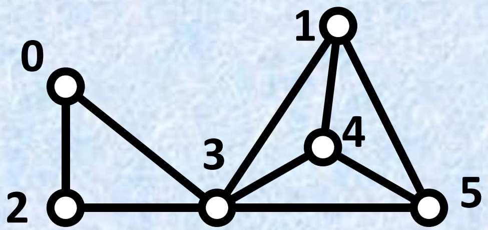
Graph



- **Nodes, Vertices**
- Servers, cities...
- Persons, people...
- Objects in comp. science
- ... etc.

- **Edges**
- Connections, roads...
- Personal relations
- Relations among objects
- ... etc.

Usual graph representations

nodes = indices		Node degrees		Lists of neighbours				
0	.....	2	.....	2	3			
1	.....	3	.....	3	4	5		
2	.....	2	.....	0	3			
3	.....	5	.....	0	2	1	4	5
4	.....	3	.....	1	3	5		
5	.....	3	.....	1	3	4		

1D/2D array, vector, ArrayList...

Less obvious, more effective

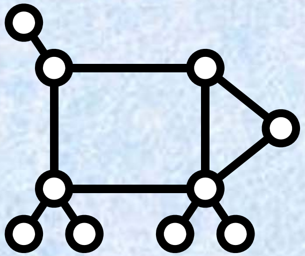
Adjacency matrix

Nodes = indices	0	1	2	3	4	5
0	0	0	1	1	0	0
1	0	0	0	1	1	1
2	1	0	0	1	0	0
3	1	1	1	0	1	1
4	0	1	0	1	0	1
5	0	1	0	1	1	0

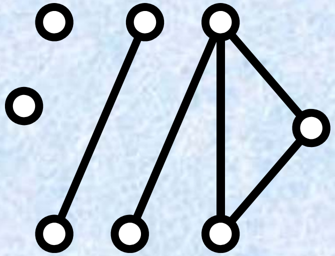
2D array, matrix

Plain, obvious, less effective

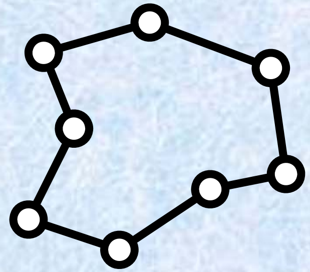
Small graph zoo



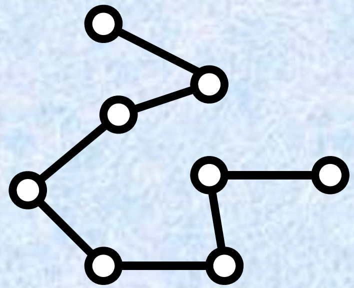
- **Connected graph**



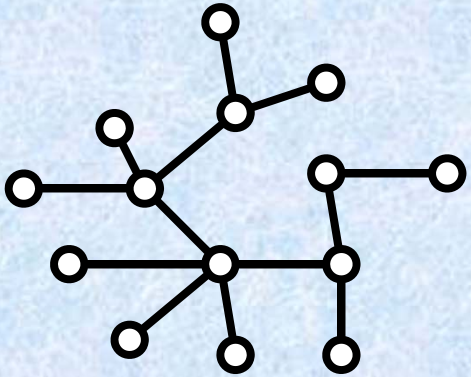
- **Disconnected**
- **graph**



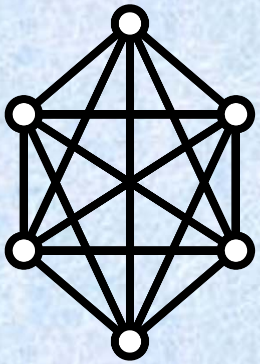
- **Cycle / circle**
- N nodes,
  N edges



