CS2040S
Data Structures and Algorithms
(e-learning edition)

Graphs!

(Part 3)

Representing a Graph

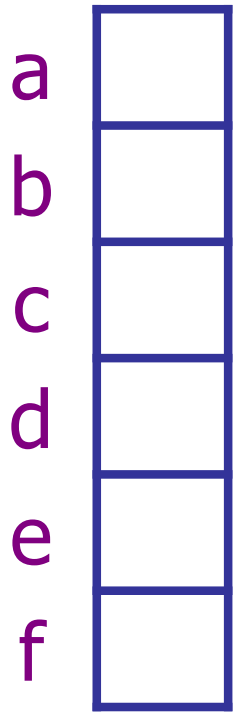
Graph consists of:

- Nodes
- Edges

Representing a Graph

Graph consists of:

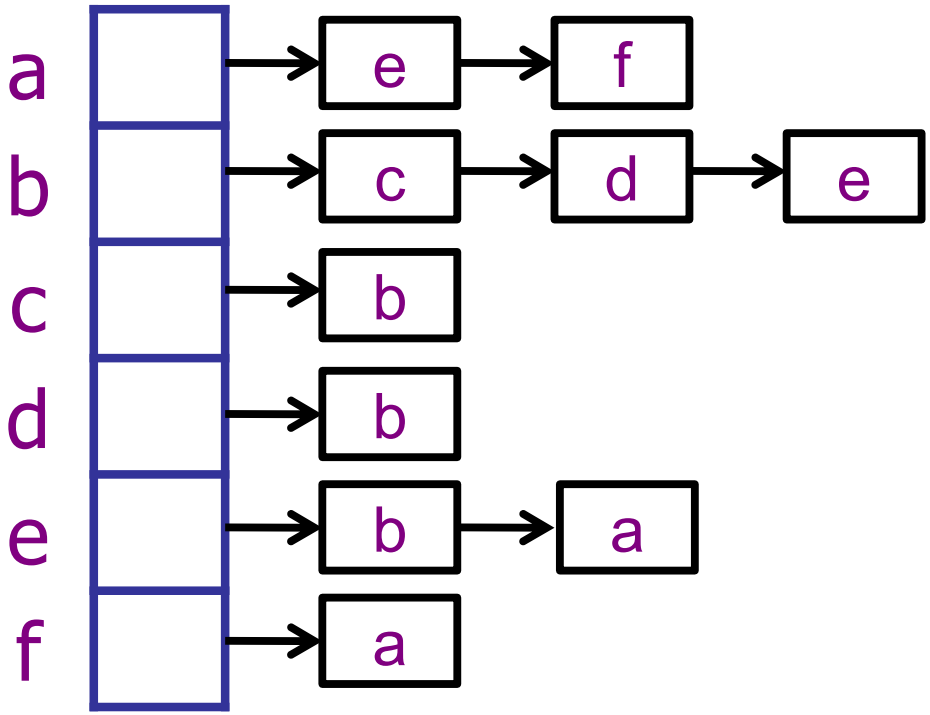
- Nodes: stored in an array
- Edges

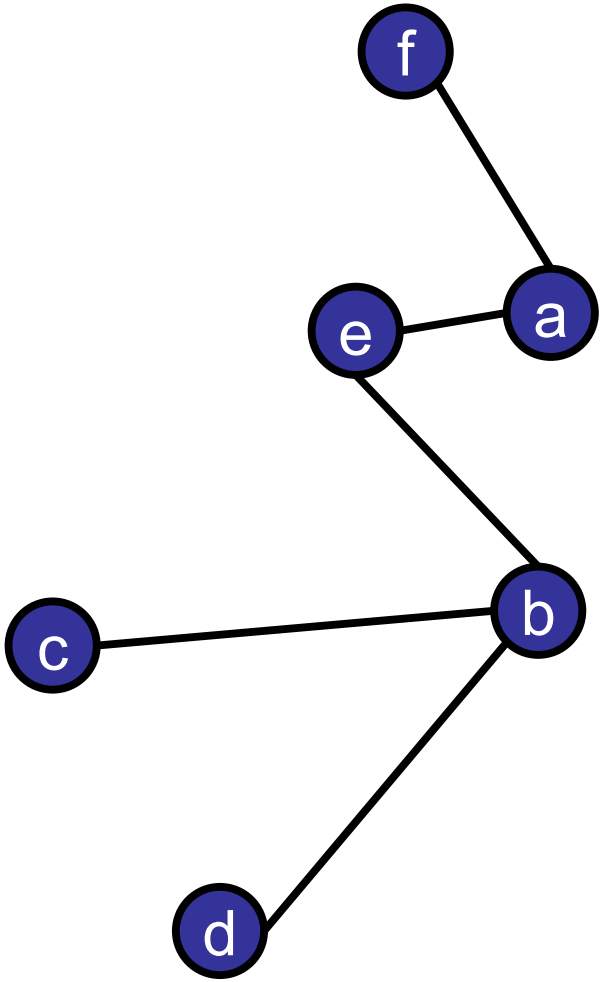


