Graph ADT
Graph Reps

COMP2521 24T3

Graphs (I) Introduction to Graphs

Hao Xue

cs2521@cse.unsw.edu.au

graph fundamentals graph ADT graph representations

Types of Graphs Graph Terminology

Graph ADT

Graph Reps

Graph Fundamentals

Types of Graphs Graph Terminolo

Graph ADT

Graph Reps

Up to this point, we've seen a few collection types...

lists: a linear sequence of items each node is connected to its next node

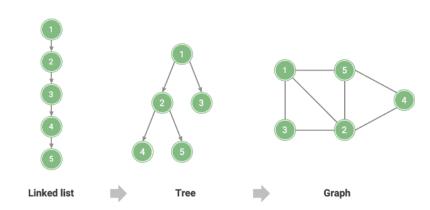
trees: a branched hierarchy of items each node is connected to its child node(s)

what if we want something more general? each node is connected to arbitrarily many nodes

Types of Graphs Graph Terminology

Graph ADT

Graph Reps



Graph has the highest degree of freedom, but it is also the most complex

Graphs Types of Graphs Graph Terminole

Graph Termino
Graph ADT

Graph Reps

Many applications need to model relationships between items.

... on a map: cities, connected by roads

... on the Web: pages, connected by hyperlinks

... in a game: states, connected by legal moves

... in a social network: people, connected by friendships

... in scheduling: tasks, connected by constraints

... in circuits: components, connected by traces

... in networking: computers, connected by cables

... in programs: functions, connected by calls

... etc. etc. etc.

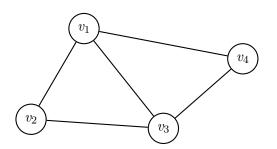
Graphs Types of Graphs

Types of Graphs
Graph Terminology
Graph ADT

Graph Reps

A graph is a data structure consisting of:

- A set of vertices V
 - Also called nodes
- A set of edges *E* between pairs of vertices



$$V = \{v_1, v_2, v_3, v_4\}$$

$$E = \{(v_1, v_2), (v_1, v_3), (v_1, v_4), (v_2, v_3), (v_3, v_4)\}$$

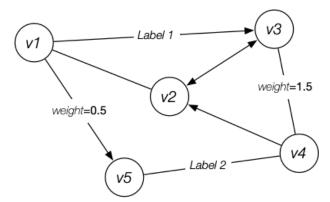
Graphs Types of Graphs Graph Terminolo

Graph Terminolo
Graph ADT
Graph Reps

Vertices are distinguished by a unique identifier.

ullet In this course, usually an integer between 0 and |V|-1

Edges may be (optionally) directed, weighted and/or labelled.

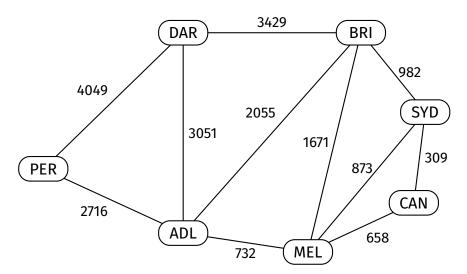


Types of Graphs Graph Terminology

Graph ADT

Graph Reps

Example: Australian cities and roads



Graphs Types of Graphs Graph Terminology

Graph ADT Graph Reps

Questions we could answer with a graph:

- Is there a way to get from *A* to *B*?
- What is the *best* way to get from *A* to *B*?
- In general, what vertices can we reach from *A*?
- Is there a path that lets me visit all vertices?
- Can we form a tree linking all vertices?
- Are two graphs "equivalent"?

Graph problems are generally more complex to solve than linked list problems:

- Items are not ordered
- Graphs may contain cycles
- Concrete representation is more complex

Graphs Types of Graphs

Graphs
Types of Graphs
Graph Terminology
Graph ADT

Graph Reps

Graphs can be a combination of these types:

```
undirected or directed
unweighted or weighted
disconnected or connected
without loops or with loops
non-multigraph or multigraph
```

... and others ...