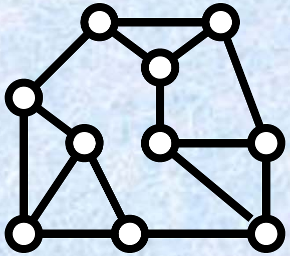
- **Path**
- N nodes,
  N—1 edges



- **Tree**
- Connected
- N nodes, N—1 edges
- is bipartite

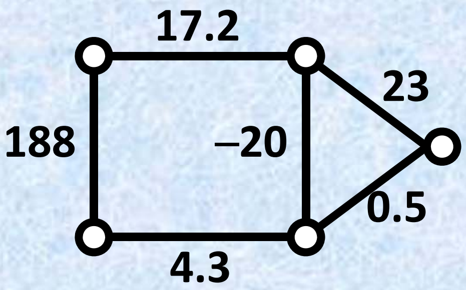


- **Complete graph**
- N nodes
- $(N^2 - N)/2$ edges

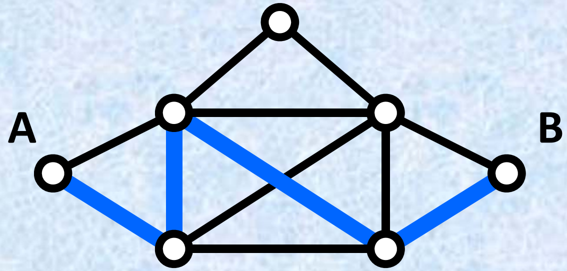


- **Regular graph**
- All node degrees
  are the same

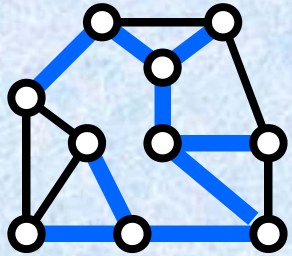
Small graph zoo



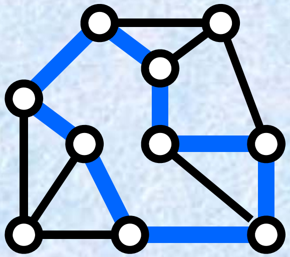
- **Weighted graph**
- Each edge has its cost (length, weight)



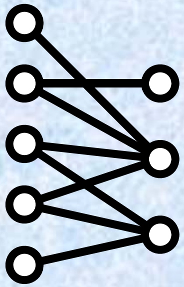
- **Path between A and B**
- Path visits each node at most once



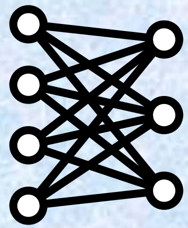
- **Spanning tree**
- subgraph which is a tree and it contains all nodes



- **Cycle in a graph**
- path which first and last node are the same

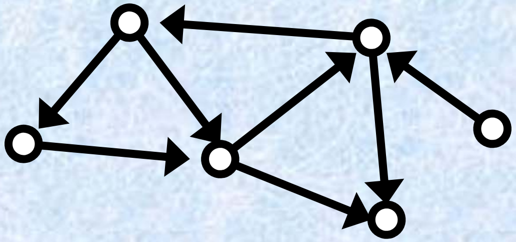


- **Bipartite graph**
- two-colorable
- cycles only of even length
- No edges inside partitions

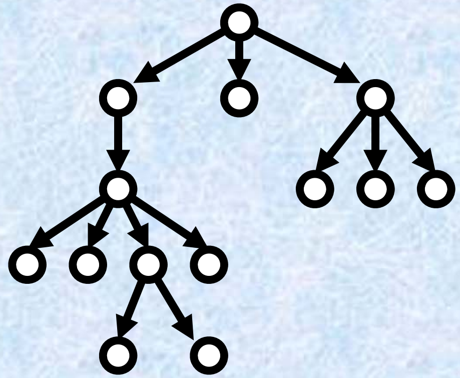


- **Complete bipartite graph**
- M and N nodes in partitions
- M x N edges

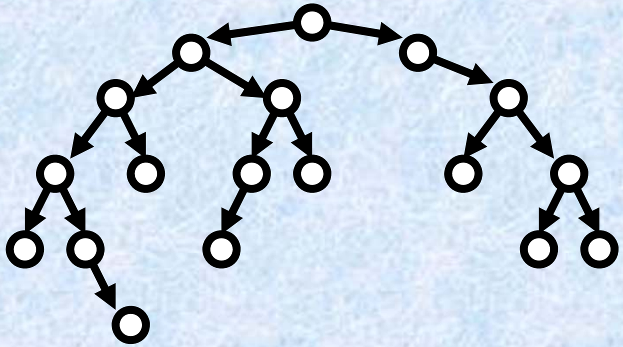
Small graph zoo



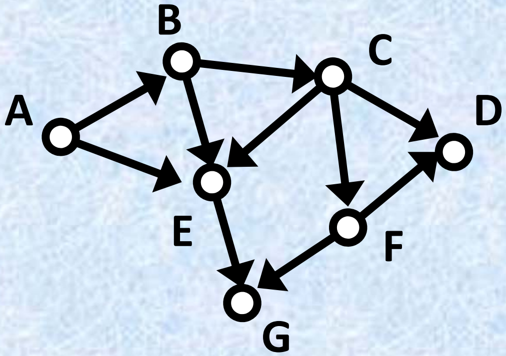
❖ **Directed graph**



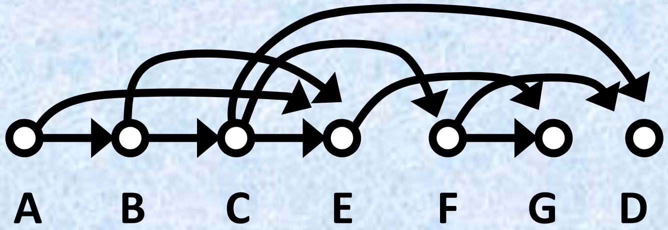
❖ **Rooted tree**



❖ **Binar rooted tree**



❖ **Directed acyclic graph (DAG)**
❖ No Directed loops



❖ **Same DAG in topological order**

Some apparently innocuous problems related to graphs.

Easy problem = a complete solution may be taught in bachelor courses.

Hard problem = a complete solution is unknow to this day.
(However, there are often satisfactory approximate solutions.
Typically, they are quite advanced)

Clay Mathematics Institute
http://www.claymath.org/millennium-problems/rules-millennium-prizes
Offers prize **1 000 000 $** for a complete solution of any of those hard questions.
The prize exists since the year 2000.
Nobody has claimed it yet....

