Lecture hall scheduling (single day)

Course 1	9:15 - 11:30
Course 2	10:00 - 12:00
Course 3	11:30 - 1:00
Course 4	12:15 - 2:00

Allocate classrooms, no two courses are assigned same room at same time

How many classrooms do we need? (Minimum)

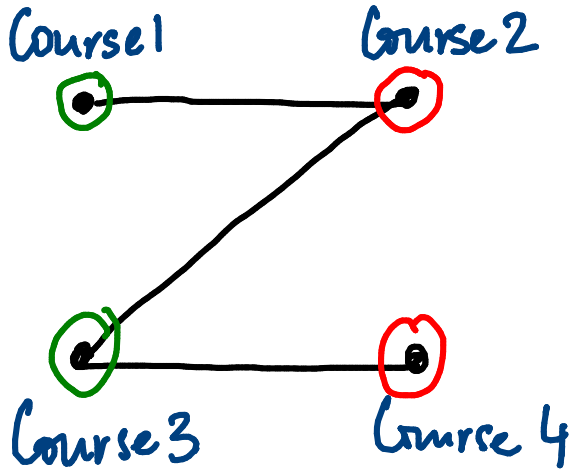
Here 2: Room 1 : Course 1,3

Room 2 : Course 2,4

Sort courses by starting time and scan left to right

Each new course – check if a room is free, else add a new room $O(n^2)$?

Also sort by end time ...



Edge if timing overlap

"Colour" the graph
Colour each vertex
Edge has diff colours at endpoints

Min # colours = Chromatic No.

Graphs are useful for modelling.

Country divided into states

Map colouring:
Neighbouring states are coloured different

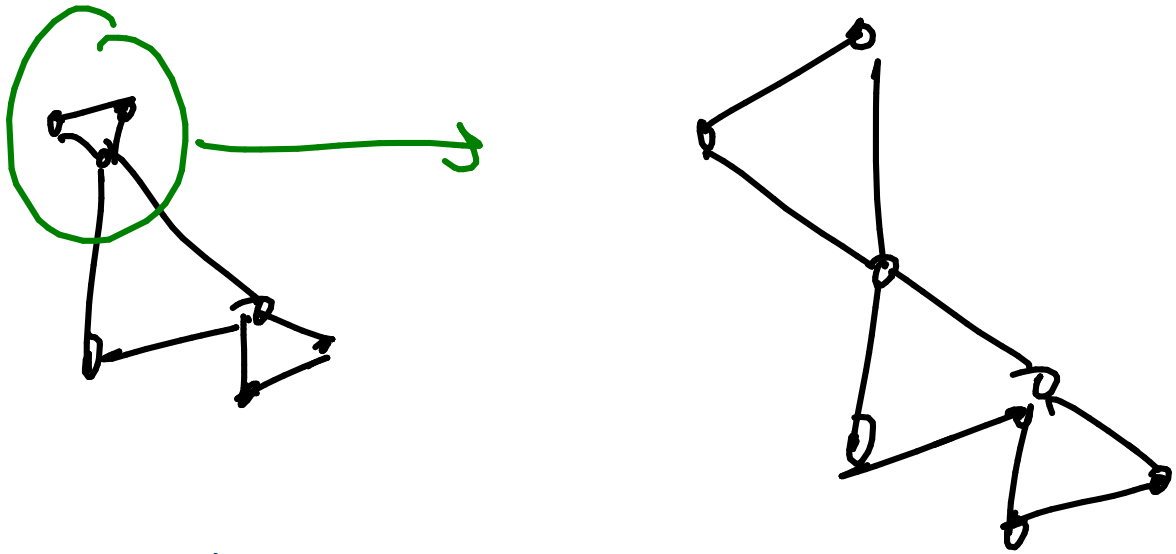
Graph:
Vertex per state
Edge: neighbour

Map colouring = Graph colouring

PLANAR GRAPH - can be drawn with no edges crossing

4 colours suffice

Moving to graphs – "redraw" etc

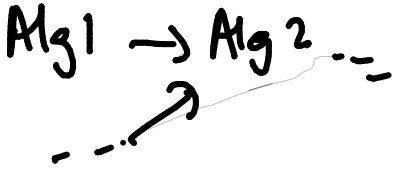


Other examples:

