



SCHOOL OF ENGINEERING
VANDERBILT UNIVERSITY

CS 3250
Algorithms
Lecture 4

Graphs: Introduction to Graphs



Announcements

HW1 has two different parts (Part A and Part B)

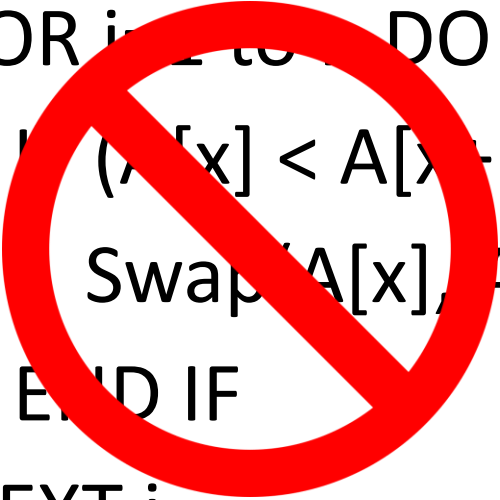
- **HW1A** - Calibration Quiz on Brightspace. Average = 95%
- **HW1B** – Due Wednesday, January 24th by 9AM. There are two subparts to HW1B:
 1. **Brightspace Quiz on Asymptotic Analysis.** Warning: This quiz is designed to be tricky. You get two attempts. The higher score is recorded. Each attempt is timed (2 hours).
 2. **Written Portion.** Submit typed PDF to Gradescope.



Some Tips For Your 1st Written HW

- In 3250, we don't write code or pseudocode.

```
FOR i = 1 TO n DO  
  IF (A[i] < A[i+1]) THEN  
    Swap(A[i], A[i+1])  
  END IF  
NEXT i
```



- **Instead, we write this:** For each of the n elements of the array, compare it to the element to the right of it. If they are out of order, swap the elements.

Some Tips For Your 1st Written HW

1. **Re-read and revise** until you are satisfied that your answers are clear, correct and concise.
2. **Don't use pseudocode** or language specific code.
3. **Be careful with built-in routines.** It must be universal AND you need to know how much work it does under the hood (e.g., binary search).
4. **Be careful with hashing approaches.** It's easy to get burned if you don't know table size, collision resolution scheme, load capacity or worst-case situation.



Proving Worst Case of $\Theta(g(x))$: Tips

- **Example:** What is the worst-case running time of this algorithm in Θ ?

```
match = false, i = 0, j = 1
while !match and i < n-1
    while !match and j < n
        if A[i] == A[j]
            then match = true
        j++
    Loop
    i++, j = i+1
Loop
output match
```

Proving Worst Case of $\Theta(g(x))$: Tips

- **Thinking:** What is the worst-case running time of this algorithm in Θ ?
- In the worst case we compare every element with every other element to find a match (or discover there is none).
- **The** outer loop and the inner while loop run to completion each time.
- This is both the lower bound for the worst-case and the upper-bound for the worst case.
- That makes this $\Theta(n^2)$ in the worst-case.



Proving Worst Case of $\Theta(g(x))$: Tips

- **Proving:** (provide exact number of comparisons):
 - First pass through the inner loop (worst-case)
 - $n-1$ compares
 - Second pass through inner loop (worst-case)
 - $n-2$ compares
 - Third time through the loop (worst-case)
 - $n-3$ compares
 - And so on...



Proving Worst Case of $\Theta(g(x))$: Tips

- Total compares worst-case is $= \sum_1^{n-1} i = \frac{n(n-1)}{2}$
- So, $f(x) = \frac{1}{2}n^2 - \frac{1}{2}n$
- By the theorem of Polynomial Orders
 - $f(x)$ is $O(n^2)$, and $f(x)$ is $\Omega(n^2)$
 - Therefore, we can conclude $f(x)$ is $\Theta(n^2)$



Proving Worst Case of $O(f(x))$: Tips

- Can we use an example to prove the **worst-case** of an algorithm is $O(f(x))$?



Proving Worst Case of $O(f(x))$: Tips

- Can we use an example to prove **worst-case** is $O(f(x))$?
- If you use an example, you are saying, “Here’s **one** example where this algorithm takes $c*f(x)$ to complete. Therefore, I confidently claim **all** data sets take no more than $c*f(x)$ to complete.”
- **Answer:** Uh, no. Proving $O()$ bounds of a worst-case running time requires an analysis of the algorithm itself.



Running Time and Use of Examples

- Hmm...what does an **example** say about the running time of an algorithm on all input?
- An example of an algorithm runs in $7(n \log n) + 5n$
- **Can we say the algorithm running time is $\Omega(n \log n)$?**



Running Time and Use of Examples

- What does an **example** say about the running time of an algorithm on all input?
- An example of an algorithm runs in $7(n \log n) + 5n$
- **Can we say the algorithm run time is $\Omega(n \log n)$?**
- **Answer:** No. There might be another example that does less work, in which case the overall running time doesn't take at least $\Omega(n \log n)$.



Running Time and Use of Examples

- What does an **example** say about the overall running time of an algorithm on all input?
- An example of an algorithm runs in $7(n \log n) + 5n$
- **Can we say the algorithm runs in $O(n \log n)$?**
- **Answer: No.** There might be another example that does worse, which means we cannot say the algorithm does AT MOST $n \log n$.