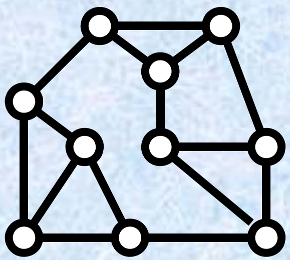
Connectivity

Is there a path between any two nodes?

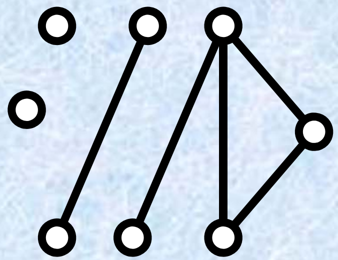
Easy problem

Algorithm: DFS, BFS, Union-Find

Complexity: DFS, BFS $O(|V|+|E|)$, Union-Find $O(|E| \cdot \alpha(|V|))$



Yes,
one connected component.



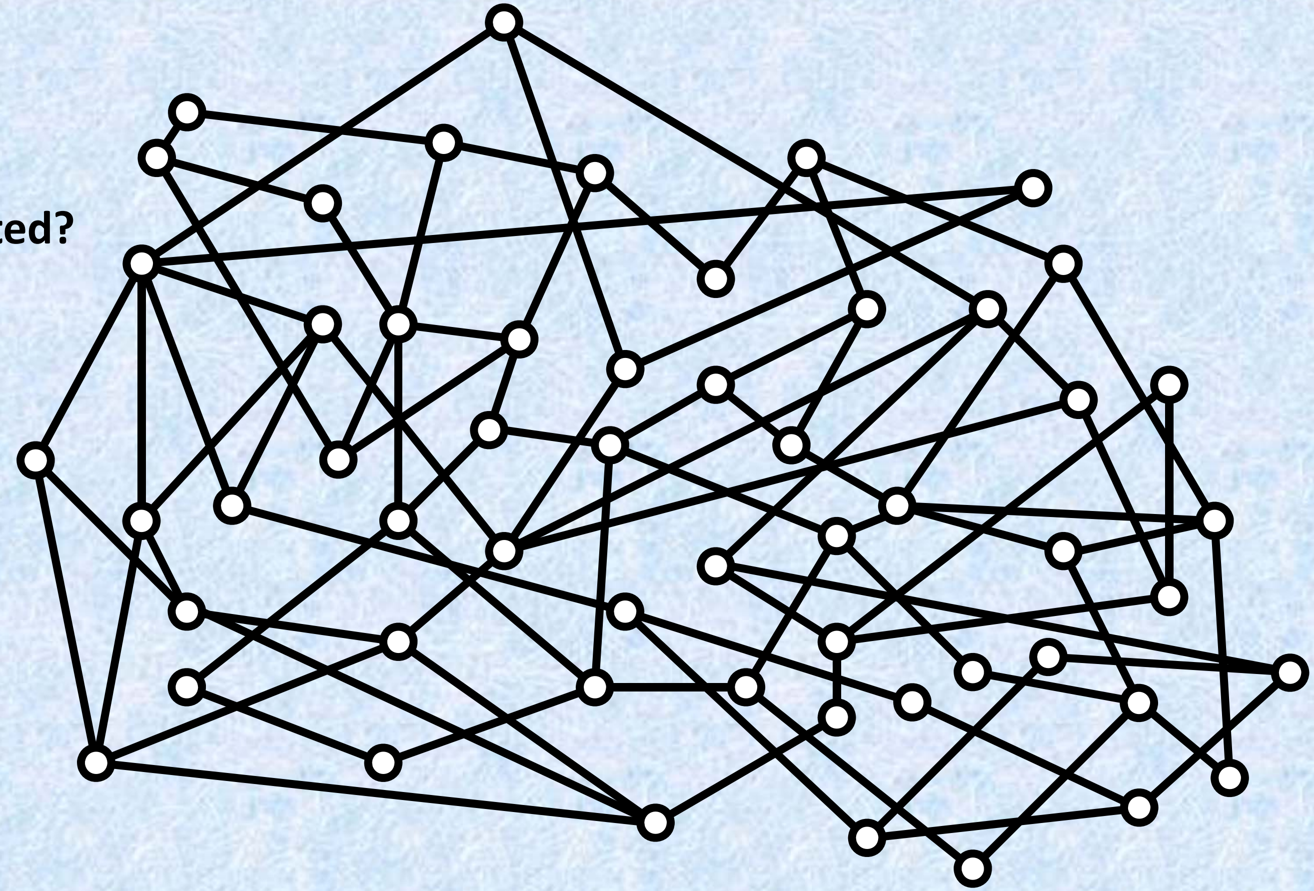
No,
four connected components.

Connectivity

Is there a path between any two nodes?

Easy problem

Is the graph connected?



