

Directed/Weighted Graphs

- Generalising Graphs
- Directed Graphs (Digraphs)
- Digraph Representation
- Weighted Graphs
- Weighted Graph Representation
- Weighted Graph Implementation

❖ Generalising Graphs

Discussion so far has considered graphs as

- V = set of vertices, E = set of edges

Real-world applications require more "precision"

- some edges are directional (e.g. one-way streets)
- some edges have a cost (e.g. distance, traffic)

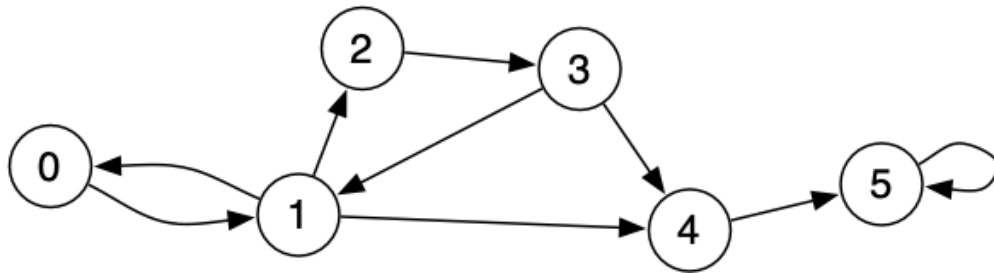
We need to consider **directed** graphs and **weighted** graphs

❖ Directed Graphs (Digraphs)

Directed graphs are ...

- graphs with V vertices, E edges (v,w)
- edge (v,w) has **source** v and **destination** w
- unlike undirected graphs, $v \rightarrow w \neq w \rightarrow v$

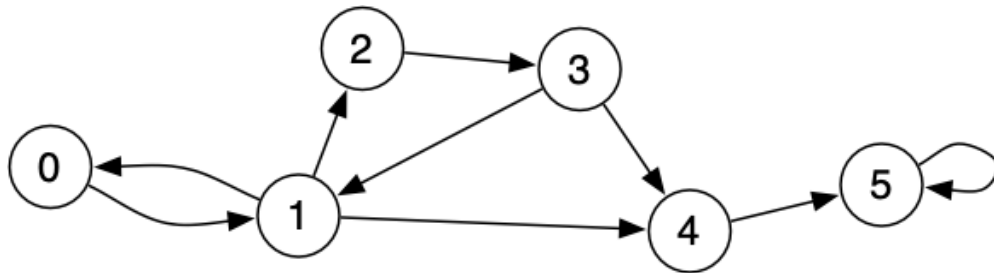
Example digraph:



COMP2521 20T2 ♦ Directed/Weighted Graphs [2/19]

❖ ... Directed Graphs (Digraphs)

Some properties of ...



- edges 1-2-3 form a cycle, edges 1-3-4 do *not* form a cycle
- vertex 5 has a self-referencing edge (5,5)
- vertices 0 and 1 reference each other, i.e. (0,1) and (1,0)
- there are no paths from 5 to any other nodes
- paths from 0→5: 0-1-2-3-4-5, 0-1-4-5, 0-1-2-3-1-4-5

❖ ... Directed Graphs (Digraphs)

Terminology for digraphs ...

Directed path: sequence of $n \geq 2$ vertices $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_n$

- where $(v_i, v_{i+1}) \in \text{edges}(G)$ for all v_i, v_{i+1} in sequence

If $v_1 = v_n$, we have a **directed cycle**

Degree of vertex: number of incident edges

- **outdegree:** $\text{deg}(v)$ = number of edges of the form $(v, _)$
- **indegree:** $\text{deg}^{-1}(v)$ = number of edges of the form $(_, v)$

❖ ... Directed Graphs (Digraphs)

More terminology for digraphs ...