Tips: Running Time and Use of Examples

- What does an example say about the running time of an algorithm on all input?
- An example data set runs in $7(n \log n) + 5n$
- **Can we say the algorithm is $\Theta(n \log n)$?**
- **Answer:** No, certainly not since this would require having both an omega and big-oh bound of $n \log n$.



When Best and Worst Don't Agree

- **Question:** Is it possible for an algorithm to have a best-case running time of $\theta(n)$ and a worst-case running time of $\theta(n^2)$?

IT WAS THE WORST
OF TIMES ☹



When Best and Worst Don't Agree

- **Question:** Is it possible for an algorithm to have a best-case running time of $\theta(n)$ and a worst-case running time of $\theta(n^2)$?
- **Answer:** Sure...why not?

IT WAS THE WORST
OF TIMES ☹



Algorithm Analysis: Summary

- Determining **worst-case bounds** of an algorithm
 1. Identify the situation which results in the worst-case for the algorithm.
 2. Analyze the algorithm and determine the Ω , O , and if applicable the appropriate Θ .
 3. **An example or sample dataset cannot help us** determine a tight worst-case bound. We need to understand the function over all input sizes n representing the worst-case scenario.



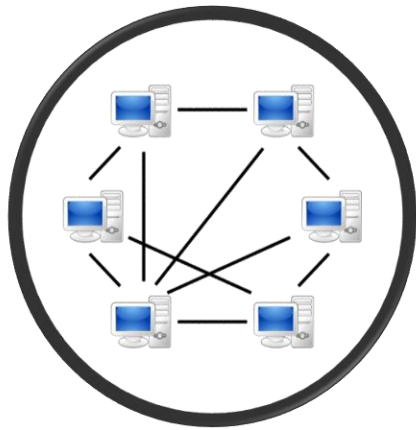
Algorithm Analysis: Summary

- Determining **best-case bounds** of an algorithm:
 1. Identify the situation which results in the best-case for the algorithm.
 2. Analyze the algorithm and determine the Ω , O , and if applicable the appropriate Θ .
 3. **An example or sample dataset cannot help us** determine a tight best-case bound. We need to understand the function over all input sizes n representing the best-case scenario.



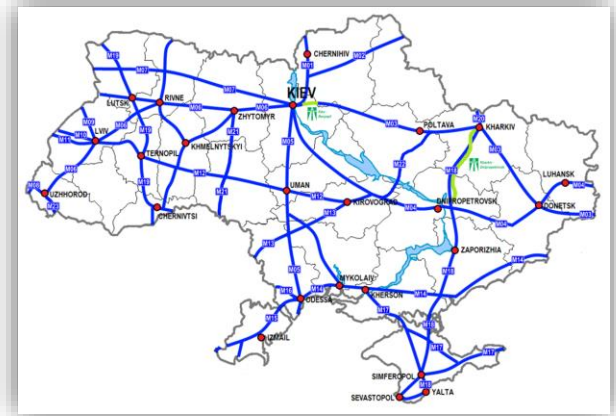
An Introduction to Graphs

- **Graphs** are one of the most important topics in Computer Science because it is the one area of computer science that intersects with virtually every other discipline.



Computer network

Social network



Highway network

An Introduction to Graphs

- I know what you're thinking... *"Oh good. I learned about graphs in CS 2201. This should be a breeze."*
- Uh, no. You have a basic knowledge of graphs. You need a **thorough** understanding of graphs.
- Graph problems often show up as technical interview questions.
- Many graph problems appear simple, when in fact, the solutions are quite subtle.



Graphs - What Do You Know?

- Which of the following statements about graph representations is always true?
 - A. An adjacency matrix representation of a graph performs slower than a linked list representation.
 - B. An adjacency matrix representation of a graph performs faster than a linked list representation.
 - C. An adjacency matrix representation of a graph performs the same as a linked list representation.
 - D. None of the above



Graphs - What Do You Know?

- Which of the following statements about graph representations is always true?
 - A. An adjacency matrix representation of a graph performs slower than a linked list representation
 - B. An adjacency matrix representation of a graph performs faster than a linked list representation.
 - C. An adjacency matrix representation of a graph performs the same as a linked list representation.
 - D. None of the above**
- By the end of today's lecture, you'll understand why.