Undirected Graphs

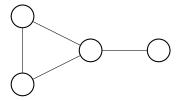
Graphs

Types of Graphs Graph Terminology

Graph ADT

Graph Reps

In an undirected graph, edges do not have direction.

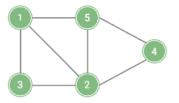


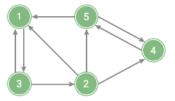
Types of Graphs Graph Terminology

Graph ADT

Graph Reps

In a directed graph or digraph, each edge has a direction.





Undirected graph

(Edges without direction)

Directed graph

(Edges with direction)

Weighted Graphs

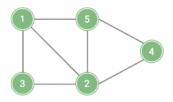
Graphs

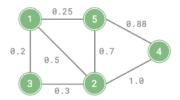
Types of Graphs Graph Terminology

Graph ADT

Graph Reps

In a weighted graph, each edge has an associated weight. For example, road maps, networks.





Unweighted graph

(All edges are equivalent)

Weighted graph

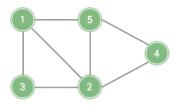
(Edges have weight attributes)

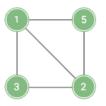
Connected Graphs

Types of Graphs
Graph Terminology
Graph ADT

Graph Reps

In a connected graph, it is possible to reach any other vertex starting from an arbitrary vertex.







Connected graph

(All vertices are reachable)

Disconnected graph

(Some vertices are unreachable)

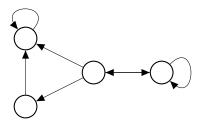
Types of Graphs **Graph Terminology**

Graph ADT

Graph Reps

A loop is an edge from a vertex to itself.

Depending on the context, a graph may or may not be allowed to have loops.

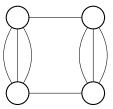


Types of Graphs Graph Terminology

Graph ADT

Graph Reps

In a multigraph, multiple edges are allowed between two vertices. For example, call graphs, maps.



Multigraphs will not be considered in this course.

Simple Graphs

Graphs
Types of Graphs
Graph Terminology

Graph ADT
Graph Reps

A simple graph is an undirected graph with no loops and no multiple edges.

For now, we will only consider simple graphs.

Simple Graphs

Graphs

Types of Graphs Graph Terminology

Graph ADT

Graph Reps

Question:

For a simple graph with \it{V} vertices, what is the $\it{maximum}$ possible number of edges?

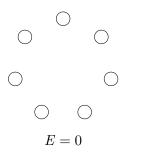
Types of Graphs

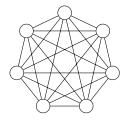
Graph ADT

Graph Reps

Question:

For a simple graph with \it{V} vertices, what is the $\it{maximum}$ possible number of edges?





$$E = V(V - 1)/2$$

Complete Graph Note on notation:

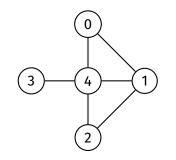
The number of vertices |V| and the number of edges |E| are normally written as V and E for simplicity.

Graphs
Types of Graphs
Graph Terminology
Graph ADT

Graph Reps

Two vertices v and w are adjacent if an edge e := (v, w) connects them; we say e is incident on v and w.

The degree of a vertex $v (\deg(v))$ is the number of edges incident on v.



deg(0) = 2 deg(1) = 3 deg(2) = 2deg(3) = 1

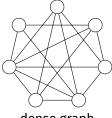
 $\deg(4) = 4$

Graph Terminology

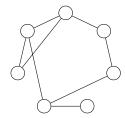
Graph Reps

The ratio E:V can vary considerably. Graph ADT

> If E is closer to V^2 , the graph is dense. If E is closer to V, the graph is sparse.



dense graph



sparse graph

Knowing whether a graph is dense or sparse will affect our choice of representation and algorithms.

