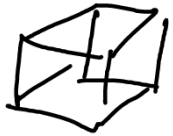
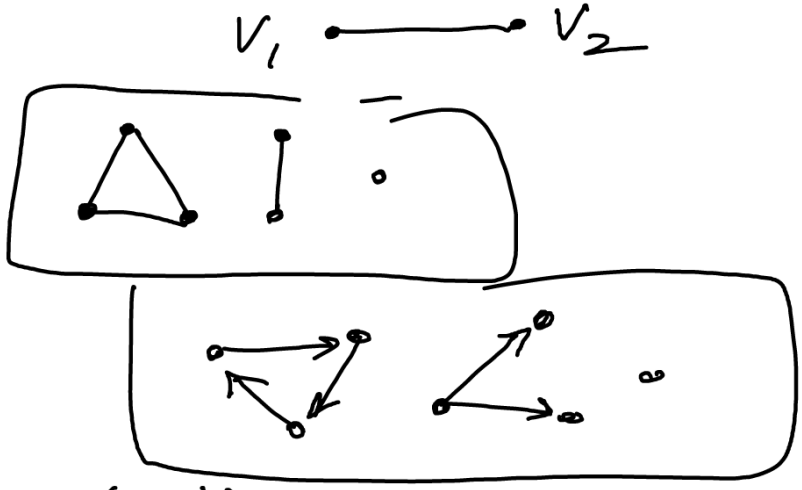
Graphs

Sample exam available today

A set of vertices (or nodes) and edges connecting pairs of vertices.

undirected graph





directed graph (digraph)

$V_1 \longrightarrow V_2$

edge has source and sink/destination

self-edges, duplicate edges: sometimes

Powerful abstraction!

	vertices	edges	
— Program objects	= objects	pointers	(directed)
— Games / puzzles	states	moves	
— Scheduling events	events	conflict	(undirected)
— Any binary relation	objects	relationships	

Problem ⟶ Graph + graph algorithms toolbox + intractability results

Notation set of nodes/vertices V (size = $|V|$)
" edges E

Directed Graph: $E = \{ (v_1, v_2), (v_3, v_4), \dots \}$

$v_1 \longrightarrow v_2$
$v_3 \longrightarrow v_4$
⋮

$(v_1, v_2) \in E$ means $v_1 \longrightarrow v_2$ (Also: $v_1 \prec v_2$, $v_2 \succ v_1$ — predecessor)

Undirected Graph

$E = \{ \{v_1, v_2\}, \{v_3, v_4\}, \ldots \}$

$\{v_1, v_2\} \in E$ means $v_1 \text{ —— } v_2$

Adjacent nodes $v_1 \sim v_2$

connected by an edge

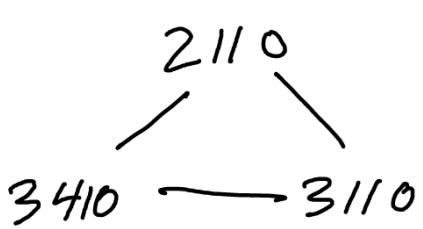
$v_1 \longrightarrow v_2$

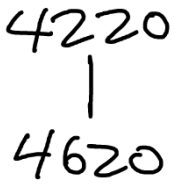
$v_2 \longrightarrow v_1$

$v_1 \text{ —— } v_2$

Directed graphs: trees, lists, game states

Undirected graphs: schedule events





valid schedule?
has no adjacent vertices.

