Representing Graphs

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

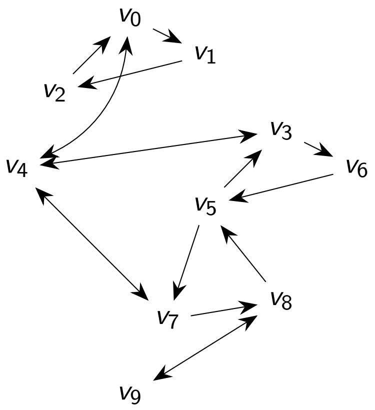
Programming, Data Structures and Algorithms using Python

Week 4

Working with graphs

- Graph $G = (V, E)$
  - V — set of vertices
  - $E \subseteq V \times V$ — set of edges
- A path is a sequence of vertices $v_1, v_2, \ldots, v_k$ connected by edges
  - For $1 \leq i < k$, $(v_i, v_{i+1}) \in E$
- Vertex v is reachable from vertex u if there is a path from u to v

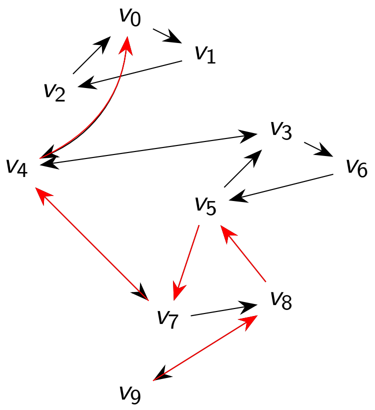
Airline routes



Working with graphs

- Graph $G = (V, E)$
  - V — set of vertices
  - $E \subseteq V \times V$ — set of edges
- A path is a sequence of vertices $v_1, v_2, \ldots, v_k$ connected by edges
  - For $1 \leq i < k$, $(v_i, v_{i+1}) \in E$
- Vertex v is reachable from vertex u if there is a path from u to v
- Looking at the picture of G, we can "see" that v_0 is reachable from v_9
- How do we represent this picture so that we can compute reachability?

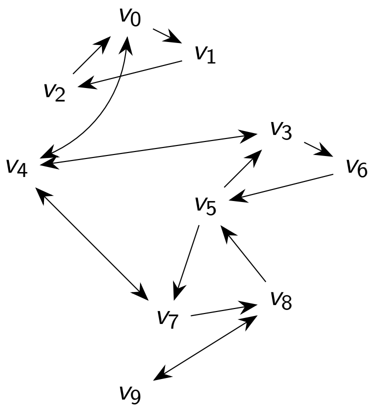
Airline routes



Adjacency matrix

- Let $|V| = n$
  - Assume $V = \{0, 1, \ldots, n-1\}$
  - Use a table to map actual vertex "names" to this set

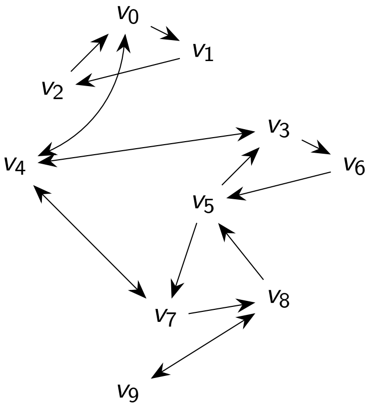
Airline routes



Adjacency matrix

- Let $|V| = n$
  - Assume $V = \{0, 1, \ldots, n-1\}$
  - Use a table to map actual vertex "names" to this set
- Edges are now pairs (i, j), where $0 \leq i, j < n$
  - Usually assume $i \neq j$, no self loops

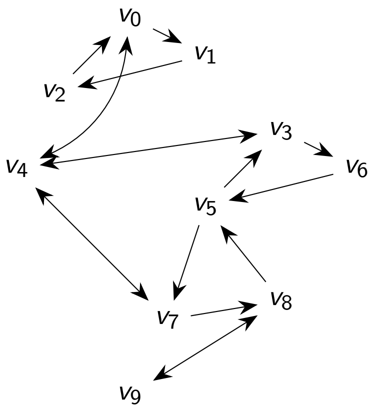
Airline routes



Adjacency matrix

- Let $|V| = n$
  - Assume $V = \{0, 1, \ldots, n-1\}$
  - Use a table to map actual vertex "names" to this set
- Edges are now pairs (i, j), where $0 \leq i, j < n$
  - Usually assume $i \neq j$, no self loops
- Adjacency matrix
  - Rows and columns numbered $\{0, 1, \ldots, n-1\}$
  - $A[i, j] = 1$ if $(i, j) \in E$