Graph Terminology

(111)

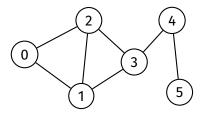
Types of Graphs
Graph Terminology
Graph ADT

Graph Reps

A path is
a sequence of vertices where
each vertex has a edge to the next in the
sequence

A path is simple if it has no repeating vertices

A cycle is a path where only the first and last vertices are the same 0-1-2-0, 1-2-3-1, 0-1-3-2-0

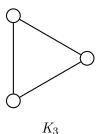


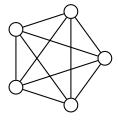
Graphs
Types of Graphs
Graph Terminology
Graph ADT

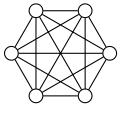
Graph Reps

A complete graph is a graph where every vertex is connected to every other vertex via an edge.

In a complete graph, $E=\frac{1}{2}\,V(\,V-1)$.





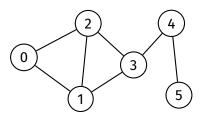


Types of Graphs
Graph Terminology

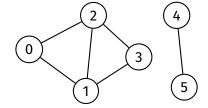
Graph ADT

Graph Reps

A connected graph is a graph where there is a path from every vertex to every other vertex.



Connected graph



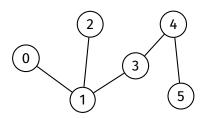
Disconnected graph

Types of Graphs
Graph Terminology
Graph ADT

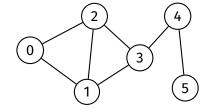
Graph Reps

A tree is a connected graph with no cycles.

A tree has exactly one path between each pair of vertices.



Tree

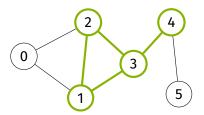


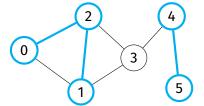
Not a tree

Graphs
Types of Graphs
Graph Terminology
Graph ADT

Graph Reps

A subgraph of a graph ${\cal G}$ is a graph that contains a subset of the vertices of ${\cal G}$ and a subset of the edges between these vertices.





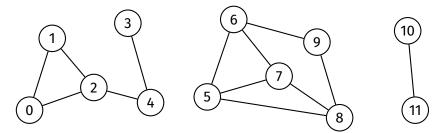
Graphs
Types of Graphs
Graph Terminology

Graph ADT

Graph Reps

A connected component is a maximally connected subgraph.

A connected graph has one connected component — the graph itself. A disconnected graph has two or more connected components.

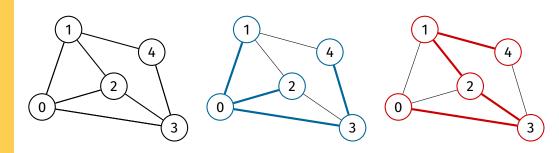


Graphs
Types of Graphs
Graph Terminology
Graph ADT

Graph Reps

A spanning tree of a graph ${\cal G}$ is a subgraph that contains all the vertices of ${\cal G}$ and is a single tree.

Spanning trees only exist for connected graphs.

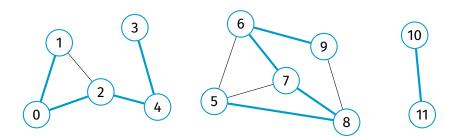


Graphs
Types of Graphs
Graph Terminology

Graph ADT

Graph Reps

A spanning forest of a graph ${\cal G}$ is a subgraph that contains all the vertices of ${\cal G}$ and contains one tree for each connected component.



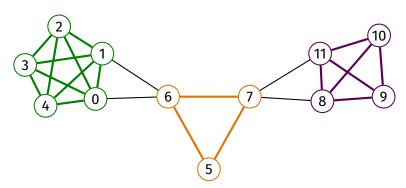
Graphs
Types of Graphs
Graph Terminology

Graph ADT

Graph Reps

A clique is a complete subgraph.