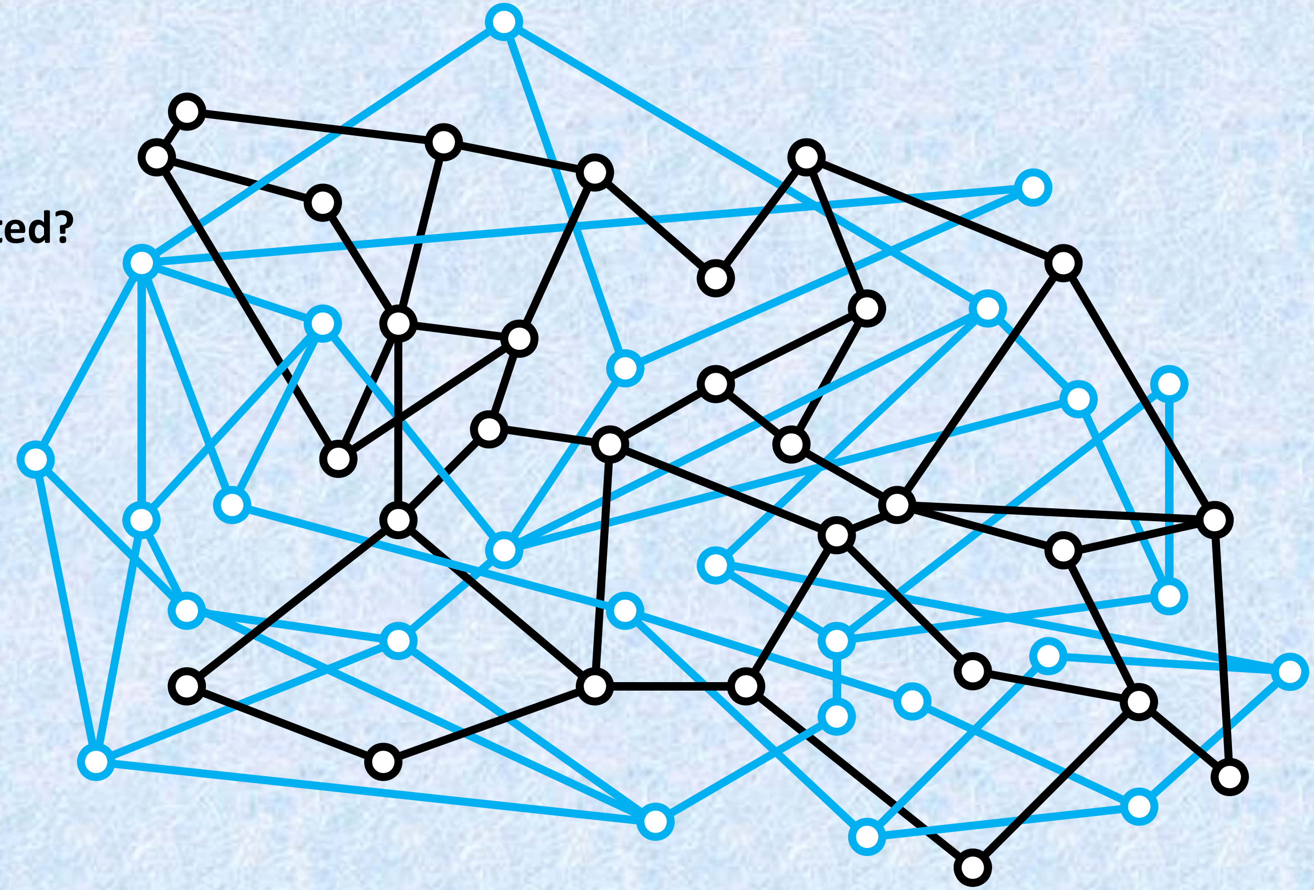
Connectivity

Is there a path between any two nodes?

Easy problem

Is the graph connected?

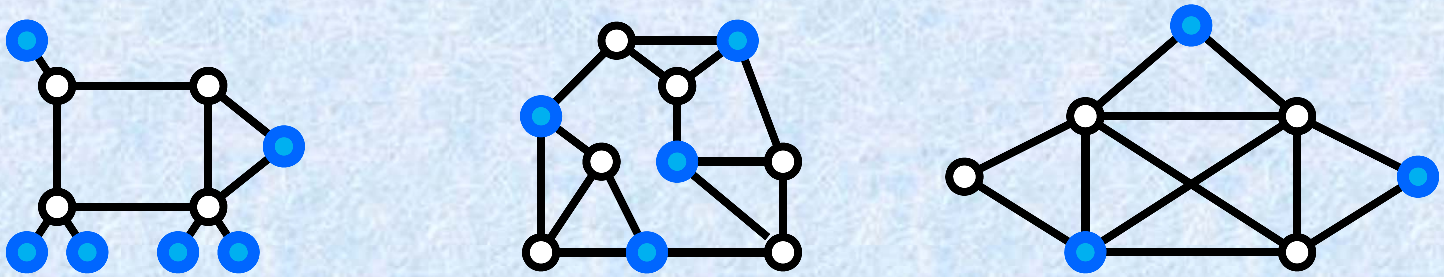
No,
it consists of
two components.



