Design and Analysis of Algorithms

Lecture 05 – Graph Related Concepts

Graphs

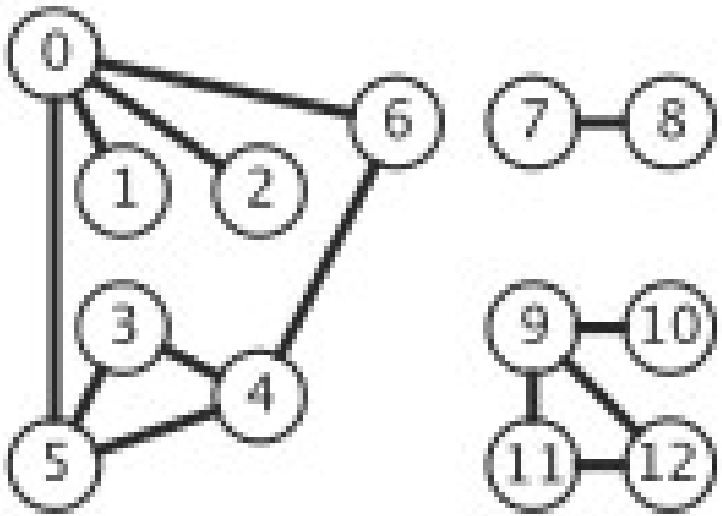
A graph is a set of vertices and a collection of edges that each connect a pair of vertices.

We prefer to use the vertexes instead of edges to define a graph.

The reason for that is that we might prefer to define edges to represent any kind of relationship.

Example. Edges could represent trade between vertices (nations). There are zero, one or two edges between each pair of vertices. Then defining the graph in the non-existence of edges and multiplicity of edges would be problematic.

We index the vertices from 0 to v-1 in a v-vertex graph.



Graphs

Some definitions.

When an edge connects two vertices, we say that the vertices are adjacent to one another and that the edge is incident on both vertices.

Exercise. Which vertices are adjacent to Vertices 3,4, and 5 in the graph?

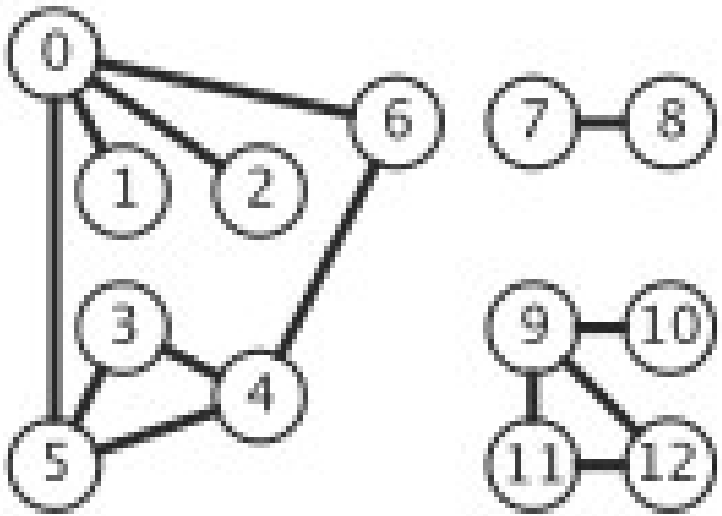
A self-loop is an edge that connects a vertex to itself.

Two edges are parallel if they connect the same pair of vertices.

The degree of a vertex is the number of edges incident on it.

Exercise. Which vertex has the highest degree o the graph displayed?

Discussion. Can a vertex have a degree of zero? What is the meaning of this?



Graphs

Some definitions.

A sub-graph is a subset of a graph's edges (and associated vertices) that constitutes a graph.

Example. There are three sub-graphs on the graph displayed.

