Algorithmic Approaches for Biological Data, Lecture #16

Katherine St. John

City University of New York
American Museum of Natural History

30 March 2016

Outline

- Networks & Graphs

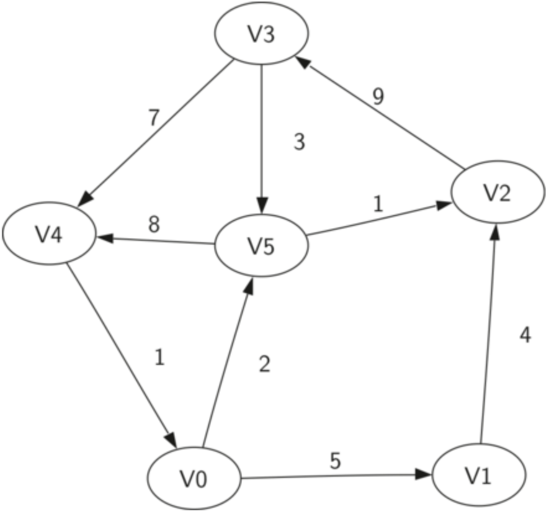
Outline

- Networks & Graphs
- Standard Representations: Adjacency Lists and Adjacency Matrices

Outline

- Networks & Graphs
- Standard Representations: Adjacency Lists and Adjacency Matrices
- Reframing Biology Questions

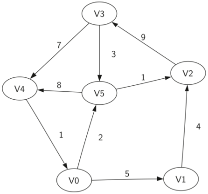
Networks & Graphs



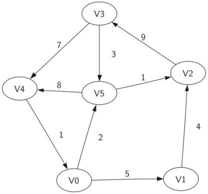
Problem Solving with Algorithms and Data Structures

Networks & Graphs

- Graphs (networks)

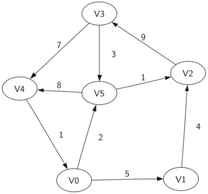


Networks & Graphs



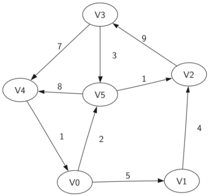
- Graphs (networks) have vertices (nodes)

Networks & Graphs



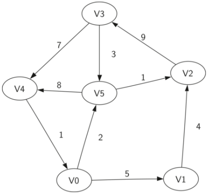
- Graphs (networks) have vertices (nodes) and edges (lines, branches) connecting them.

Networks & Graphs



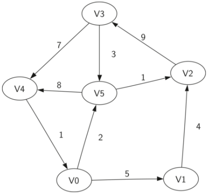
- Graphs (networks) have vertices (nodes) and edges (lines, branches) connecting them.
- Edges can have weights.

Networks & Graphs



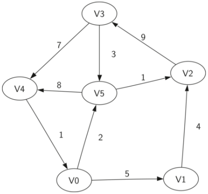
- Graphs (networks) have vertices (nodes) and edges (lines, branches) connecting them.
- Edges can have weights.
- Widely used model in math, routing, biology, etc.

Networks & Graphs



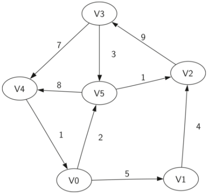
- Graphs (networks) have vertices (nodes) and edges (lines, branches) connecting them.
- Edges can have weights.
- Widely used model in math, routing, biology, etc.
- In example: $G = (V, E)$ where:

Networks & Graphs



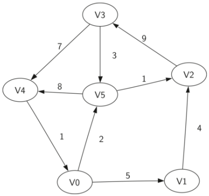
- Graphs (networks) have vertices (nodes) and edges (lines, branches) connecting them.
- Edges can have weights.
- Widely used model in math, routing, biology, etc.
- In example: $G = (V, E)$ where:
  - $V = \{V0, V1, V2, V3, V4, V5\}$

Networks & Graphs



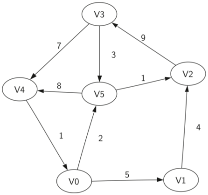
- Graphs (networks) have vertices (nodes) and edges (lines, branches) connecting them.
- Edges can have weights.
- Widely used model in math, routing, biology, etc.
- In example: $G = (V, E)$ where:
  - $V = \{V0, V1, V2, V3, V4, V5\}$
  - $E = \{(V0, V1, 5), (V1, V2, 4), (V2, V3, 9), (V3, V4, 7), (V4, V0, 1), (V0, V5, 2), (V5, V4, 8), (V3, V5, 3), (V5, V2, 1)\}$

Networks & Graphs



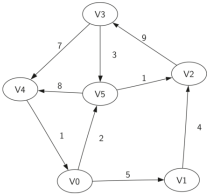
- Graphs (networks) have vertices (nodes) and edges (lines, branches) connecting them.
- Edges can have weights.
- Widely used model in math, routing, biology, etc.
- In example: $G = (V, E)$ where:
  - $V = \{V0, V1, V2, V3, V4, V5\}$
  - $E = \{(V0, V1, 5), (V1, V2, 4), (V2, V3, 9), (V3, V4, 7), (V4, V0, 1), (V0, V5, 2), (V5, V4, 8), (V3, V5, 3), (V5, V2, 1)\}$
  - Since edges have a direction, called a directed graph.