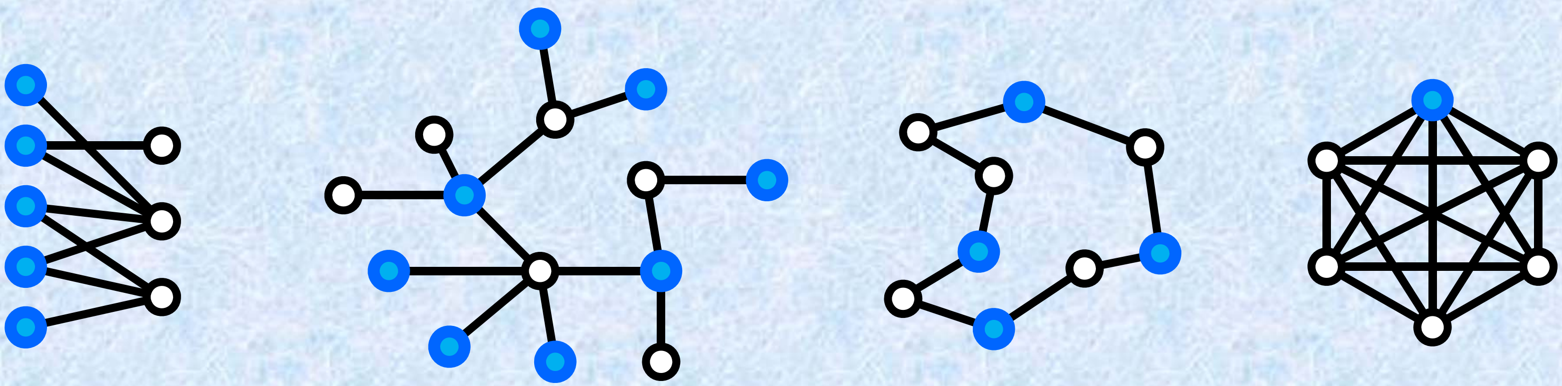
Independence

Maximum size of a set of nodes in which no two nodes are adjacent.

Hard problem in general



Easy problem on graphs with some particular structure



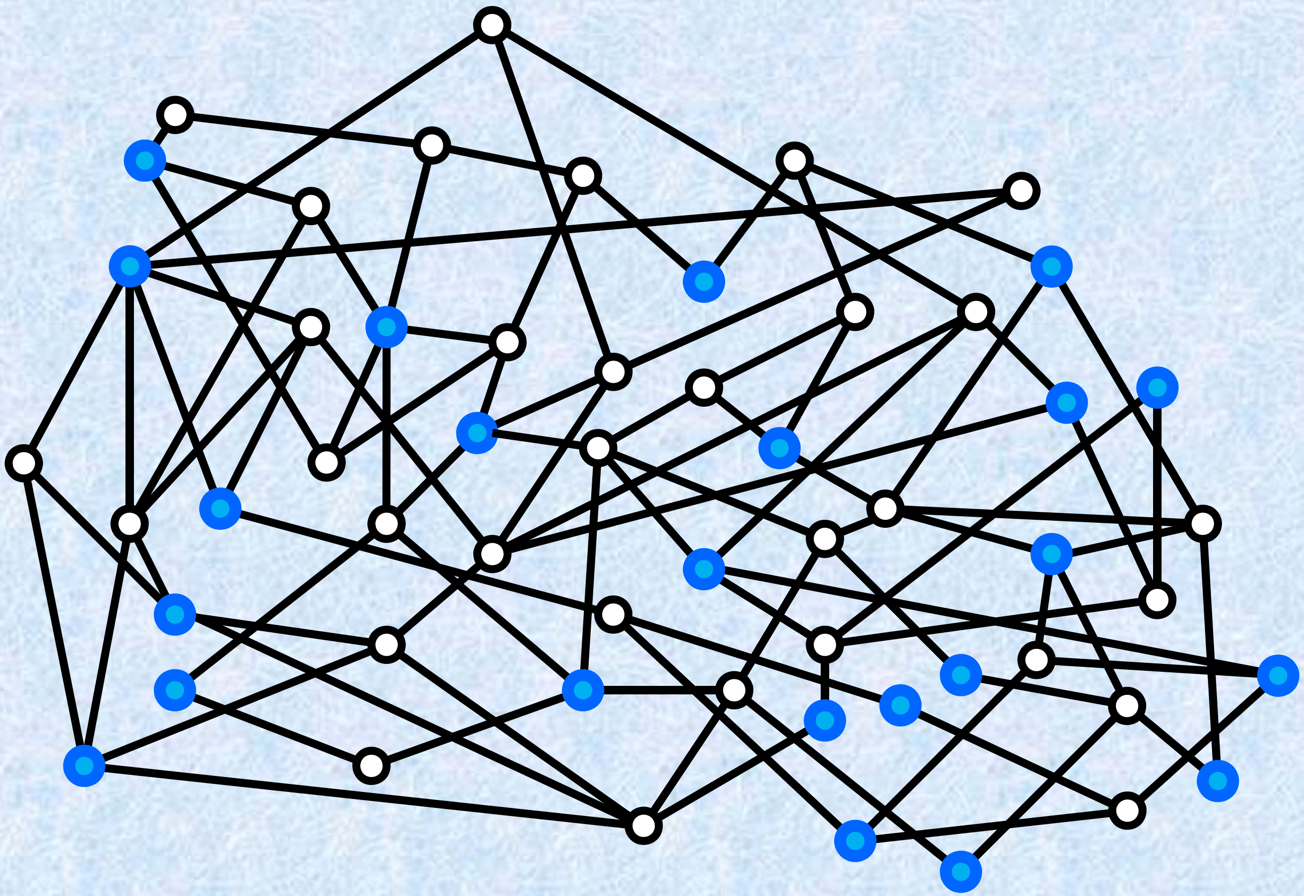
- **Bipartite graph**
- **Tree** is always bipartite
- **Cycle**
- **Complete graph**

Independence

Maximum size of a set of nodes in which no two nodes are adjacent.