Reachability:

- w is reachable from v if \exists directed path v, \dots, w

Strong connectivity:

- every vertex is reachable from every other vertex

Directed acyclic graph (DAG):

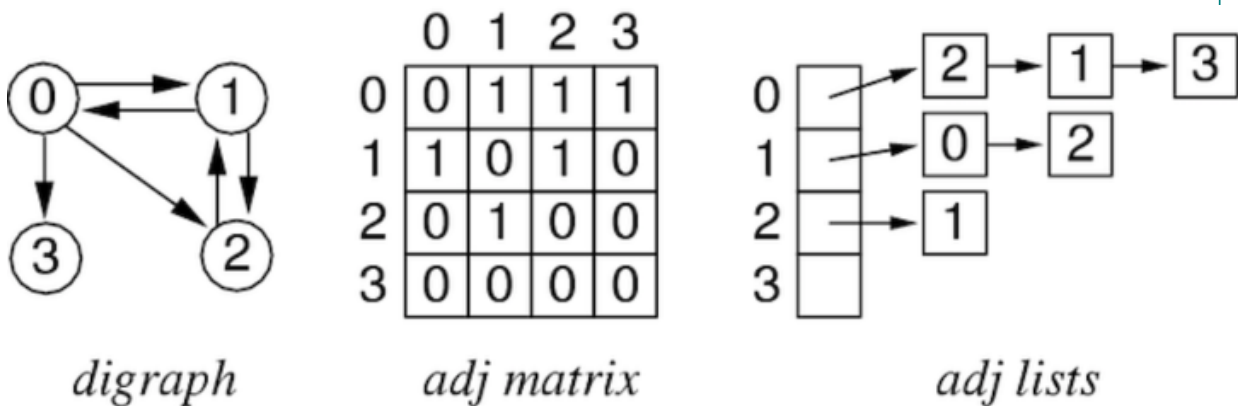
- contains no directed cycles

❖ Digraph Representation

Similar set of choices as for undirectional graphs:

- array of edges (directed)
- vertex-indexed adjacency matrix (non-symmetric)
- vertex-indexed adjacency lists

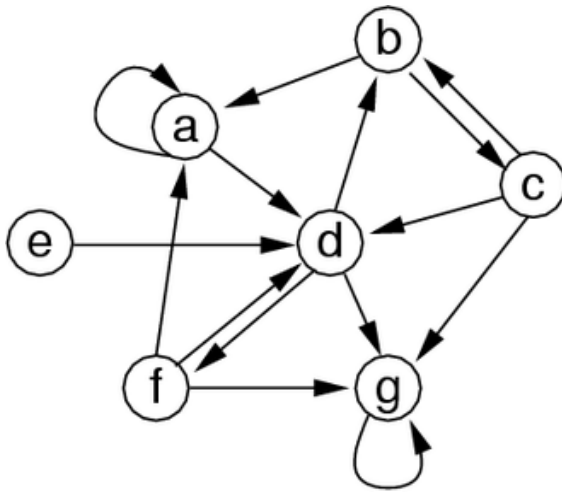
V vertices identified by $0..V-1$



COMP2521 20T2 ♦ Directed/Weighted Graphs [6/19]

❖ ... Digraph Representation

Example digraph and adjacency matrix representation:



| | a | b | c | d | e | f | g |
|---|---|---|---|---|---|---|---|
| a | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| b | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| c | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| d | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| e | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| f | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| g | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Undirectional \Rightarrow symmetric matrix

Directional \Rightarrow non-symmetric matrix

Maximum #edges in a digraph with V vertices: V^2

❖ ... Digraph Representation

Costs of representations: (where degree $\deg(v)$ = #edges leaving v)

| | array of edges | adjacency matrix | adjacency list |
|-----------------------|-------------------|---------------------|-------------------|
| space usage | E | V^2 | $V+E$ |
| insert edge | E | 1 | 1 |
| exists edge (v,w) ? | E | 1 | $\deg(v)$ |
| get edges leaving v | E | V | $\deg(v)$ |