Networks & Graphs



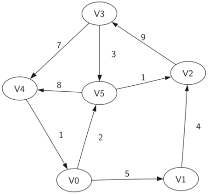
- Paths are a sequence of vertices in graph, each connected to the next by an edge.

Networks & Graphs



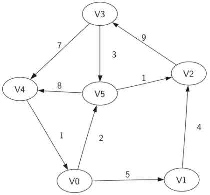
- Paths are a sequence of vertices in graph, each connected to the next by an edge.
  Example: there is a path from $V1$ to $V4$.

Networks & Graphs



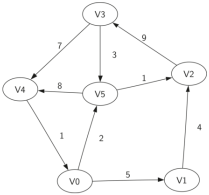
- Paths are a sequence of vertices in graph, each connected to the next by an edge.
  Example: there is a path from $V1$ to $V4$.
- A cycle is a path that starts and ends at the same vertex.

Networks & Graphs



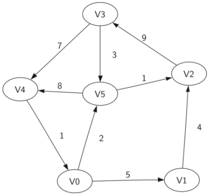
- Paths are a sequence of vertices in graph, each connected to the next by an edge.
  Example: there is a path from $V1$ to $V4$.
- A cycle is a path that starts and ends at the same vertex.
  Example: $(V5, V2, V3, V5)$ is a cycle.

Networks & Graphs



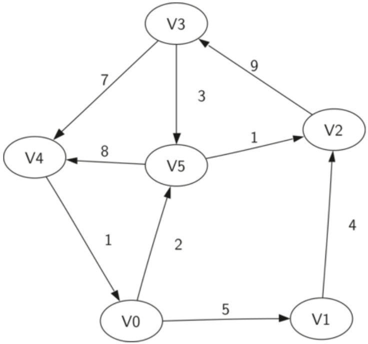
- Paths are a sequence of vertices in graph, each connected to the next by an edge.
  Example: there is a path from $V1$ to $V4$.
- A cycle is a path that starts and ends at the same vertex.
  Example: $(V5, V2, V3, V5)$ is a cycle.
- A graph with no cycles is called an acyclic graph.

Networks & Graphs



- Paths are a sequence of vertices in graph, each connected to the next by an edge.
  Example: there is a path from $V1$ to $V4$.
- A cycle is a path that starts and ends at the same vertex.
  Example: $(V5, V2, V3, V5)$ is a cycle.
- A graph with no cycles is called an acyclic graph.
- A directed graph with no cycles is called a directed acyclic graph (DAG).

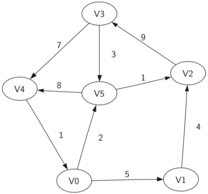
Representing Graphs in the Computer: Adjacency Matrix



	V0	V1	V2	V3	V4	V5
V0		5				2
V1			4			
V2				9		
V3					7	3
V4	1					
V5			1		8	

Problem Solving with Algorithms and Data Structures

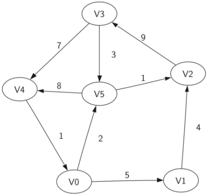
Representing Graphs in the Computer: Adjacency Matrix



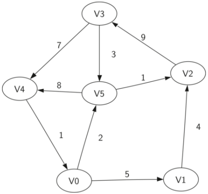
	V0	V1	V2	V3	V4	V5
V0		5				2
V1			4			
V2				9		
V3					7	3
V4	1					
V5			1		8	

- In Python, can use a list of lists or a `numpy` array.

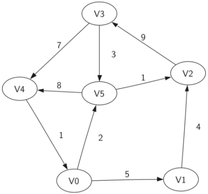
Representing Graphs in the Computer: Adjacency Matrix



