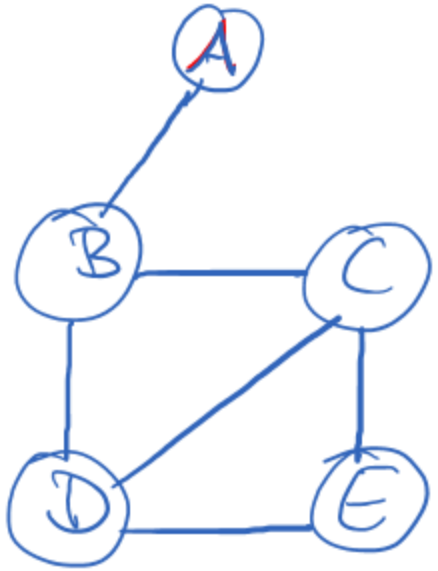
Graphs

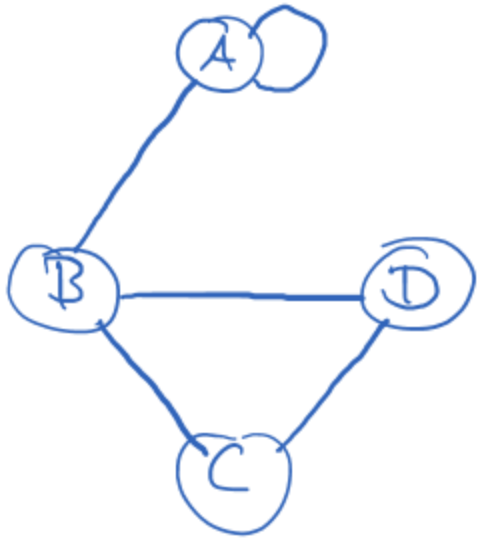
What is a Graph?

- Graphs consist of
  - Set of nodes (or vertices)
  - Set of links (or edges)
- Two nodes connected with direct edge are
  - Neighbors or adjacent nodes

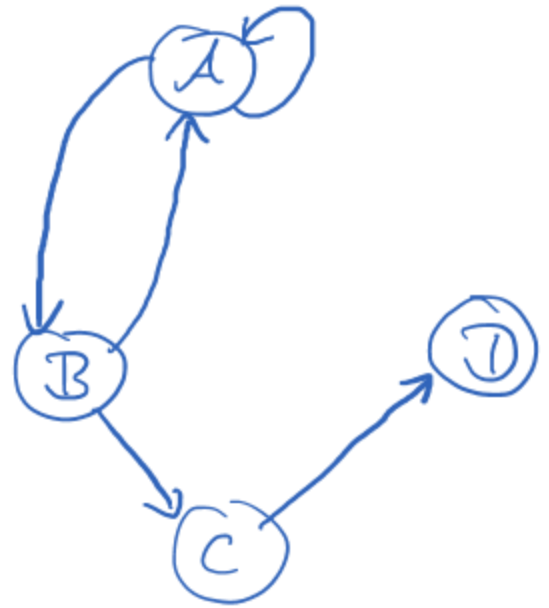


Two Types of Graphs

undirected



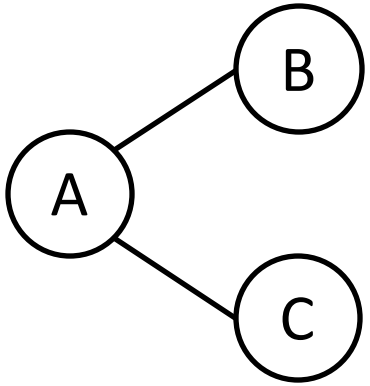
directed



Source → target

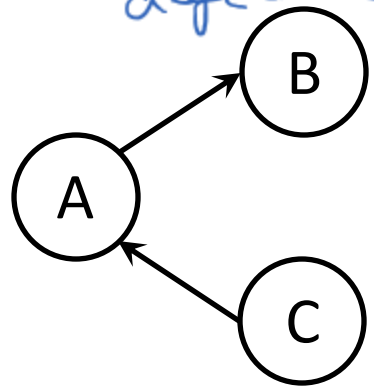
Degree of a Node

The number of edges connected to a node.