A clique is non-trivial if it has 3 or more vertices.



Graph ADT

Graph Reps

Graph ADT

Graphs
Graph ADT

Graph Reps

What do we need to represent? What operations do we need to support?

Graph ADT
Graph Reps

What do we need to represent?

A set of vertices $V:=\{v_1,\cdots,v_n\}$ A set of edges $E:=\{(v,w)\,|\,v,w\in V\}$

What operations do we need to support?

create/destroy graph add/remove edges get #vertices, #edges check if an edge exists

Graph ADT Operations

Graphs

Graph ADT

Graph Reps

create/destroy

create a graph free memory allocated to graph

query

get number of vertices get number of edges check if an edge exists

update

add edge remove edge

We will extend this ADT with more complex operations later.

```
Graph ADT
Graph Reps
```

```
typedef struct graph *Graph;

// vertices denoted by integers 0..V-1
typedef int Vertex;

/** Creates a new graph with nV vertices */
Graph GraphNew(int nV);

/** Frees all memory allocated to a graph */
void GraphFree(Graph g);
```

```
Graph ADT
Graph Reps
```

```
/** Returns the number of vertices in a graph */
int GraphNumVertices(Graph g);

/** Returns the number of edges in a graph */
int GraphNumEdges(Graph g);

/** Returns true if there is an edge between the given vertices
    and false otherwise */
bool GraphIsAdjacent(Graph g, Vertex v, Vertex w);
```

```
Graph ADT
```

Graph Reps

```
/** Inserts an edge into a graph */
void GraphInsertEdge(Graph g, Vertex v, Vertex w);
/** Removes an edge from a graph */
void GraphRemoveEdge(Graph g, Vertex v, Vertex w);
```

Graph ADT

Graph Reps

Adjacency Matrix Adjacency List Array of Edges

Graph Representations

Graph Representations

Graphs

Graph ADT

Graph Reps

Adjacency List Array of Edges Summary 3 main graph representations:

Adjacency Matrix

Edges defined by presence value in $V \times V$ matrix

Adjacency List

Edges defined by entries in array of \it{V} lists

Array of Edges

Explicit representation of edges as (v, w) pairs

We'll consider these representations for unweighted, undirected graphs.

Adjacency Matrix

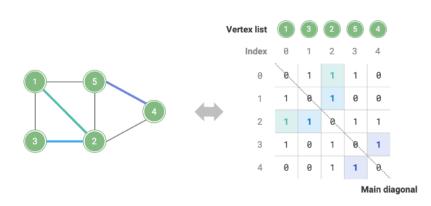
Graphs

Graph ADT

Graph Reps

Adjacency Matrix Adjacency List Array of Edges

A $V \times V$ matrix; each cell represents an edge.



Graph

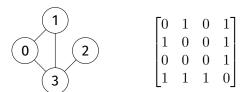
Adjacency matrix

Graph ADT

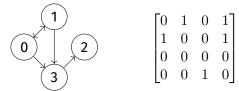
Graph Reps Adiacency Matrix

Adjacency List Array of Edges Summary

A $V \times V$ matrix; each cell represents an edge.



undirected



directed

Adjacency Matrix

Implementation in C

Graphs

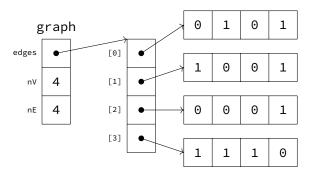
Graph ADT

Graph Reps

Adjacency Matrix Adjacency List Array of Edges Summary

```
struct graph {
   int nV;
   int nE;
   bool **edges;
};
```

3



Adjacency Matrix

Advantages and Disadvantages

Graphs

Graph ADT

Graph Reps
Adiacency Matrix

Adjacency List Array of Edges Summary

Advantages

Efficient edge insertion/deletion and adjacency check (O(1))

Disadvantages

Huge memory usage $(O(V^2))$ sparse graph \Rightarrow wasted space! undirected graph \Rightarrow wasted space!