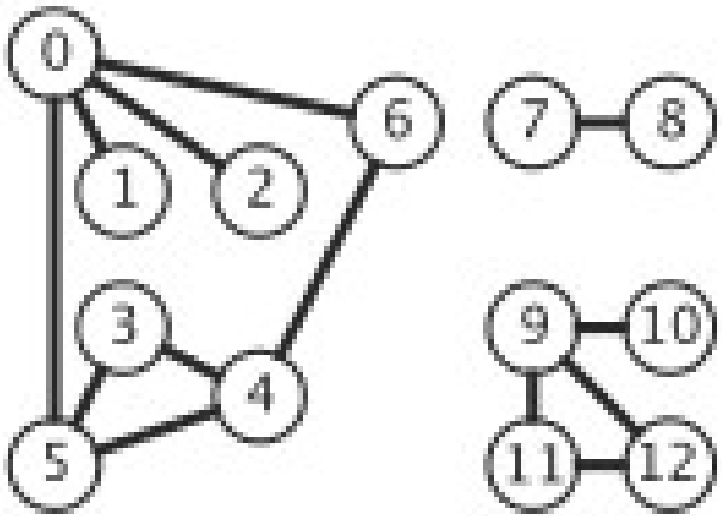
A path in a graph is a sequence of vertices connected by edges, with no repeated edges.

A simple path is a path with no repeated vertices.

We say that one vertex is connected to another if there exists a path that contains both of them.

A cycle is a path (with at least one edge) whose first and last vertices are the same.

Exercise. Identify the cycles on the sub-graph consisting of vertices 9-10-11-12.



Graphs

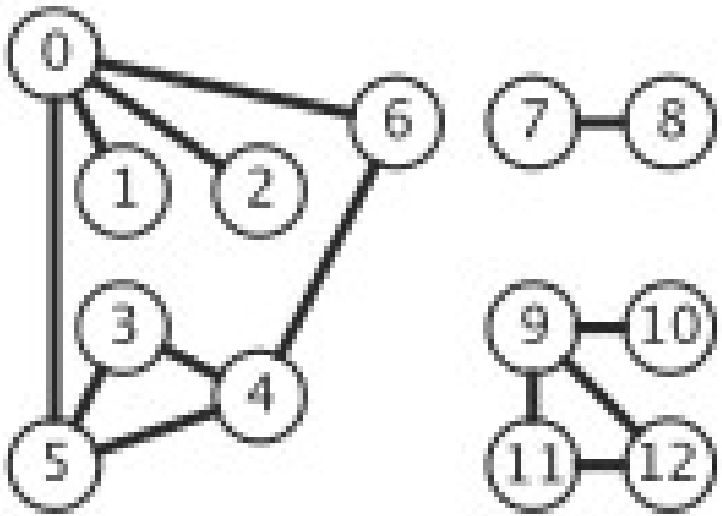
Some definitions.

A simple cycle is a cycle with no repeated vertices (other than the requisite repetition of the first and last vertices).

Exercise. For the three sub-graphs displayed, can you identify simple cycles that cover all vertices in the sub-graph?

Discussion. Can there be more than one cycle (simple or non-simple) that covers all items in a sub-graph.

The length of a path or a cycle is its number of edges.



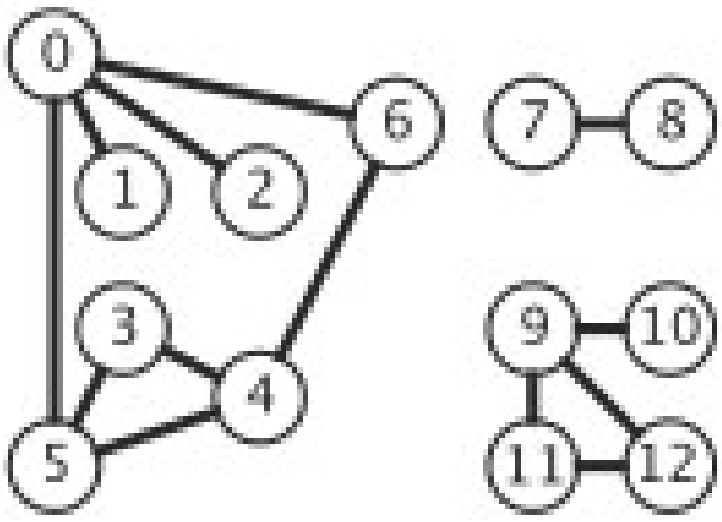
Graphs

Some definitions.

