THE GRAPH REPRESENTATION

ADJACENCY MATRIX

USE A MATRIX WITH ROWS AND COLUMNS

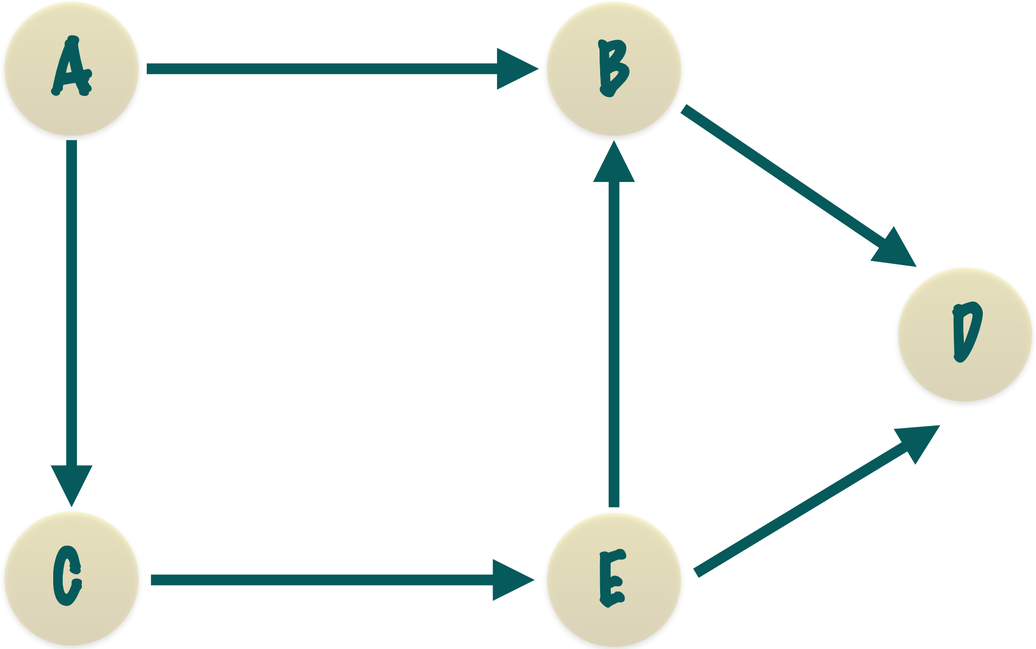
A MATRIX IS JUST A TABLE

THE ROW LABELS AND THE COLUMN LABELS REPRESENT THE VERTICES

EACH CELL REPRESENTS THE RELATIONSHIP BETWEEN THE VERTICES I.E. THE EDGES

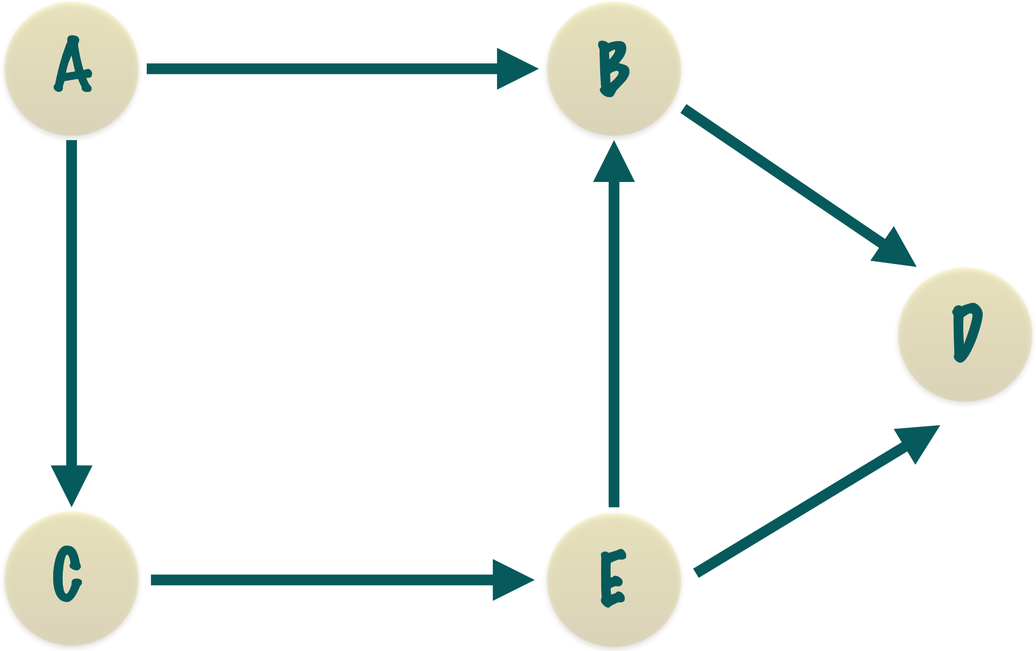
