Directed Graphs

- Directed Graphs
- Representation
- Search
- Topological Sorting
- Directed Acyclic Graphs
- Strongly Connected Components
- Implicit Graphs

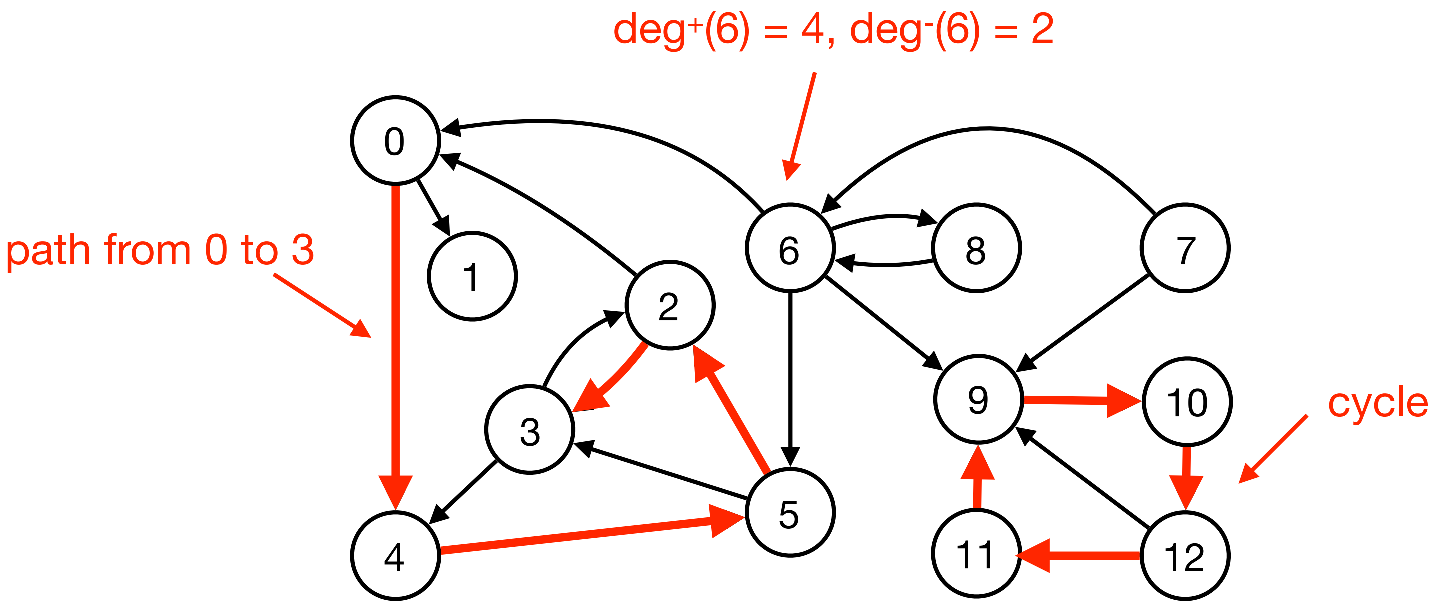
Philip Bille

Directed Graphs

- Directed Graphs
- Representation
- Search
- Topological Sorting
- Directed Acyclic Graphs
- Strongly Connected Components
- Implicit Graphs

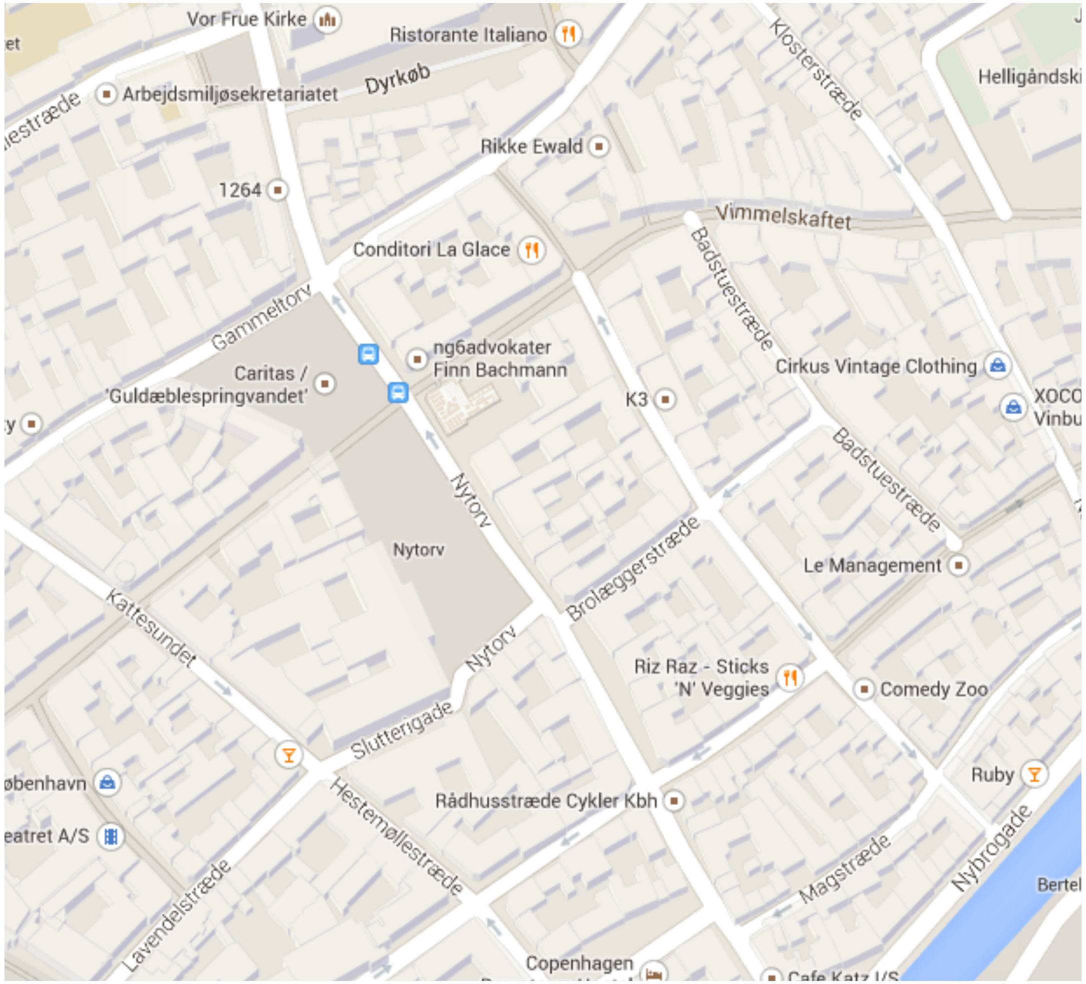
Directed Graphs

- Directed graph. Set of vertices pairwise joined by directed edges.



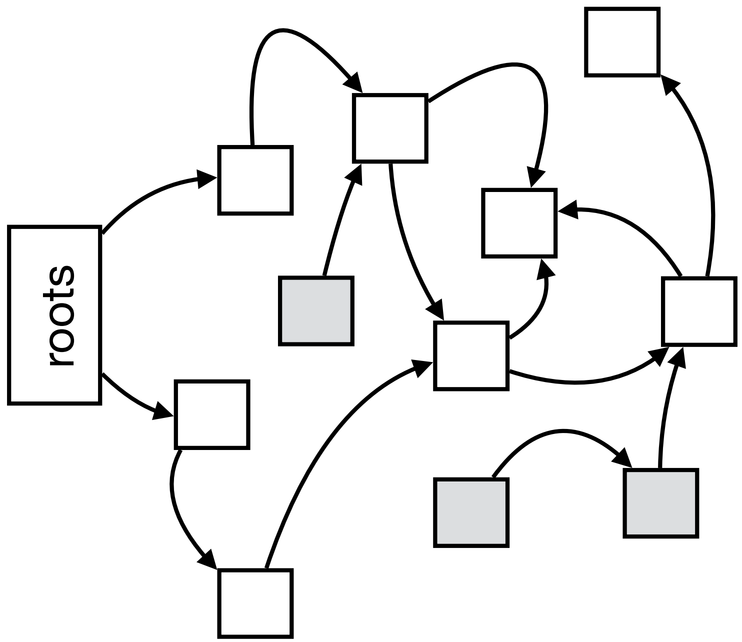
Road Networks

- Vertex = intersection, edge = (one-way) road.



