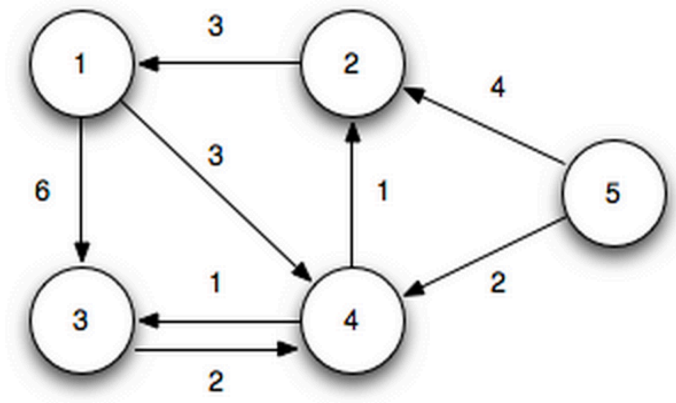
Algorithm design techniques

- divide and conquer
- incremental
- Dynamic Programming & Greedy

Use Graph Algorithms (esp. shortest paths) as examples

- Graph Representation

Graphs



$G = (V,E)$

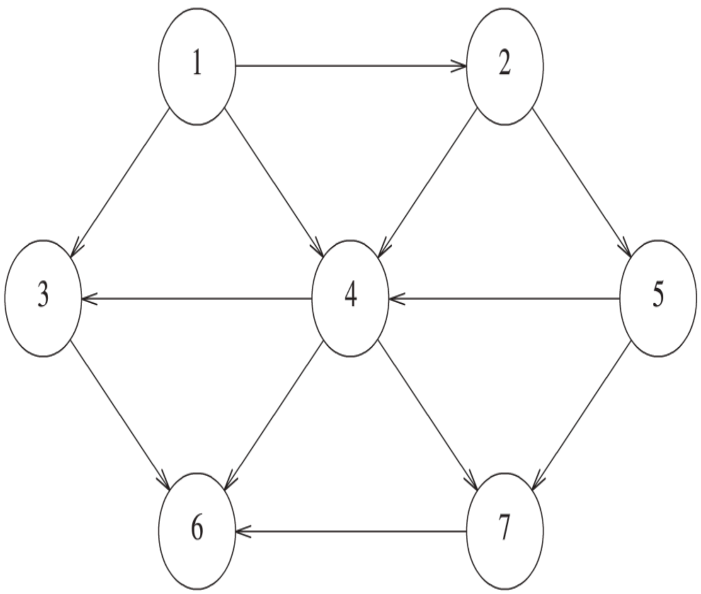
V the vertices of the graph $\{v_1, v_2, ..., v_n\}$

E the edges; E a subset of V x V

A cost function – c_{ij} is the cost/ weight of the edge (v_i, v_j)

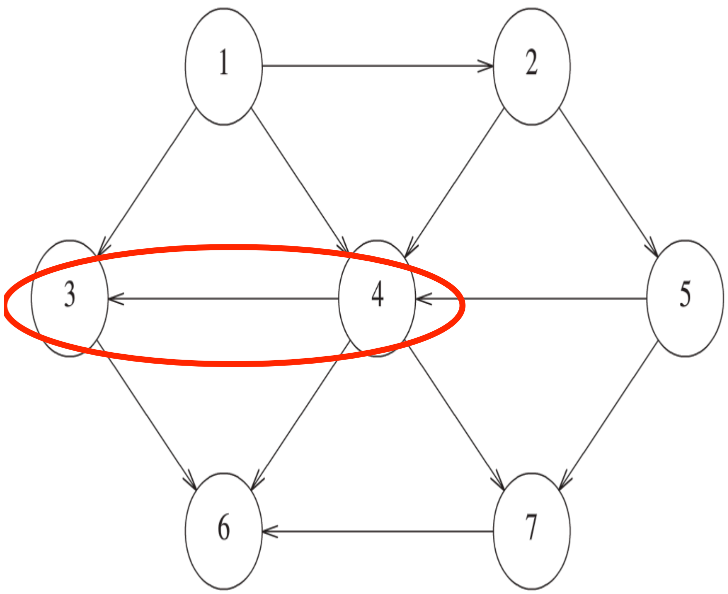
Graph G=(V,E): representation

1. Adjacency Matrix – a |V| x |V| matrix, with the [i,j]'th entry representing the edge from the i'th to the j'th vertex
2. Adjacency List – an array of linked lists of length |V|, with the i'th entry denoting the edges from the i'th vertex