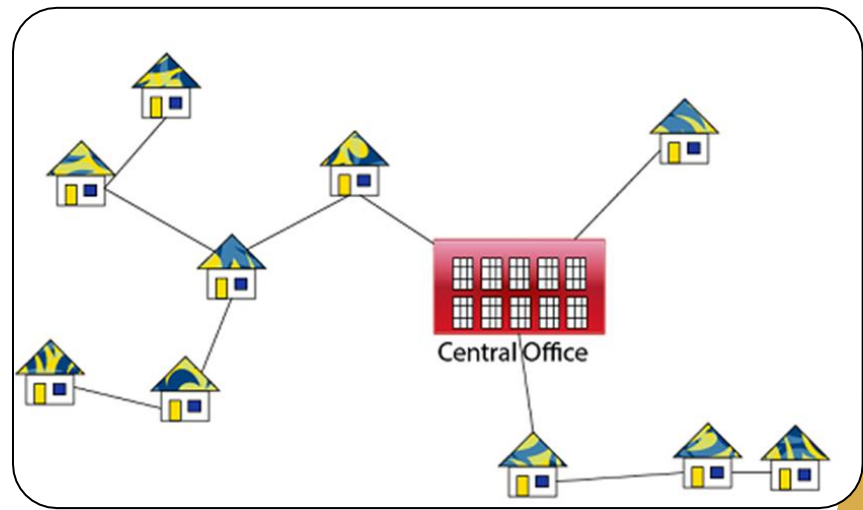
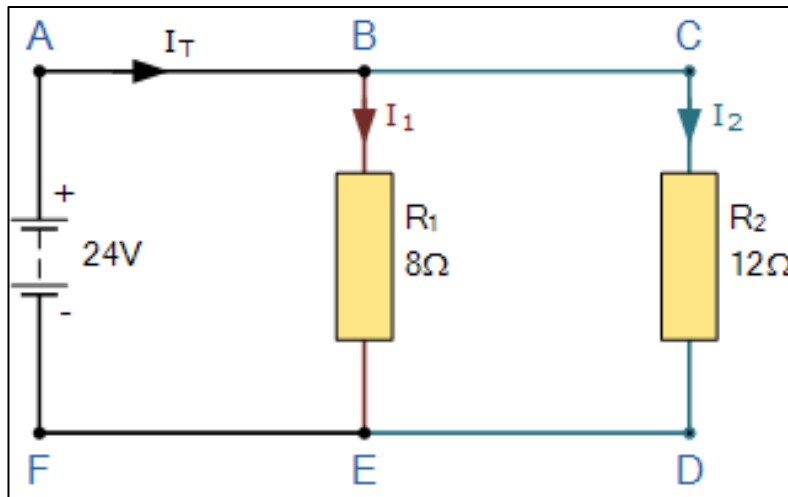


An Introduction to Graphs

- **Technical Interview Tip:** Most graph problems have already been studied and have efficient algorithms. Turns out a small number of graph algorithms solve **many** graph problems. So...
- It is usually **not a good idea** to approach a graph problem from scratch and invent your own fancy strategy. That won't impress your instructor (or your technical interviewer).
- It is better to identify and **recognize** the type of graph problem and use or modify a well-known algorithm.

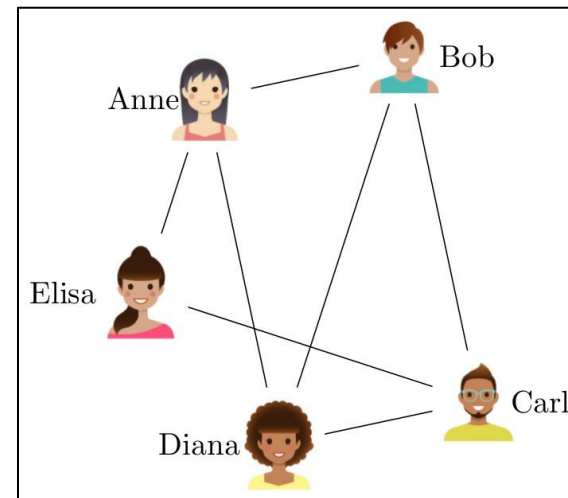
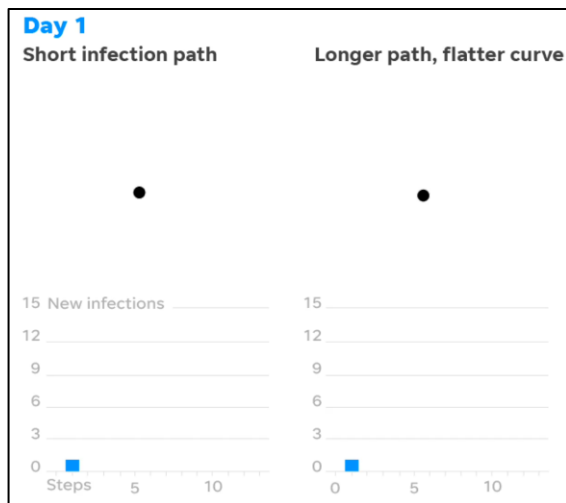
Graphs in Action

- **Hardware:** Circuit boards, networks, cables, etc.
- Applications in graph theory:
 - Where are the cycles located (Kirchhoff's Law)?
 - What is the minimum length of cable/pipe/etc. needed to connect all the houses?