deg(A) = 2
deg(B) = 1
deg(C) = 1

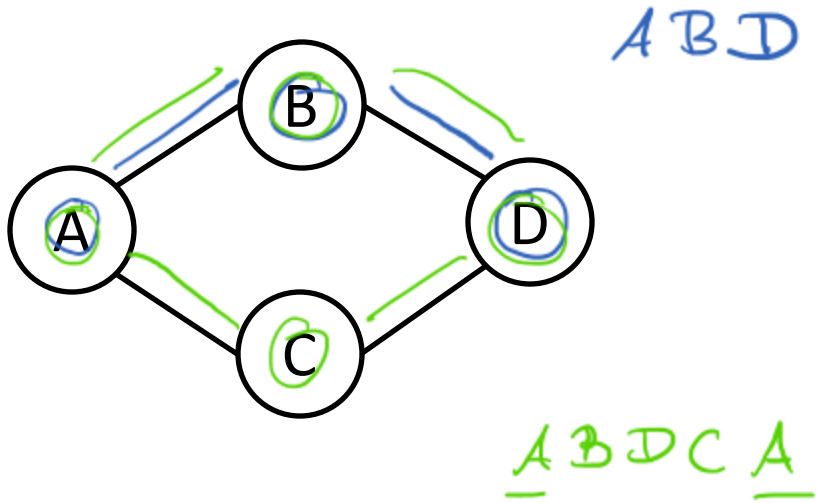
deg(A) = 2
deg(B) = 1
deg(C) = 1

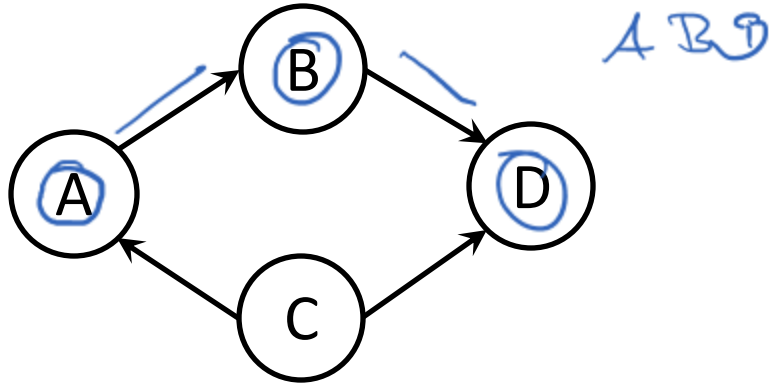


indeg(A) = 1
indeg(B) = 1
indeg(C) = 0

outdeg(A) = 1
outdeg(B) = 0
outdeg(C) = 1

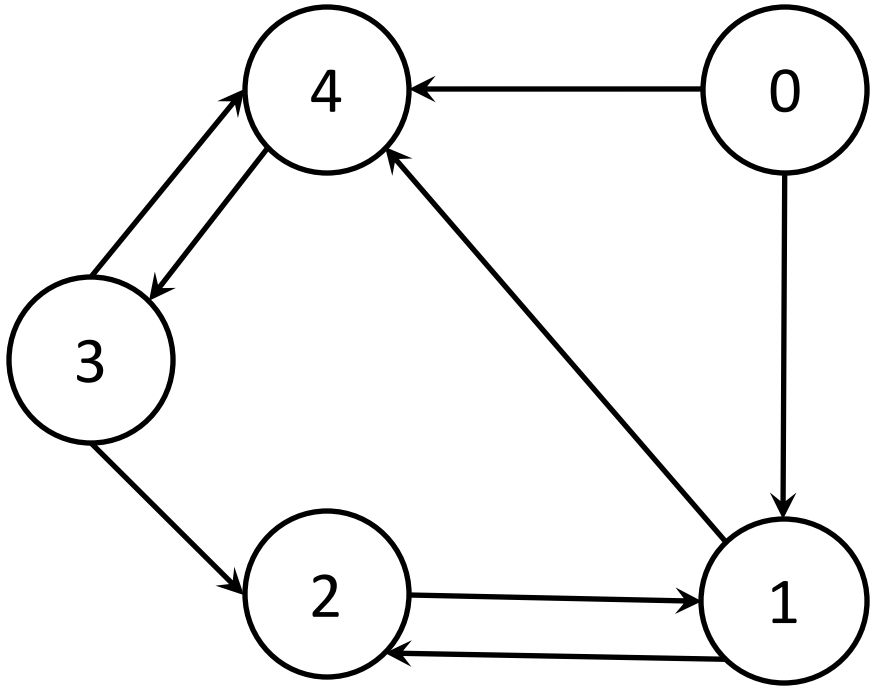
Paths in a Graph





- Cycle: A path that returns to a previously visited node
- Cyclic graph: contains at least one cycle
- Acyclic graph: contains no cycle