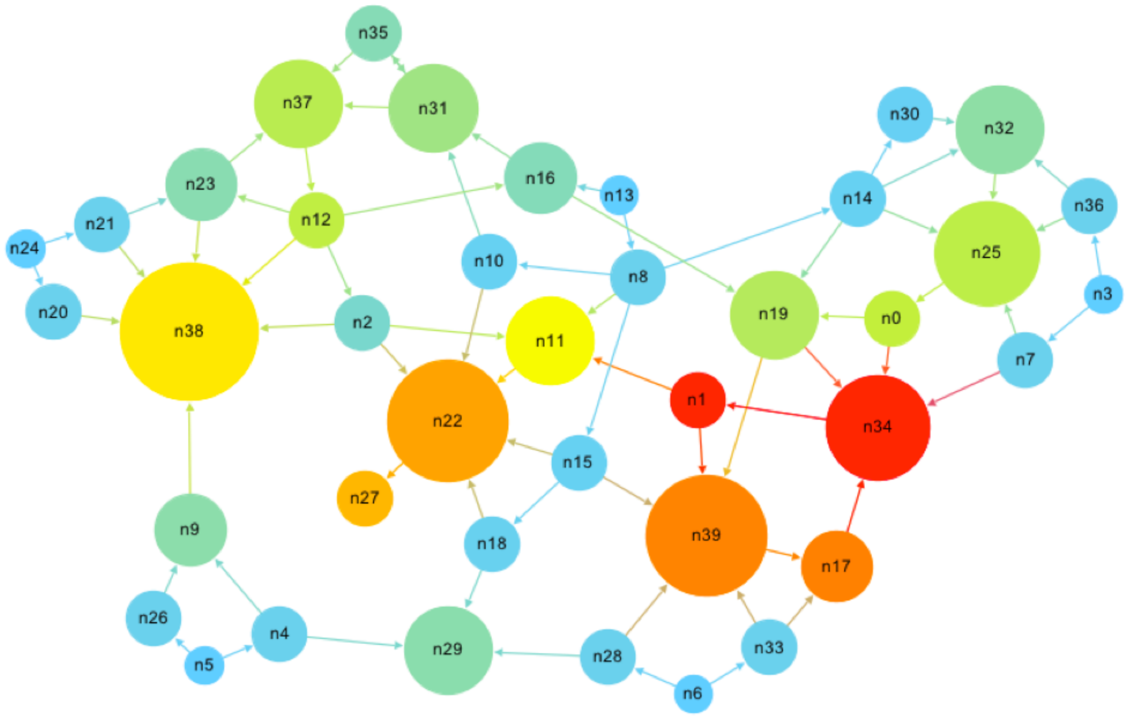
Garbage Collection

- Vertex = object, edge = pointer/reference.
- Which objects are reachable from a root?



WWW

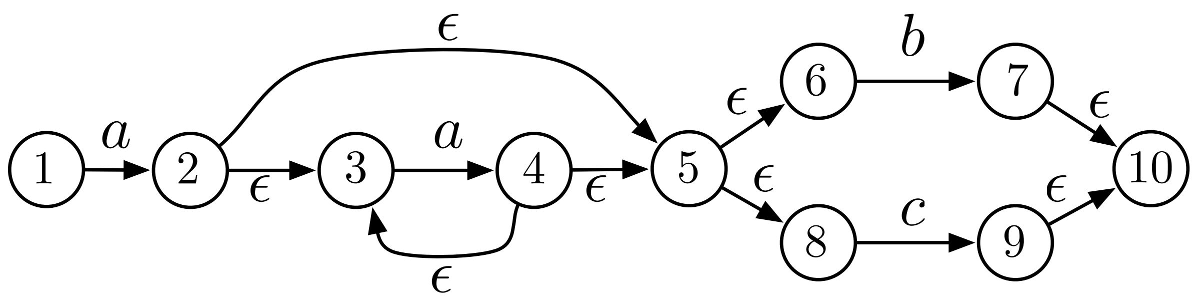
- Vertex = homepage, edge = hyperlink.
- Web Crawling
- PageRank



http://computationalculture.net/article/what_is_in_pagerank

Automata and Regular Expressions

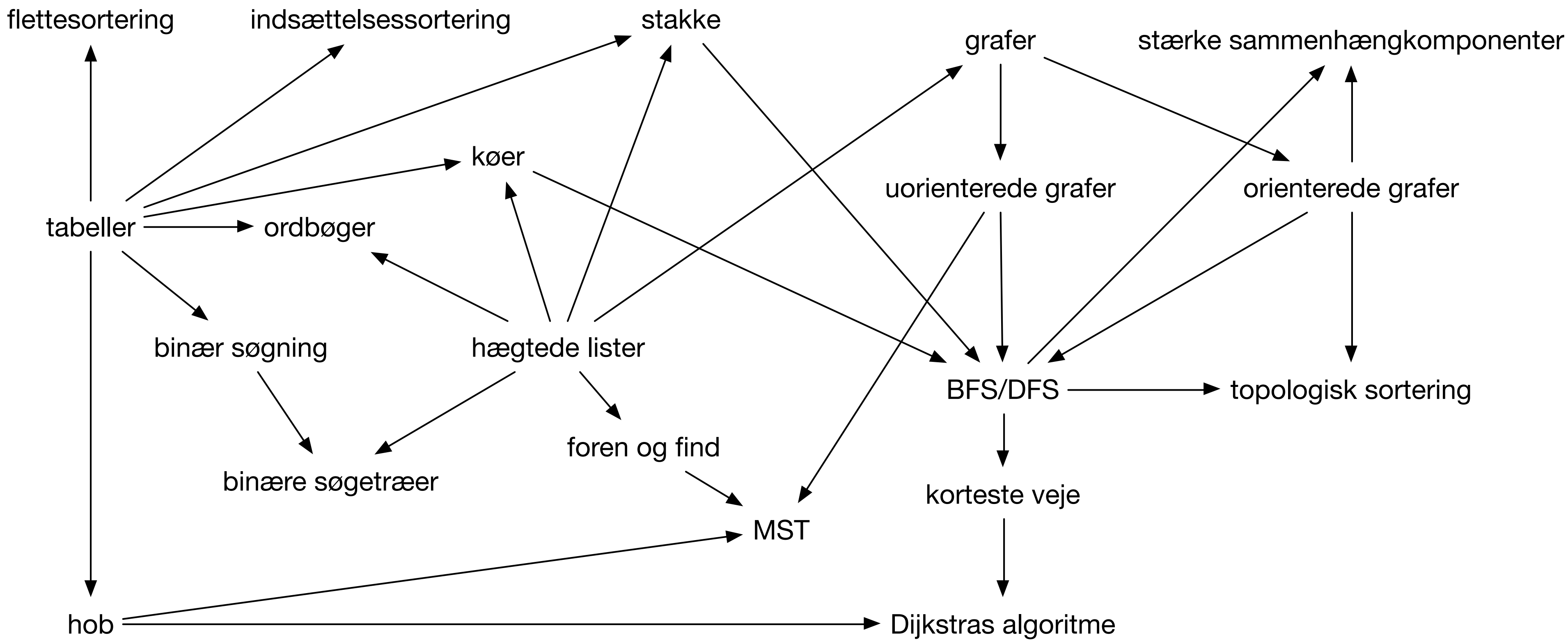
- Vertex = state, edge = state transition.
- Does the automaton accept “aab” = is there a path from 1 to 10 that matches "aab"?
- Regular expressions can be represented as automata.