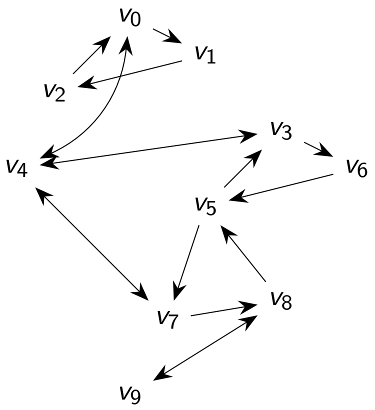
Airline routes



Adjacency matrix

- Adjacency matrix
  - Rows and columns numbered $\{0, 1, \ldots, n-1\}$
  - $A[i,j] = 1$ if $(i,j) \in E$

Airline routes



Adjacency matrix

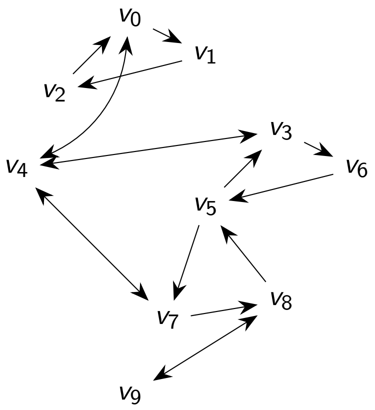
- Adjacency matrix
  - Rows and columns numbered $\{0, 1, \ldots, n-1\}$
  - $A[i,j] = 1$ if $(i,j) \in E$

```
edges = [(0,1),(0,4),(1,2),(2,0),
         (3,4),(3,6),(4,0),(4,3),
         (4,7),(5,3),(5,7),
         (6,5),(7,4),(7,8),
         (8,5),(8,9),(9,8)]

import numpy as np
A = np.zeros(shape=(10,10))

for (i,j) in edges:
    A[i,j] = 1
```

Airline routes

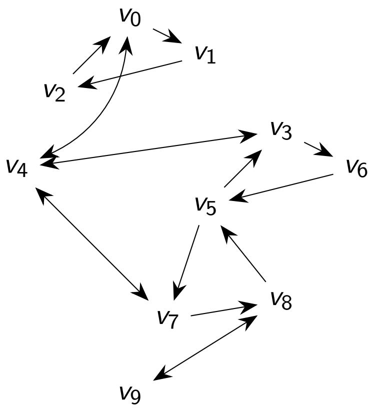


Adjacency matrix

- Adjacency matrix
  - Rows and columns numbered $\{0, 1, \ldots, n-1\}$
  - $A[i,j] = 1$ if $(i,j) \in E$

	0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	1	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	0	1	0	0
6	0	0	0	0	0	1	0	0	0	0
7	0	0	0	0	1	0	0	0	1	0
8	0	0	0	0	0	1	0	0	0	1
9	0	0	0	0	0	0	0	0	1	0

Airline routes

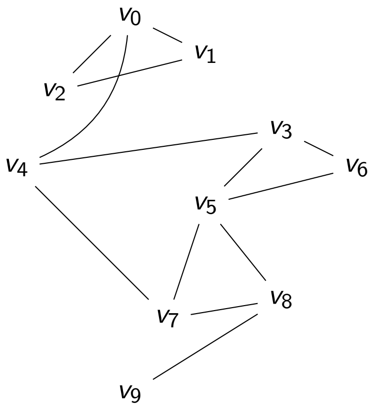


Adjacency matrix

- Undirected graph
  - $A[i,j] = 1$ iff $A[j,i] = 1$
  - Symmetric across main diagonal

	0	1	2	3	4	5	6	7	8	9
0	0	1	1	0	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	1	1	1	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	1	0	0	1	0
8	0	0	0	0	0	1	0	1	0	1
9	0	0	0	0	0	0	0	0	1	0