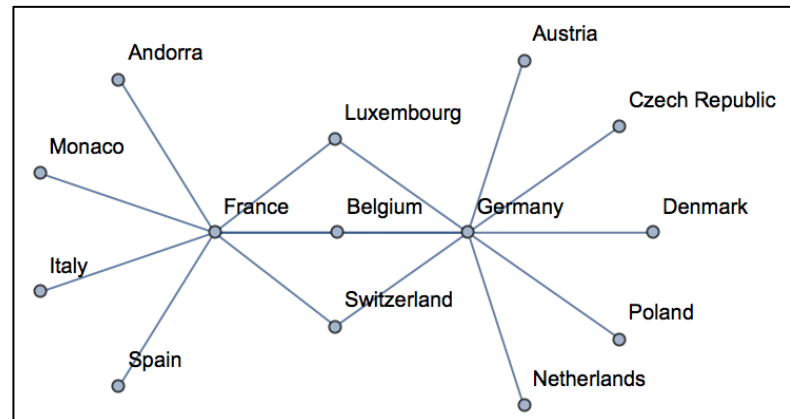
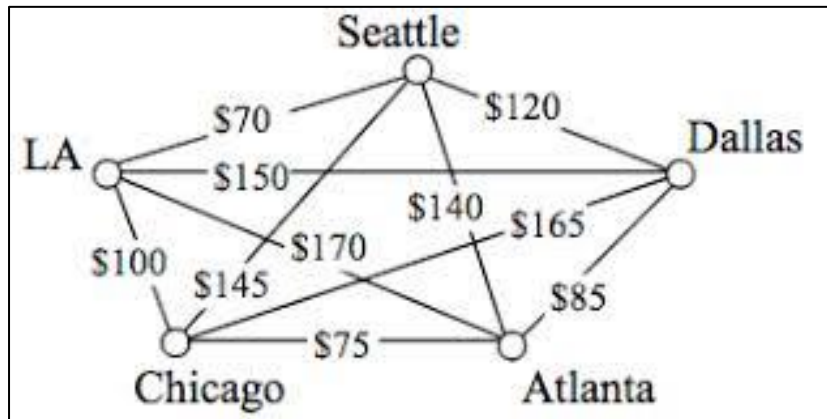
Graphs in Action

- **Social Networks:** Instagram, Twitter, Covid-19, etc.
- Is Anne friends with Bob?
- Who else might Anne want to connect with?
- Who has the most friends?
- Can social bubbles minimize rate of infection?



Graphs in Action

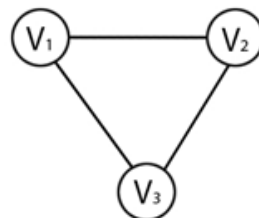
- **Transportation:** GPS, road networks, airplane routes, shipping routes, truck routes, etc.
- Can I get from Monaco to Denmark?
- What is the cheapest flight from LA to Dallas?
- France is having severe weather. How else can I can fly from Spain to Germany?



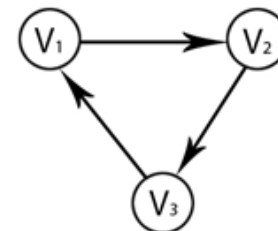
Graphs: A Quick Review

- A **graph** $G = (V, E)$ is defined by a set of vertices V and a set of edges E consisting of ordered or unordered pairs of vertices from V .
- A graph is **undirected** if edge $(v_1, v_2) \in E$ implies edge (v_2, v_1) is also $\in E$. Otherwise, graph is **directed**.
- Airplane networks are usually undirected.
- Street networks usually directed (one-way)

Undirected Graph

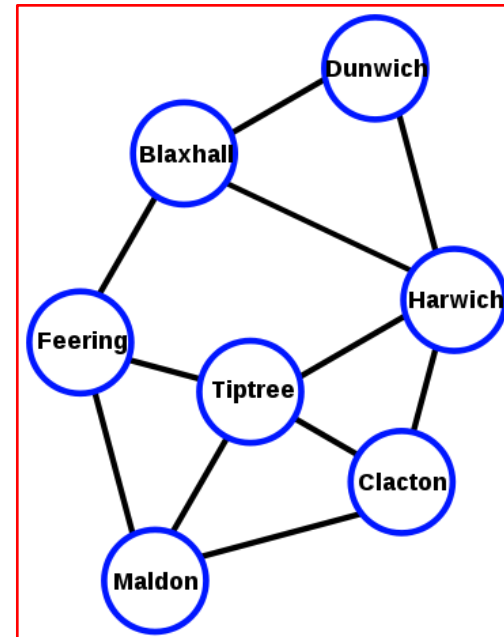
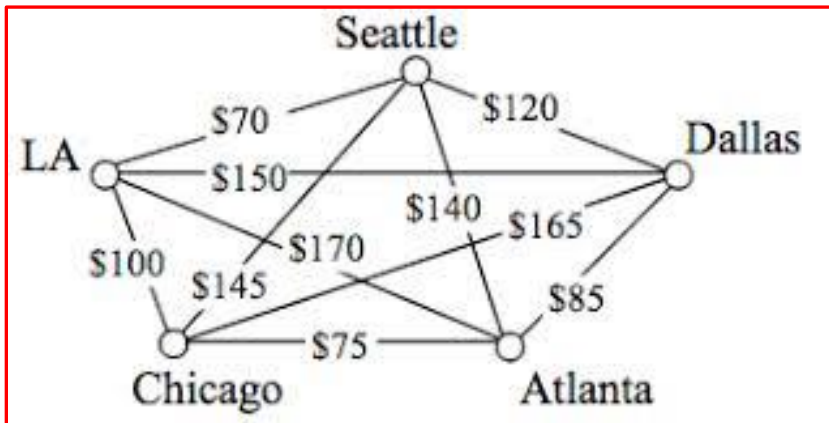