Adjacency List

Graph consists of:

- Nodes: stored in an array
- Edges: linked list per node





Adjacency List in Java

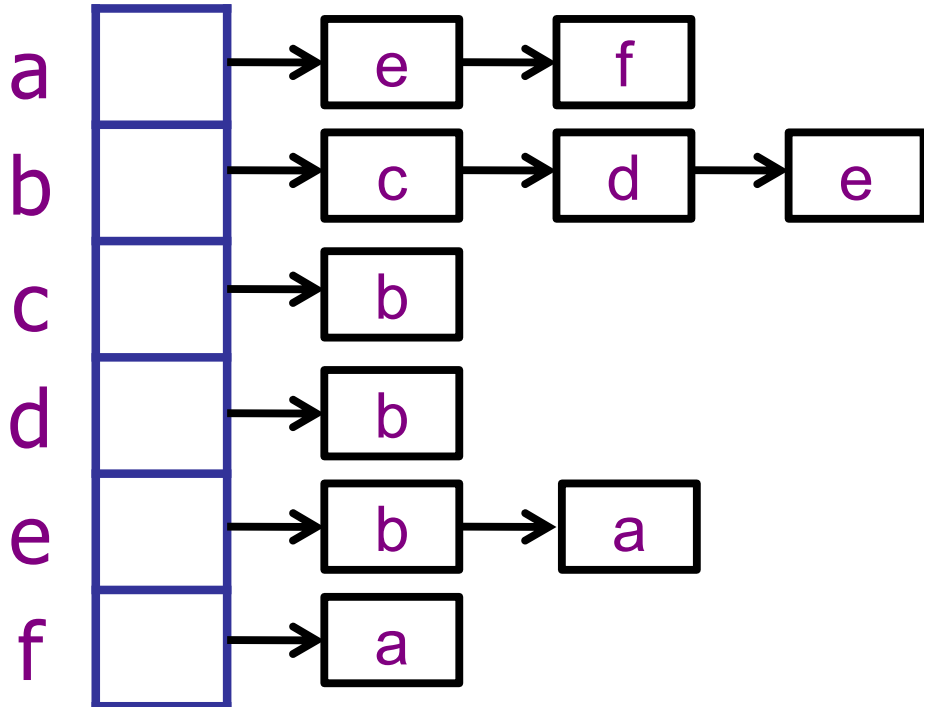
```
class NeighborList extends LinkedList<Integer> {
}


class Node {
  int key;
  NeighborList nbrs;
}

class Graph {
  Node[] nodeList;



}
```