THE GRAPH REPRESENTATION

ADJACENCY MATRIX



THE GRAPH REPRESENTATION

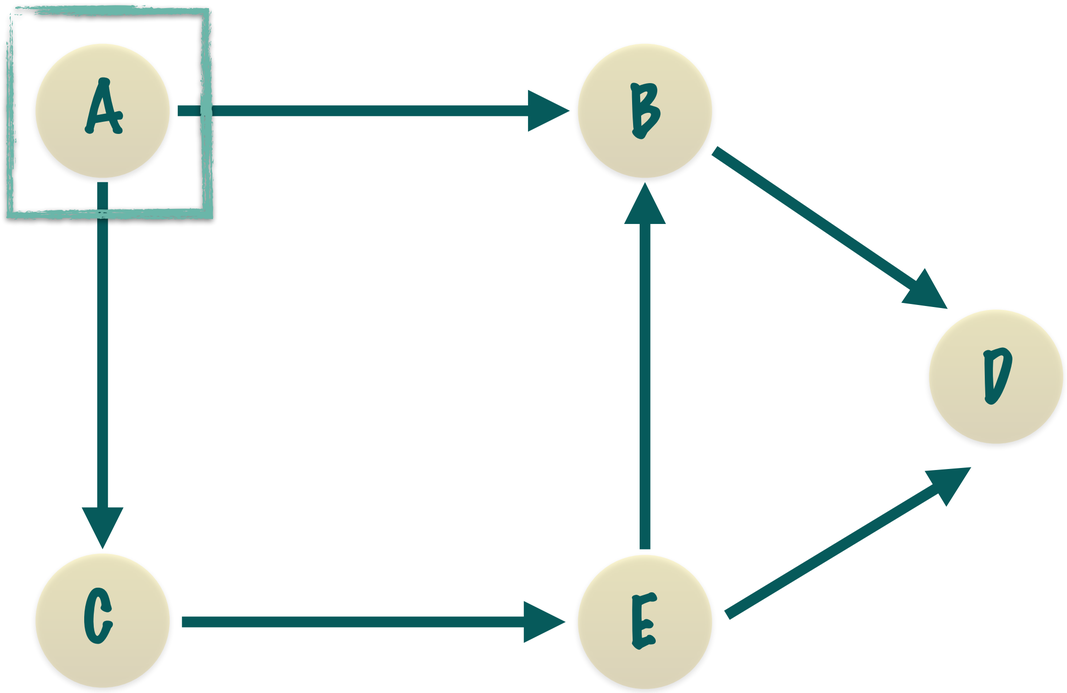
ADJACENCY MATRIX



	A	B	C	D	E

THE GRAPH REPRESENTATION

ADJACENCY MATRIX

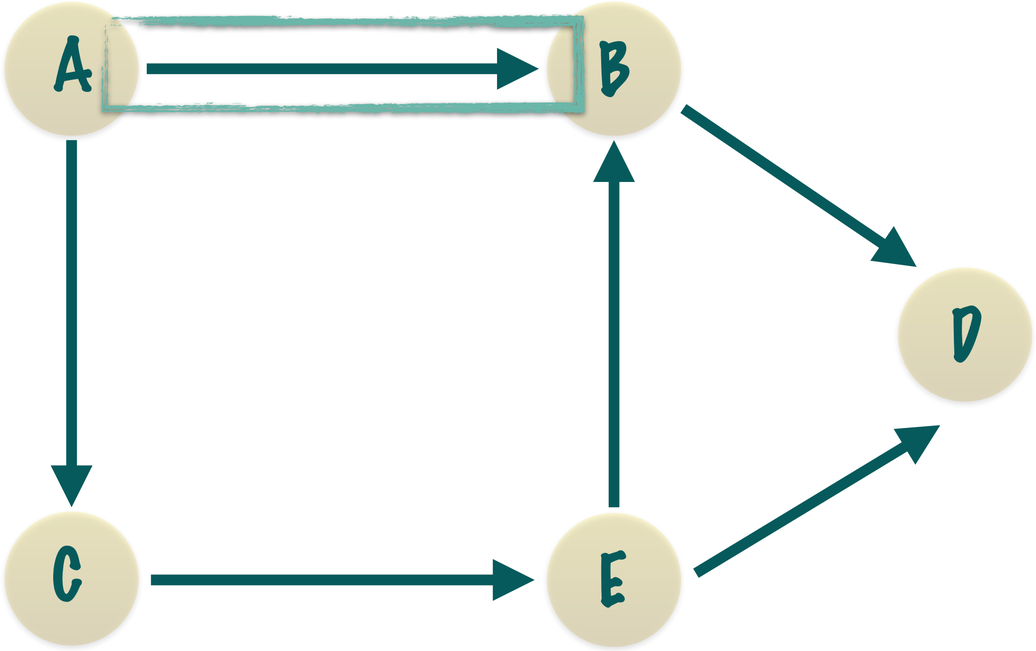


	A	B	C	D	E
A					
B					
C					
D					
E					

THE GRAPH REPRESENTATION

ADJACENCY MATRIX