Airline routes, all routes bidirectional



Computing with the adjacency matrix

- Neighbours of i — column j with entry 1
  - Scan row i to identify neighbours of i
  - Neighbours of 6 are $[3, 5]$

```
def neighbours(AMat,i):
    nbrs = []
    (rows,cols) = AMat.shape
    for j in range(cols):
        if AMat[i,j] == 1:
            nbrs.append(j)
    return(nbrs)
```

```
neighbours(A,7)
```

```
[4, 5, 8]
```

Undirected airline routes

	0	1	2	3	4	5	6	7	8	9
0	0	1	1	0	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	1	1	1	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	1	0	0	1	0
8	0	0	0	0	0	1	0	1	0	1
9	0	0	0	0	0	0	0	0	1	0

Computing with the adjacency matrix

- Neighbours of i — column j with entry 1
  - Scan row i to identify neighbours of i
  - Neighbours of 6 are $[3, 5]$

- Directed graph

Directed airline routes

	0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	1	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	0	1	0	0
6	0	0	0	0	0	1	0	0	0	0
7	0	0	0	0	1	0	0	0	1	0
8	0	0	0	0	0	1	0	0	0	1
9	0	0	0	0	0	0	0	0	1	0

Computing with the adjacency matrix

- Neighbours of i — column j with entry 1
  - Scan row i to identify neighbours of i
  - Neighbours of 6 are $[3, 5]$
- Directed graph
  - Rows represent outgoing edges

Directed airline routes

	0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	1	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	0	1	0	0
6	0	0	0	0	0	1	0	0	0	0
7	0	0	0	0	1	0	0	0	1	0
8	0	0	0	0	0	1	0	0	0	1
9	0	0	0	0	0	0	0	0	1	0

Computing with the adjacency matrix

- Neighbours of i — column j with entry 1
  - Scan row i to identify neighbours of i
  - Neighbours of 6 are $[3, 5]$
- Directed graph
  - Rows represent outgoing edges
  - Columns represent incoming edges

Directed airline routes

	0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	1	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	0	1	0	0
6	0	0	0	0	0	1	0	0	0	0
7	0	0	0	0	1	0	0	0	1	0
8	0	0	0	0	0	1	0	0	0	1
9	0	0	0	0	0	0	0	0	1	0

Computing with the adjacency matrix

- Neighbours of i — column j with entry 1
  - Scan row i to identify neighbours of i
  - Neighbours of 6 are $[3, 5]$
- Directed graph
  - Rows represent outgoing edges
  - Columns represent incoming edges
- Degree of a vertex i
  - Number of edges incident on i

    $\text{degree}(6) = 2$

Undirected airline routes

	0	1	2	3	4	5	6	7	8	9
0	0	1	1	0	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	1	1	1	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	1	0	0	1	0
8	0	0	0	0	0	1	0	1	0	1
9	0	0	0	0	0	0	0	0	1	0

Computing with the adjacency matrix

- Neighbours of i — column j with entry 1
  - Scan row i to identify neighbours of i
  - Neighbours of 6 are $[3, 5]$
- Directed graph
  - Rows represent outgoing edges
  - Columns represent incoming edges
- Degree of a vertex i
  - Number of edges incident on i

    $\text{degree}(6) = 2$
  - For directed graphs, outdegree and indegree

    $\text{indegree}(6) = 1$, $\text{outdegree}(6) = 1$

Directed airline routes

	0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	1	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	0	1	0	0
6	0	0	0	0	0	1	0	0	0	0
7	0	0	0	0	1	0	0	0	1	0
8	0	0	0	0	0	1	0	0	0	1
9	0	0	0	0	0	0	0	0	1	0

Checking reachability

- Is Delhi (0) reachable from Madurai (9)?

Undirected airline routes

	0	1	2	3	4	5	6	7	8	9
0	0	1	1	0	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	1	1	1	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	1	0	0	1	0
8	0	0	0	0	0	1	0	1	0	1
9	0	0	0	0	0	0	0	0	1	0

Checking reachability

- Is Delhi (0) reachable from Madurai (9)?
- Mark 9 as reachable

Undirected airline routes

	0	1	2	3	4	5	6	7	8	9
0	0	1	1	0	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	1	1	1	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	1	0	0	1	0
8	0	0	0	0	0	1	0	1	0	1
9	0	0	0	0	0	0	0	0	1	0

