Lecture 21: Graphs

CS 2110

April 9, 2026

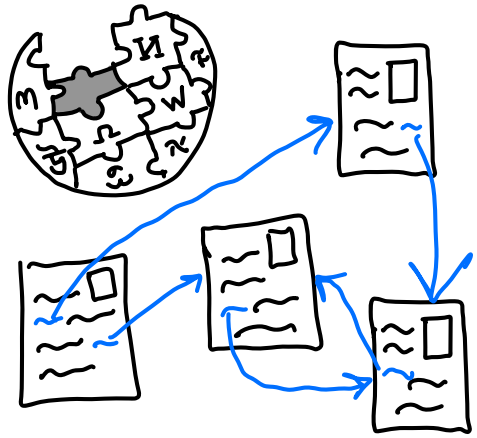
Today's Learning Outcomes

88. Translate between a formal description (list of vertices, edges) and an illustration of a (weighted) graph.

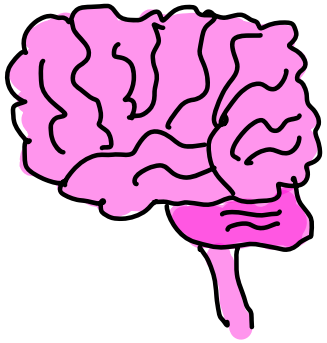
89. Identify substructures in graphs such as neighborhoods, paths, and cycles both visually and programmatically.

90. Describe adjacency list and matrix representations of a graph and compare the space/time complexities of operations on each of these representations.

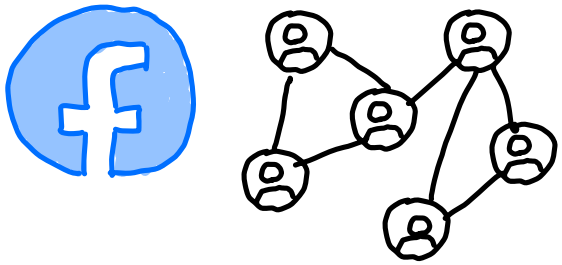
Modeling Structure in Real-World Settings