A RELATION SHIP FROM A TO B IS REPRESENTED BY CELL VALUES

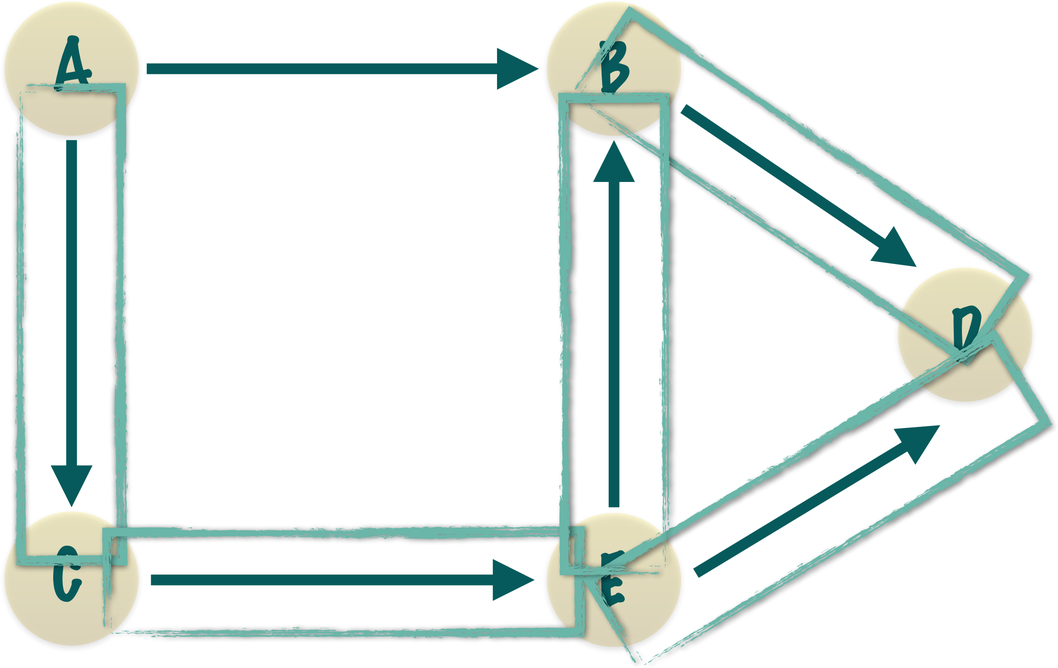


	A	B	C	D	E
A		1			
B					
C					
D					
E					

A VALUE OF 1 OR TRUE IN (ROW A, COLUMN B) INDICATES AN EDGE FROM A TO B

THE GRAPH REPRESENTATION

ADJACENCY MATRIX

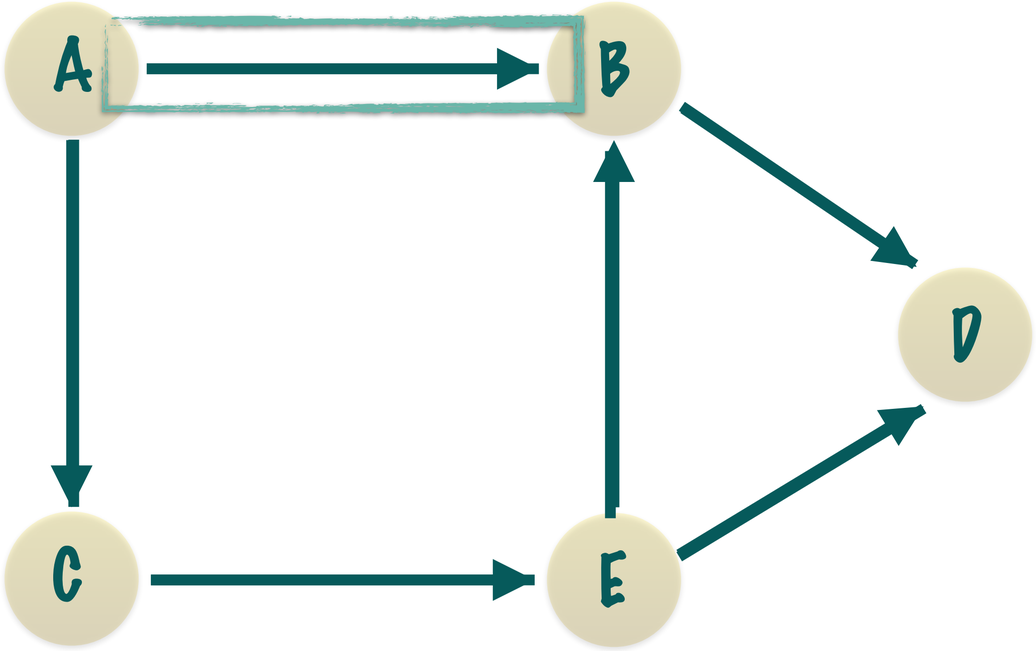


	A	B	C	D	E
A	0	1	1	0	0