Airline routes – check connectivity

Course dependencies

Directions



Algorithmic manipulation of graphs

$G = (V, E)$ V: Vertices — n $\{1, 2, \ldots, n\}$

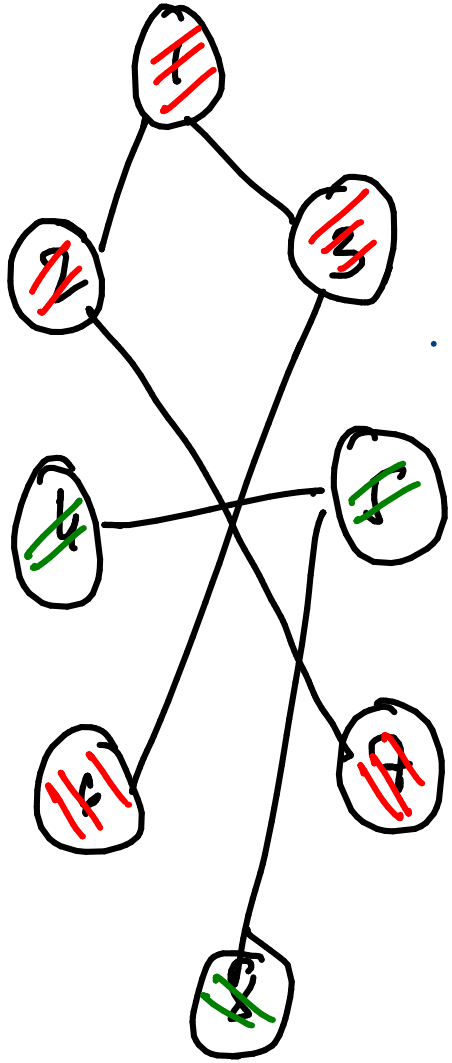
E: Edges — m

Undirected $E \subseteq V \times V$ $(i, j) \in E$ iff $(j, i) \in E$

Directed E is not symmetric in general

Loops? Assume (i, i) is not an edge

Most of our examples do not have loops



Can I go from 1 to 8?

No. How?

Start at 1.

Mark/colour 1 and everything connected to 1

Keep extending the marking via edges till you can't colour any more vertices

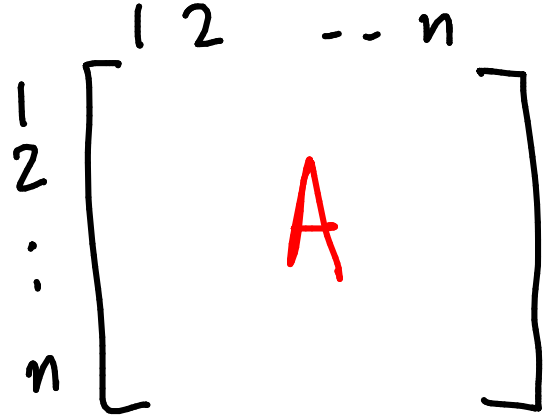
How do we code this?

Represent the graph

$n \times n$ matrix

$A[i,j] = 1$ iff $(i,j) \in E$,

0 otherwise



ADJACENCY MATRIX

Is (i,j) an edge? Check $A[i,j] == 1$ — $O(1)$

Degree(i) = # neighbours? No of 1's in row i — $O(n)$

Collect list of neighbours of i — Scan row i — $O(n)$

Formalize earlier algorithm

EXPLORE
- Fix a start node
- Colour it
- Examine all neighbours, colour them if not coloured

"Explore" each coloured node once

Each vertex has two distinct attributes

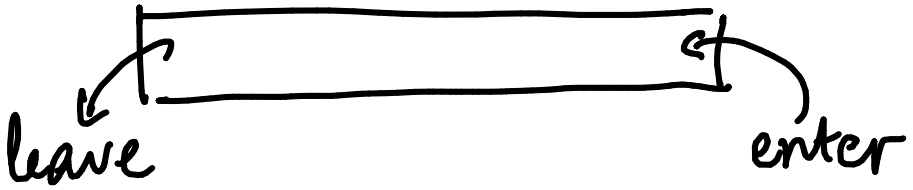
Coloured — Has been reached from Start

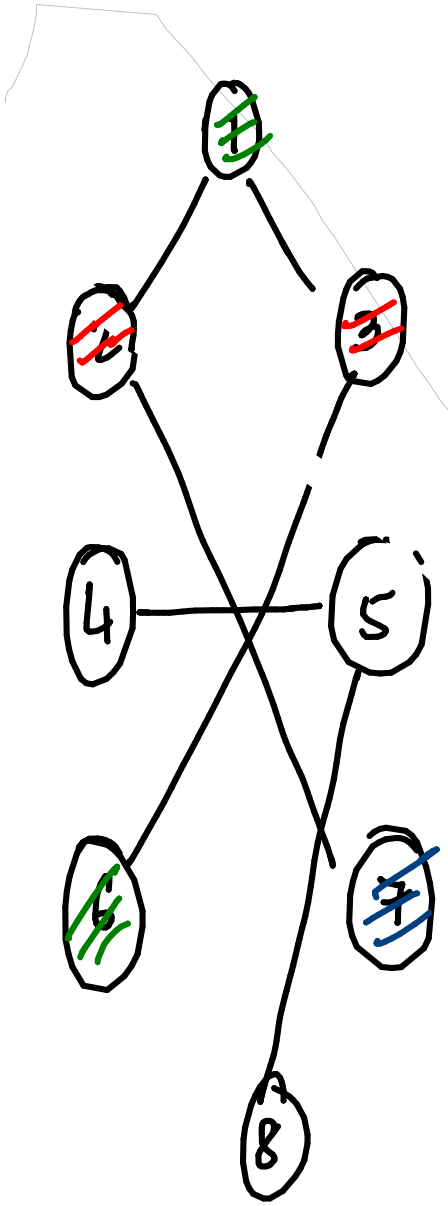
Explored — Its neighbours have been coloured

Each vertex is explored after it is coloured

When we colour a vertex for the first time "queue it up" for exploration

Queue:





Start at 1 [BREADTH FIRST SEARCH]

Colour 1, put in queue

Pull 1 out of queue, colour/add 2,3

Pull 2, colour/add 7 (not 1)

Pull 3, colour/add 6 (not 1)

Pull 7, no change

Pull 6, no change

Empty queue - stop!

Queue | ~~1~~ ~~2~~ ~~3~~ ~~7~~ ~~6~~ |

Colours indicate iterations

Assign same "colour" to all