A graph is connected if there is a path (simple or non-simple) from every vertex to every other vertex.

Exercise. Is the graph displayed connected?

A graph that is not connected consists of a set of connected components, which are maximal connected sub-graphs.



Graphs

Some definitions.

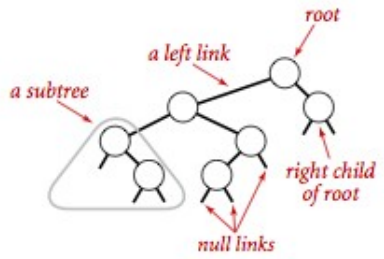
An acyclic graph is a graph with no cycles.

A tree is an acyclic connected graph.

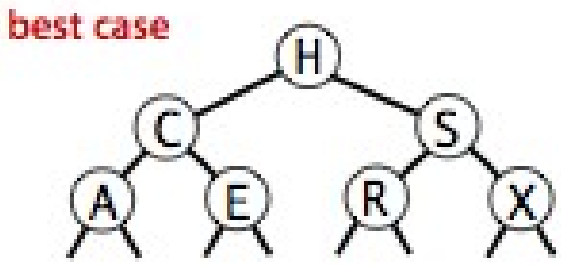
Exercise. Try to find a cycle in the tree with H at its root.

Discussion. Why is a tree always acyclic?

Discussion. Why is a tree always connected?



Anatomy of a binary tree



Graphs

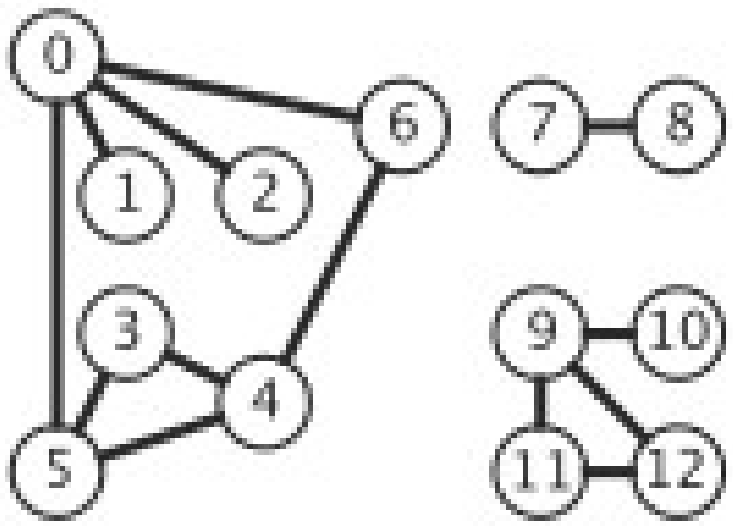
Some definitions.

A forest is a disjoint set of trees.

A spanning tree of a connected graph is a sub-graph that contains all of that graph's vertices and is a single tree.

A spanning forest of a graph is the union of the spanning trees of its connected components.

Exercise. Try to draw spanning trees to represent the three sub-graphs displayed.



Graphs

Some definitions.

A bipartite graph is a graph whose vertices we can divide into two sets such that all edges connect a vertex in one set with a vertex in the other set.