B	0	0	0	1	0
C	0	0	0	0	1
D	0	0	0	0	0
E	0	1	0	1	0

THE GRAPH REPRESENTATION

ADJACENCY MATRIX

WHAT ABOUT AN UNDIRECTED GRAPH?

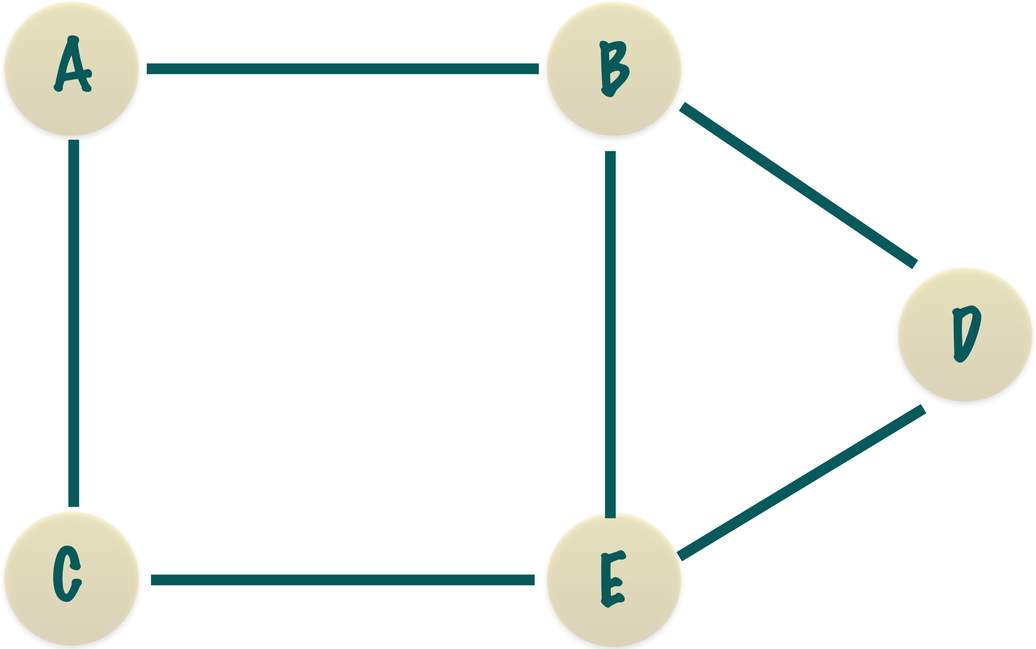


	A	B	C	D	E
A	0	1	1	0	0
B	0	0	0	1	0
C	0	0	0	0	1
D	0	0	0	0	0
E	0	1	0	1	0

THE GRAPH REPRESENTATION

ADJACENCY MATRIX

WHAT ABOUT AN UNDIRECTED GRAPH?