Graph G=(V,E): representation

1. Adjacency Matrix – a |V| x |V| matrix, with the [i,j]'th entry representing the edge from the i'th to the j'th vertex
2. Adjacency List – an array of linked lists of length |V|, with the i'th entry denoting the edges from the i'th vertex



	1	2	3	4	5	6	7
1	0	1	1	1	0	0	0
2	0	0	0	1	1	0	0
3	0	0	0	0	0	1	0
4	0	0	1	0	0	1	1
5	0	0	0	1	0	0	1
6	0	0	0	0	0	0	0
7	0	0	0	0	0	1	0

Graph G=(V,E): representation

1. Adjacency Matrix – a |V| x |V| matrix, with the [i,j]'th entry representing the edge from the i'th to the j'th vertex
2. Adjacency List – an array of linked lists of length |V|, with the i'th entry denoting the edges from the i'th vertex

Weighted graph: the matrix entries denote the edge-weights

Some sentinel value (depends on application) for non-existent edges

- E.g., shortest-path problems: ∞

Values along the diagonal

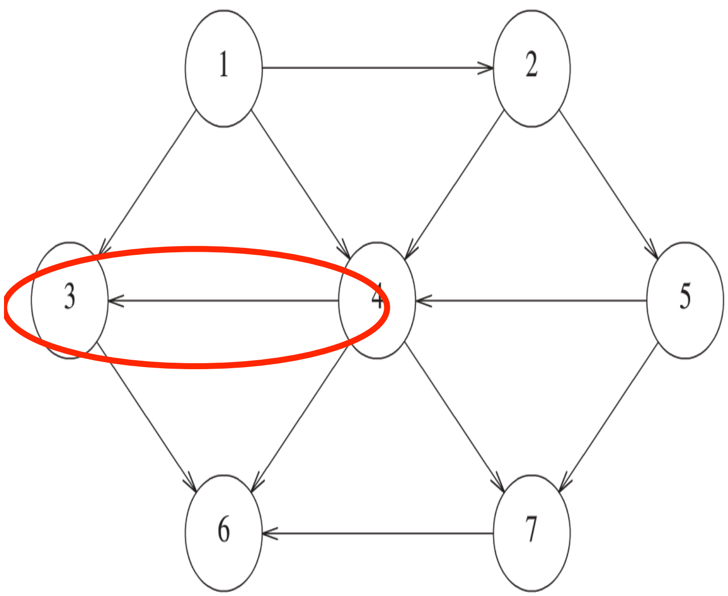
Undirected graph: symmetric along diagonal