Adjacency List

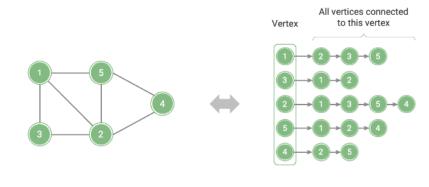
Graphs

Graph ADT

Graph Reps

Adjacency Matrix Adjacency List

Array of Edges



Graph

Adjacency list

Graph ADT

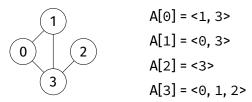
Graph Reps

Adjacency Matrix

Adjacency List

Array of Edges

Array of V lists



undirected

directed

Adjacency List

Implementation in C

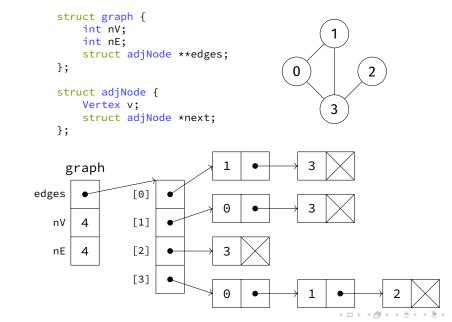
Graphs

Graph ADT

Graph Reps

Adjacency Matrix

Adjacency List Array of Edges



Graph ADT

Graph Reps
Adiacency Matrix

Adiacency List

Array of Edges

Advantages

Space-efficient for sparse graphs O(V+E) memory usage

Disadvantages

Inefficient edge insertion/deletion (O(V)) (matters less for sparse graphs)

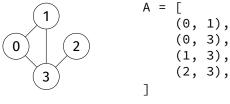
Graph ADT

Graph Reps Adjacency Matrix

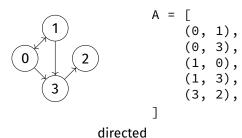
Array of Edges

Summany

Edges represented by an array of edge structs (pairs of vertices)



undirected



Array of Edges

Implementation in C

Graphs

Graph ADT

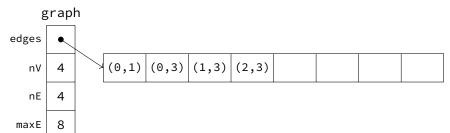
Graph Reps Adjacency Matrix

Adjacency List Array of Edges

andy or Eugh

```
struct graph {
   int nV;
   int nE;
   int maxE;
   struct edge *edges;
};

struct edge {
   Vertex v;
   Vertex w;
};
```



Array of Edges

Advantages and Disadvantages

Grapns

Graph ADT

Graph Reps
Adiacency Matrix

Adjacency List

Array of Edges

Advantages

Very space-efficient for sparse graphs where E < V

Disadvantages

 $\begin{array}{c} \text{Inefficient} \\ \text{edge insertion/deletion (}O(E)\text{)} \end{array}$

Summary of Graph Representations

Graphs

Graph ADT

Graph Reps
Adjacency Matrix
Adjacency List
Array of Edges
Summary

	Adjacency Matrix	Adjacency List	Array of Edges
Space usage	$O(V^2)$	O(V+E)	O(E)
Create	$O(V^2)$	O(V)	O(1)
Destroy	O(V)	O(V+E)	O(1)
Insert edge	O(1)	O(V)	O(E)
Remove edge	O(1)	O(V)	O(E)
Is adjacent	O(1)	O(V)	$O(E)^*$
Degree	O(V)	O(V)	$O(E)^*$

^{*} Can be $O(\log E)$ if the array is ordered and both directions of each edge are stored in an undirected graph

Graph ADT

Graph Reps

Adjacency Matrix Adjacency List Array of Edges Summary https://forms.office.com/r/zEqxUXvmLR