Adjacency Matrix Example 1

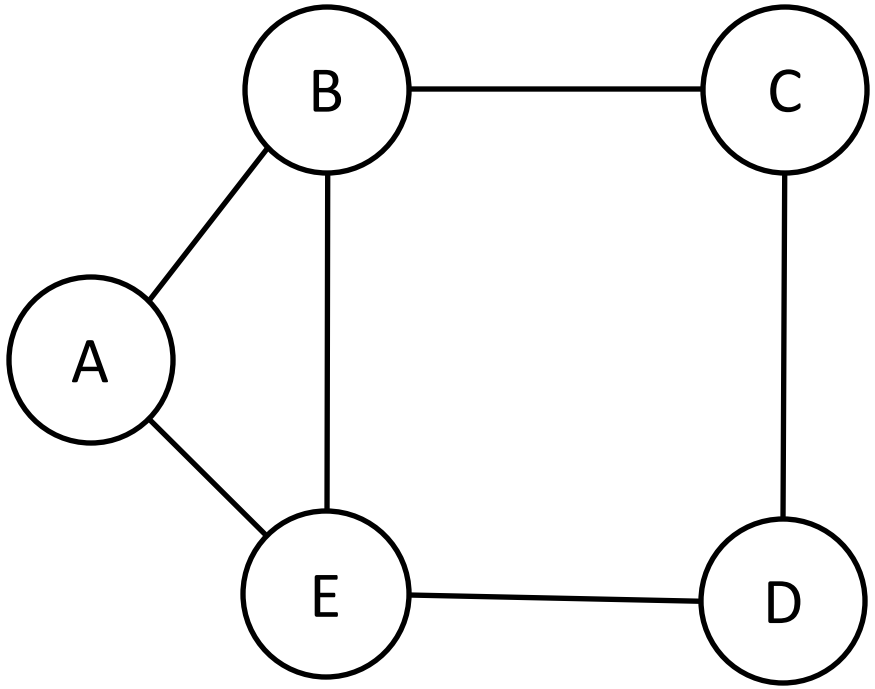


Columns: target

rows: source

	0	1	2	3	4
0	F	T	F	F	T
1	F	F	T	F	T
2	F	T	F	F	F
3	F	F	T	F	T
4	F	F	F	T	F

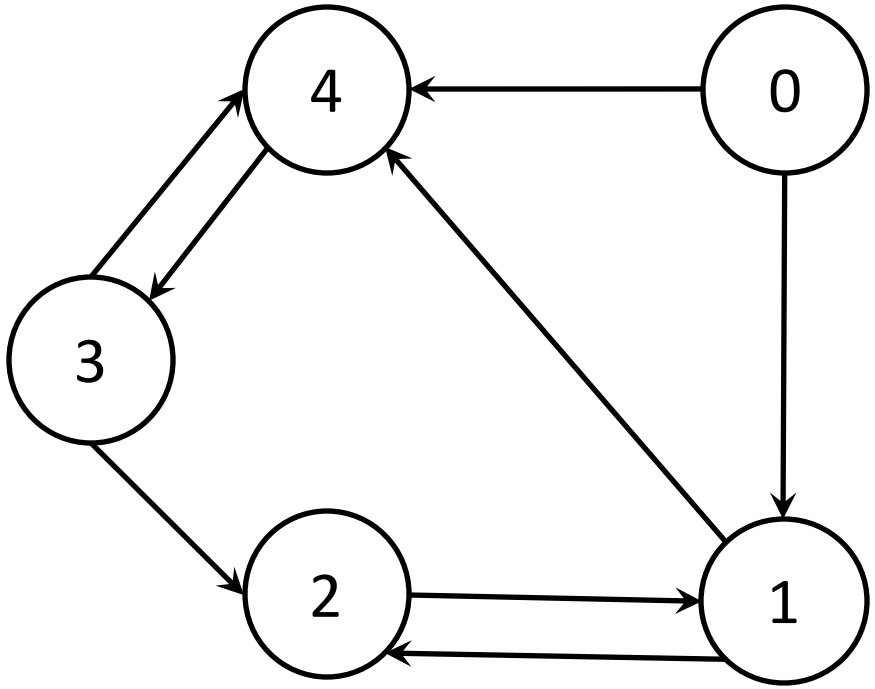
Adjacency Matrix Example 2

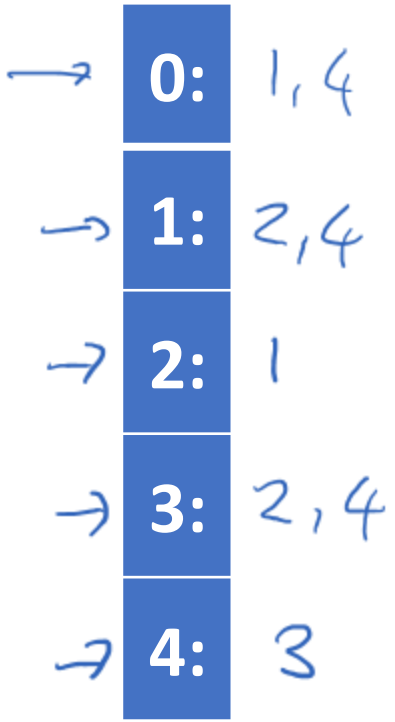


	A	B	C	D	E
A	F	T	F	F	T
B	T	F	T	F	T
C	F	T	F	T	F
D	F	F	T	F	T
E	T	T	F	T	F

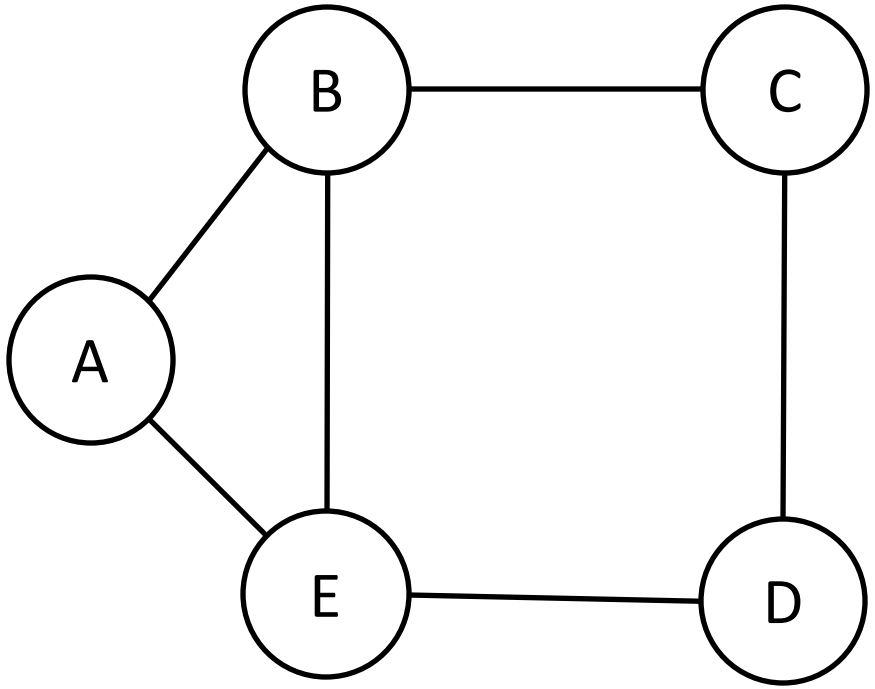
undirected: symmetric at the diagonal