adjacent: 5

in-degree 2

out-degree of vertex: # exiting edges

in-degree of vertex: # entering edges

degree: # adjacent vertices

} directed graphs

out-degree: 3

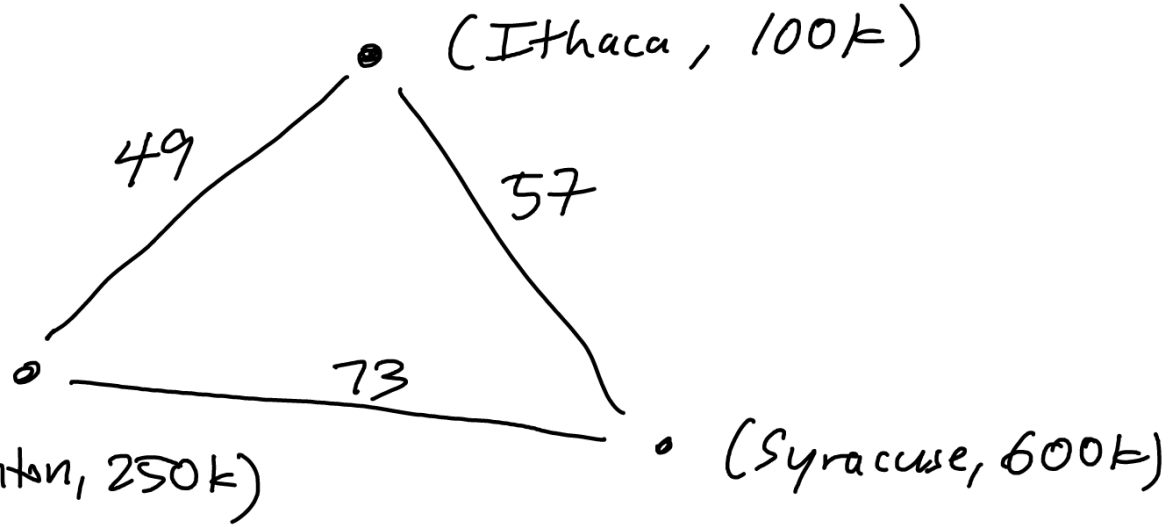
A-E: 1-5

Labeled Graphs

Nodes, edges can have labels

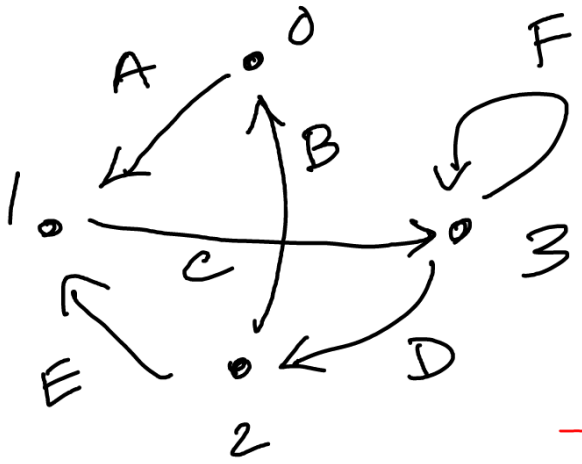
Weighted graph : edge labels are numbers.

Ex/



Representations

Adjacency matrix



m[src][dest] =
- whether edge src→dest exists.
- or label for edge
  e.g. Maybe<String>

	0	1	2	3
0	/	A	/	/
1	/	/	/	C
2	B	E	/	/
3	/	/	D	F

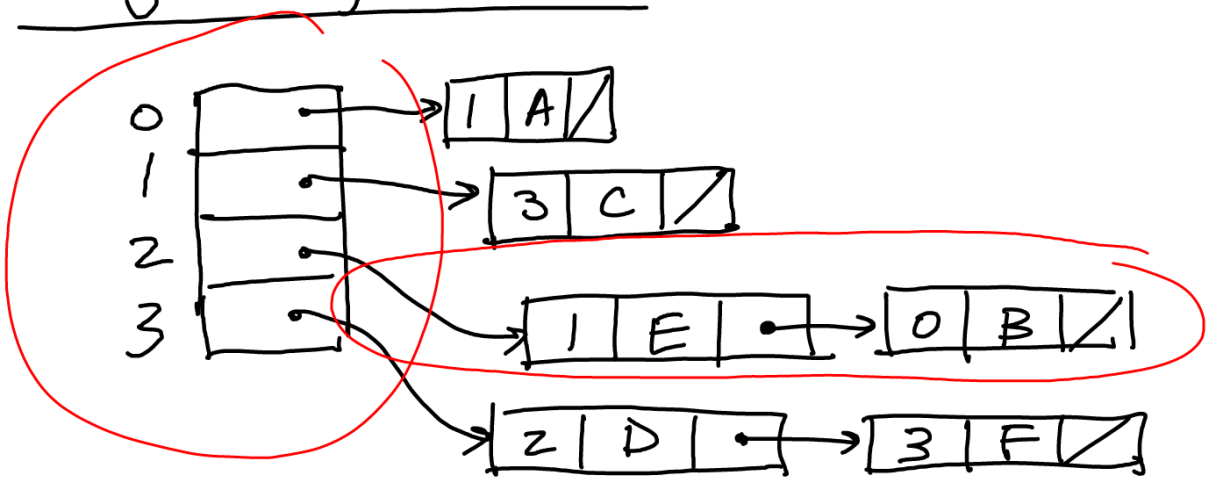
A: O(1)
B: O(lg V)
C: O(V)
D: O(V²)
E: O(E)