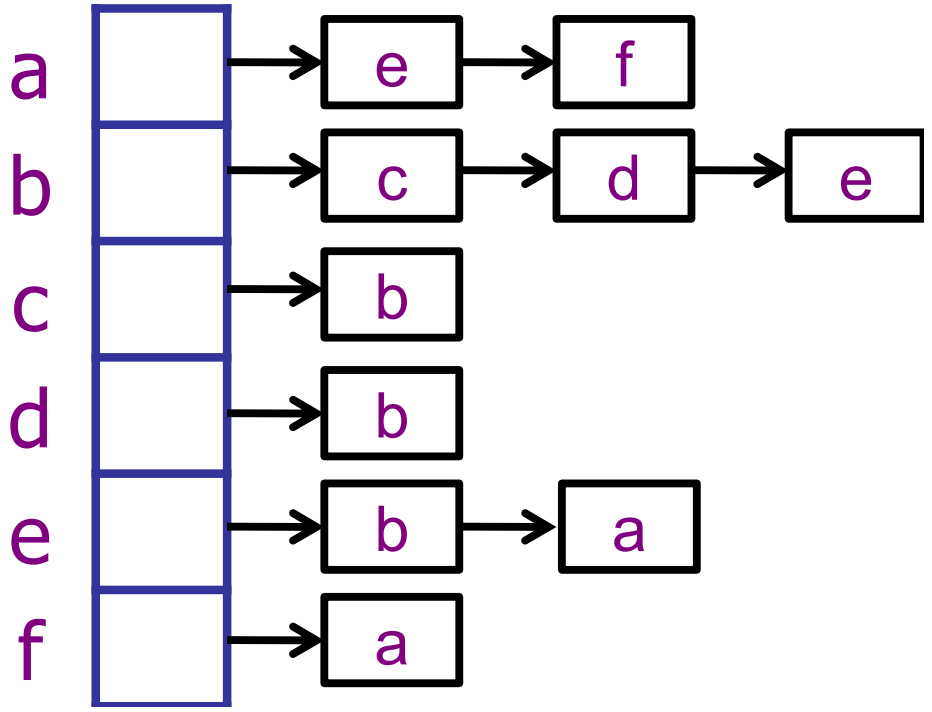
Adjacency List in Java

```
class Graph{
    List<List<Integer>> nodes;

}
```

More concise code is not *always* better…

- Harder to read
- Harder to debug
- Harder to extend



Representing a Graph

Graph consists of:

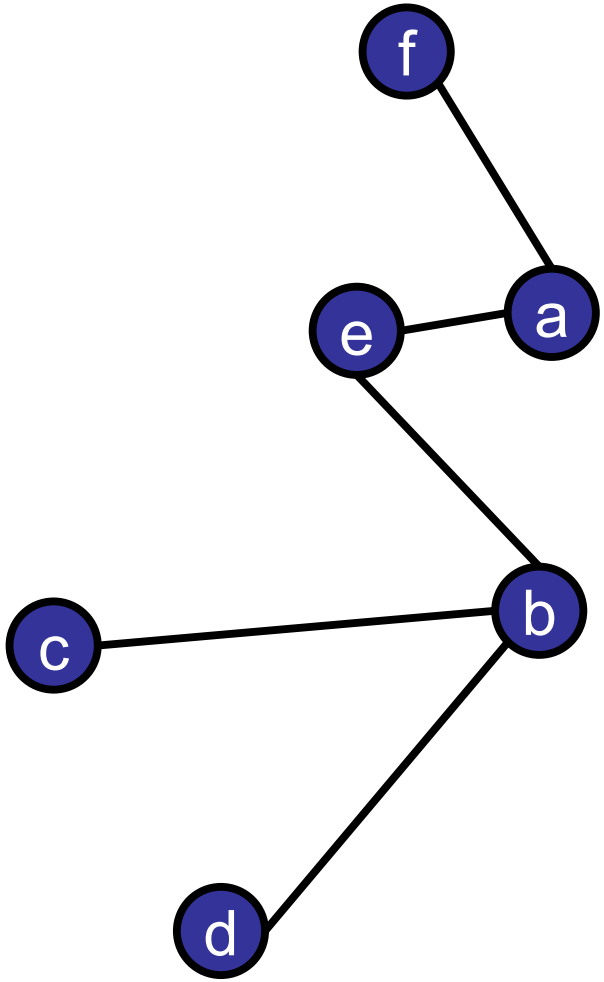
- Nodes
- Edges = pairs of nodes

Adjacency Matrix

Graph consists of:

- Nodes
- Edges = pairs of nodes

	a	b	c	d	e	f
a	0	0	0	0	**1**	**1**
b	0	0	**1**	**1**	**1**	0
c	0	**1**	0	0	0	0
d	0	**1**	0	0	0	0
e	**1**	**1**	0	0	0	0
f	**1**	0	0	0	0	0



Adjacency Matrix

Graph represented as:

$A[v][w] = 1$ iff $(v,w) \in E$

Neat property:

- A^2 = length 2 paths

	a	b	c	d	e	f
a	0	0	0	0	**1**	**1**
b	0	0	**1**	**1**	**1**	0
c	0	**1**	0	0	0	0
d	0	**1**	0	0	0	0
e	**1**	**1**	0	0	0	0
f	**1**	0	0	0	0	0