Directed Graph



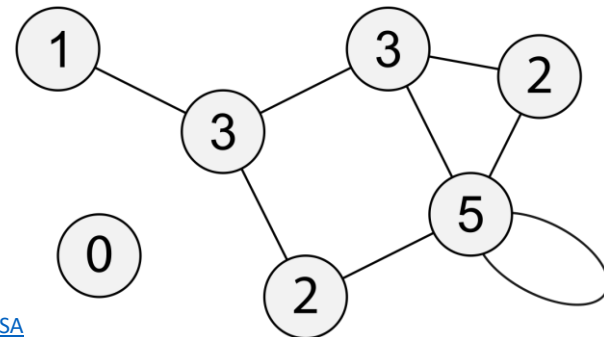
Graphs: A Quick Review

- In a **weighted** graph $G = (V, E)$ each edge is assigned a weight or cost.
- A social network is typically unweighted
- An airplane network is often weighted
- A weighted graph can be directed or undirected



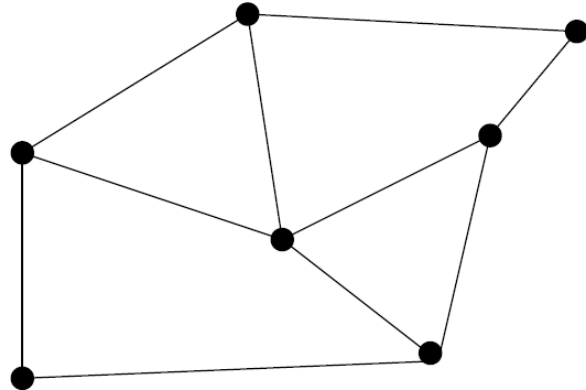
Graphs: A Quick Review

- A **self-loop** is an edge (x, x) involving only one vertex.
- Self-loops can unnecessarily complicate an algorithm and are often not included in the graph as a matter of common sense.
- **Example:** A plane that departs Nashville, goes nowhere and then returns to Nashville).
- By default, assume our graph problems will not include self-loops.

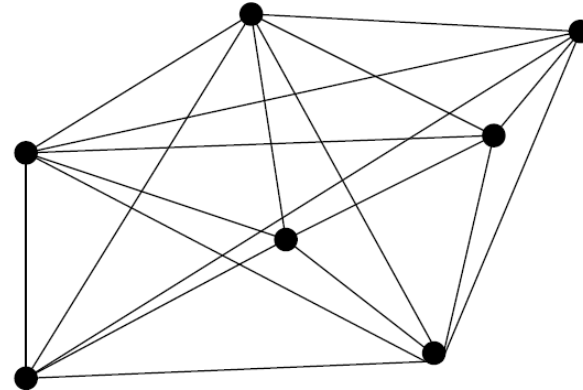


Graphs: A Quick Review

- Graphs may be considered **sparse** or **dense** depending on the number of edges in the graph relative to the number of vertices.
- Sparse graphs typically have a linear number of edges.



Sparse Graph



Dense Graph

Graphs: A Quick Review

- Around how many edges are there in a very **dense** graph?
 - A. Approximately $|V|$
 - B. Approximately $|V|^2$
 - C. Approximately $|V|^3$
 - D. None of the above