Ex: How many of them in this graph? more than 23?

Hard problem

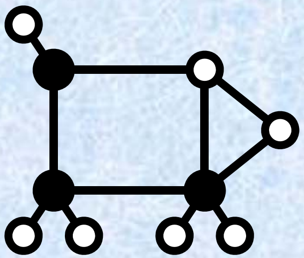
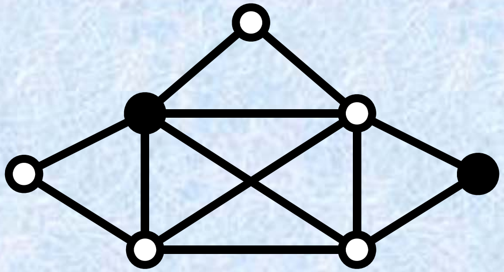
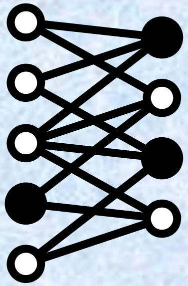


Dominance

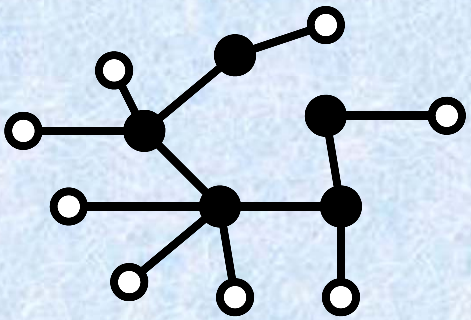
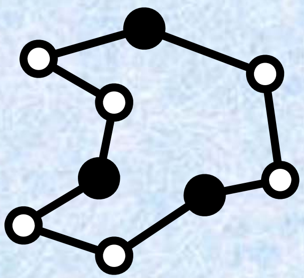
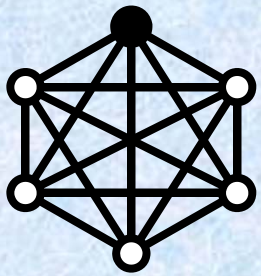
Maximum size of such set M of nodes that each node in the graph is either in M or is a neighbour of some node in M.

Ex. Fire station must either in a village or it in the immediately neighbour village. How many fire stations are enough to serve the region?

Hard problem

Easy problem on graphs with some particular structure

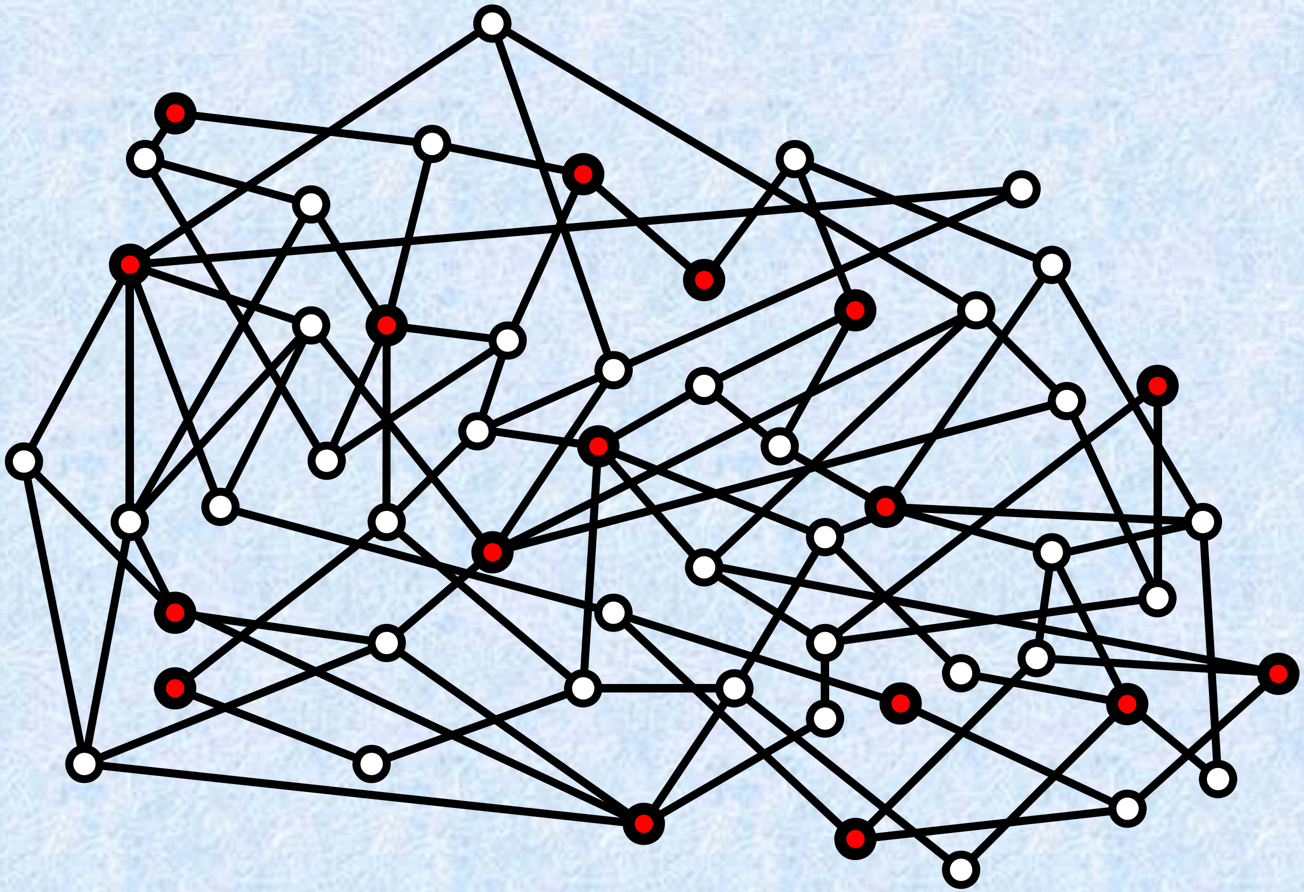
- ❖ **Tree,** apply Dynamic programming
- ❖ **Circle**
- ❖ **Complete graph**