	A	B	C	D	E
A	0	1	1	0	0
B	1	0	0	1	1
C	1	0	0	0	1
D	0	1	0	0	1
E	0	1	1	1	0

THE GRAPH REPRESENTATION

ADJACENCY MATRIX

```
class Graph {
   int[][] adjacencyMatrix;
   int numVertices;
}
```

	A	B	C	D	E
A	0	1	1	0	0
B	1	0	0	1	1
C	1	0	0	0	1
D	0	1	0	0	1
E	0	1	1	1	0

THE GRAPH

REPRESENTATION

ADJACENCY MATRIX

NOW LET'S SEE SOME CODE...

THE ADJACENCY MATRIX

```
public class AdjacencyMatrixGraph implements Graph {

    private int[][] adjacencyMatrix;
    private GraphType graphType = GraphType.DIRECTED;
    private int numVertices = 0;

    public AdjacencyMatrixGraph(int numVertices, GraphType graphType) {
        this.numVertices = numVertices;
        this.graphType = graphType;

        adjacencyMatrix = new int[numVertices][numVertices];

        for (int i = 0; i < numVertices; i++) {
            for (int j = 0; j < numVertices; j++) {
                adjacencyMatrix[i][j] = 0;
            }
        }
    }
```

THIS IMPLEMENTS THE GRAPH INTERFACE - THE USE OF THE ADJACENCY MATRIX IS AN IMPLEMENTATION DETAIL

SET UP A V X V MATRIX TO HOLD THE VERTICES AND EDGES RELATIONSHIP

THIS CAN BE A DIRECTED OR UNDIRECTED GRAPH

INITIALIZE THE MATRIX, AND OTHER INFORMATION IN THE CONSTRUCTOR

ADJACENCY MATRIX - ADD EDGE

```java
@Override
public void addEdge(int v1, int v2) {
    if (v1 >= numVertices || v1 < 0 || v2 >= numVertices || v2 < 0) {
        throw new IllegalArgumentException("Vertex number is not valid");
    }

    adjacencyMatrix[v1][v2] = 1;
    if (graphType == GraphType.UNDIRECTED) {
        adjacencyMatrix[v2][v1] = 1;
    }
}
```

SPECIFY THE VERTICES THE EDGE CONNECTS - V1 IS THE SOURCE VERTEX AND V2 IS THE DESTINATION VERTEX

ENSURE THE VERTICES ARE VALID

SET THE CELL AT ROW V1 AND COLUMN V2 TO 1

IF THE GRAPH IS UNDIRECTED THEN THE CONNECTION GOES BOTH WAYS - SET ROW V2 AND COLUMN V1 TO 1 AS WELL

ADJACENCY MATRIX - GET ADJACENT VERTICES

```java
@Override
public List<Integer> getAdjacentVertices(int v) {
    if (v >= numVertices || v < 0) {
        throw new IllegalArgumentException("Vertex number is not valid");
    }

    List<Integer> adjacentVerticesList = new ArrayList<>();
    for (int i = 0; i < numVertices; i++) {
        if (adjacencyMatrix[v][i] == 1) {
            adjacentVerticesList.add(i);
        }
    }

    // Always return the vertices in ascending order.
    Collections.sort(adjacentVerticesList);

    return adjacentVerticesList;
}
```

ENSURE THE VERTEX IS VALID

SET UP THE LIST TO HOLD THE ADJACENT VERTICES

IF 1 IS PRESENT IN THE CELL IT MEANS THAT THE VERTEX V IS DIRECTLY CONNECTED TO ANOTHER VERTEX

SORT THE VERTICES IN ASCENDING ORDER SO RETURN VALUES ARE CONSISTENT