Checking reachability

- Is Delhi (0) reachable from Madurai (9)?
- Mark 9 as reachable
- Mark each neighbour of 9 as reachable

Undirected airline routes

	0	1	2	3	4	5	6	7	8	9
0	0	1	1	0	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	1	1	1	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	1	0	0	1	0
8	0	0	0	0	0	1	0	1	0	1
9	0	0	0	0	0	0	0	0	1	0

Checking reachability

- Is Delhi (0) reachable from Madurai (9)?
- Mark 9 as reachable
- Mark each neighbour of 9 as reachable

Undirected airline routes

	0	1	2	3	4	5	6	7	8	9
0	0	1	1	0	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	1	1	1	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	1	0	0	1	0
8	0	0	0	0	0	1	0	1	0	1
9	0	0	0	0	0	0	0	0	1	0

Checking reachability

- Is Delhi (0) reachable from Madurai (9)?
- Mark 9 as reachable
- Mark each neighbour of 9 as reachable
- Systematically mark neighbours of marked vertices

Undirected airline routes

	0	1	2	3	4	5	6	7	8	9
0	0	1	1	0	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	1	1	1	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	1	0	0	1	0
8	0	0	0	0	0	1	0	1	0	1
9	0	0	0	0	0	0	0	0	1	0

Checking reachability

- Is Delhi (0) reachable from Madurai (9)?
- Mark 9 as reachable
- Mark each neighbour of 9 as reachable
- Systematically mark neighbours of marked vertices

Undirected airline routes

	0	1	2	3	4	5	6	7	8	9
0	0	1	1	0	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	1	1	1	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	1	0	0	1	0
8	0	0	0	0	0	1	0	1	0	1
9	0	0	0	0	0	0	0	0	1	0

Checking reachability

- Is Delhi (0) reachable from Madurai (9)?
- Mark 9 as reachable
- Mark each neighbour of 9 as reachable
- Systematically mark neighbours of marked vertices

Undirected airline routes

	0	1	2	3	4	5	6	7	8	9
0	0	1	1	0	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	1	1	1	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	1	0	0	1	0
8	0	0	0	0	0	1	0	1	0	1
9	0	0	0	0	0	0	0	0	1	0

Checking reachability

- Is Delhi (0) reachable from Madurai (9)?
- Mark 9 as reachable
- Mark each neighbour of 9 as reachable
- Systematically mark neighbours of marked vertices

Undirected airline routes

	0	1	2	3	4	5	6	7	8	9
0	0	1	1	0	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	1	1	1	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	1	0	0	1	0
8	0	0	0	0	0	1	0	1	0	1
9	0	0	0	0	0	0	0	0	1	0

Checking reachability

- Is Delhi (0) reachable from Madurai (9)?
- Mark 9 as reachable
- Mark each neighbour of 9 as reachable
- Systematically mark neighbours of marked vertices

Undirected airline routes

	0	1	2	3	4	5	6	7	8	9
0	0	1	1	0	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	1	1	1	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	1	0	0	1	0
8	0	0	0	0	0	1	0	1	0	1
9	0	0	0	0	0	0	0	0	1	0

Checking reachability

- Is Delhi (0) reachable from Madurai (9)?
- Mark 9 as reachable
- Mark each neighbour of 9 as reachable
- Systematically mark neighbours of marked vertices

Undirected airline routes

	0	1	2	3	4	5	6	7	8	9
0	0	1	1	0	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	1	1	1	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	1	0	0	1	0
8	0	0	0	0	0	1	0	1	0	1
9	0	0	0	0	0	0	0	0	1	0

Checking reachability

- Is Delhi (0) reachable from Madurai (9)?
- Mark 9 as reachable
- Mark each neighbour of 9 as reachable
- Systematically mark neighbours of marked vertices

Undirected airline routes

	0	1	2	3	4	5	6	7	8	9
0	0	1	1	0	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	1	1	1	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	1	0	0	1	0
8	0	0	0	0	0	1	0	1	0	1
9	0	0	0	0	0	0	0	0	1	0

Checking reachability

- Is Delhi (0) reachable from Madurai (9)?
- Mark 9 as reachable
- Mark each neighbour of 9 as reachable
- Systematically mark neighbours of marked vertices
- Stop when 0 becomes marked