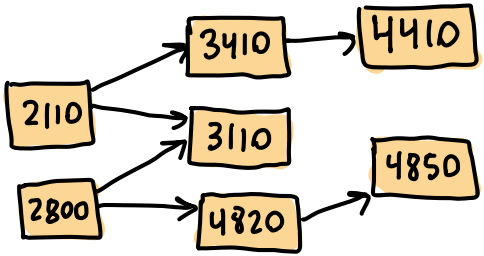
Wikipedia Links



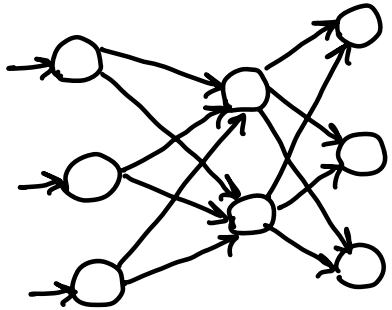
Neural Networks

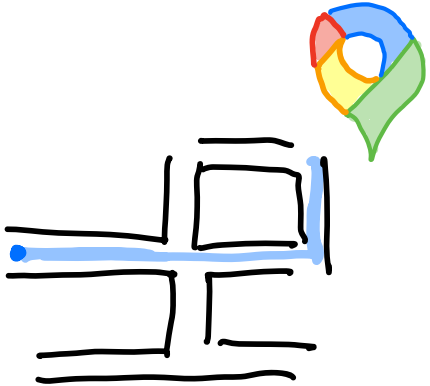


Social Networks



Course Planning

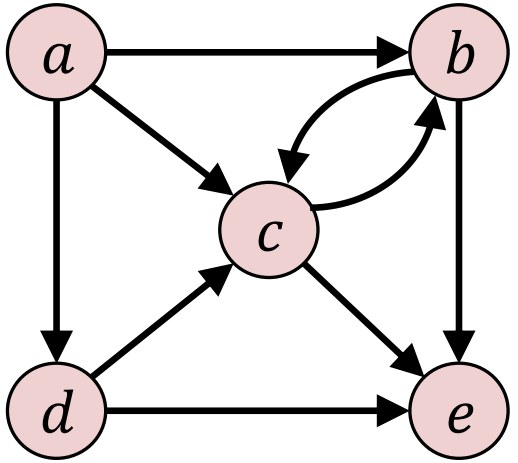




Route Planning

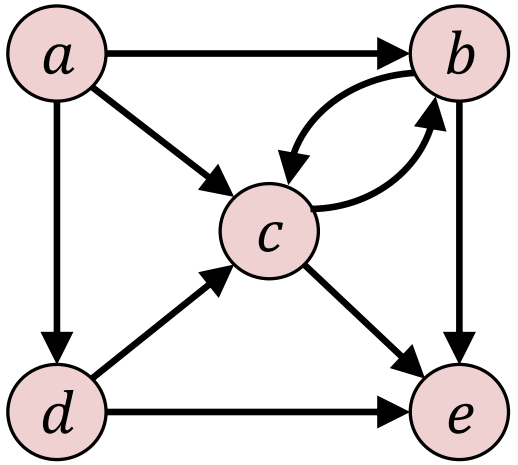
Directed Graphs

A directed graph is a structure described by two sets.

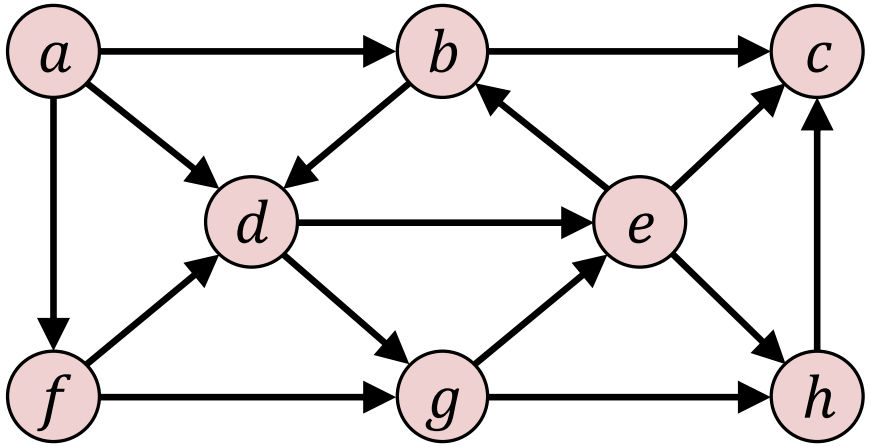


Other Graph Varieties

Neighborhoods in Graphs

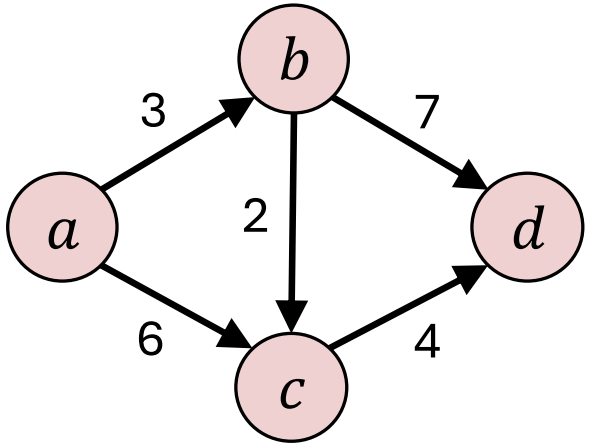


Paths and Cycles



Labeled Graphs

Sometimes, we label vertices/edges with extra info



A Graph ADT

Brainstorm: What operations should a `Graph` support?