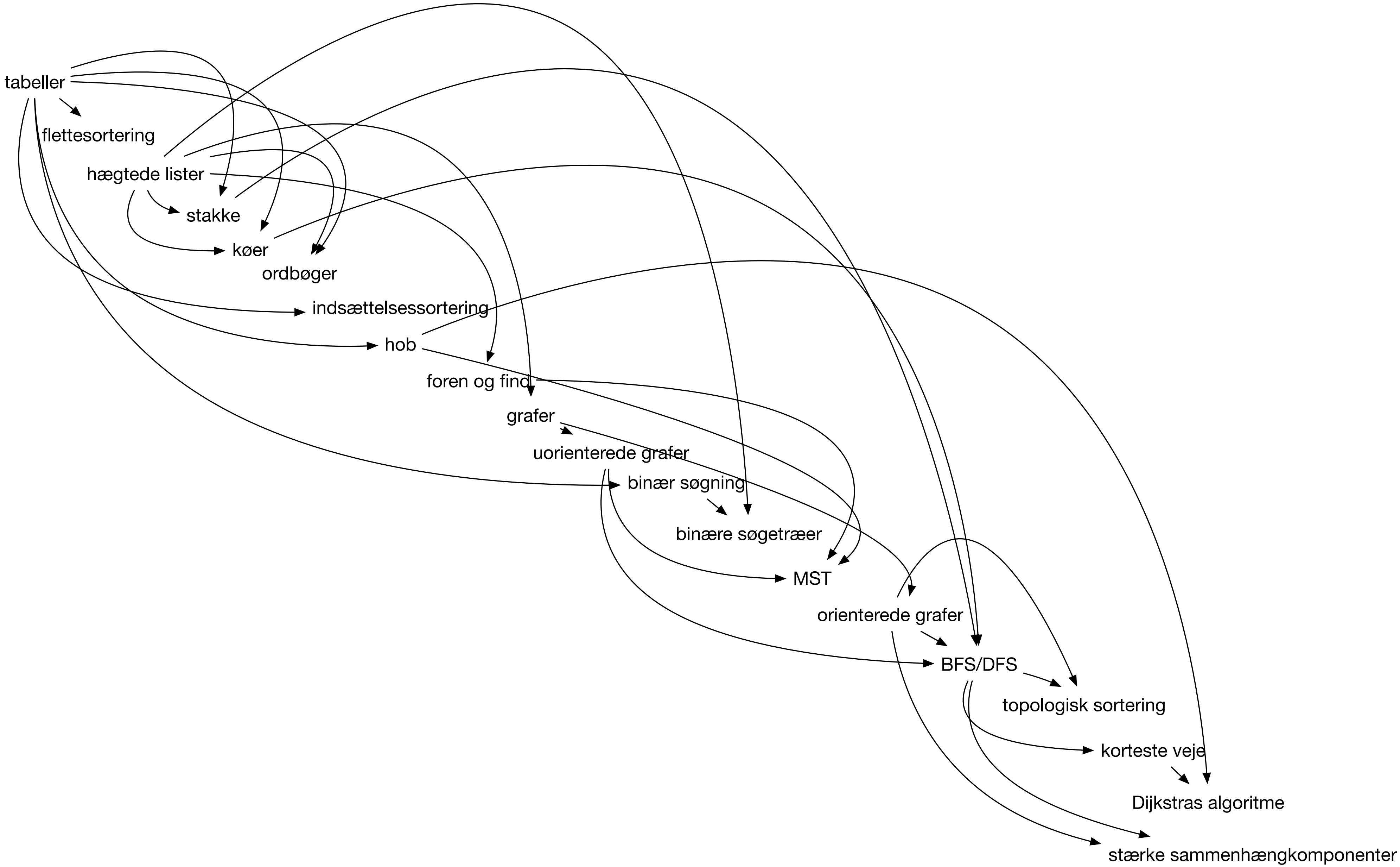
R = a·(a*)·(b|c)

Dependencies

- Vertices = topics, edge = dependency.
- Are there any cyclic dependencies? Can we find an ordering of vertices that avoids cyclic dependencies?



Dependencies

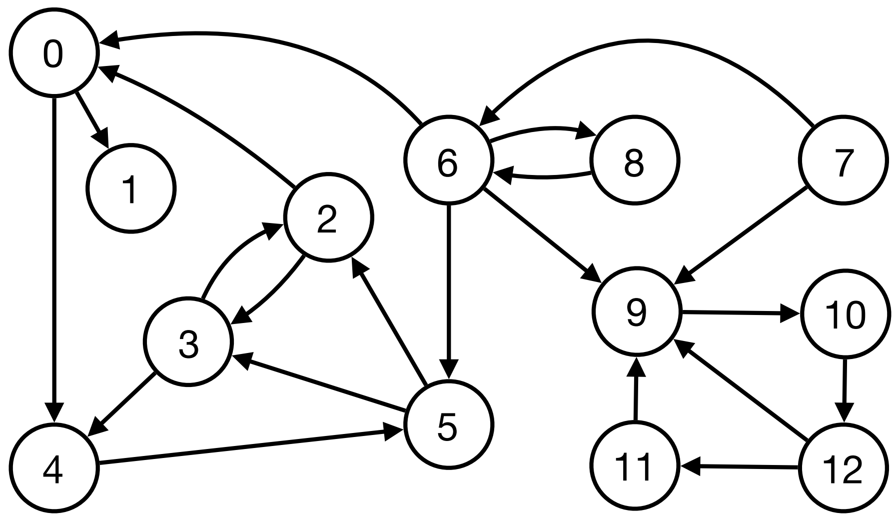


Applications

Graph	Vertices	Edges
internet	homepage	hyperlink
transport	intersection	one-way road
scheduling	job	precedence relation
disease outbreak	person	infects relation
citation	paper	citation
object graph	objects	pointers/references
object hierarchy	class	inheritance
control-flow	code	jump

Directed Graphs

- Lemma. $\sum_{v \in V} \deg^{-}(v) = \sum_{v \in V} \deg^{+}(v) = m$.
- Bevis. Every edge has exactly one start and end vertex.



Algorithmic Problems on Directed Graphs

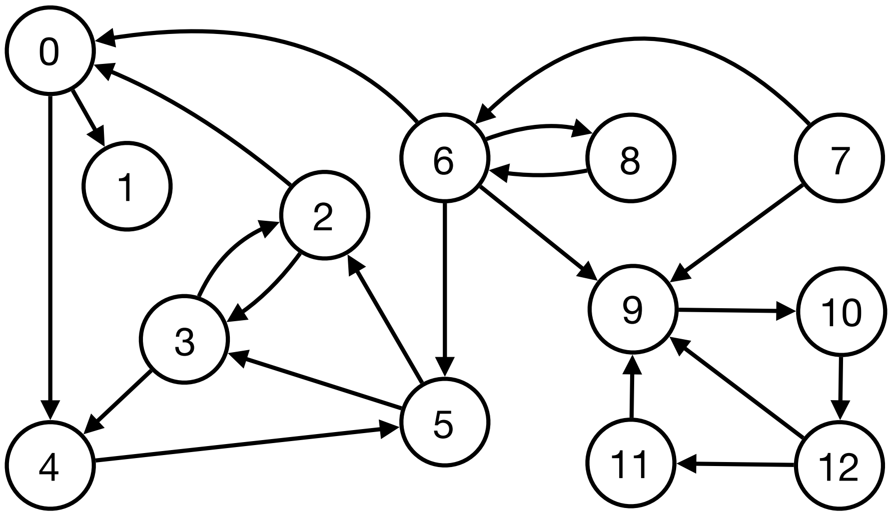
- Path. Is there a path from s to t?
- Shortest path. What is the shortest path from s to t.
- Directed acyclic graph. Is there a cycle in the graph?
- Topological sorting. Can we order the vertices such that all edges are directed in same direction?
- Strongly connected component. Is there a path between all pairs of vertices?
- Transitive closure. For which vertices is there a path from v to w?

Directed Graphs

- Directed Graphs
- Representation
- Search
- Topological Sorting
- Directed Acyclic Graphs
- Strongly Connected Components
- Implicit Graphs

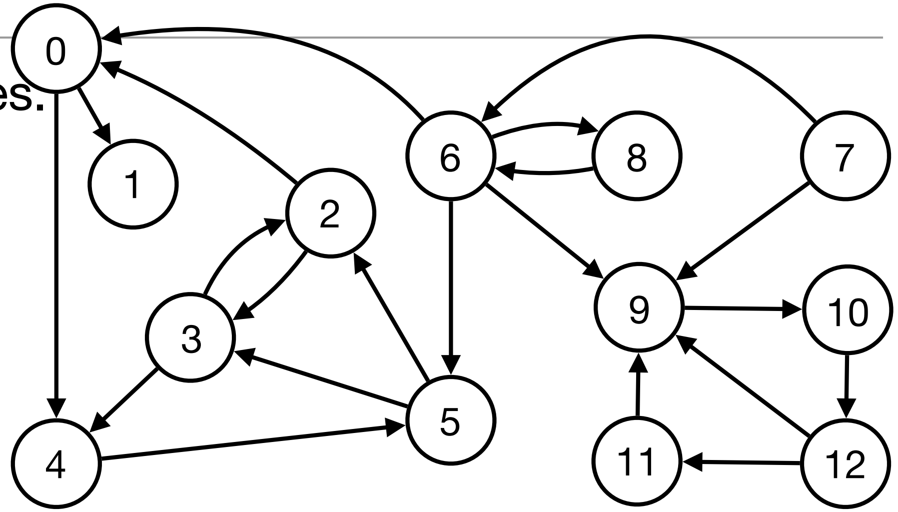
Representation

- G directed graph with n vertices and m edges.
- Representation. We need the following operations on directed graphs.
  - POINTSTO(v, u): determine if v points to u.
  - NEIGHBORS(v): return all vertices that v points to.
  - INSERT(v, u): add edge (v, u) to G (unless it is already there).