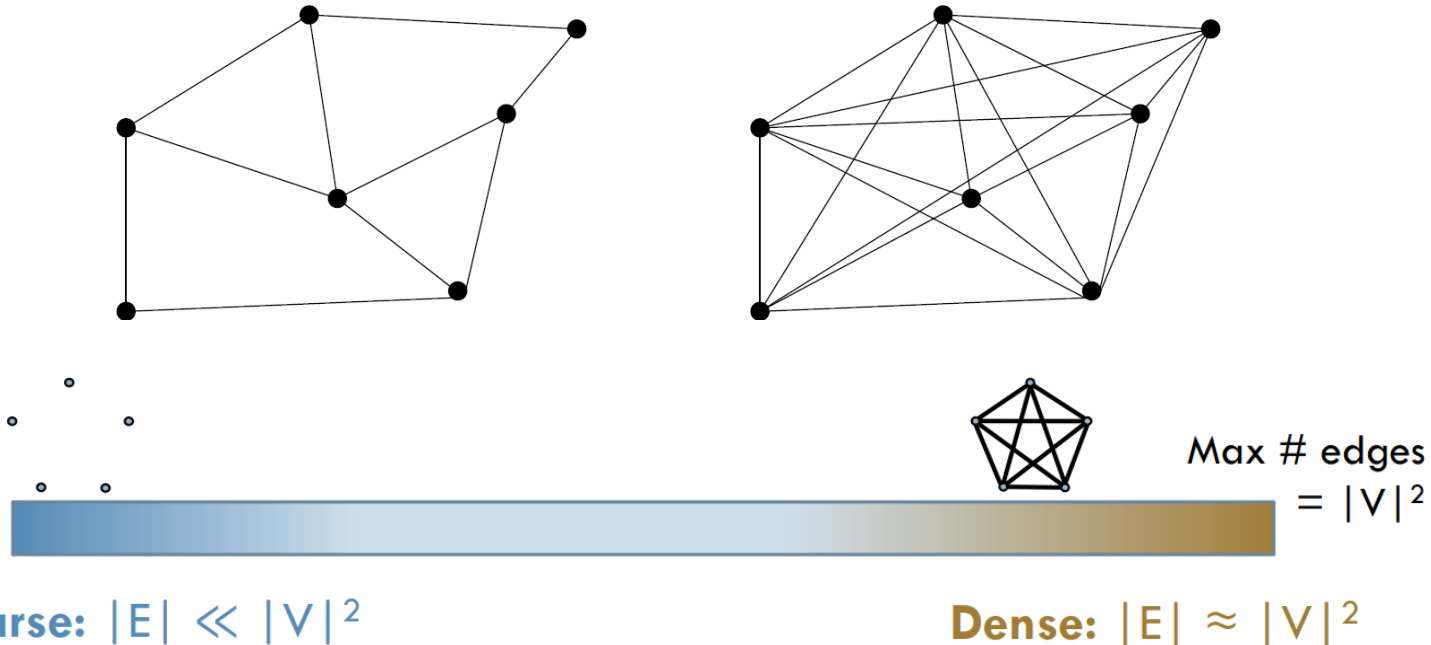
Graphs: A Quick Review

- Around how many edges are there in a very dense graph?
 - A. Approximately $|V|$
 - B. Approximately $|V|^2$**
 - C. Approximately $|V|^3$
 - D. None of the above



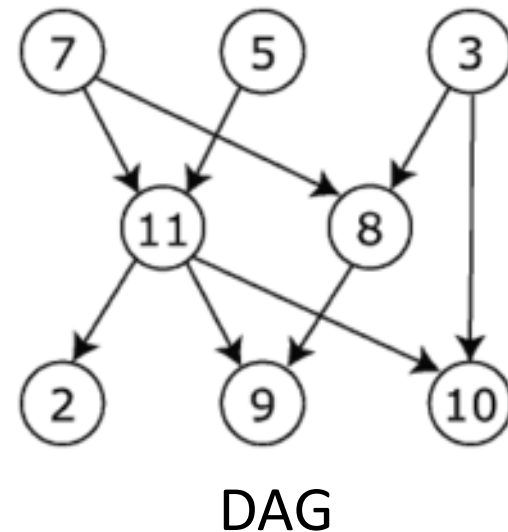
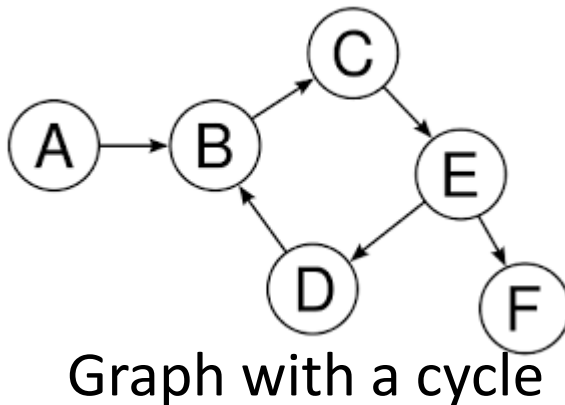
Graphs: A Quick Review

- Graphs are considered **dense** when a majority of the possible number of vertex pairs have edges.
- Dense graphs typically have a quadratic number of edges ($\approx |V|^2$)



Graphs: A Quick Review

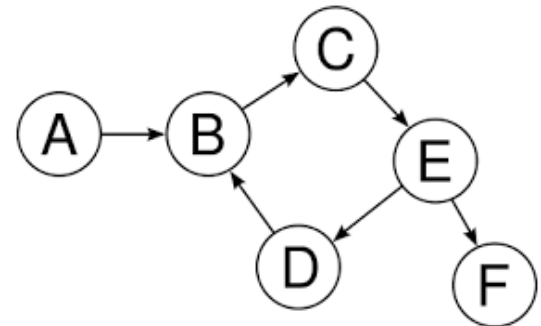
- The **degree** of a vertex is the number of edges adjacent to it. For a directed graph, we talk about **in-degree** and **out-degree**.
- A **path** is a sequence of edges connecting two vertices
- A **cycle** is a path with distinct edges that begins and ends at the same vertex (e.g., think dog chasing its tail).
- A **DAG** is a directed acyclic graph.



Graphs: A Quick Review

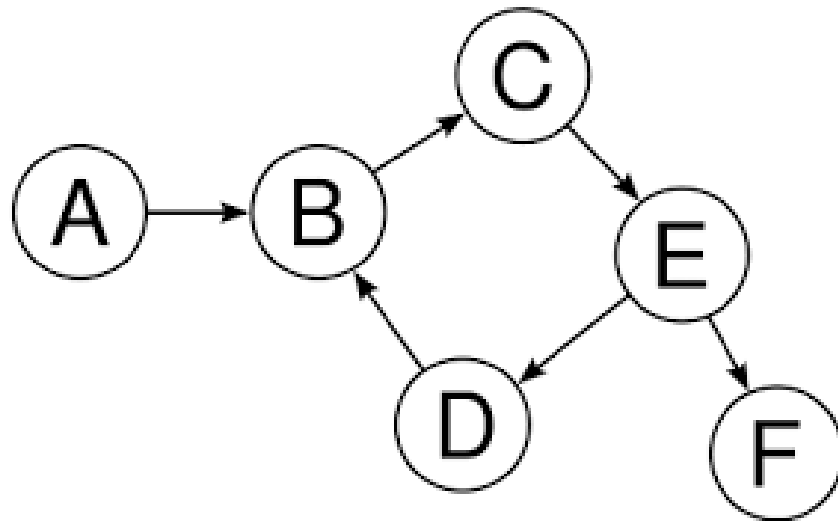
- In an **undirected** graph, we say vertex v is a **neighbor** of vertex u or **adjacent** to vertex u if there is an edge connecting v to u .
- In a **directed** graph, we say vertex A is **adjacent** to vertex B if there is an edge from A to B .
- In a **directed** graph, most authors distinguish between in-neighbor and out-neighbor.
- FYI - some authors use the opposite direction edge to mean adjacent. For programmers, it makes sense to use our definition.