11

Dominance

Ex. Fire station must either in a village or it in the immediately neighbour village. Are 17 fire stations are enough to serve the region?

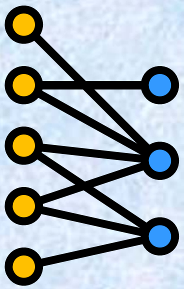
Hard problem



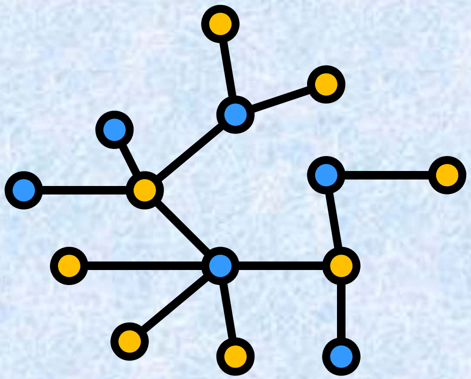
Colorability, chromatic number

Minimum number of colors needed to color each node so that any two neighbours have different color.

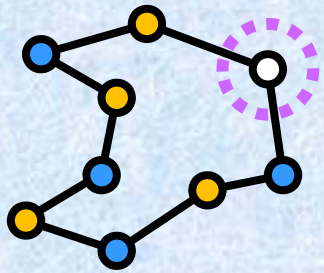
Is 2 colors enough? -- Easy problem.   Graph must be bipartite.



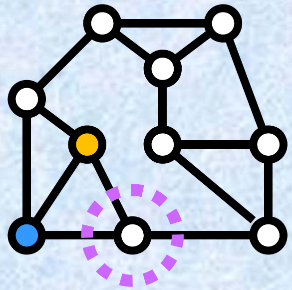
2 colors , bipartite graph



2 colors for any tree



2 colors are not enough in a cycle of odd length.



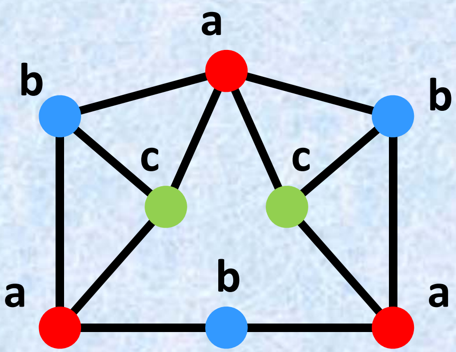
2colors are not enough, there is a cycle of odd length in the graph

Is graph bipartite? Apply BFS.
Mark by 1 all nodes in odd distance from the start and mark by 0 all nodes in even distance from start. If any two nodes with the same mark are connected by an edge, the graph is not bipartite (two-colorable).

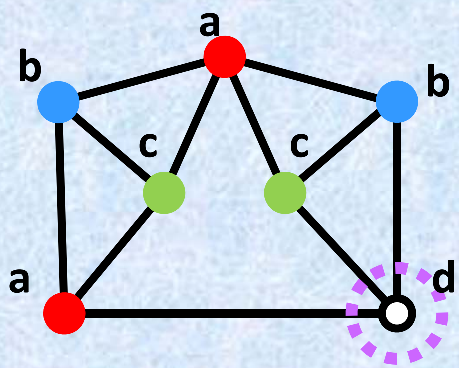
Colorability, chromatic number

Minimum number of colors needed to color each node so that any two neighbours have different color.