Undirected airline routes

	0	1	2	3	4	5	6	7	8	9
0	0	1	1	0	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	1	1	1	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	1	0	0	1	0
8	0	0	0	0	0	1	0	1	0	1
9	0	0	0	0	0	0	0	0	1	0

Checking reachability

- Is Delhi (0) reachable from Madurai (9)?
- Mark 9 as reachable
- Mark each neighbour of 9 as reachable
- Systematically mark neighbours of marked vertices
- Stop when 0 becomes marked
- If marking process stops without target becoming marked, the target is unreachable

Undirected airline routes

	0	1	2	3	4	5	6	7	8	9
0	0	1	1	0	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	1	1	1	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	1	0	0	1	0
8	0	0	0	0	0	1	0	1	0	1
9	0	0	0	0	0	0	0	0	1	0

Checking reachability

- Mark source vertex as reachable
- Systematically mark neighbours of marked vertices
- Stop when target becomes marked

Undirected airline routes

	0	1	2	3	4	5	6	7	8	9
0	0	1	1	0	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	1	1	1	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	1	0	0	1	0
8	0	0	0	0	0	1	0	1	0	1
9	0	0	0	0	0	0	0	0	1	0

Checking reachability

- Mark source vertex as reachable
- Systematically mark neighbours of marked vertices
- Stop when target becomes marked
- Need a strategy to systematically explore marked neighbours

Undirected airline routes

	0	1	2	3	4	5	6	7	8	9
0	0	1	1	0	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	1	1	1	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	1	0	0	1	0
8	0	0	0	0	0	1	0	1	0	1
9	0	0	0	0	0	0	0	0	1	0

Checking reachability

- Mark source vertex as reachable
- Systematically mark neighbours of marked vertices
- Stop when target becomes marked
- Need a strategy to systematically explore marked neighbours
- Two primary strategies
  - Breadth first — propagate marks in "layers"
  - Depth first — explore a path till it dies out, then backtrack

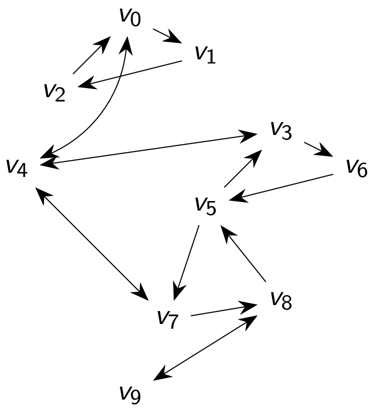
Undirected airline routes

	0	1	2	3	4	5	6	7	8	9
0	0	1	1	0	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	1	1	1	0
6	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	1	1	0	0	1	0
8	0	0	0	0	0	1	0	1	0	1
9	0	0	0	0	0	0	0	0	1	0

Adjacency lists

- Adjacency matrix has many 0's
  - Size is n^2, regardless of number of edges
  - Undirected graph: $|E| \leq n(n-1)/2$
  - Directed graph: $|E| \leq n(n-1)$
  - Typically $|E|$ much less than n^2

Airline routes



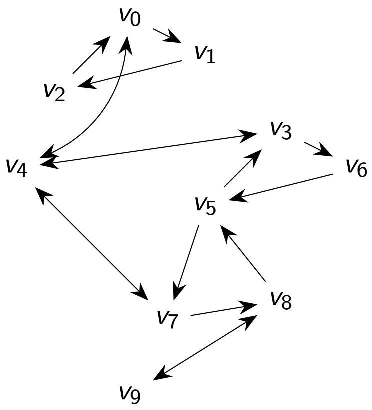
Adjacency lists

- Adjacency matrix has many 0's
  - Size is n^2, regardless of number of edges
  - Undirected graph: $|E| \leq n(n-1)/2$
  - Directed graph: $|E| \leq n(n-1)$
  - Typically $|E|$ much less than n^2
- Adjacency list
  - List of neighbours for each vertex

0	[1,4]
1	[2]
2	[0]
3	[4,6]
4	[0,3,7]

5	[3,7]
6	[5]
7	[4,8]
8	[5,9]
9	[8]