Adjacency Matrix

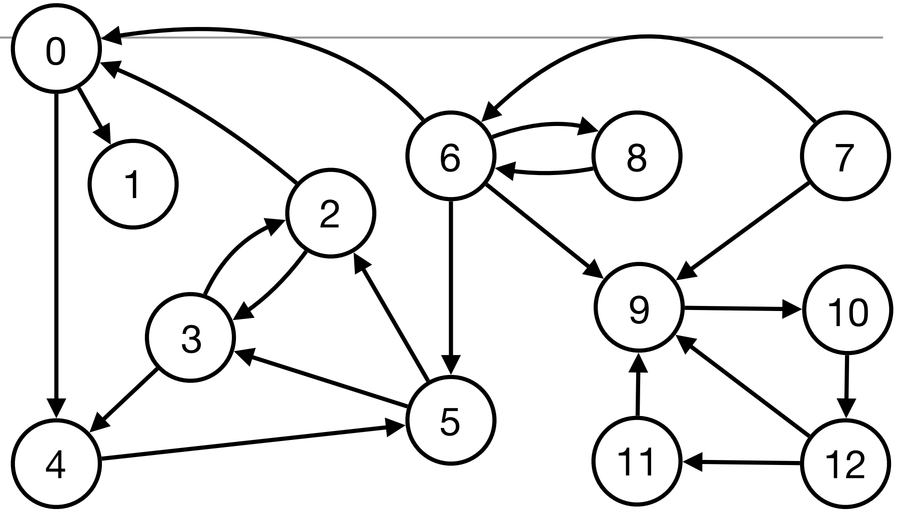
- Directed graph G with n vertices and m edges.
- Adjacency matrix.
  - 2D n × n array A.
  - A[i,j] = 1 if i points to j, 0 otherwise.
- Space. O(n^2)
- Time.
  - POINTSTO in O(1) time.
  - NEIGHBORS(v) in O(n) time.
  - INSERT(v, u) in O(1) time.

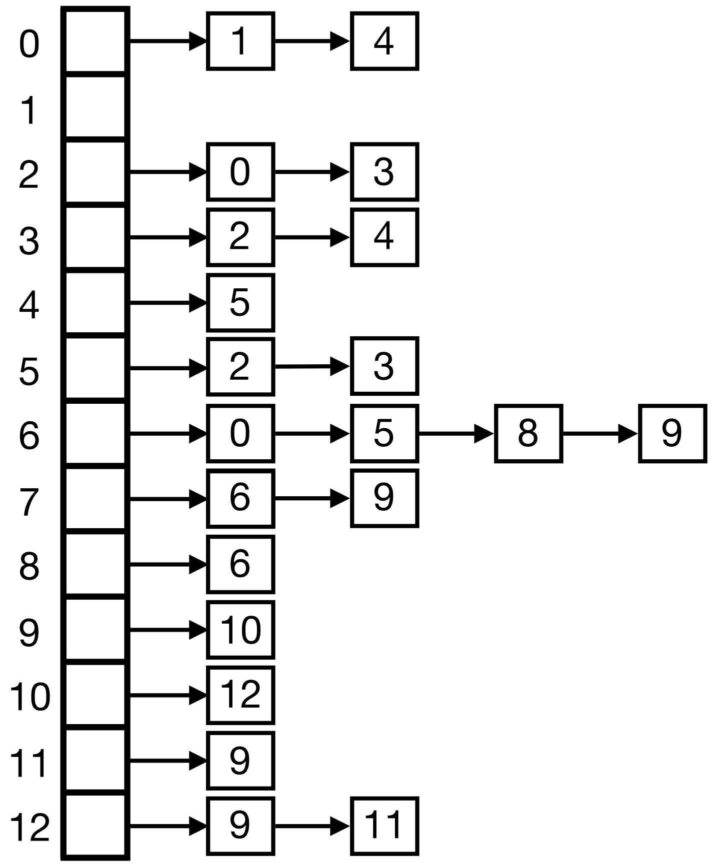


	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	1	0	0	0	0	0	0	0	0	0
3	0	0	1	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	1	1	0	0	0	0	0	0	0	0	0
6	1	0	0	0	0	1	0	0	1	1	0	0	0
7	0	0	0	0	0	0	1	0	0	1	0	0	0
8	0	0	0	0	0	0	1	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	0	0	0	1	0	0	0
12	0	0	0	0	0	0	0	0	0	1	0	1	0

Adjacency List

- Directed graph G with n vertices and m edges.
- Adjacency list.
    - Array A[0..n-1].
    - A[i] is a linked of all nodes that i points to.
- Space. $O(n + \sum_{v \in V} \deg^+(v)) = O(n + m)$
- Time.
    - POINTSTO, NEIGHBORS and INSERT in O(deg(v)) time.





Repræsentation

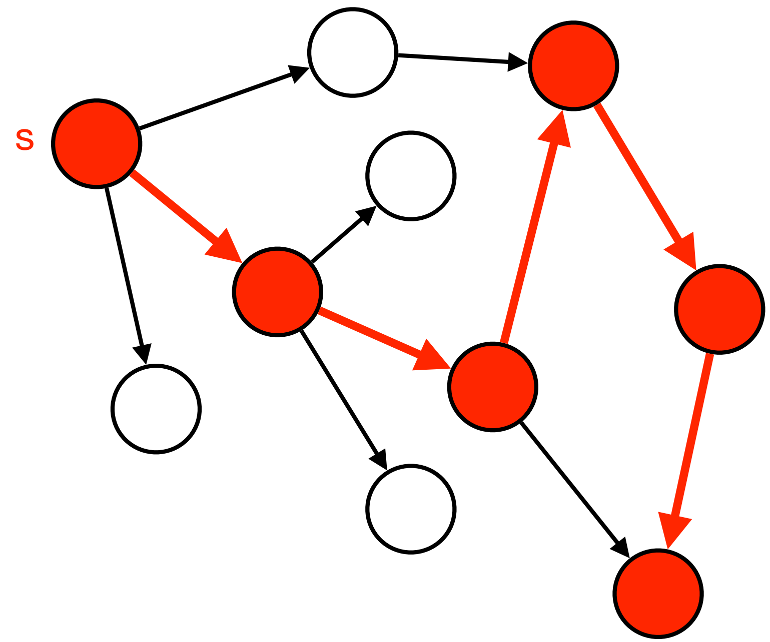
Data structure	POINTSTO	NEIGHBORS	INSERT	Space
adjacency matrix	O(1)	O(n)	O(1)	$O(n^2)$
adjacency list	$O(\deg^+(v))$	$O(\deg^+(v))$	$O(\deg^+(v))$	O(n+m)

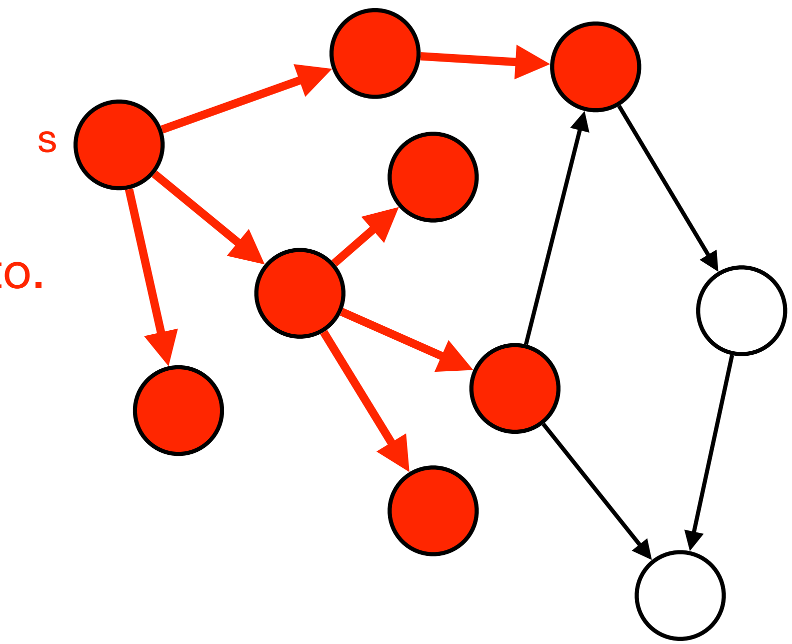
Directed Graphs

- Directed Graphs
- Representation
- Search
- Topological Sorting
- Directed Acyclic Graphs
- Strongly Connected Components
- Implicit Graphs

Søgning

- Depth first search from s.
  - Unmark all vertices and visit s.
  - Visit vertex s:
    - Mark v.
    - Visit all unmarked neighbors that v points to recursively.
- Breadth first search from s.
  - Unmark all vertices and initialize queue Q.
  - Mark s and Q.ENQUEUE(s).
  - While Q is not empty:
    - v = Q.DEQUEUE().
    - For each unmarked neighbor u that v points to.
      - Mark u.
      - Q.ENQUEUE(u).
- Time. O(n + m)



