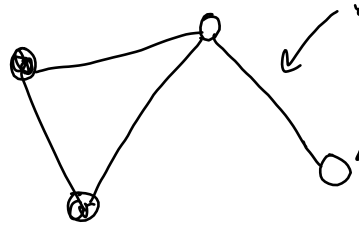
Graphs 1

Monday, March 23, 2020 10:58 AM

Today:

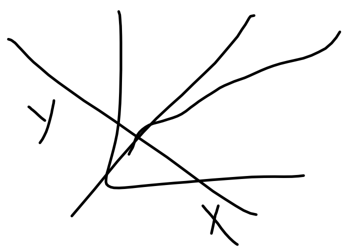
- Graphs terminology
- Modelling problems
- Representing graphs
- Graph traversals / graph search



edges / arcs

nodes, vertices

vertex



$G = (V, E)$

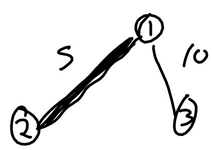
V: set of vertices,

$E \subseteq V^2$

$E = \{ (v_1, v_2) \ldots \}$

"there is an edge between v_1 and v_2 in G"

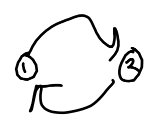
other features
- attributes about the nodes or about the edges aka label
- directed or undirected.



undirected graph:

$(v_1, v_2) \in E \iff (v_2, v_1) \in E$

in a directed graph



- ok to have edge between (v_1, v_2) and (v_2, v_1)

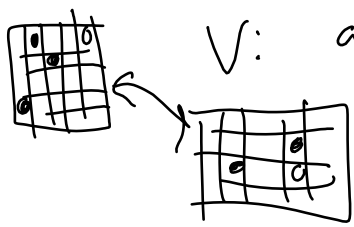
"multigraph"

- Simple undirected graph:
  - at most one edge (v_1, v_2)
  - no self loops
- trees

model a problem with a graph

example	what are the nodes?	what are the edges?	what are the attributes	directed
s	states	transitions	symbols	yes

DFA				
Roadmaps	junctions	segments	speed limit	no – only two way yes if one way roads
Chess?	gamestates & whose move	moves	?	yes
FB friends	people	friend relation (mutual)	duration? relationship status	no

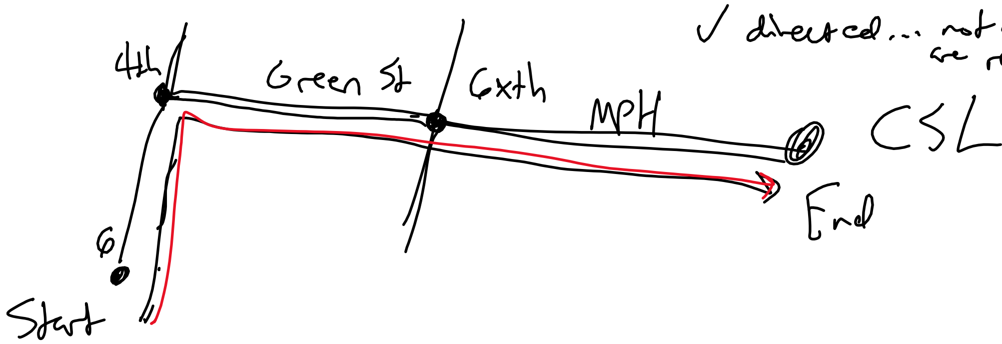


V: all possible game states

E: (v_1, v_2) iff

v_2 is the result of applying a valid chess move.

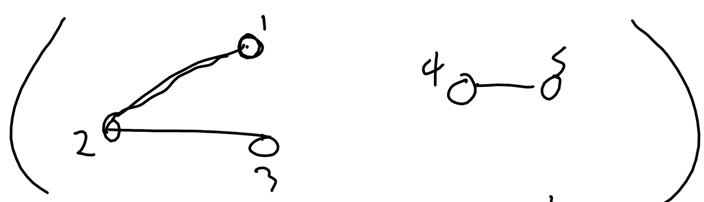
✓ directed... not all moves are reversible.