	V0	V1	V2	V3	V4	V5
V0		5				2
V1			4			
V2				9		
V3					7	3
V4	1					
V5			1		8	

- In Python, can use a list of lists or a `numpy` array.
- ```
  import numpy as np
  adjMatrix = np.zeros(6,6)
  adjMatix[0,1] = 5
  adjMatix[0,5] = 2
  adjMatix[1,2] = 4
  adjMatix[2,3] = 9
  adjMatix[3,4] = 7
  adjMatix[3,5] = 3
  adjMatix[4,0] = 1
  adjMatix[5,2] = 1
  adjMatix[5,4] = 8
  ```
  ```

Representing Graphs in the Computer: Adjacency Matrix



	V0	V1	V2	V3	V4	V5
V0		5				2
V1			4			
V2				9		
V3					7	3
V4	1					
V5			1		8	

- In Python, can use a list of lists or a `numpy` array.
- ```
  import numpy as np
  adjMatrix = np.zeros(6,6)
  adjMatix[0,1] = 5
  adjMatix[0,5] = 2
  adjMatix[1,2] = 4
  adjMatix[2,3] = 9
  adjMatix[3,4] = 7
  adjMatix[3,5] = 3
  adjMatix[4,0] = 1
  adjMatix[5,2] = 1
  adjMatix[5,4] = 8
  ```
- Need to keep track of the node names separately.
  ```

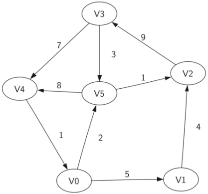
Representing Graphs in the Computer: Adjacency Matrix



	V0	V1	V2	V3	V4	V5
V0		5				2
V1			4			
V2				9		
V3					7	3
V4	1					
V5			1		8	

- Advantages:

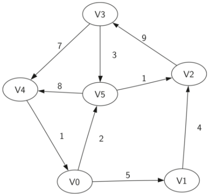
Representing Graphs in the Computer: Adjacency Matrix



	V0	V1	V2	V3	V4	V5
V0		5				2
V1			4			
V2				9		
V3					7	3
V4	1					
V5			1		8	

- Advantages:
  - Checking if an edge occurs is quick.

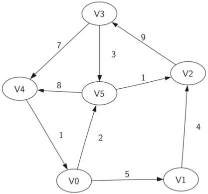
Representing Graphs in the Computer: Adjacency Matrix



	V0	V1	V2	V3	V4	V5
V0		5				2
V1			4			
V2				9		
V3					7	3
V4	1					
V5			1		8	

- Advantages:
  - Checking if an edge occurs is quick.
  - Can check connectivity by matrix multiplication (explained in lab).

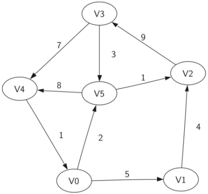
Representing Graphs in the Computer: Adjacency Matrix



	V0	V1	V2	V3	V4	V5
V0		5				2
V1			4			
V2				9		
V3					7	3
V4	1					
V5			1		8	

- Advantages:
  - Checking if an edge occurs is quick.
  - Can check connectivity by matrix multiplication (explained in lab).
- Disadvantages:

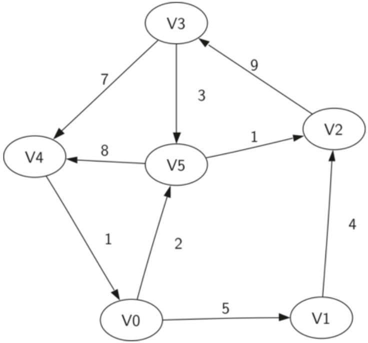
Representing Graphs in the Computer: Adjacency Matrix

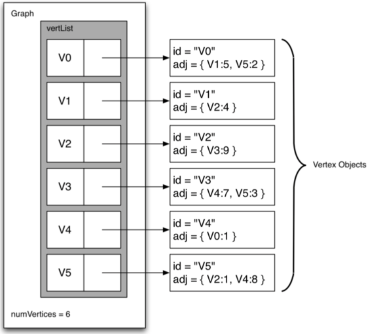


	V0	V1	V2	V3	V4	V5
V0		5				2
V1			4			
V2				9		
V3					7	3
V4	1					
V5			1		8	

- Advantages:
  - Checking if an edge occurs is quick.
  - Can check connectivity by matrix multiplication (explained in lab).
- Disadvantages:
  - Always the same size ($n \times n$) even if there are few edges.

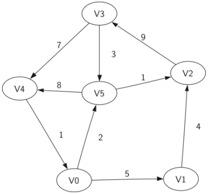
Representing Graphs in the Computer: Adjacency List

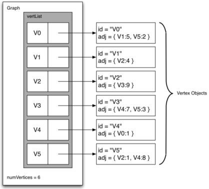




Problem Solving with Algorithms and Data Structures

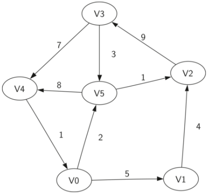
Representing Graphs in the Computer: Adjacency List

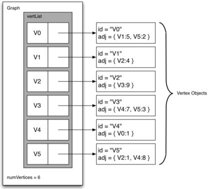




- In Python, can use a dictionary to store lists of tuples.