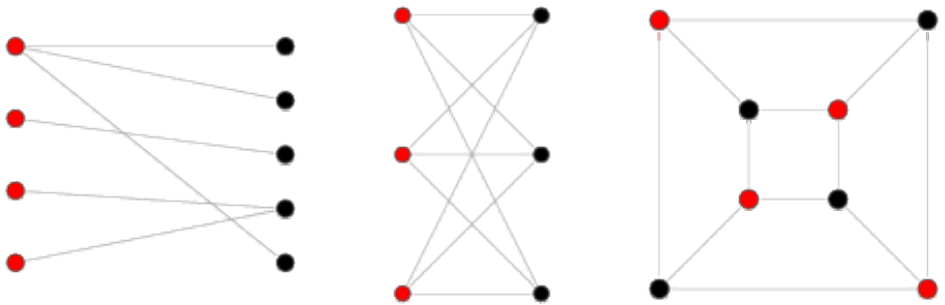
Bipartite graphs are also called bi-graphs.

Bipartite graphs are equivalent to two-colorable graphs.

All acyclic graphs are bipartite. Hence, trees are bipartite.

A cyclic graph is bipartite if and only if all its cycles are of even length.

Exercise. Search for a proof of this even hypothesis.



Graphs

Some definitions.

We usually end up with bi-partite graphs when modeling relations between two different classes of objects.

Therefore it is a common and important type of graphs.

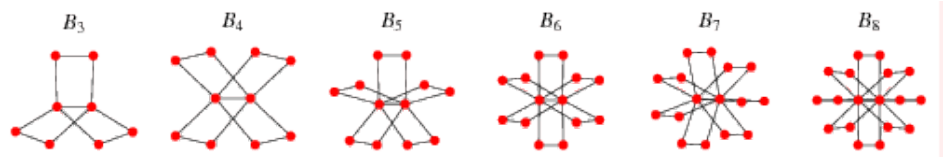
Example. Departments in a company, and employees. There is a relationship between each employee and a department they are assigned to. There could be also relationship between the same employee and multiple department because of work history or working hours being split (based on model). The other way around is also possible because a department could consist of many employees.

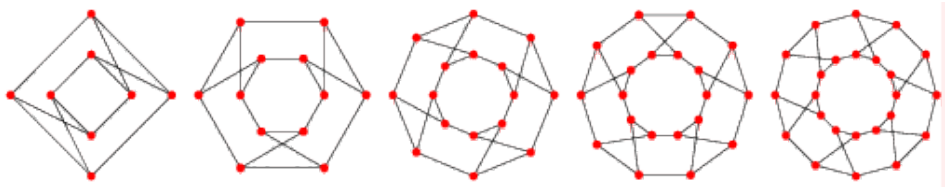
Some examples of bi-partite graphs, other than trees.

- Book graphs are graphs formed by multiple cycles sharing an edge.
- Crossed-prism graphs.

There are some typical tasks in bi-partite graphs.

- Testing for bipartedness.
- For a non-bipartite graph, determining if there exists k edges, which if removed would end up with a bi-partite graph.

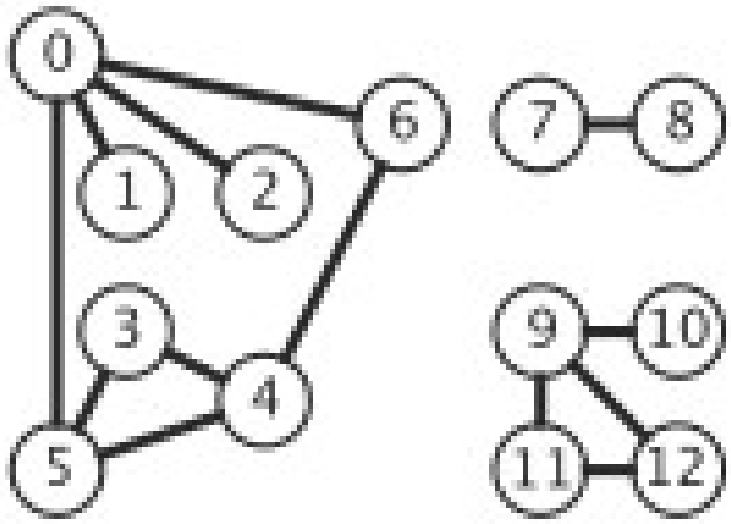




Graphs

To implement graphs as a data structure useful for graph algorithms.