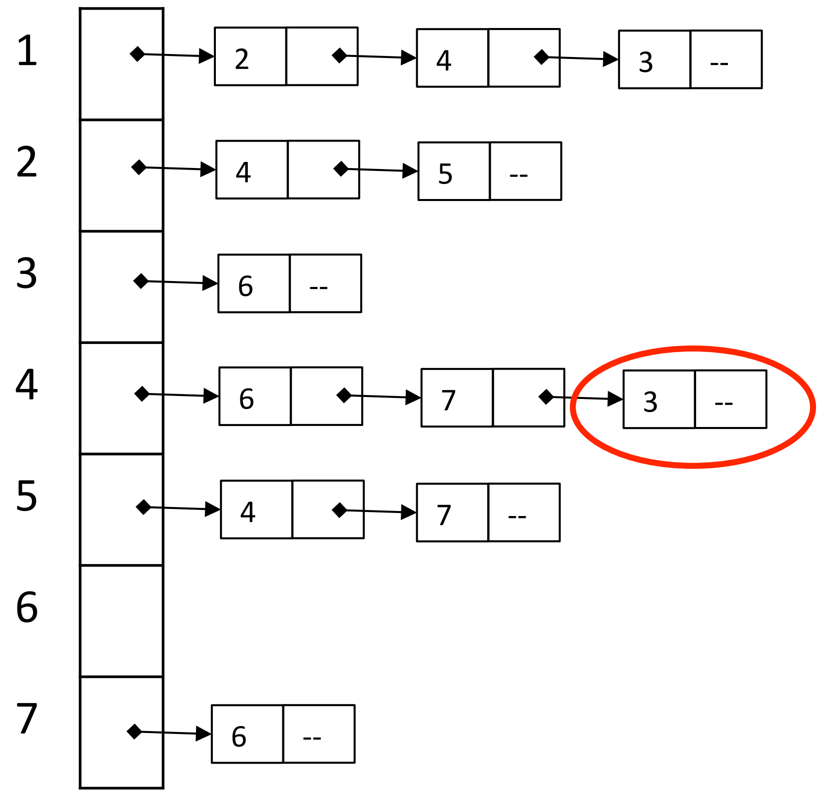
Memory requirement: $\Theta(|V|^2)$

- OK for dense graphs; too much for sparse graphs
- Road networks; social n'works; etc. tend to be sparse

Graph G=(V,E): representation

1. Adjacency Matrix – a |V| x |V| matrix, with the [i,j]'th entry representing the edge from the i'th to the j'th vertex
2. Adjacency List – an array of linked lists of length |V|, with the i'th entry denoting the edges from the i'th vertex





Graph G=(V,E): representation

1. Adjacency Matrix – a |V| x |V| matrix, with the [i,j]'th entry representing the edge from the i'th to the j'th vertex
2. Adjacency List – an array of linked lists of length |V|, with the i'th entry denoting the edges from the i'th vertex

Weighted graph: the list entries contain the edge-weights as well

The order of the edges within a list is irrelevant

Undirected graph: each edge appears in two lists

Memory requirement: $\Theta(|V| + |E|)$

- linear in the size of the graph

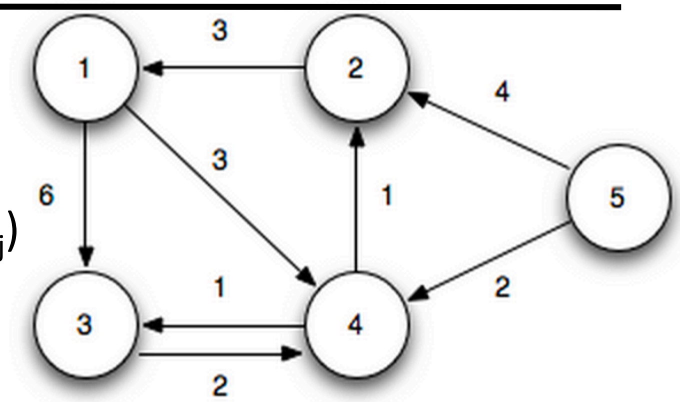
Graphs

G = (V,E)

V the vertices of the graph $\{v_1, v_2, ..., v_n\}$

E the edges; E a subset of V x V

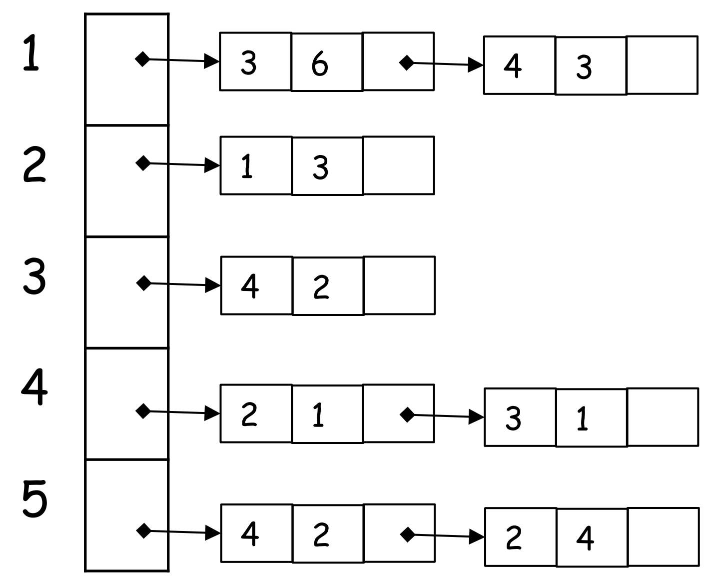
A cost function – c_{ij} is the cost/ weight of the edge (v_i, v_j)