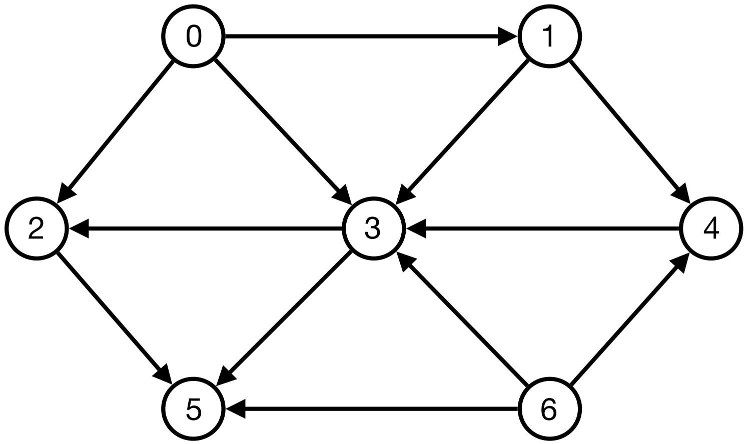


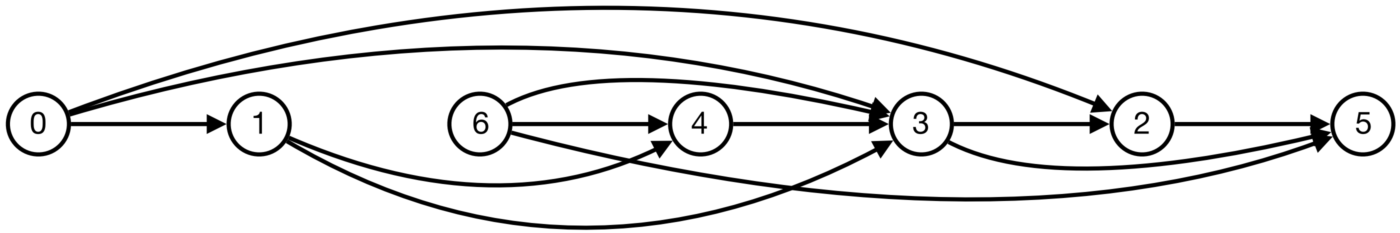
Directed Graphs

- Directed Graphs
- Representation
- Search
- Topological Sorting
- Directed Acyclic Graphs
- Strongly Connected Components
- Implicit Graphs

Topological Sorting

- Topological sorting. Ordering of vertices $v_0, v_1, \ldots, v_{n-1}$ from left to right such that all edges are directed to the right.

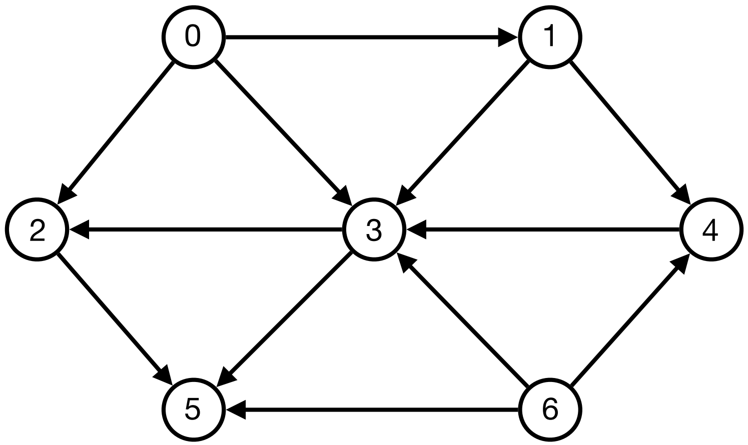


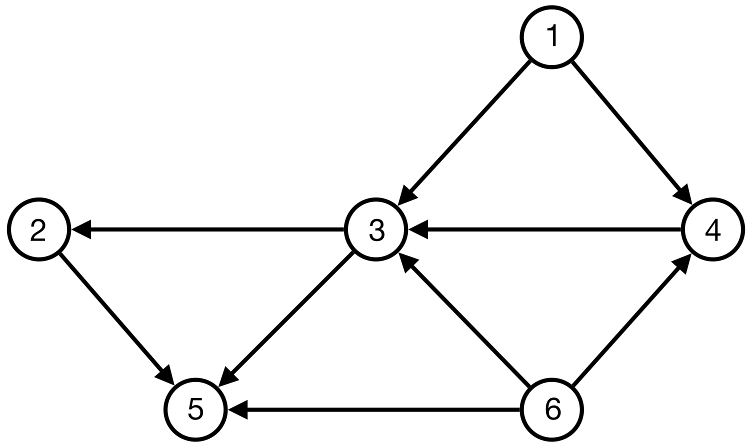


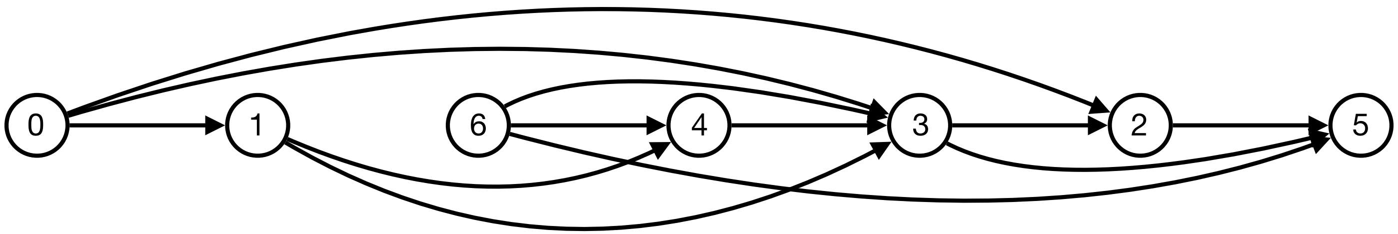
- Challenge. Compute a topological sorting or determine that none exists.

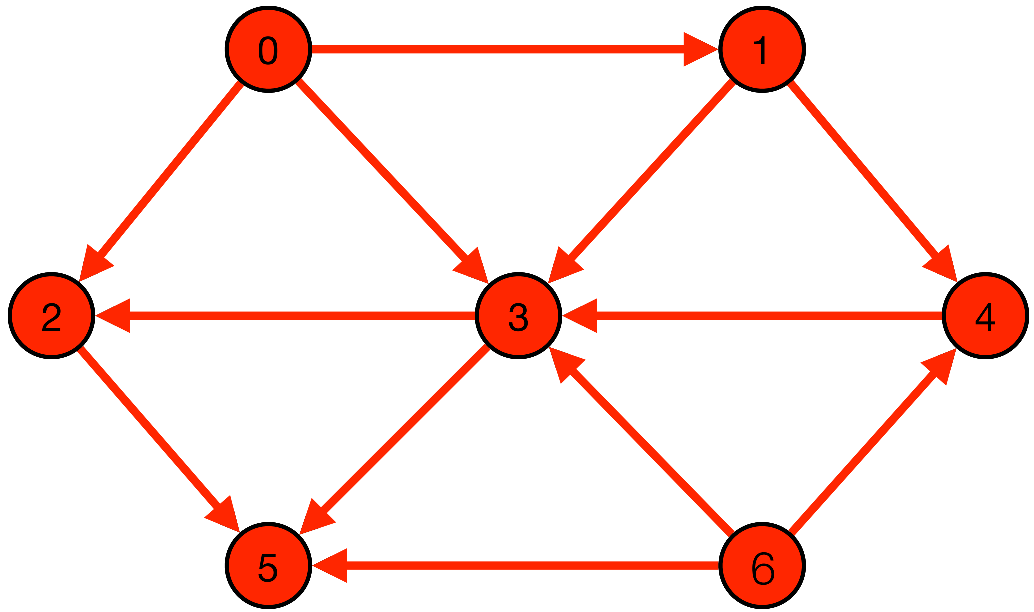
Topological Sorting

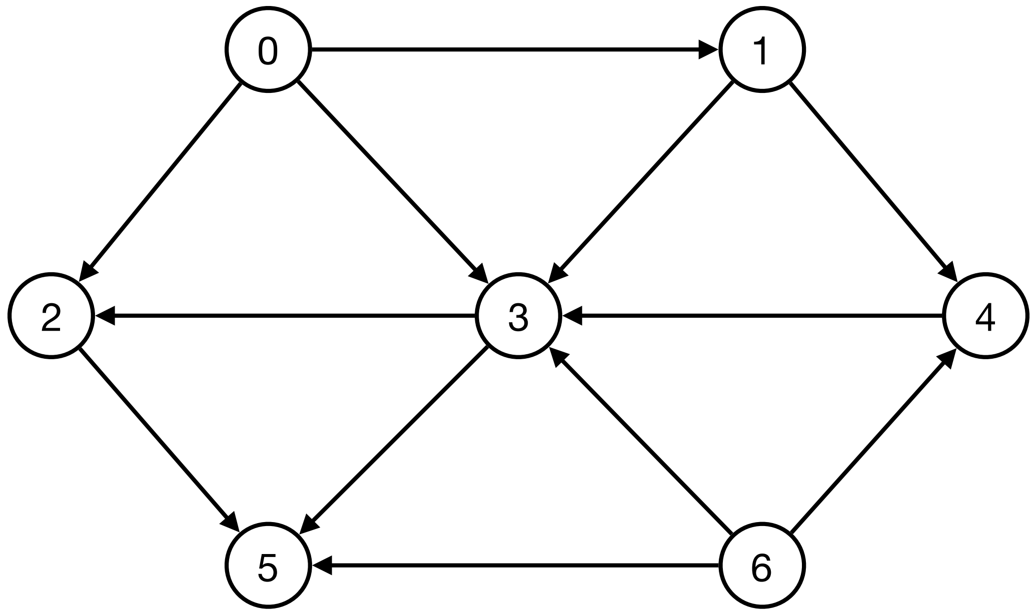
- Algorithm.
  - Find v with in-degree 0.
  - Output v and recurse on G - {v}.