- We need to be able to add vertices, individually or in bulk.
- We need to be able to add an edge, specifying the two vertices.
- It is easier for many tasks to know the number of edges and vertices.
- We need to be able to get all adjacent vertices to a given vertex (or be able to iterate over them).



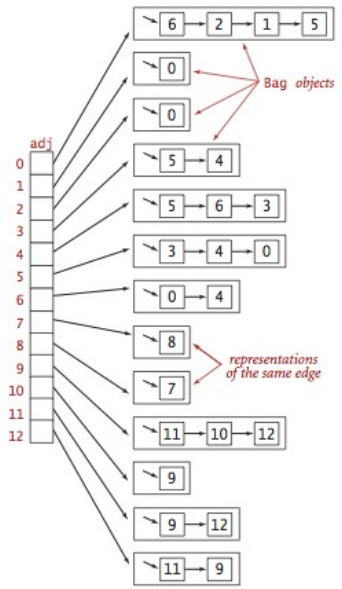
Graphs

How to implement such a data structure?

The simplest way is to use a two dimensional data structure.

- A list of bags (sets) would suffice.
- Adding a vertex – Adding to the list.
- Adding an edge – Adding the second vertex to the set associated with the first vertex.
- Number of vertices – Size of list.
- Number of edges – Sum of sizes of sets.
- Adjacent vertices – Get the associated set of vertices or an iterator to the set.

Note that in Java and similar languages, the vertices will be independent objects and the list and all sets will be composed of references to the vertices.



Adjacency-lists representation (undirected graph)

Graphs

How to iterate over all edges in a graph?

Depth-first search (DFS) is a classic recursive method for systematically examining each of the vertices and edges in a graph.

For each vertex.

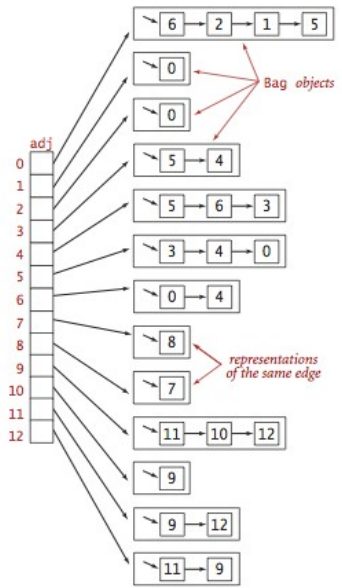
If it has already been visited, pass.

If not

Mark it as having been visited.

Visit (recursively) all the vertices that are adjacent to it and that have not yet been marked.

Exercise. Starting from vertex 0, please execute the algorithm.



Adjacency-lists representation (undirected graph)

Graphs

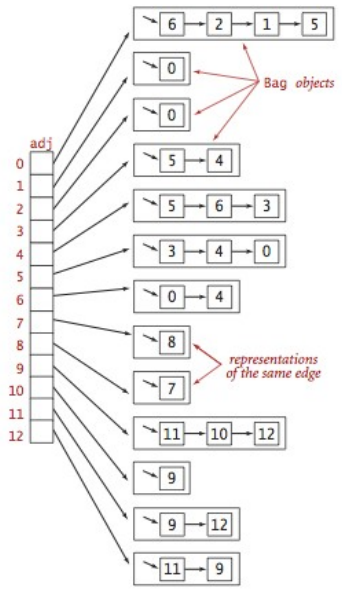
How to find a path between two vertices.

Modify depth-first search.

Mark all edges as you search.

Back-track your steps when you reach the second vertex.

Discussion. A path that you find this way is not necessarily the shortest path. Why?