Overall, adjacency list representation is best

- real graphs tend to be sparse
(large number of vertices, small average degree $\deg(v)$)
- algorithms frequently iterate over edges from v

❖ Weighted Graphs

Graphs so far have considered

- edge = an association between two vertices/nodes
- may be a precedence in the association (directed)

Some applications require us to consider

- a **cost** or **weight** of an association
- modelled by assigning values to edges (e.g. positive reals)

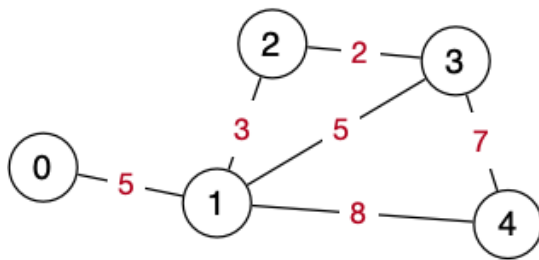
❖ ... Weighted Graphs

Weighted graphs are ...

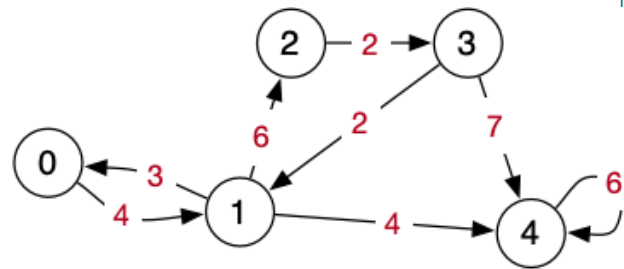
- graphs with V vertices, E edges (s,t)
- each edge (s,t,w) connects vertices s and t and has weight w

Weights can be used in both directed and undirected graphs.

Example weighted graphs:



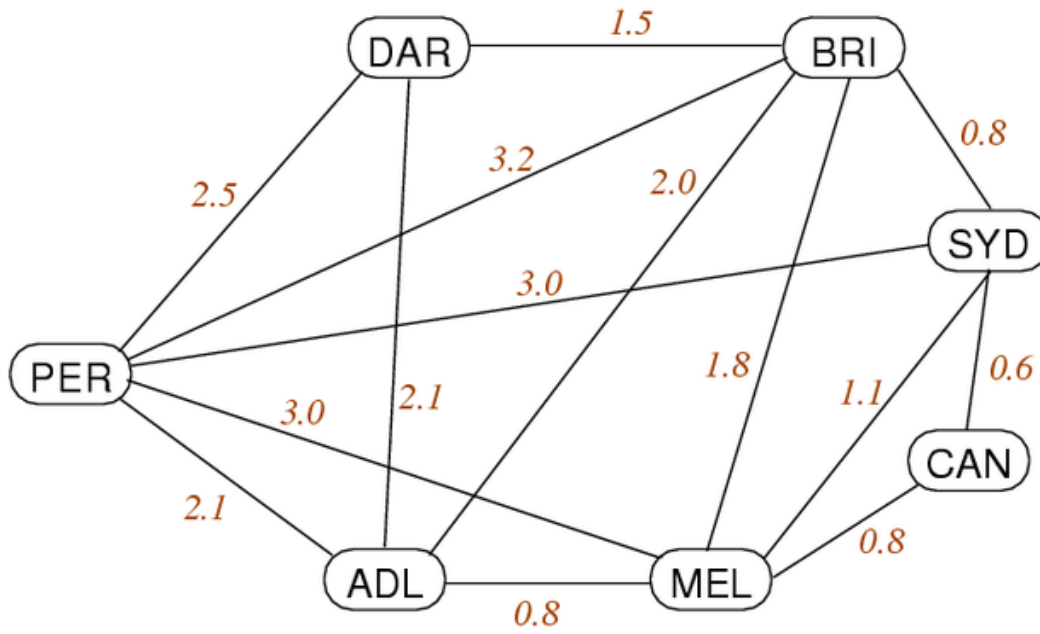
Weighted Graph



Directed Weighted Graph

❖ ... Weighted Graphs

Example: major airline flight routes in Australia



Representation: edge = direct flight; weight = approx flying time (hours)

❖ ... Weighted Graphs

Weights lead to minimisation-type questions, e.g.