Vertex A is adjacent to B



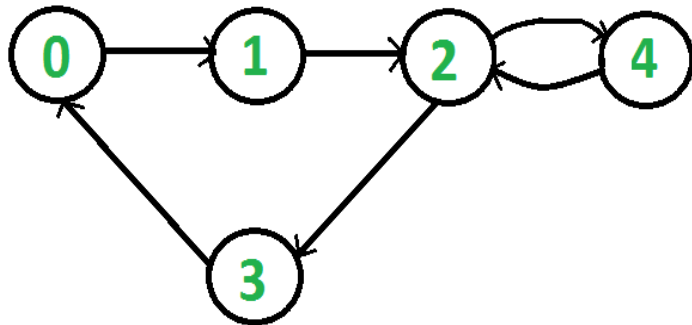
Graphs: A Quick Review

- According to our definition, in the graph below:
 - A is adjacent to B.
 - B is not adjacent to A.
 - B is an out-neighbor of A.
 - A is an in-neighbor of B.

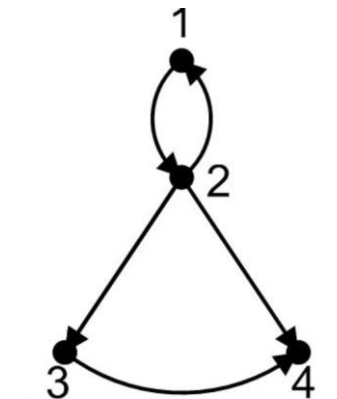


Graphs: A Quick Review

- A graph is **connected** if there is a path between any two vertices.
- A directed graph is **strongly connected** if there is a directed path between any two vertices.
- A directed graph is **weakly connected** if there is a path between any two vertices when ignoring edge direction.



Strongly Connected

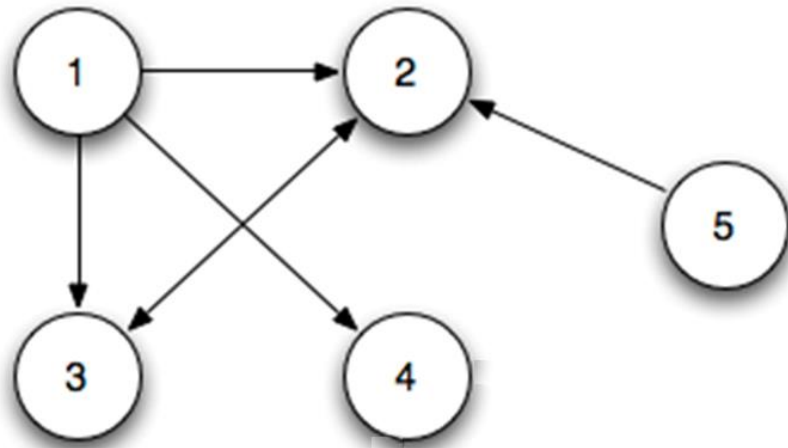


Weakly connected graph

Data Structures for Graphs

There are two main data structures used to represent graph $G = (V, E)$ which contains n vertices and m edges.

1. Adjacency Matrix - We can represent G using an $n \times n$ matrix M , where element $M[i, j]$ contains a 1 if (i, j) is an edge in G , and 0 if it is not. For a **weighted graph**, the edge weight is stored in lieu of storing a 1.



	1	2	3	4	5
1	0	1	1	1	0
2	0	0	1	0	0
3	0	1	0	0	0
4	0	0	0	0	0
5	0	1	0	0	0

Data Structures for Graphs

1. Adjacency Matrix –

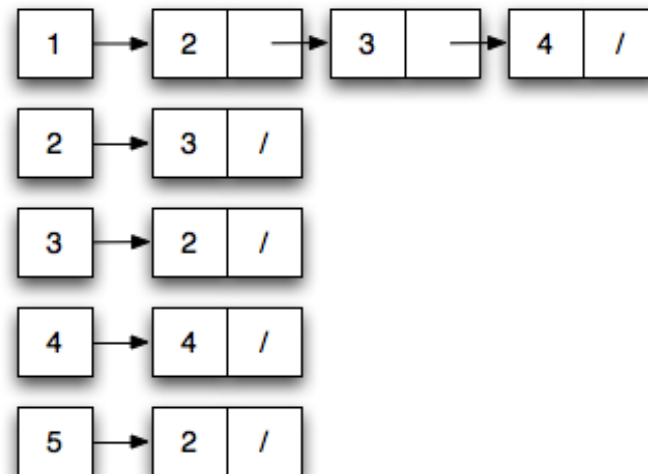
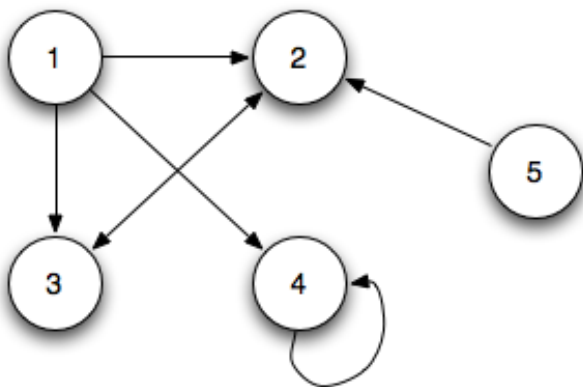
- **Space:** An array of $n \times n$ is needed $\Theta(n^2)$ regardless of a sparse or dense graph in terms of number of edges.
- **Speed:** Depends on the question being asked.
 - Is there an edge from vertex 4 to vertex 8? Fast. $\Theta(1)$
 - List all the neighbors of every vertex? Slow. $\Theta(n^2)$
 - Delete vertex 4? Slow. $O(n^2)$