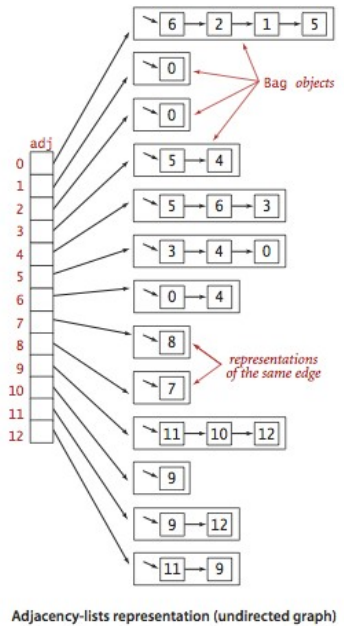


Adjacency-lists representation (undirected graph)

Graphs

Modifying DFS enables us to

- Detect cycles.
- Test bipartedness.
- Detect bridges that connect two sub-graphs.
- Detect planarity (ie. no edge crosses any other edge).



Adjacency-lists representation (undirected graph)

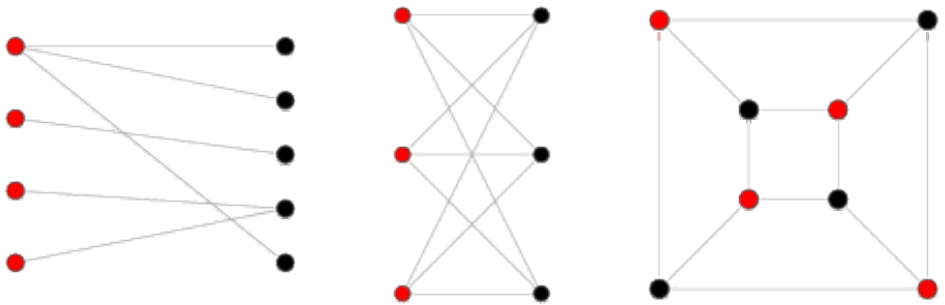
Graphs

Symbol Graphs

When we can name the vertices and edges, instead of giving them indexes, we refer to the graph as a symbol graph.

The idea is that the names are representing some symbolic relationship.

Example. Actors who have received awards. There would be a bipartite graph, with the first type of vertices named as actors, and the second type of vertices named as the types of awards (ie. Oscar, Sundance, Golden Bear, Golden Orange, etc).



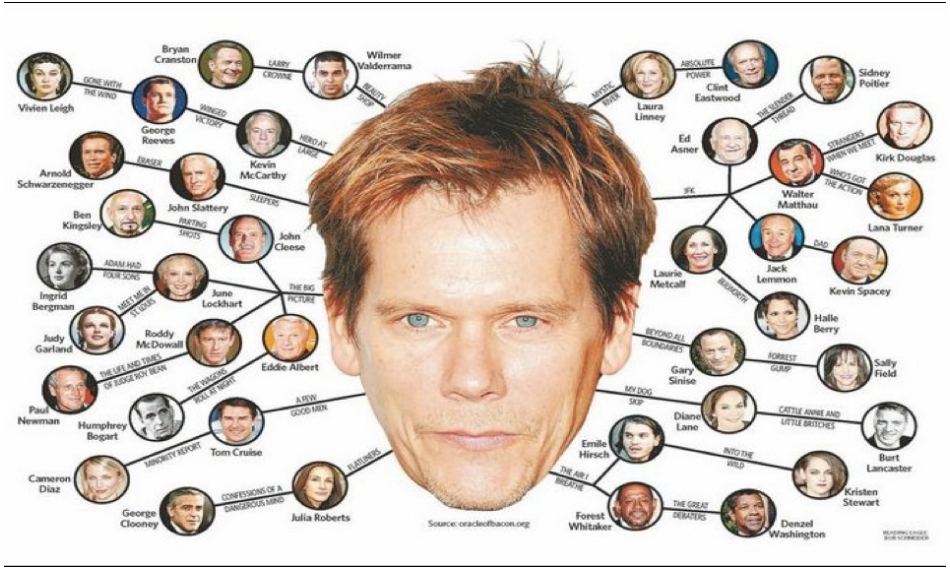
Graphs

Symbol Graphs

Example. It is assumed that anyone involved in the Hollywood film industry can be linked through their film roles to Kevin Bacon within six steps.