1. Cheapest way to connect all vertices?

- a.k.a. **minimum spanning tree** problem
- assumes: edges are weighted and undirected

2. Cheapest way to get from A to B ?

- a.k.a **shortest path** problem
- assumes: edge weights positive, directed or undirected

❖ Weighted Graph Representation

Weights can easily be added to:

- adjacency matrix representation ($0/1 \rightarrow \text{int or float}$)
- adjacency lists representation (add int/float to list node)

The edge list representation changes to list of (s, t, w) triples

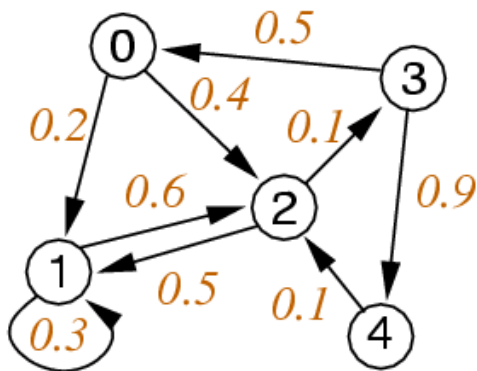
All representations can also work with directed edges

Weight values are determined by domain being modelled

- in some contexts weight could be zero or negative

❖ ... Weighted Graph Representation

Adjacency matrix representation with weights:



Weighted Digraph

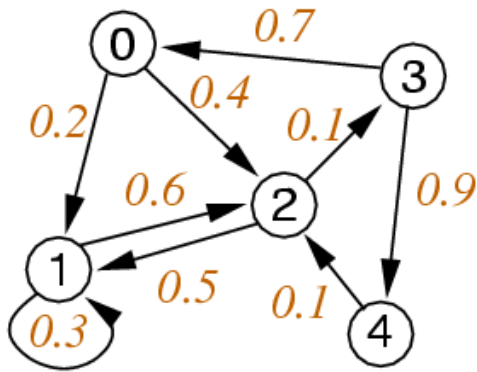
| | 0 | 1 | 2 | 3 | 4 |
|---|-----|-----|-----|-----|-----|
| 0 | * | 0.2 | 0.4 | * | * |
| 1 | * | 0.3 | 0.6 | * | * |
| 2 | * | 0.5 | * | 0.1 | * |
| 3 | 0.5 | * | * | * | 0.9 |
| 4 | * | * | 0.1 | * | * |

Adjacency Matrix

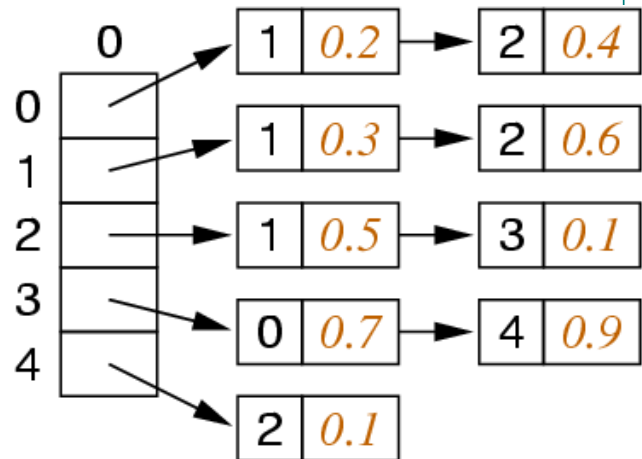
Note: need distinguished value to indicate "no edge".