	1	2	3	4	5
1	0	1	1	1	0
2	0	0	1	0	0
3	0	1	0	0	0
4	0	0	0	0	0
5	0	1	0	0	0

Data Structures for Graphs

There are two main data structures used to represent graph $G = (V, E)$ which contains n vertices and m edges.

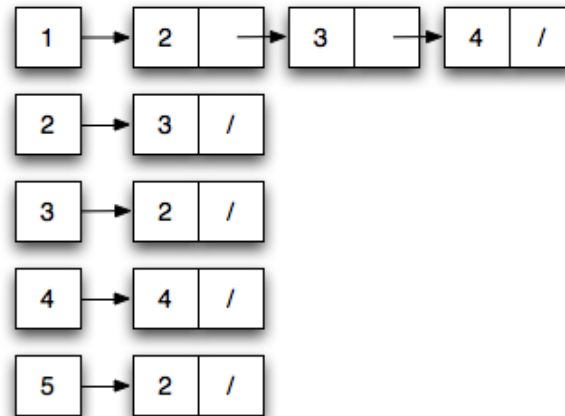
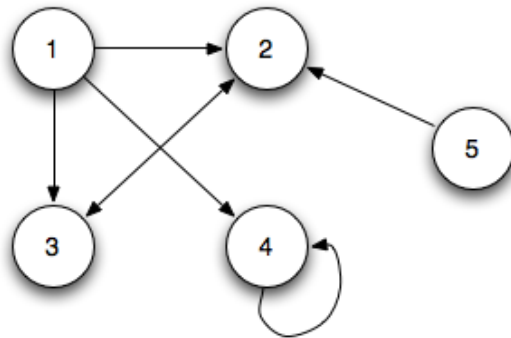
2. Adjacency List - An adjacency list consists of an array of $|V|$ pointers, where the i^{th} element points to a linked list of all the edges incident to vertex i . For a **weighted graph** store the edge weight as an additional field in each edge node in the adjacency list.



Data Structures for Graphs

2. Adjacency List –

- **Space:** $O(|V| + |E|) = O(V + E) = O(\max(V, E))$
 - Cardinality symbol usually not included.
- **Speed:** Depends on the question being asked.
 - Is there an edge from vertex 4 to vertex 8? $O(V)$ since a vertex might be adjacent to all other vertices.
 - List all the neighbors of every vertex? $\Theta(V + E)$



Graphs – Now What Do You Know?

- In most cases, is an adjacency matrix representation the fastest way to solve a graph problem?
 - A. Yes, of course.
 - B. Definitely not.
 - C. It just depends on the situation.



Graphs – Now What Do You Know?

- In most cases, is an adjacency matrix representation the fastest way to solve a graph problem?
 - A. Yes, of course.
 - B. Definitely not.
 - C. It just depends on the situation.**



On Your Own: Adjacency List

Question: Answer these without drawing the graph

Given a directed graph, assume that:

- The neighbor lists are linked lists.
- You can't discover whether vertex a is adjacent to vertex f without first traversing i and g in the adjacency list.
- You may assume you can use pointers to keep track of where you are in the lists.
- A null list indicates no neighbors.

Vertex	Neighbors (linked list)
a	i g f
b	a f d
c	b d e
d	null
e	f
f	d e h
g	f i
h	null
i	h c

Questions:

1. List all of the neighbors f.
2. Is vertex b one of c's neighbors?
3. Is there a path from a to e?



Graph Representations: Adjacency Matrix

- **Question:** Try the same questions with an adjacency matrix. Explain your answers:
 1. List the neighbors of f: **d, e, h**
 2. Is b the neighbor of c? **yes**
 3. Is there a path from a to e? If so, what is it?
Yes. a->i->c->e



Adjacency Matrix vs. Adjacency List

FACE OFF

Task	Winner
Does edge (x, y) exist?	Adjacency Matrix
What is the degree of vertex 4?	Adjacency List
Add/Delete an edge	Adjacency Matrix
Traverse the graph	Adjacency List
Better for most problems	Adjacency List

That's All For Now...

- Coming to a Slideshow Near You Soon...
 1. BFS Tree of Discovery
 2. Analysis of the BFS algorithm
 3. Applications of BFS algorithm

That's All For Now