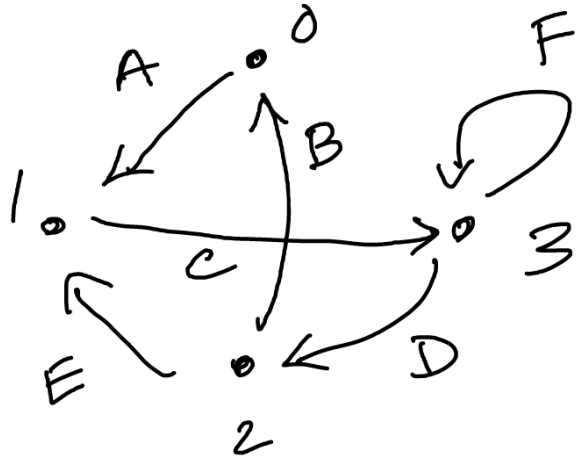
good for dense graphs

$E = O(V^2)$

- Testing $x \rightarrow y$?
- Find all incoming/outgoing edges? $O(V)$
- space? $O(V^2)$

Adjacency list





- Test $x \to y$?
- Find outgoing edges?
  $O(V)$
- Find incoming edges
  $O(V+E)$
- Space: $O(V+E)$

A: $O(1)$
B: $O(\lg V)$
C: $O(V)$
D: $O(V^2)$
E: $O(E)$

$\Rightarrow$ good for sparse graph $E \ll V^2$

Speed up $x \to y$?
use hash table.
$\Rightarrow$ HashMap<V, E> $\Rightarrow O(1)$

Build dual graph where $x \to y$ in dual if $y \to x$ in graph.

Paths & Cycles

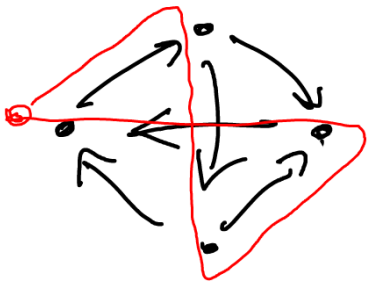
Walk: sequence $(v_0, v_1, \ldots, v_p)$ where $v_i \to v_{i+1}$ for all $i \in [0, p)$

same: "path"

length = #edges $= p$

(simple) path: walk with no repeated nodes (exc. maybe $v_0 = v_p$)

cycle: simple path where $v_0 = v_p$



length of longest cycle: (1–5)

4

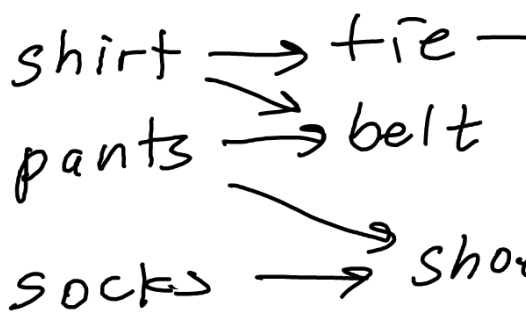
DAG directed graph with no cycles — acyclic

directed acyclic graph = DAG (linked list, tree)

Common use: dependencies

$x \rightarrow y$: "x must happen before y"

Ex/ Men's formal dressing problem



DAG $\Leftrightarrow$ Can be dressed

dressing plan = topological sort of graph (all edges go fwd)

pants, shirt, belt, socks, tie, jacket, shoes

or: shirt, tie, jacket, socks, pants, belt, shoes

Other graph problems

- Reachability: what nodes lie on a path from some source
- Shortest path: from a source node to all nodes
  " all source nodes

↑ efficiently solvable

Not:

- Hamiltonian cycle: is there a length-$|V|$ cycle?
- Graph coloring: $v_1 \sim v_2$ if v_1 & v_2 conflict.
  if v_i are jobs, how many time slots are needed.