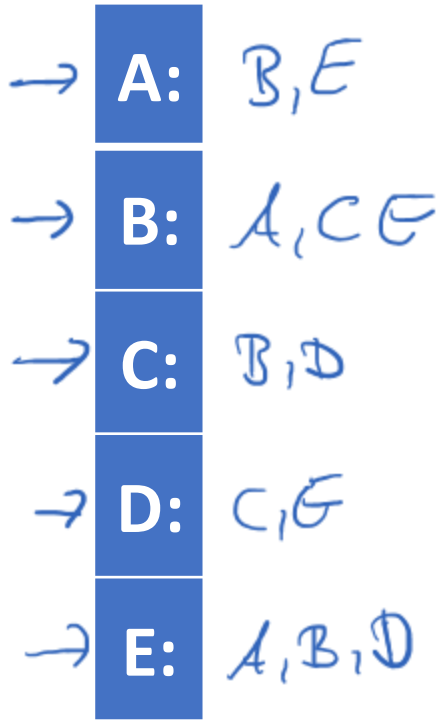
Adjacency List Example 1





Adjacency List Example 2





Implementation of Graphs

node type

```
public class Graph <T> {
        boolean[][] adjacencyMatrix;
        Map<T,Integer> nodeIndices;

        OR

        Map<T,Graphnode<T>> vertexTable;
        protected class Graphnode<T> {
                protected T data;
                protected List<Graphnode<T>> adjacencyList;
        }
}
```