CS 315 Lecture 1 Graph Intro

Introduction to the course

Late Policy:

- <= 12 hours after deadline
  - 10% reduction
- > 12 <= 24 hours after deadline
  - 20% reduction
- > 24 <= 48 hours after deadline
  - 30% reduction
- > 48 hours after deadline
  - Won't be graded

Course Topics:

- Programming and problem solving, with applications
- **Algorithms:** step-by-step procedure for solving a problem
- **Topics:**
  - Graphs:
    - BFS
    - DFS
    - Prim
    - Kruskal
    - Dijkstra
    - etc
  - Dynamic Programming:
    - Fibonacci
    - Matrix chain multiplication
    - Common sequence
    - Knapsack
    - etc

- Divide and Conquer:
  - Maximum-sub array
  - Strassen
  - Multiplying polynomials
- Greedy:
  - Activity Selection
  - Job scheduling
  - Huffman coding
- Advanced Topics:
  - Network flow string matching
- Algorithm Complexity:
  - Np Hardness

Why Study Algorithms?

- Their impact is broad and far reaching
  - Tons of fields from internet to biology
- Become a proficient programmer:
  - "Difference between a good and bad programmer, bad programmers worry about the code, good ones worry about the data structures and their relationships" (Linus Torvalds(Creator of linux))
  - Algorithms + Data Structures = Programs
- Used by almost every company:
  - Apple
  - Amazon
  - Morgan Stanley
  - IBM
  - Netflix
  - Microsoft
  - Etc

Graph Introduction

Graphs

- **Graph:** Set of vertices connected pairwise by **edges**
- Why study graph algorithms?
  - Thousands of practical applications
  - Hundreds of graph algorithms known

- Examples:
  - **Transportation networks:**
    - Vertex: subway stop
    - Edge: direct route
  - **Protein Interaction Network:**
    - Vertex: protein
    - Edge: interaction
  - **Facebook Friends:**
    - Vertex: person
    - Edge: social relationships
  - **Twitter:**
    - Vertex: account
    - Edge: follower

Directed Graphs

- Terminology:
  - **Graph:** Set of vertices connected pairwise by edges
  - **Directed:** has an in and out degree because the edges have a direction
  - **Path:** Sequence of vertices connected by **directed** edges, with no repeated edges
  - **Connected:** Two vertices are **connected** if there is a **directed** path between them
  - **Directed Cycle: Directed** path whose first and last vertices are the same
  - **Out Degree:** number of edges pointing out from a vertex
  - **In Degree:** number of edges pointing into a vertex
- Example:
  - **Road Map:**
    - Vertices: Intersections
    - Edges: Roads

Undirected Graphs

- Terminology:
  - **Graph:** Set of vertices connected pairwise by edges
  - **Path:** Sequence of vertices connected by **directed** edges, with no repeated edges
  - **Connected:** Two vertices are **connected** if there is a **directed** path between them
  - **Cycle:** path whose first and last vertices are the same
  - **Degree:** number of edges connected to a vertex
- Example:

- **Facebook Friends:**
    - Vertices: profile
    - Edges: friendships

Some Graph-processing Problems

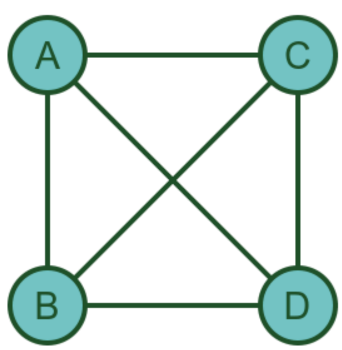
- **S-t path:** Find a path between s and t
- **Shortest s-t path:** Find the shortest path between s and t
- **Cycle:** Find a cycle
- **Euler Cycle:** Find a cycle that uses each edge exactly once
- **Hamilton Cycle:** Find a cycle that uses each vertex exactly once
- **Bioconnectivity:** Find a vertex whose removal disconnects the graph
- **Planarity:** Draw in the plane with no crossing edges
- **Graph Isomorphism:** Find an isomorphism between two graphs

Vertex Representation

- Use integers between 0 and $V-1$(or between 1 and V)
- Applications: convert between names and integers with symbol table

Graph Representation:

- **Adjacency Matrix:** V by V boolean array for each edge $v->w$ in graph: $adj[v][w] = true$

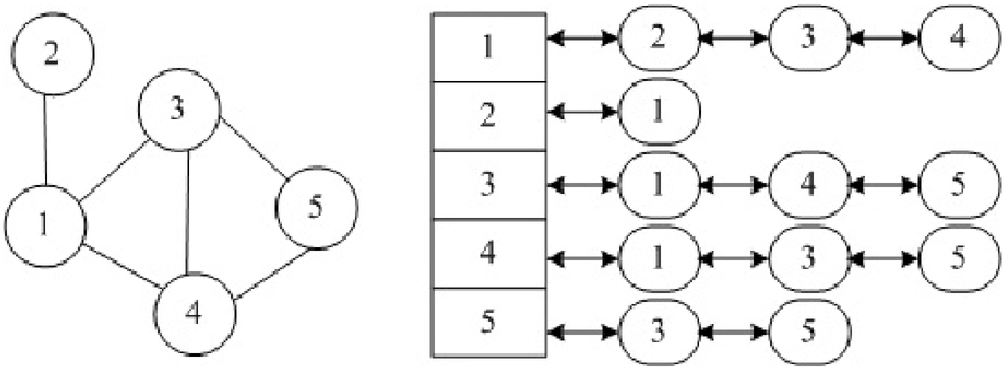


Complete graph

	A	B	C	D
A	0	1	1	1
B	1	0	1	1
C	1	1	0	1
D	1	1	1	0

- 1 for a connection
- 0 for no connection
- **Problem:** hard to resize and show for large amounts of data

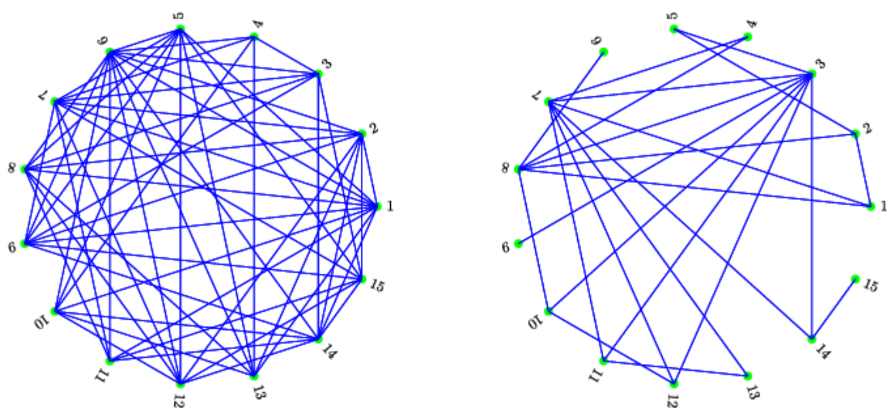
- **Adjacency Lists:** maintain vertex-indexed array of lists



  - **Question:** How long to iterate over vertices adjacent to v ?
    - Complexity will be $O(outdegree)$ of v

In Practice

- Most real world graphs are **sparse**
- **Sparse:** huge number of vertices and small average vertex degree



Representation	Space	Insert edge from v to w	Edge from v to w	Iterate over vertices pointing from v
Adjacency matrix	V^2	1^t $(t =$ dissal	1	V
Adjacency lists	$E + V$	1	$outdegree(v)$	$outdegree(v)$