Adjacency Matrix

To find out if c and d are 2-hop neighbors:

- Let $B = A^2$.
- $B[c, d] = A[c, .] \bullet A[., d]$

- $B[c, d] = 1$ iff

  $A[c, x] == A[x, d]$

  for some x.

	a	b	c	d	e	f
a	0	0	0	0	**1**	**1**
b	0	0	**1**	**1**	**1**	0
c	0	**1**	0	0	0	0
d	0	**1**	0	0	0	0
e	**1**	**1**	0	0	0	0
f	**1**	0	0	0	0	0

Adjacency Matrix

Graph represented as:

$A[v][w] = 1$ iff $(v,w) \in E$

Neat properties:

- A^2 = length 2 paths
- A^4 = length 4 paths

	a	b	c	d	e	f
a	0	0	0	0	**1**	**1**
b	0	0	**1**	**1**	**1**	0
c	0	**1**	0	0	0	0
d	0	**1**	0	0	0	0
e	**1**	**1**	0	0	0	0
f	**1**	0	0	0	0	0

Adjacency Matrix

Graph represented as:

$A[v][w] = 1$ iff $(v,w) \in E$

Neat properties:

- A^2 = length 2 paths
- A^4 = length 4 paths
- A^∞ = Google pagerank

	a	b	c	d	e	f
a	0	0	0	0	**1**	**1**
b	0	0	**1**	**1**	**1**	0
c	0	**1**	0	0	0	0
d	0	**1**	0	0	0	0
e	**1**	**1**	0	0	0	0
f	**1**	0	0	0	0	0

Adjacency Matrix in Java

Graph represented as:

A[v][w] = 1 iff (v,w) $\in$ E

```
class Graph {

  boolean[][] m_adjMatrix;


}
```

	a	b	c	d	
a	0	0	0	0	
b	0	0	**1**	**1**	
c	0	**1**	0	0	
d	0	**1**	0	0	
e	**1**	**1**	0	0	
f	**1**	0	0	0	

Adjacency Matrix in Java

Graph represented as:

A[v][w] = 1 iff (v,w) $\in$ E

```
class Graph {

  Node[][] m_adjMatrix;


}
```

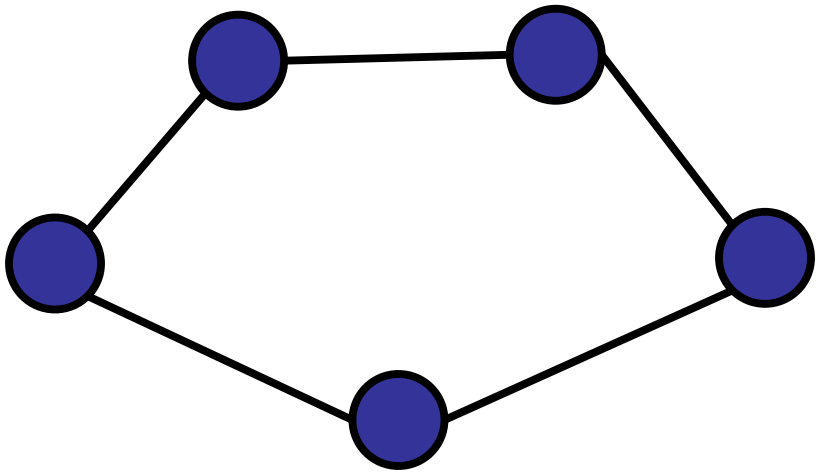
	a	b	c	d	
a	0	0	0	0	
b	0	0	**1**	**1**	
c	0	**1**	0	0	
d	0	**1**	0	0	
e	**1**	**1**	0	0	
f	**1**	0	0	0	

Trade-offs

Adjacency Matrix vs. Array?

For a cycle, which representation is better?