Query: Access info

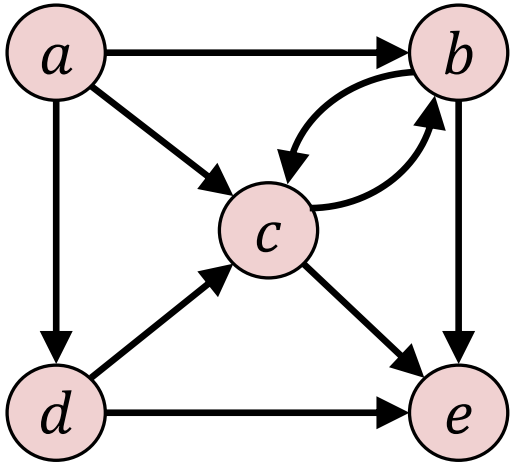
Mutate: Change

Iterate

Coding Demo: Graph ADTs

Adjacency Matrix Representation

Keep track of edges using 2D array / list structure



tail

head

	a	b	c	d	e
a					
b					
c					
d					
e					

Coding Demo: AdjMatrixGraph

Adjacency Matrix Operation Complexities

```
class AdjMatrixGraph implements Graph<...> {
   record AdjMatrixEdge(...) implements Edge<...> { }

   class AdjMatrixVertex implements Vertex<...> {
      private String label;
      private int index;
   }

   private HashMap<String, AdjMatrixVertex> vertices;
   private ArrayList<ArrayList<AdjMatrixEdge>> edges;

   // methods
}
```

Iterate over neighbors:

Count vertices:

Count edges:

Check for an edge:

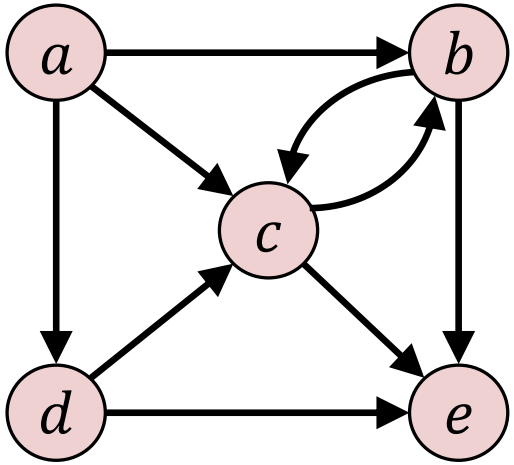
Add vertex:

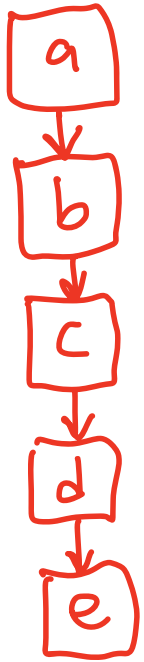
Add edge:

Iterate over all edges:

Adjacency List Representation (Classical)

Maintain a list of all vertices, and have each vertex maintain a list of all its neighbors





Adjacency List Representation (Improved)

Iterating over nested list structure is slow

- linear scan over vertices to find tail's neighbors list
- linear scan over neighbor list to search for edge

Coding Demo: `AdjListGraph`

Adjacency List Operation Complexities

```
class AdjListGraph implements Graph<...> {
   record AdjListEdge(...) implements Edge<...> { }

   static class AdjListVertex implements Vertex<...> {
      String label;
      HashMap<String, AdjListEdge> outEdges;
   }

   private HashMap<String, AdjListVertex> vertices;

   // methods
}
```

Iterate over neighbors:

Count vertices:

Count edges:

Check for an edge:

Add vertex:

Add edge:

Iterate over all edges:

Comparing Representations

Representation	Memory Usage	Time to Check for an Edge	Time to Iterate over Neighborhood	Time to Iterate over all Edges
Adj Matrix				
Adj List (Linked)				
Adj List (Map)				