❖ ... Weighted Graph Representation

Adjacency lists representation with weights:



Weighted Digraph

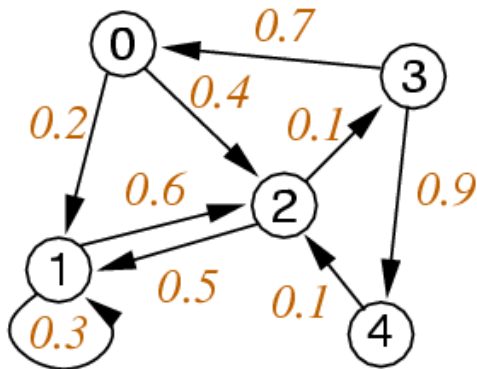


Adjacency Lists

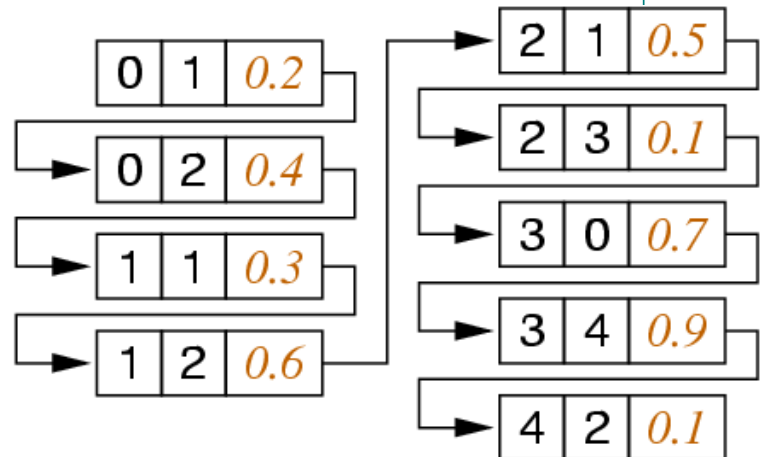
Note: if undirected, each edge appears twice with same weight

❖ ... Weighted Graph Representation

Edge array / edge list representation with weights:



Weighted Digraph



Edge List

Note: not very efficient for use in processing algorithms, but does give a possible representation for min spanning trees or shortest paths

❖ Weighted Graph Implementation

Changes to previous graph data structures to include weights:

WGraph.h

```
// edges are pairs of vertices (end-points) plus weight
typedef struct Edge {
    Vertex v;
    Vertex w;
    int weight;
} Edge;

// returns weight, or 0 if vertices not adjacent
int adjacent(Graph, Vertex, Vertex);
```

Note: here, we assume all weights are positive, but not required

❖ ... Weighted Graph Implementation

WGraph.c (assuming adjacency matrix representation)

```
typedef struct GraphRep {  
    int **edges; // adjacency matrix storing weights  
                // 0 if nodes not adjacent  
    int  nV;     // #vertices  
    int  nE;     // #edges  
} GraphRep;  
  
bool adjacent(Graph g, Vertex v, Vertex w) {  
    assert(valid graph, valid vertices)  
    return (g->edges[v][w] != 0);  
}
```

❖ ... Weighted Graph Implementation

More **WGraph.c**

```
void insertEdge(Graph g, Edge e) {
    assert(valid graph, valid edge)
    // edge e not already in graph
    if (g->edges[e.v][e.w] == 0) g->nE++;
    // may change weight of existing edge
    g->edges[e.v][e.w] = e.weight;
    g->edges[e.w][e.v] = e.weight;
}

void removeEdge(Graph g, Edge e) {
    assert(valid graph, valid edge)
    // edge e not in graph
    if (g->edges[e.v][e.w] == 0) return;
    g->edges[e.v][e.w] = 0;
    g->edges[e.w][e.v] = 0;
    g->nE--;
}
```

COMP2521 20T2 ♦ Directed/Weighted Graphs [19/19]

Produced: 28 Jun 2020