CS 340 - lecture 23 - Oct 30

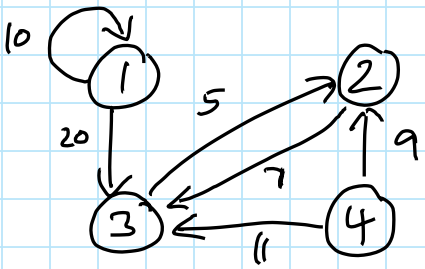
How to represent graphs?

- adjacency matrix (dense graphs - many edges)
- adjacency list (sparse graphs - few edges)

choice of representation depends on problem to be solved

adjacency matrix A:

two-dimensional array of size $|V| \times |V|$; entry $A[v][w]$ is TRUE if $(v,w) \in E$ and FALSE otherwise



A:

	1	2	3	4
1	T	F	T	F
2	F	F	T	F
3	F	T	F	F
4	F	T	T	F

column 4 has no T in it $\Rightarrow$ vertex 4 has no incoming edge

for weighted graphs, A contains cost values instead of "T": $A[v][w]$ is $c(v,w)$ if $(v,w) \in E$

-11- is some special value otherwise

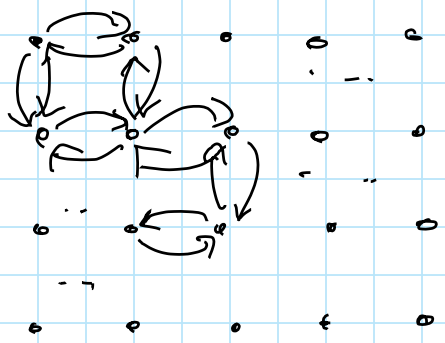
(∞, $-\infty$, 0, F, ..., depending on appl.)

A:

	1	2	3	4
1	10	∞	20	∞
2	∞	∞	7	∞
3	∞	5	∞	∞
4	∞	9	11	∞

Note:

- The max. number of edges a graph $G=(V,E)$ can have is $|V|^2$.
- Typically, graphs of interest have substantially fewer edges.



$\rightsquigarrow |E| \approx 4 \cdot |V| = \Theta(|V|)$
$= o(|V|^2)$

assume $|V| = 5{,}000$

$\rightsquigarrow |E| \approx 20{,}000$

but adjacency matrix is of size

$25{,}000{,}000$

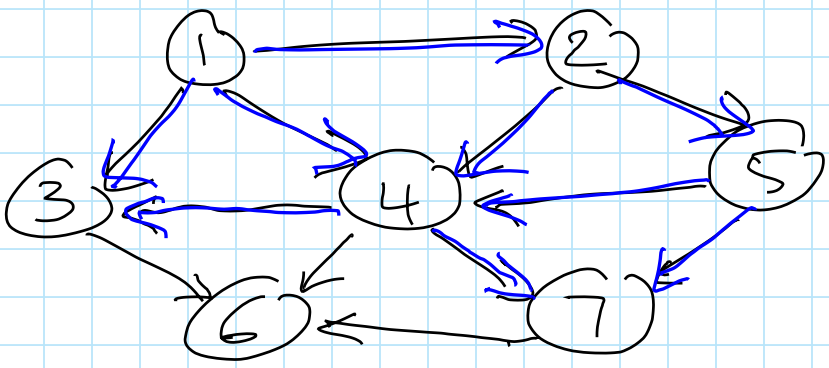
$\rightsquigarrow$ sometimes other representations are less "wasteful".

adjacency list:

for each vertex $v \in V$ keep a list of all adjacent vertices

$\rightsquigarrow$ space complexity: $\Theta(|V|+|E|)$

Example 42.