✓1. Adjacency list
2. Adjacency matrix
3. Equivalent



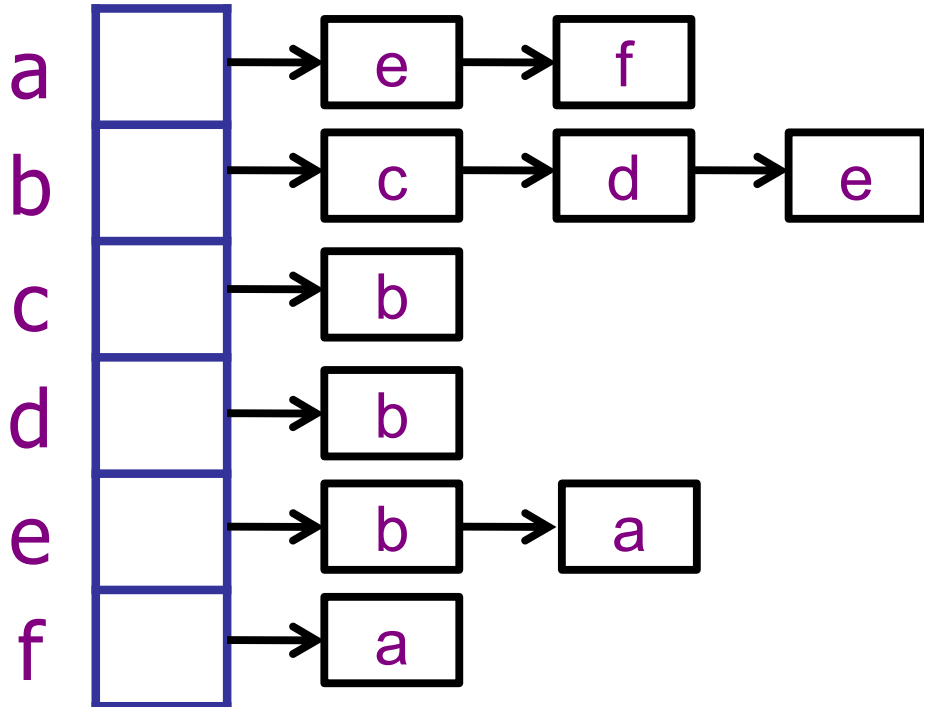
Adjacency List

Memory usage for graph G = (V, E):

- array of size |V|
- linked lists of size |E|

Total: O(V + E)

For a cycle: O(V)



Adjacency Matrix

Memory usage for graph G = (V, E):

- array of size |V|*|V|

Total: $O(V^2)$

For a cycle: $O(V^2)$

	a	b	c	d	e	f
a	0	0	0	0	**1**	**1**
b	0	0	**1**	**1**	**1**	0
c	0	**1**	0	0	0	0
d	0	**1**	0	0	0	0
e	**1**	**1**	0	0	0	0
f	**1**	0	0	0	0	0

For a clique, which representation is better?

1. Adjacency matrix
2. Adjacency list
3. Equivalent

Adjacency List vs. Matrix

Memory usage for graph $G = (V, E)$:

- Adjacency List: $O(V + E)$
- Adjacency Matrix: $O(V^2)$

For a cycle: $O(V)$ vs. $O(V^2)$

For a clique: $O(V + E) = O(V^2)$ vs. $O(V^2)$

Adjacency List vs. Matrix

Memory usage for graph $G = (V, E)$:

- Adjacency List: $O(V + E)$
- Adjacency Matrix: $O(V^2)$

For a cycle: $O(V)$ vs. $O(V^2)$

For a clique: $O(V + E) = O(V^2)$ vs. $O(V^2)$

Base rule: if graph is dense then use an adjacency matrix; else use an adjacency list.

dense: $|E| = \theta(V^2)$

Which representation for Facebook Graph?
Query: Are Bob and Joe friends?

1. Adjacency List
2. ✓ Adjacency Matrix
3. Equivalent

List: (much) better space.
Matrix: somewhat faster

Which representation for Facebook Graph?
Query: List all my friends?

✓1. Adjacency List
2. Adjacency Matrix
3. Equivalent

Trade-offs

Adjacency Matrix:

- Fast query: are v and w neighbors?
- Slow query: find me any neighbor of v.
- Slow query: enumerate all neighbors.

Adjacency List:

- Fast query: find me any neighbor.
- Fast query: enumerate all neighbors.
- Slower query: are v and w neighbors?

Graph Representations

Key questions to ask:

- Space usage: is graph dense or sparse?
- Queries: what type of queries do I need?
  - Enumerate neighbors?
  - Query relationship?

Roadmap

Today: Graph Basics

- What is a graph?
- Modeling problems as graphs.
- Graph representations (list vs. matrix)
- Searching graphs (DFS / BFS)