Airline routes



Adjacency lists

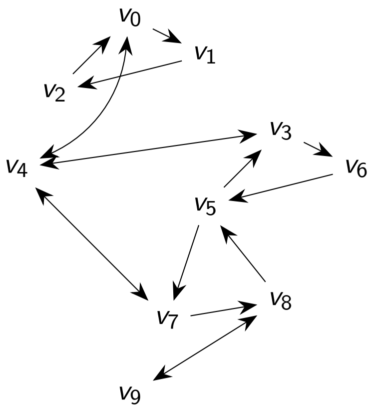
- Adjacency matrix has many 0's
  - Size is n^2, regardless of number of edges
  - Undirected graph: $|E| \leq n(n-1)/2$
  - Directed graph: $|E| \leq n(n-1)$
  - Typically $|E|$ much less than n^2
- Adjacency list

```
AList = {}
for i in range(10):
    AList[i] = []
for (i,j) in edges:
    AList[i].append(j)

print(AList)
```

```
{0: [1, 4], 1: [2], 2: [0], 3: [4, 6], 4: [0, 3, 7],
5: [3, 7], 6: [5], 7: [4, 8], 8: [5, 9], 9: [8]}
```

Airline routes



Comparing representations

- Adjacency list typically requires less space

	0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	1	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	0	1	0	0
6	0	0	0	0	0	1	0	0	0	0
7	0	0	0	0	1	0	0	0	1	0
8	0	0	0	0	0	1	0	0	0	1
9	0	0	0	0	0	0	0	0	1	0

0	[1,4]
1	[2]
2	[0]
3	[4,6]
4	[0,3,7]

5	[3,7]
6	[5]
7	[4,8]
8	[5,9]
9	[8]

Comparing representations

- Adjacency list typically requires less space
- Is j a neighbour of i?
  - Check if $A[i,j] = 1$ in adjacency matrix
  - Scan all neighbours of i in adjacency list

	0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	1	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	0	1	0	0
6	0	0	0	0	0	1	0	0	0	0
7	0	0	0	0	1	0	0	0	1	0
8	0	0	0	0	0	1	0	0	0	1
9	0	0	0	0	0	0	0	0	1	0

0	[1,4]
1	[2]
2	[0]
3	[4,6]
4	[0,3,7]

5	[3,7]
6	[5]
7	[4,8]
8	[5,9]
9	[8]

Comparing representations

- Adjacency list typically requires less space
- Is j a neighbour of i?
  - Check if $A[i,j] = 1$ in adjacency matrix
  - Scan all neighbours of i in adjacency list
- Which are the neighbours of i?
  - Scan all n entries in row i in adjacency matrix
  - Takes time proportional to (out)degree of i in adjacency list

	0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	1	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	0	1	0	0
6	0	0	0	0	0	1	0	0	0	0
7	0	0	0	0	1	0	0	0	1	0
8	0	0	0	0	0	1	0	0	0	1
9	0	0	0	0	0	0	0	0	1	0

0	[1,4]
1	[2]
2	[0]
3	[4,6]
4	[0,3,7]

5	[3,7]
6	[5]
7	[4,8]
8	[5,9]
9	[8]

Comparing representations

- Adjacency list typically requires less space
- Is j a neighbour of i?
  - Check if $A[i,j] = 1$ in adjacency matrix
  - Scan all neighbours of i in adjacency list
- Which are the neighbours of i?
  - Scan all n entries in row i in adjacency matrix
  - Takes time proportional to (out)degree of i in adjacency list
- Choose representation depending on requirement

	0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	1	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	1	0	0	0
4	1	0	0	1	0	0	0	1	0	0
5	0	0	0	1	0	0	0	1	0	0
6	0	0	0	0	0	1	0	0	0	0
7	0	0	0	0	1	0	0	0	1	0
8	0	0	0	0	0	1	0	0	0	1
9	0	0	0	0	0	0	0	0	1	0

0	[1,4]
1	[2]
2	[0]
3	[4,6]
4	[0,3,7]

5	[3,7]
6	[5]
7	[4,8]
8	[5,9]
9	[8]

Summary

- To operate on graphs, we need to represent them
- Adjacency matrix
  - $n \times n$ matrix, $AMat[i,j] = 1$ iff $(i,j) \in E$
- Adjacency list
  - Dictionary of lists
  - For each vertex i, $AList[i]$ is the list of neighbours of i
- Can systematically explore a graph using these representations
  - For reachability, propagate marking to all reachable vertices