Abstract questions about graphs

– Is there a path between v_1 and v_2

G =



e.g. no path b/w 1 and 4

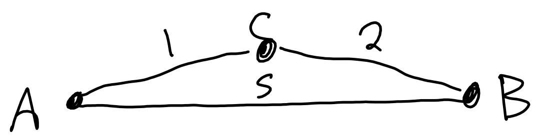
source/start ↓ sink/end ↓

Defn: Path between v_0 and v_k is a sequence of nodes $(v_0, v_1, \ldots v_k)$ s.t.

for all i in 0 K-1, $(v_i, v_{i+1}) \in E$

- related: what's the shortest path?

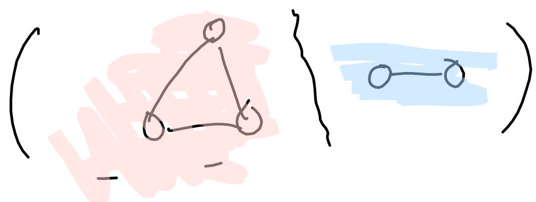
  if there are weights on edges, may be interest smallest weight path



$A \rightsquigarrow B$ of length 1

$A \rightsquigarrow B$ shortest weighted path is 3

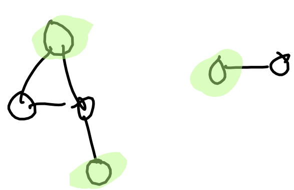
- Connectedness

G =


Def'n unconnected graph: $\exists\, v_1, v_2$ s.t. no path from $v_1 \rightsquigarrow v_2$

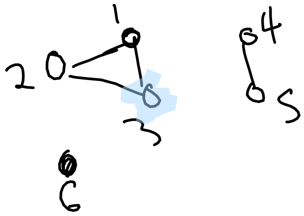
Count the number of connected components

- Independent sets set of nodes $\{v_1, \dots, v_n\}$ s.t. no pair has an edge b/w them



Representing graphs

- Adjacency matrix $E \subseteq V^2$



deg(3)?

	1	2	3	4	5
1	0	1	1	0	0
2	1	0	1	0	0
3	1	1	0	0	0
4	0	0	0	0	1
5	0	0	0	1	0

- List of lists (array of arrays)

For each node, store a list of neighbors

1: [2,3]
2: [1,3]
3: [1,2]
4: [5]
5: [4]
→ 6: []

Metrics: $G = (V, E)$

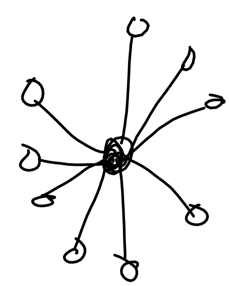
→ $n = |V|$ # of nodes

→ $m = |E|$ # of edges

Constraints: $m \leq n^2$
(for a simple graph)

$O(n+m)$ is a smaller bound $O(n^2)$
if $m \lll n^2$
"much smaller than"

	storage	running time for deg()
adj. mat	n^2 $\|V\|^2$	$O(n)$
list of lists	$O(n+m)$	$O(n+m)$
array of arrays	$O(n+m)$	$O(1)$

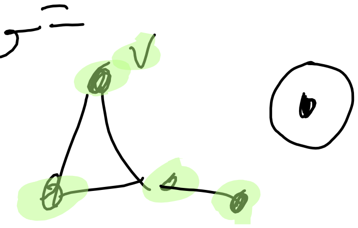


degree($\underset{\uparrow \text{node}}{v}$): # of neighbors

Traversals:

Given v, find all nodes in $\underline{\text{Conn}}(v)$

G =



$\text{Conn}(v): \{ v' \in V \mid \exists \text{ path } v \rightsquigarrow v' \}$

Visited := ["False" for every $v \in V$] // size n array

- Explore(v):
  // Starting from v, "visit" every node reachable from v
  if v not visited: mark v
  for neighbor w ∈ neighbors(v):
  if visited[w] = False:
  set visited[w] := True
  Explore(w)