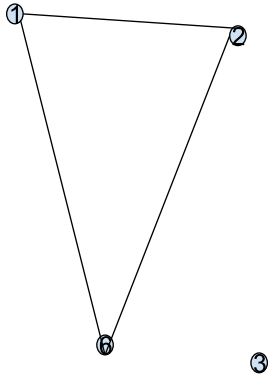


Introduction to Graphs



A graph is a bunch of **vertices** (V) connected by **edges** (E).

In the example above:

V is {1,2,3,4,5,6}. Each vertex, or **node**, belongs to the set V.

E is {(1,2), (1,6), (2,6), (4,5)}. Each edge belongs to the set E.

Vertex 3 is **isolated** since it connects to nothing.

Incident: the vertices of an edge. Vertex 2 and 6 are incident to the (2,6) edge.

Degree: number of edges connecting to a vertex. Vertex

1 has a degree of 2, since it is incident to the edges (1,2) and (1,6).

Two vertices are **adjacent** if an edge connects them. Vertex 4 is adjacent to vertex 5.

If each edge had a number with it (like a cost or length of the edge), the graph is **edge weighted**.

If each vertex had a number with it (like height of that vertex), the graph is **vertex weighted**.

Self loop: an edge looks like (x,x) (the vertex has an edge that loops on itself).

How many edges are possible?

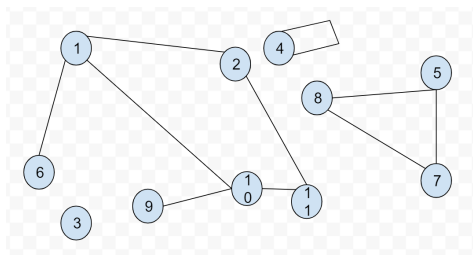
- If there are N vertices, there can be $\frac{N(N-1)}{2}$ edges.
 - If there are a lot of edges, the graph is **dense**.
 - If there are few edges, the graph is **sparse**.

Undirected graph: graph where the edges go both ways (no arrows).

Directed graph: graph where the edges go only 1 way. (One-way arrows on the lines). These edges are called **arcs**.

- In directed graphs, the **out degree** for a vertex is how many arcs start from the vertex (point outward).
- **In degree** is how many arcs end at the vertex (point inward).

Paths and Cycles



Notice the Edge (4,4) is a self-loop.

A **path** is a sequence of traveling along edges to get from vertex A to vertex B . For example, $(6,1,2,11,10,9)$ is a path. That path traveled along 5 edges, so the length of the path is 5. This path contains the vertices $6,1,2,11,10$, and 9 and the edges $(6,1)$, $(1,2)$, $(2,11)$, $(11,10)$, and $(10,9)$.

If a path can take you from vertex A to B , vertex B is **reachable** from vertex A . For example, Vertex 11 is reachable from vertex 1.

A **simple path** contains no vertex more than once.

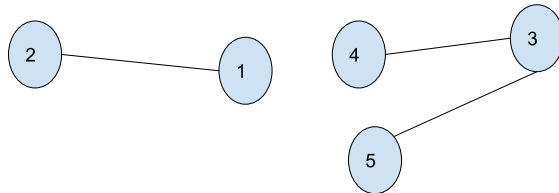
A **cycle** is a path that starts and ends at the same vertex. $(7,5,8,7)$ is a cycle.

- A cycle cannot be a simple path, because the start/end vertex appears twice.
- However, a **simple cycle** contains no vertex *except* the start/end more than once.

Representing Graphs

Three main ways.

For a graph with N vertices.