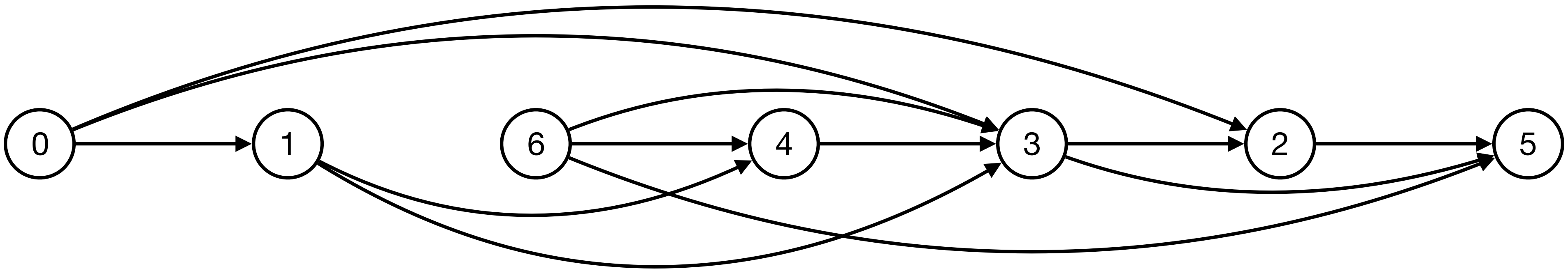






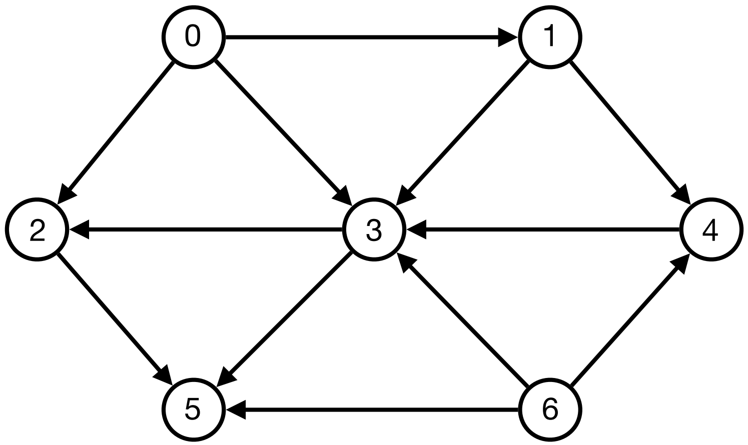

0
1
2
3
4
5
6

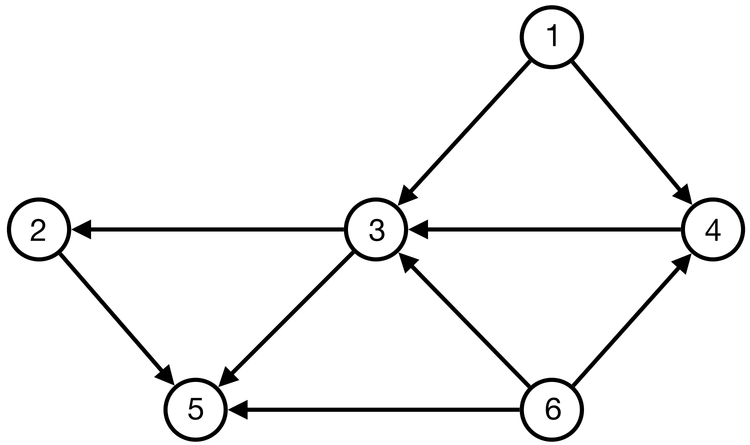

0
1
2
3
4
5
6

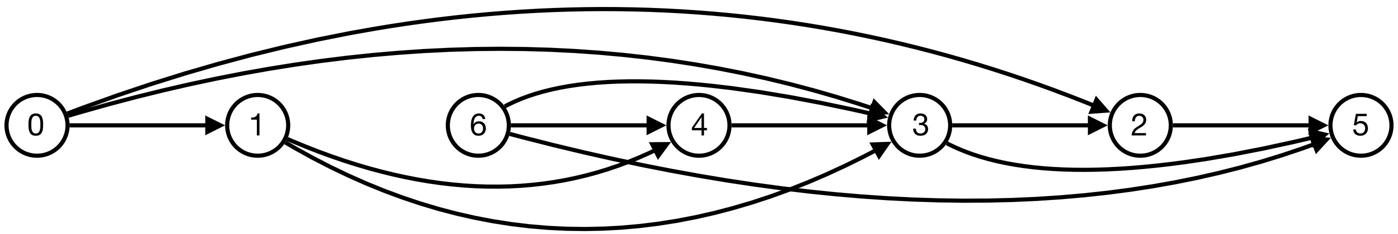

0
1
6
4
3
2
5

Topological Sorting

- Correctness?
- Lemma. G has topological sorting $\Longleftrightarrow$ G has vertex v with in-degree 0 and G - {v} has topological sorting.