Adjacency Matrix

	1	2	3	4	5
1	0	∞	6	3	∞
2	3	0	∞	∞	∞
3	∞	∞	0	2	∞
4	∞	1	1	0	∞
5	∞	4	∞	2	0

Adjacency List



Graphs

Memory – O(|V|^2) vs O(|V| + |E|)

Does a particular edge exist? – O(|1|) vs O(min(|V|,|E|))

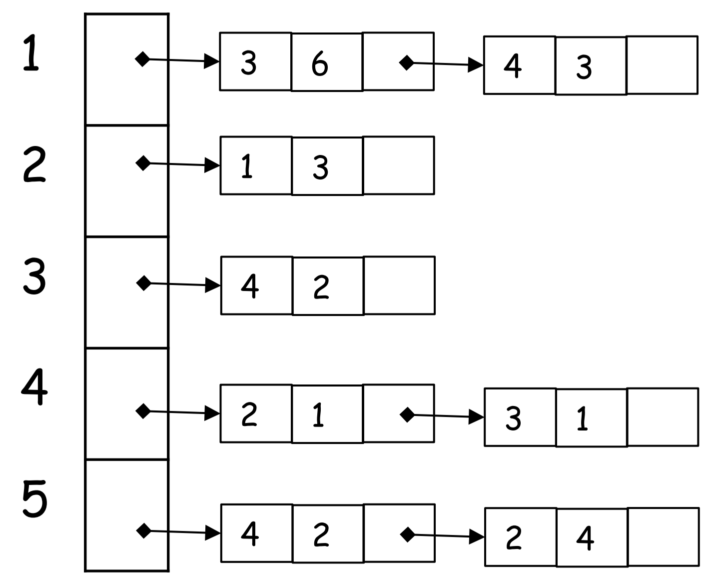
Outdegree of a particular vertex – O(|V|) vs O(min(|V|,|E|))

Indegree of a particular vertex – O(|V|) vs O(max(|V|,|E|))

Adjacency Matrix

	1	2	3	4	5
1	0	∞	6	3	∞
2	3	0	∞	∞	∞
3	∞	∞	0	2	∞
4	∞	1	1	0	∞
5	∞	4	∞	2	0

Adjacency List



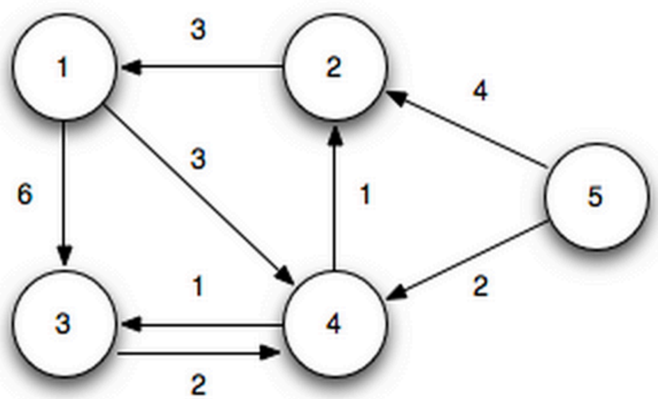
Graphs

22.1-5

The ***square*** of a directed graph $G = (V, E)$ is the graph $G^2 = (V, E^2)$ such that $(u, v) \in E^2$ if and only G contains a path with at most two edges between u and v. Describe efficient algorithms for computing G^2 from G for both the adjacency-list and adjacency-matrix representations of G. Analyze the running times of your algorithms.

Is there an edge between vertices 3 and 2 in G^2?

Is there an edge between vertices 3 and 5 in G^2?



	1	2	3	4	5
1	0	∞	6	3	∞
2	3	0	∞	∞	∞
3	∞	∞	0	2	∞
4	∞	1	1	0	∞
5	∞	4	∞	2	0