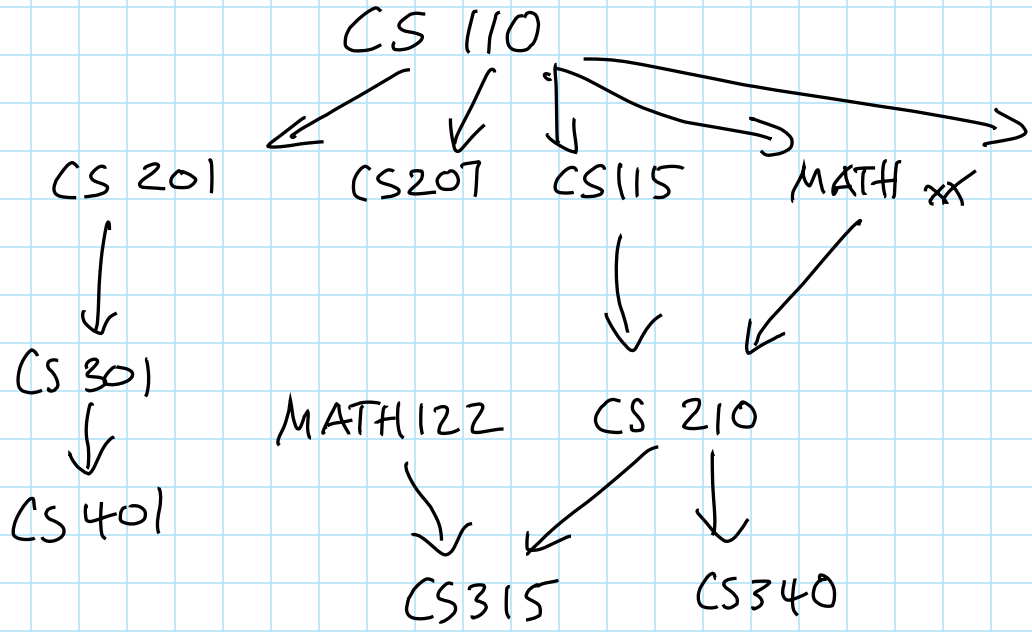


1	2	3	4	5	6	7
2	4	6	3	4		6
4	5		6	7		
3			7			

(empty)

4.2. Topological Sort

Example 43. class prereq. structure as graph



produce a list of all nodes such that a course appears only after (to the right of) all its prereq.'s.

⇝ topological sort / ordering

Definition 15. A topological sort/ordering of a directed graph $G = (V, E)$ is a list $v_1, \ldots, v_z$ of all vertices in V in which

$i < j$ implies that there is no path from v_j to v_i.

Properties

- Not every graph has a topological sort.
  Graphs containing cycles don't.
- A graph may have more than one top sort.

Example 42 cfd.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
a top. sort:	1,	2,	5,	4,	3,	7,	6
another top. sort:	1,	2,	5,	4,	7,	3,	6

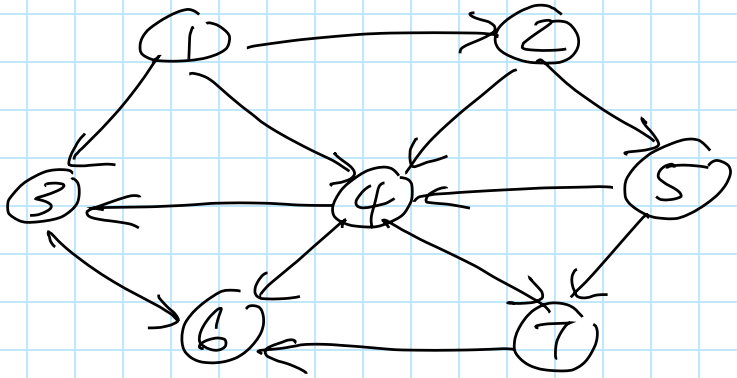
Definition 16. Let $G = (V, E)$ be a directed graph, $v \in V$. The in-degree of v in G is the number of $w \in V$ for which $(w, v) \in E$.
"number of in-coming edges to v"

idea for top. sort:

- maintain in-degrees for all vertices in a list, store those with in-degree zero separately
- output a vertex w/ in-degree 0, remove it, and decrement in-degrees of all its adjacent vertices
- repeat

Assume graph is initially given as adjacency list!

Example 42 cfd.



	1	2	3	4	5	6	7	output
in-deg:	0	1	2	3	1	3	2	1
	[0]	0	1	2	1	3	2	2
	[0]	[0]	1	1	0	3	2	5
	[0]	[0]	1	0	[0]	3	1	4
	[0]	[0]	0	[0]	[0]	2	0	7 (3)
	[0]	[0]	0	[0]	[0]	1	[0]	3
	[0]	[0]	[0]	[0]	[0]	0	[0]	6

time complexity (when starting from adjacency list):
$O(|V| + |E|)$