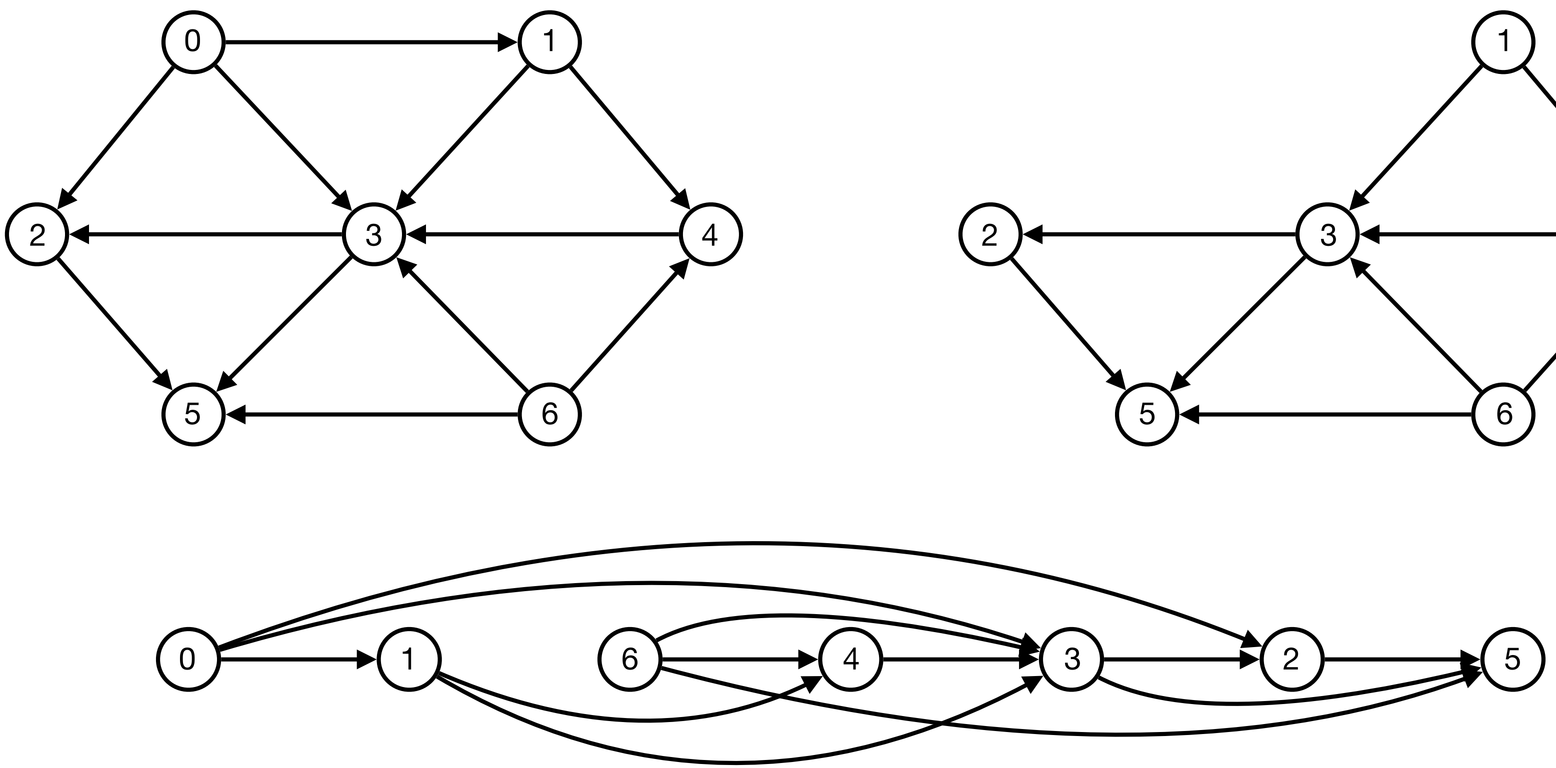






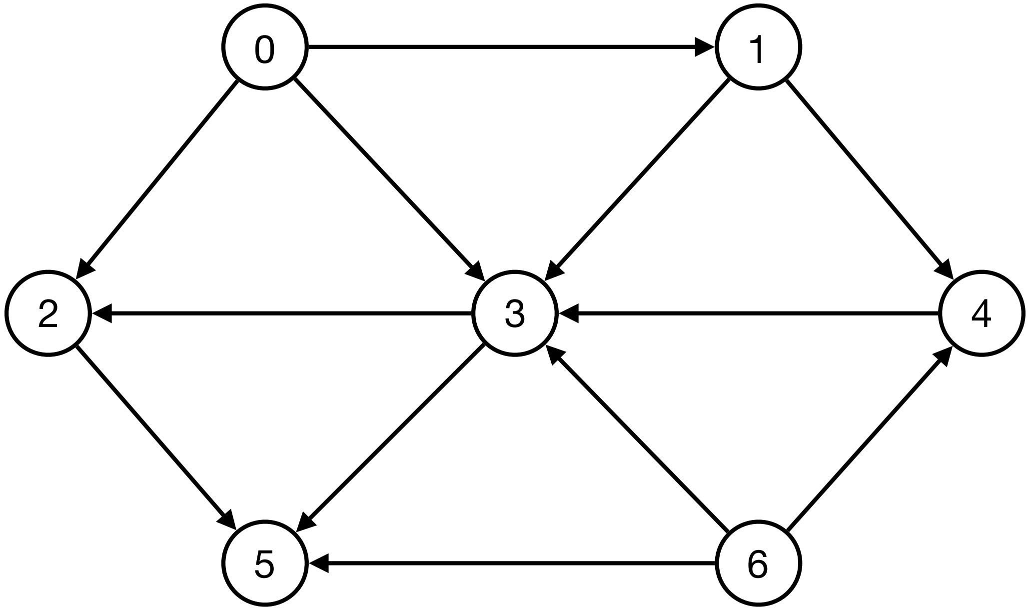
Topological Sorting

- Challenge. How do we implement algorithm efficiently on adjacency list representation?



Topological Sorting

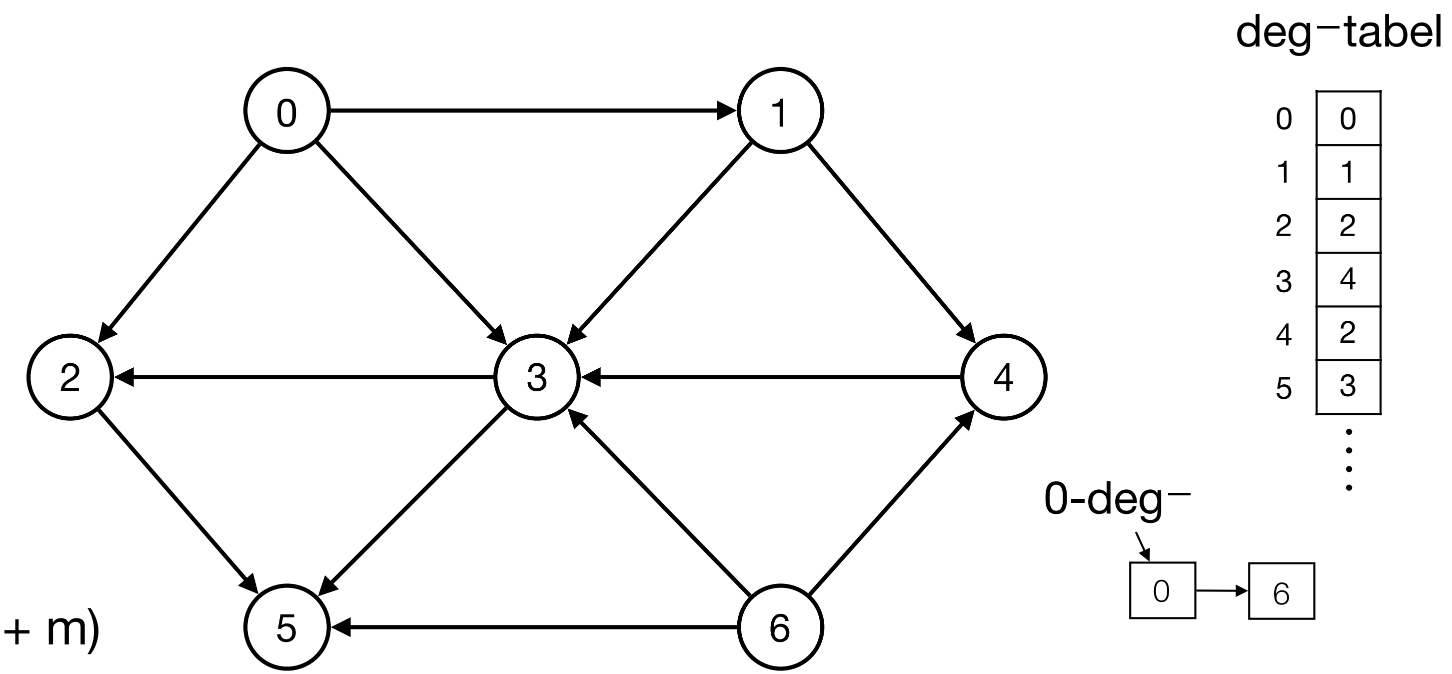
- Solution 1. Construct reverse graph G^R.
  - Search in adjacency list representation of G^R to find vertex v with in-degree 0.
  - Remove v and edges out of v.
  - Put v leftmost.
  - Repeat.



- Time per vertex.
  - Find vertex v with in-degree 0: O(n).
  - Remove edges out of v: $O(\deg^+(v))$
- Total time. $O(n^2 + \sum_{v \in V} \deg^+(v)) = O(n^2 + m) = O(n^2)$.

Topological Sorting

- Solution 2. Maintain in-degree of every vertex + linked list of all vertices with in-degree 0.
  - Remove v and edges out of v.
  - Put v leftmost.
  - Repeat



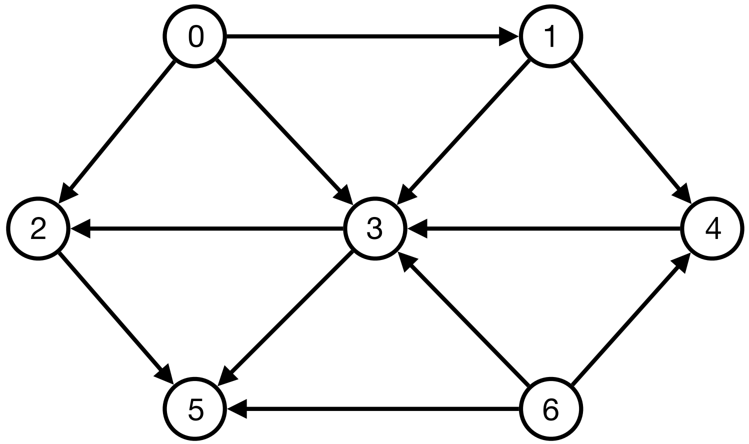
- Initialization. O(n + m)
- Time per vertex.
  - Remove vertex v with in-degree 0: O(1).
  - Remove edges out of v: $O(\deg^+(v))$
- Total time. $O(n + \sum_{v \in V} \deg^+(v)) = O(n + m) = O(n + m)$.