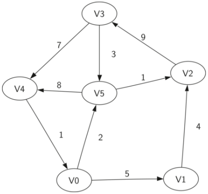
Representing Graphs in the Computer: Adjacency List

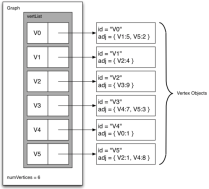




- In Python, can use a dictionary to store lists of tuples.
-
```
import numpy as np
adjList = {}
adjList["V0"] = [("V1",5), ("V5",2)]
adjList["V1"] = [("V2",4)]
adjList["V2"] = [("V3",9)]
adjList["V3"] = [("V4",7), ("V5",3)]
adjList["V4"] = [("V0",1)]
adjList["V5"] = [("V2",1), ("V4",8)]
```

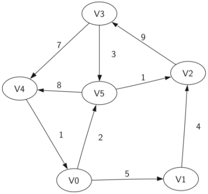
Representing Graphs in the Computer: Adjacency List

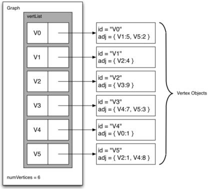




- In Python, can use a dictionary to store lists of tuples.
-
```
import numpy as np
adjList = {}
adjList["V0"] = [("V1",5), ("V5",2)]
adjList["V1"] = [("V2",4)]
adjList["V2"] = [("V3",9)]
adjList["V3"] = [("V4",7), ("V5",3)]
adjList["V4"] = [("V0",1)]
adjList["V5"] = [("V2",1), ("V4",8)]
```
- Can look up each list of adjacencies in the dictionary using the vertex label as the key.

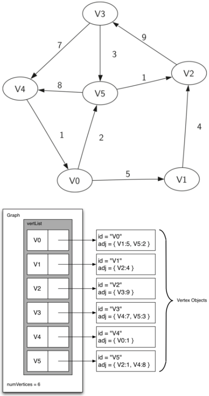
Representing Graphs in the Computer: Adjacency List





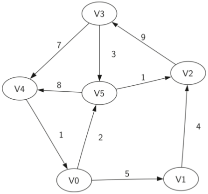
- Advantages:

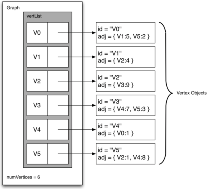
Representing Graphs in the Computer: Adjacency List



- Advantages:
  - More space efficient for sparsely connected graphs

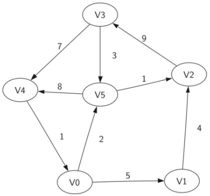
Representing Graphs in the Computer: Adjacency List

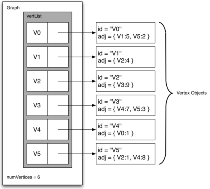




- Advantages:
  - More space efficient for sparsely connected graphs
- Disadvantages:

Representing Graphs in the Computer: Adjacency List

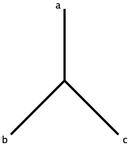




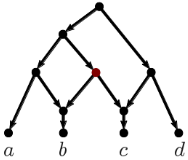
- Advantages:
  - More space efficient for sparsely connected graphs
- Disadvantages:
  - Could be costly to find adjacencies if a vertex has many neighbors.

In Pairs

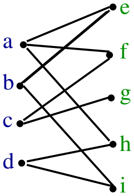
In pairs/triples, represent the following graphs in the computer:



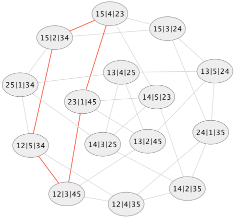
(1)



(2)

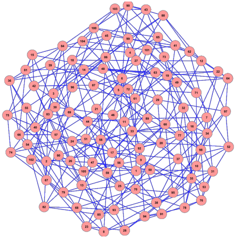


(3)



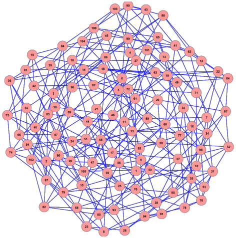
(4)

Recap



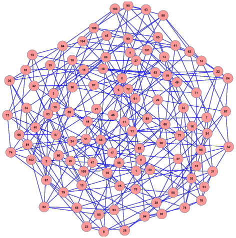
- Lab today: connectivity & storing trees

Recap



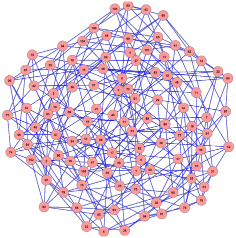
- Lab today: connectivity & storing trees
- Using `networkx` in lab today
  (for displaying graphs).

Recap



- Lab today: connectivity & storing trees
- Using `networkx` in lab today
  (for displaying graphs).
- Email lab reports to `kstjohn@amnh.org`

Recap



- Lab today: connectivity & storing trees
- Using `networkx` in lab today (for displaying graphs).
- Email lab reports to `kstjohn@amnh.org`
- Challenges available at `rosalind.info`