| | Edge List | Adjacency Matrix | Adjacency List | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----------------|---|---|--|----------------|----------------|----------------|----------------|----------------|----------------|---|---|---|---|---|----------------|---|---|---|---|---|----------------|---|---|---|---|---|----------------|---|---|---|---|---|----------------|---|---|---|---|---|--|----------------|---|----------------|---|----------------|-----|----------------|---|----------------|---|
| What is it? | A list of every edge. AKA: list of all the vertices paired up by a line. | 2d array, size N by N. Entry [i][j] is 1 if there is an edge (i,j), otherwise 0. | List of length N. The <i>ith</i> entry in the list is a list of all adjacent vertices to <i>ith</i> vertex | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Example | (1,2), (3,4), (3,5) | <div><i>Matrix A:</i><table><tr><td></td><td>V₁</td><td>V₂</td><td>V₃</td><td>V₄</td><td>V₅</td></tr><tr><td>V₁</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>V₂</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>V₃</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr><tr><td>V₄</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>V₅</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr></table></div> | | V ₁ | V ₂ | V ₃ | V ₄ | V ₅ | V ₁ | 0 | 1 | 0 | 0 | 0 | V ₂ | 1 | 0 | 0 | 0 | 0 | V ₃ | 0 | 0 | 0 | 1 | 1 | V ₄ | 0 | 0 | 1 | 0 | 0 | V ₅ | 0 | 0 | 1 | 0 | 0 | <table><tr><td>V₁</td><td>2</td></tr><tr><td>V₂</td><td>1</td></tr><tr><td>V₃</td><td>4,5</td></tr><tr><td>V₄</td><td>3</td></tr><tr><td>V₅</td><td>3</td></tr></table> | V ₁ | 2 | V ₂ | 1 | V ₃ | 4,5 | V ₄ | 3 | V ₅ | 3 |
| | V ₁ | V ₂ | V ₃ | V ₄ | V ₅ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| V ₁ | 0 | 1 | 0 | 0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| V ₂ | 1 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| V ₃ | 0 | 0 | 0 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| V ₄ | 0 | 0 | 1 | 0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| V ₅ | 0 | 0 | 1 | 0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| V ₁ | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| V ₂ | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| V ₃ | 4,5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| V ₄ | 3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| V ₅ | 3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| In Java: | ArrayList<Pair> | int[N][N] | ArrayList<ArrayList<Integer>> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | | |
|-----------|--|---|-----------------|
| Pros/Cons | (+)Easy to code (-)Finding degree is hard (-)Deleting edge could be hard | (+)Easy to code (-)Lots of space (+)Finding adjacent easy | (-)Hard to code |
|-----------|--|---|-----------------|

We would need to make the pair class, which holds two instance variables. Or we could store it in an ArrayList of arrays with 2 elements, etc.

Graph Representation, Method 4: ***Java is slow, so this is one way to represent the graph for Java and not get time limit problems:

<https://docs.google.com/document/d/1gXvlyYXNDRkNfVQhsIcLEIgHbQGvw1hEM9z9OwQMZGg/edit?usp=sharing>

“It is sometimes helpful to use the fact that the (i,j) entry of the adjacency matrix raised to the k -th power gives the number of paths from vertex i to vertex j consisting of exactly k edges / length k .”

- The 2D array is an $N \times N$ matrix; let's call it A .
- To find the number of paths of length 2, let's do A times A , or find A^2 .
- <https://www.mathsisfun.com/algebra/matrix-multiplying.html>

In the example above, $A \cdot A$ yields:

| | C_1 | C_2 | C_3 | C_4 | C_5 |
|---|-------|-------|-------|-------|-------|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 0 |
| 3 | 0 | 0 | 2 | 0 | 0 |
| 4 | 0 | 0 | 0 | 1 | 1 |
| 5 | 0 | 0 | 0 | 1 | 1 |

$i = 1, j = 1$. From the graph, we can see that the *only* path of length 2 that goes from vertex 1 to vertex 1 (a cycle) is $(1,2,1)$ (It takes two edges $(1,2)$ and then $(1,2)$ again, this time starting from V_2).

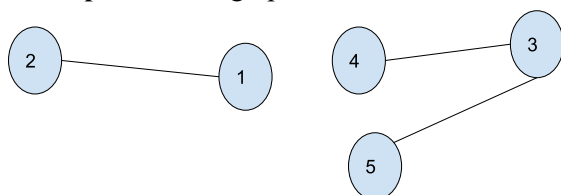
$i = 3, j = 3$. We can see that going from vertex 3 to vertex 3 with a path length of 2 is achievable in 2 different ways. On the graph, we see this is $(3,4,3)$ and $(3,5,3)$.

$i = 4, j = 5$. Going from vertex 4 to vertex 5 with a path length of 2 only happens once. Visually, we see this is $(4,3,5)$.

Connectedness

A graph is **connected** if there is a path from every vertex to every other vertex.

