is there a path between every pair of vertices?
• Is this graph connected?
  • There’s no way to get from the green graph to the red
Does this graph have a cycle?

- How many does it have?
**GRAPHS**

- Does this graph have a cycle?
  - \{ (A,E), (E,B), (B,D), (D,A) \}
  - \{ (A,B), (B,D), (D,A) \}
GRAPHS

• Paths and cycles can not have repeated vertices or edges
  • A path that can repeat vertices or edges is called a walk
  • A path that can repeat vertices but not edges is called a trail
  • A circuit is a trail that starts and ends at the same vertex
• Edges can have weights
  • This becomes important when we consider path finding algorithms
  • Usually, we consider the weights to be the costs of using a particular edge.
  • In a graph representation of the US interstate system, the I-90 edge between Seattle and Spokane may have weight 270 for miles or 4 for hours, depending on what we want to minimize!
GRAPHS

• When we consider graphs, we determine them to be either dense or sparse
  • Dense graphs are very connected, each vertex is connected to a fraction of the total vertices
  • Sparse graphs are less connected and can be more clustered, each vertex is connected to some constant number of vertices
GRAPHS

• When graphs are small, it is difficult to distinguish between the two
  • If we represent Facebook as a graph, where users are vertices and “friendships” are edges, what can we say about the graph?
    • Directed? **No**, (A,B) means (B,A)
    • Connected? **Very probably**
    • Cyclic? **Yes**, mutual friends
    • Sparse/Dense? **Sparse!** 338 average!
GRAPHS

• This “value” is called the degree of the vertex
  • If you have 338 friends, then that vertex has degree 338.
• In directed graph, we separate this into in-degree and out-degree
  • Consider Twitter, where friendship isn’t symmetric. The number of followers you have is your in-degree and the number of people you follow is your out-degree
TRAVERSALS

• Since graphs are abstractions similar to trees, we can also perform traversals.
  • If a graph is connected, i.e. there is a path between all pairs of vertices, then a traversal can output all nodes if you do it cleverly.
TRAVERSAL

- Depth-first search (prev graph with (D,G) added to make it connected
  - Traverse the tree with DFS, if there are multiple nodes to choose from, go alphabetically. Start at A.
Output: A
Current Node: A
Out-vertices: B, D, E
Output: A,B
Current Node: B
Out-vertices: D
Output: A,B, D
Current Node: D
Out-vertices: A,G
Output: A, B, D, A
Current Node: A
Out-vertices: B, D, E
Output: A, B, D, A
Current Node: A
Out-vertices: B, D, E

Oh, no! We have repeated output!
TRAVERSAL

• Depth first search needs to check which nodes have been output or else it can get stuck in loops.
  • This increases the runtime and memory constraints of the traversal
• In a connected graph, a BFS will print all nodes, but it will repeat if there are cycles and may not terminate
TRAVERSAL

• As an aside, in-order, pre-order and post-order traversals only make sense in binary trees, so they aren’t important for graphs. However, we do need some way to order our out-vertices (left and right in BST).
TRAVERSAL

• Topological ordering
  • One final ordering for graphs
  • Ordering with a focus on dependency resolutions

• Example, consider a graph where courses are vertices and edges are prerequisites. A topological ordering is any valid class order
Start with the nodes that have in-degree 0 (no prereqs).
Then eliminate that vertex (print it out) and eliminate its out edges.
What is a valid topological sort of this graph?
What is a valid topological sort of this graph?

F,C,G,D,A,E,B  
F,G,D,C,A,E,B  
F,G,C,D,A,E,B  

Is this all the valid solutions?
NEXT WEEK

• Another topological sort problem
• Weights and pathfinding
• Start Dijkstra’s algorithm