This has originated from an interview in the 90s, where Kevin Bacon claimed to have worked with anyone or those he has worked with have worked with anyone.

Question. Could you use IMDB data to prove/disprove this?



Graphs

Symbol Graphs

Bacon number is your degree of separation from Kevin bacon.

Same kind of connectedness is discussed in Mathematics community as Erdös number, being the degree of separation to famous mathematician Paul Erdös.

The Erdös-Bacon number is the sum of these two numbers.

- A few select scientists who had parts in documentaries have finite Erdös-Bacon numbers.
- A few actors who had pursued science prior to acting have co-authored papers so that they have Erdös-Bacon numbers. **Example.** Mayim Bialik has a Phd in neuroscience.
- The only non-scientist/non-actor person with Erdös-Bacon number is Elon Musk.

Graphs

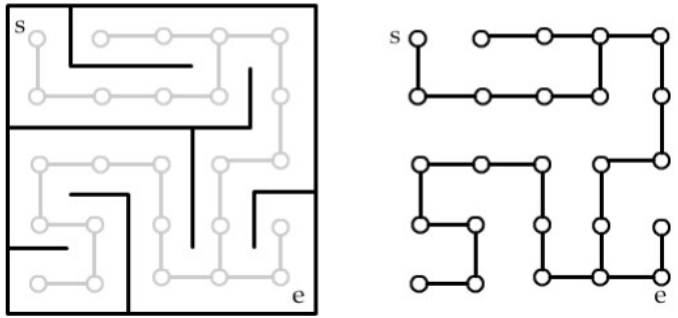
Mazes

Mazes are easily expressed as graphs.

Intersections are vertices and the paths between intersections are edges.

Question. If a maze offers only two paths at each intersection, would the representing graph be a tree? How about three paths?

Question. Is it easier to solve a maze when it is expressed as a graph?

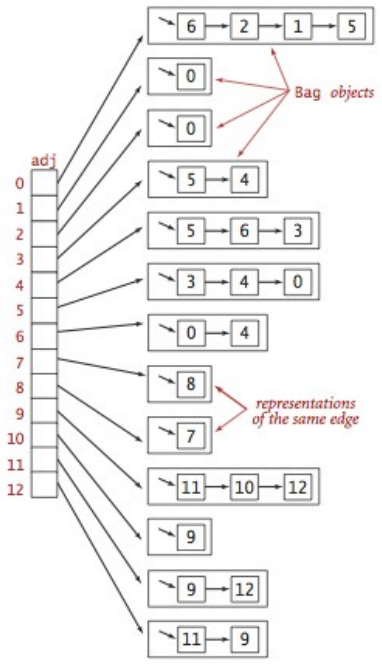


Graphs

DFS revisited.

Can we solve the same problems without recursion?

What would we need to redesign DFS as non-recursive?



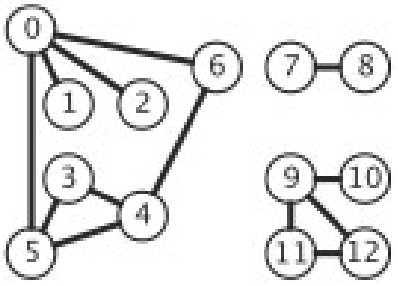
Adjacency-lists representation (undirected graph)

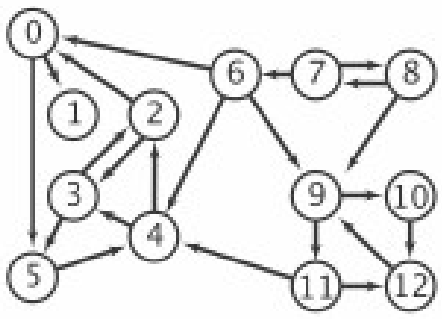
Directed Graphs

Directed Graph

A directed graph (or digraph) is a set of vertices and a collection of directed edges that each connects an ordered pair of vertices.

