## Graphs

Represent data points and the relationships between them.

That's vague.

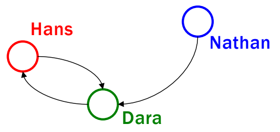
Formally:

A graph is a pair: G = (V,E)

V: set of **vertices** (aka **nodes**)

E: set of **edges**

Each edge is a pair of vertices.

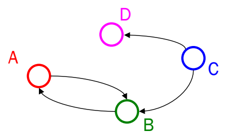



```
V = {Hans,Dara,Nathan}
E = {(Nathan,Dara),
     (Hans,Dara),
     (Dara,Hans)}
```





## Some math with edges




For a graph **G = (V,E)**:

- **|V|** = N, is the number of vertices
- **|E|** = M, is the number of edges
  - Minimum?
  - Maximum for undirected?
  - Maximum for directed?

```
V = {A, B, C, D}
E = {(C, B),
     (A, B),
     (B, A)
     (C, D)}
```

- If **(u,v) ∈ E**
  - Then **v** is a neighbor of **u**, i.e., **v** is adjacent to **u**
  - Order matters for directed edges
    - **u** is not adjacent to **v** unless **(v,u) ∈ E**

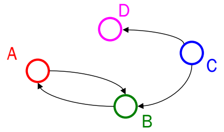




## Adjacency Matrix Properties

- Running time to:
  - Get a vertex's out-edges:
  - Get a vertex's in-edges:
  - Decide if some edge exists:
  - Insert an edge:
  - Delete an edge:
- Space requirements:
- Best for sparse or dense graphs?

| | A | B | C | D |
|---|---|---|---|---|
| A | F | T | F | F |
| B | T | F | F | F |
| C | F | T | F | T |
| D | F | F | F | F |

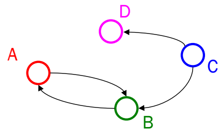


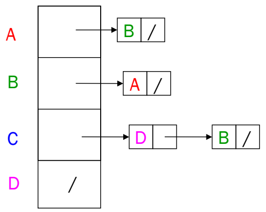


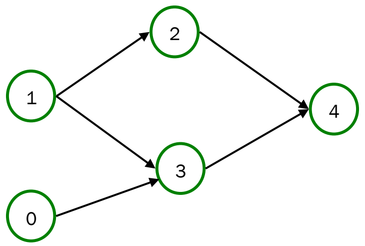



## Adjacency List

- Assign each node a number from **0** to **|V|-1**
- An array of length **|V|** in which each entry stores a list of all adjacent vertices (e.g., linked list)






Valid Topological Sorts:



38

## Topological Sort: Running time?

```
labelEachVertexWithItsInDegree();

for(ctr=0; ctr < numVertices; ctr++){
  v = findNewVertexOfDegreeZero();
  put v next in output
  for each w adjacent to v
    w.indegree--;
}
```



54

## Topological Sort(optimized): Running time?

```
labelAllAndEnqueueZeros();
for(ctr=0; ctr < numVertices; ctr++){
  v = dequeue();
  put v next in output
  for each w adjacent to v {
    w.indegree--;
    if(w.indegree==0)
      enqueue(w);
  }
}
```



57