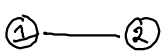
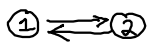
Find "directed" and cross it out.



undirected: unordered pairs



directed: ordered pairs

Graph is a set of vertices and a set of edges.

pair of vertices (and may be more)

Directed graph: $\langle \{1,2\}, \{\langle 1,2\rangle, \langle 2,1\rangle\} \rangle$

Undirected graph: $\langle \{1,2\}, \{\langle 1,2\rangle\} \rangle \approx \langle \{1,2\}, \{\langle 2,1\rangle\} \rangle$

An undirected graph can be encoded as a directed graph in which, for every $\langle v_1, v_2 \rangle$ in E, $\langle v_2, v_1 \rangle$ is also in E.

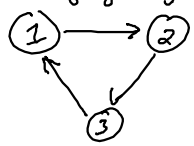
Graph ADT

Explicit vertices Implicit vertices

- X void insertVertex(V vertex)
- void insertEdge(Edge edge)
- X void removeVertex(V vertex)
- void removeEdge(V source, V destination)
- int getVertexCount()
- int getEdgeCount()
- vector<Edge> getEdges()
- vector<Edge> getOutgoingEdges(V source)
- Edge getEdge(V source, V destination)

```
class Edge {
  public:
    V source;
    V destination;
    int weight;
    L label;
}
```

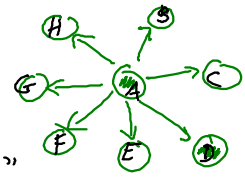
The number of outgoing edges from a vertex is its out-degree.



Graph ADT

- void insertEdge(Edge edge)
- void removeEdge(V source, V destination)
- int getVertexCount()
- int getEdgeCount()
- vector<Edge> getEdges()
- vector<Edge> getOutgoingEdges(V source)
- Edge getEdge(V source, V destination)

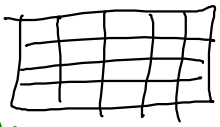
Implementations



1. Lists! "Linear Graph"
2. Dictionary<V, List<Edge>>
3. Dictionary<V, Dictionary<V, Edge>>

   Adjacency List Graph
4. Dictionary<V, int>, pair<Edge, bool>[][]

   {A↦1, B↦2, ...}



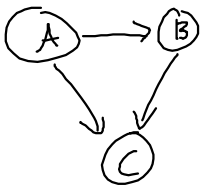
Adjacency Matrix Graph

Graph algorithms — depth-first search

```
Function reachableDFS(Graph g, V source, V destination) → bool:
    frontier ← new Stack
    frontier.insert(source)
    visited ← new Set  Dictionary
    While frontier is not empty
        V current ← frontier.remove()
        If current is not in visited:
            visited.insert(current)  any value
            If current == destination:
                Return true
            EndIf
            For each neighbor of current in g:
                If neighbor is not in visited:
                    frontier.insert(neighbor)
                EndIf
            EndFor
        EndIf
    EndWhile
```

WC $O(V^2)$

$E < V^2$



$O(V)$

$O(V)$

$O(E)$