We say that a directed edge points from the first vertex in the pair and points to the second vertex in the pair.





Directed Graphs

Some terms

The outdegree of a vertex is the number of edges pointing from it.

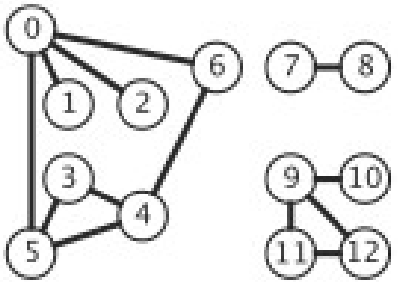
The indegree of a vertex is the number of edges pointing to it.

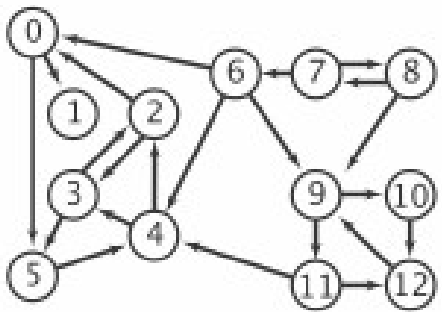
We say that a vertex w is reachable from a vertex v if there exists a directed path from v to w.

We say that two vertices v and w are strongly connected if they are mutually reachable: there is a directed path from v to w and a directed path from w to v.

A directed acyclic graph (or DAG) is a digraph with no directed cycles.

As expected DAGs have important applications.





Directed Graphs

Some terms

A directed acyclic graph (or DAG) is a digraph with no directed cycles.

As expected DAGs have important applications.

A key property of DAGs is that they have what is known as a "topological ordering", which means that the nodes of a DAG can be put into a linear sequence with the nodes given an "ordering", specifically nodes at the beginning of the sequence have a "lower value" than nodes at the end of the sequence.

Project plans, business workflows, state change diagrams, course syllabuses and similar (flowing) concepts are modeled with diagrams and/or model description markups which end up as DAGs.

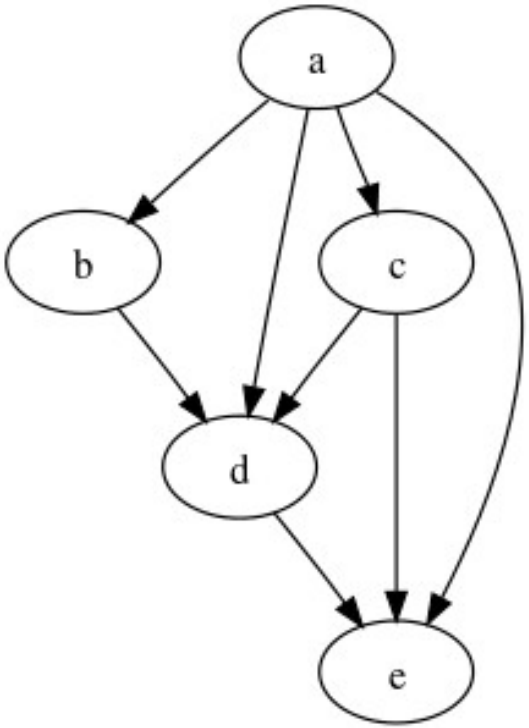
Many path optimization and scheduling algorithms make use of DAGs.

Problems on optimization of resource use such as garbage collection in the Java Virtual Machine make use of DAGs.

Blockchains make use of DAGs.

Automated meta-analysis of scientific research make use of many DAGs, each describing the relationships between parameters/concepts in a hypothesis.

Compilers make use of DAGs to optimize code.



Next Session

Next session we will be discussing minimum spanning trees and shortest path problems.