Directed Graphs

- Directed Graphs
- Representation
- Search
- Topological Sorting
- Directed Acyclic Graphs
- Strongly Connected Components
- Implicit Graphs

Directed Acyclic Graphs

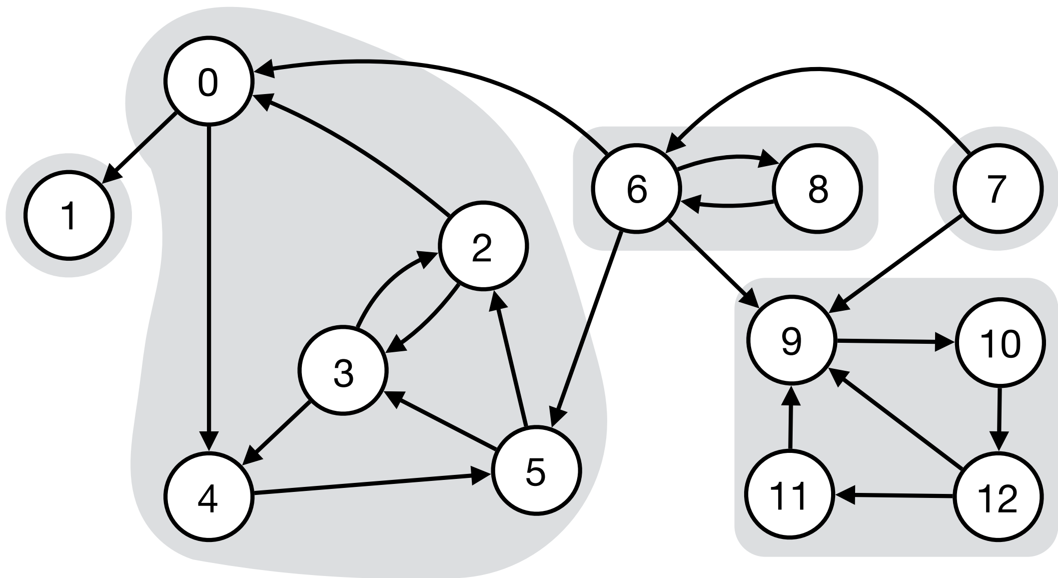
- Directed acyclic graph (DAG). G is a DAG if it contains no (directed) cycles.



- Challenge. Determine whether or not G is a DAG.
- Equivalence of DAGs and topological sorting. G is a DAG $\iff$ G has a topological sorting (see exercises).
- Algorithm.
  - Compute a topological sorting.
  - If success output yes, otherwise no.
- Time. O(n+ m)

Strongly Connected Components

- Def. v and u are strongly connected if there is a path from v to u and u to v.
- Def. A strongly connected component is a maximal subset of strongly connected vertices.



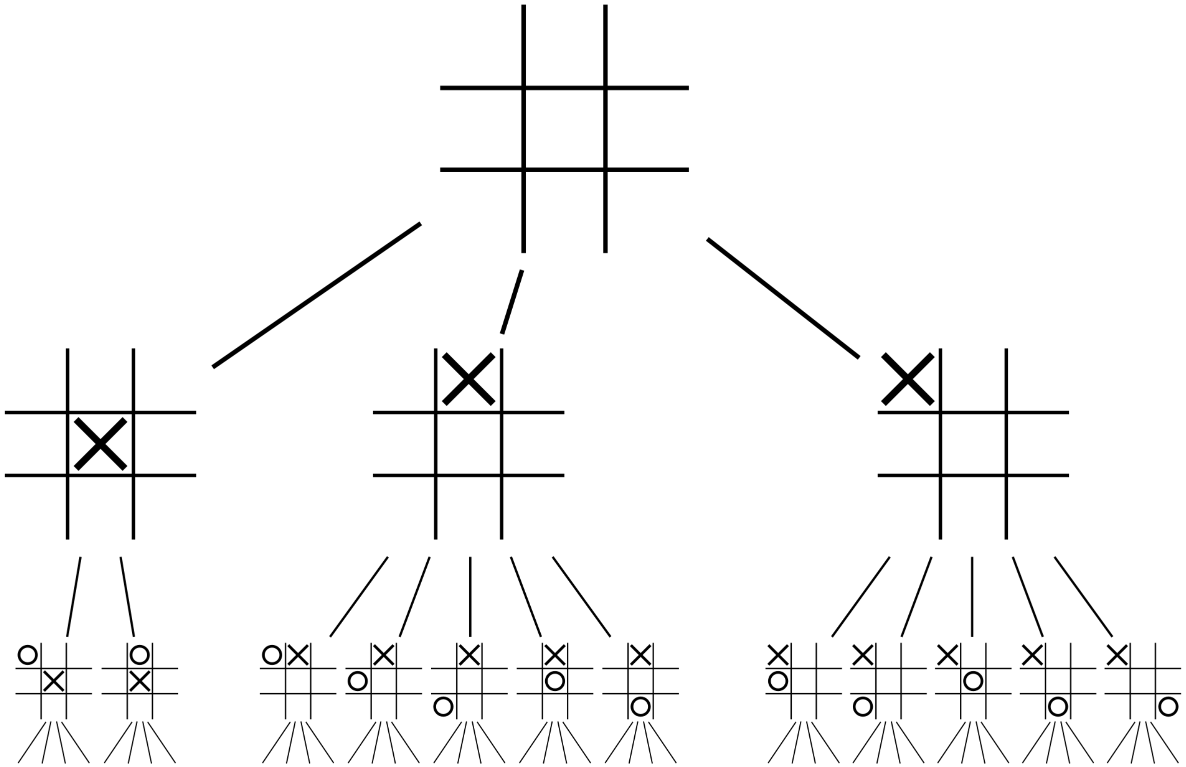
- Theorem. We can compute the strongly connected components in a graph in O(n + m) time.
- See CLRS 22.5.

Directed Graphs

- Directed Graphs
- Representation
- Search
- Topological Sorting
- Directed Acyclic Graphs
- Strongly Connected Components
- Implicit Graphs

Implicit Graphs

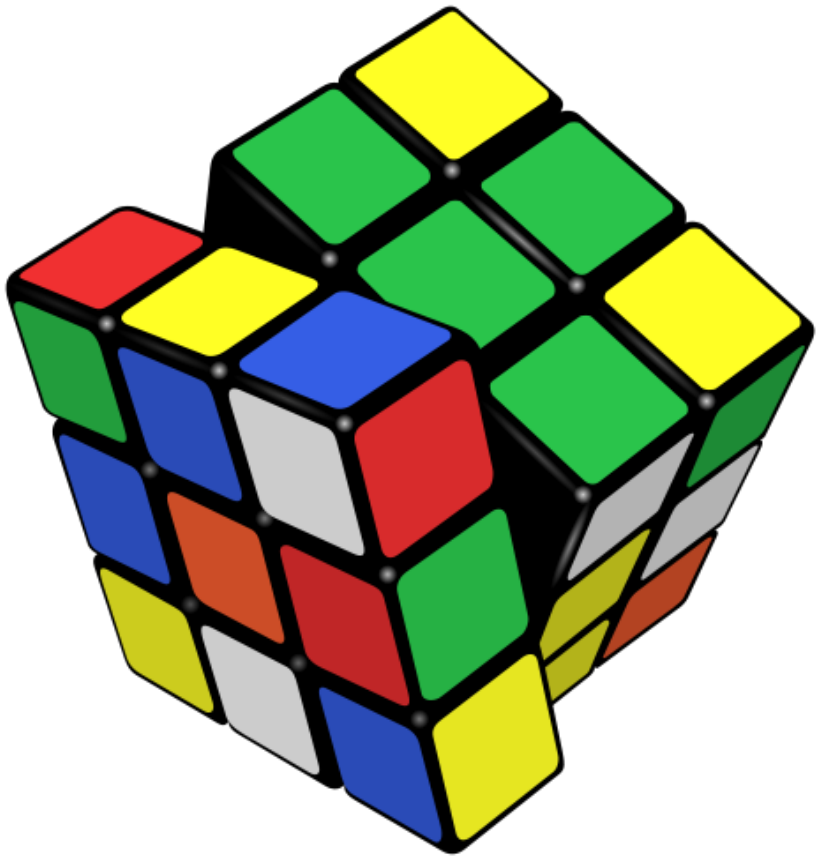
- Implicit graph. Undirected/directed graph with implicit representation.
- Implicit representation.
  - Start vertex s + algorithm to generate neighbors of a vertex.
- Applications. Games, AI, etc.



Implicit Graphs

- Rubiks cube
  - n+m = 43.252.003.274.489.856.000 ~ 43 trillions.
- What is the smallest number of moves needed to solve a cube from any starting configuration?

year	lower bound	upper bound
1981	18	52
1990	18	42
1992	18	39
1992	18	37
1995	18	29
1995	20	29
2005	20	28
2006	20	27
2007	20	26
2008	20	25
2008	20	23
2008	20	22
2010	20	20



Directed Graphs

- Directed Graphs
- Representation
- Search
- Topological Sorting
- Directed Acyclic Graphs
- Strongly Connected Components
- Implicit Graphs