A **component** of a graph is the maximal subset of a graph that acts connected.



This graph is not connected. $\{1,2\}$ and $\{3,4,5\}$ are components of the graph. $\{3,4\}$ is not a component because it could be made bigger (add 5) and still be all connected.

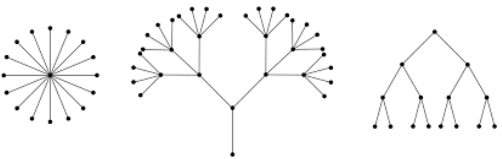
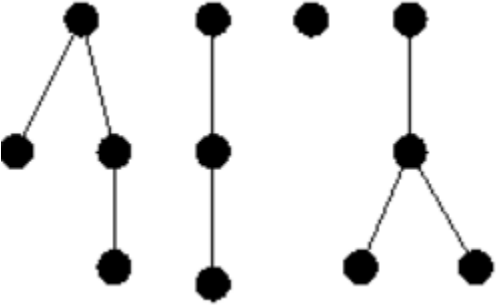
For directed graphs:

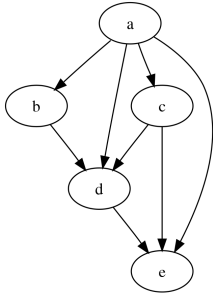
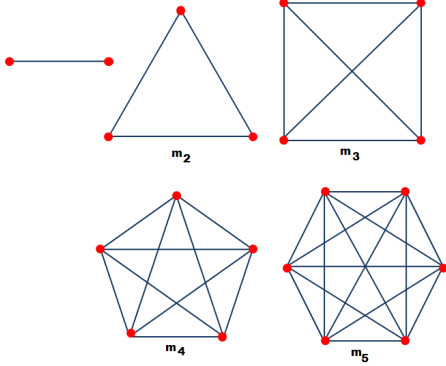
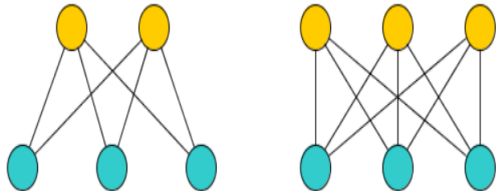
- **Strongly connected** if for all pairs of vertices A and B there is a path from A to B and B to A .
- **Strongly Connected Components** are the maximal subsets of graphs that act connected.

For a Graph G with vertices set V and Edge set E :

- G' is a **subset** if G' vertices set V' is a subset of V and G' 's edges set E' is a subset of E .
 - A possible: A G' from the example above could be $V' = \{2,1,4,3\}$ and $E' = \{\}$.
 - Notice the pairs of vertices that normally have edges do not have to be in this subgraph!
- If you take a subset of V and call it V' , and then you also take in all the edges from E that include vertices in V' , the subgraph G' is **induced** by V' .
 - Given the $V' = \{1,4,3,5\}$, it induces the G' with $E' = \{(3,5), (3,4)\}$.
 - The pairs of vertices in V' that have edges normally *must* have those edges in the induced subgraph!

Special Graphs

| Properties | Name | Image |
|---|--------|--|
| No cycles, connected. It's nice to draw these as rooted trees. Look at the rightmost and middle examples. Root is the top node, and each node has children (adjacent vertices farther away from the node), but only one parent (adjacent node closest to root). | Tree |  <p>These are 3 separate graphs.</p> |
| No cycles | Forest |  <p>This is all 1 graph</p> |

| | | |
|--|-----------|---|
| No cycles, directed | dag |  <p>This is 1 graph</p> |
| Edge between all pairs of vertices. | Complete |  <p>These are 5 separate graphs</p> |
| Graph splits into 2 groups of vertices. Every edge must have a vertex from both groups. (No edge has vertices from 1 group only) | Bipartite |  <p>These are 2 separate graphs</p> |

Example: Finding Number of Paths with Given Length

Given an undirected graph with N vertices, represented with an adjacency matrix, find the pair of vertices (i,j) for which there are the most number of paths that are of length k , start at vertex i , and end at vertex j .

All we need to do is raise the adjacency matrix to the k th power, and call that matrix A . Then report the (i,j) location of the maximum element in the matrix A .