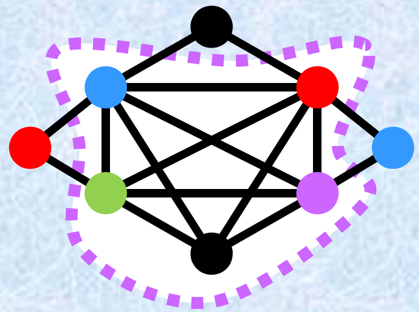
Hard problem -- Are 3 colors enough?



3 colors suffice



4 colors.
The node colors are chosen WLOG, the color of node at the bottom right cannot be any of a, b, c.



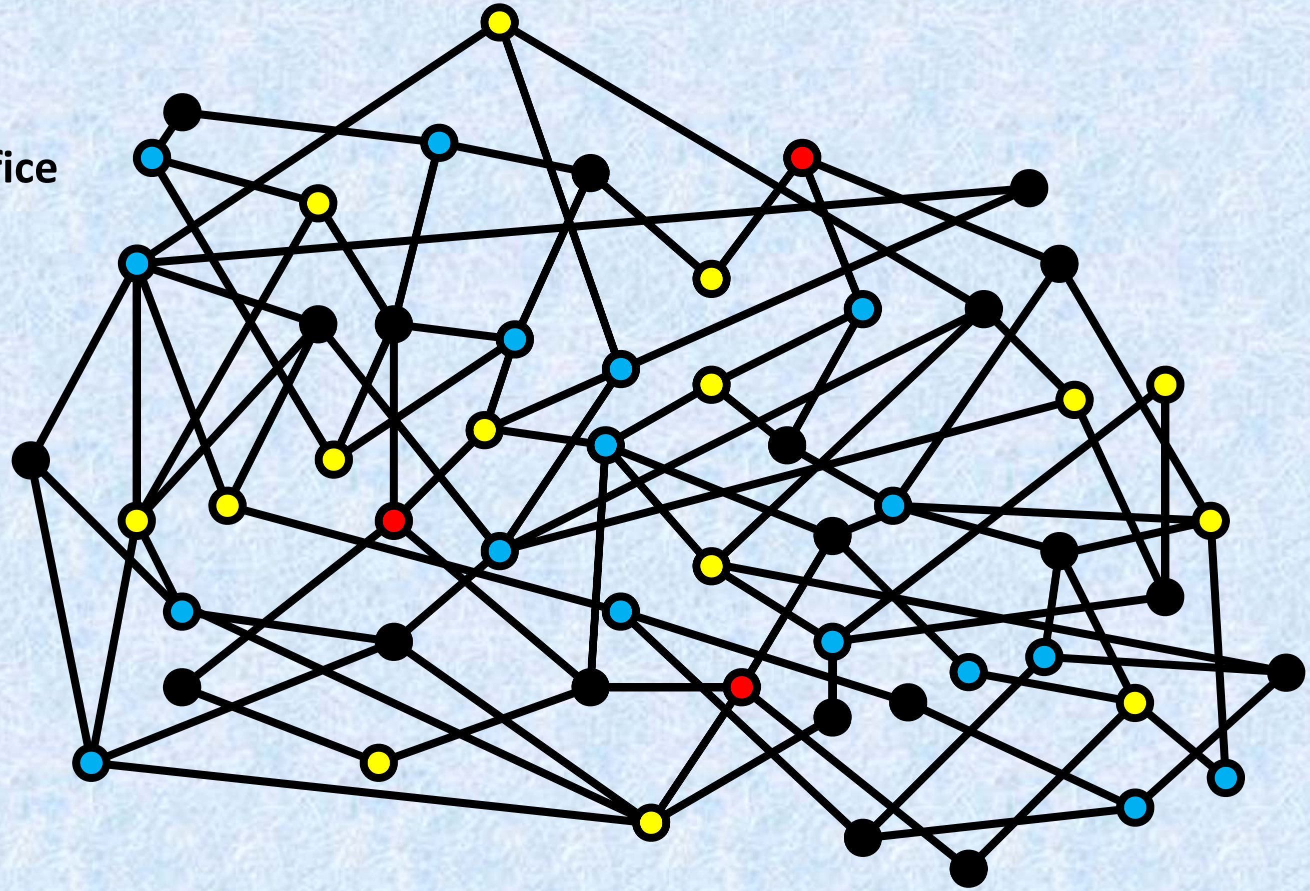
5 colors.
The graph contains a clique (complete subgraph) of size 5.
Clique detection is a hard problem.

Colorability, chromatic number

Minimum number of colors needed to color each node so that any two neighbours have different color.

Hard problem -- Are 3 colors enough?

4 colors are suffice in this graph. Maybe 3 colors would suffice too? ... ?



Shortest paths

Minimum possible number of edges(nodes) on a path from A to B.

Easy problem

Algorithms: BFS, Dijkstra, Bellman–Ford, Floyd–Warshall, Johnson...
Complexities: Polynomial, mostly less than $O(N^3)$.

Longest paths

Typically, each node/edge can be visited at most once.

Hard problem for general graphs

Easy problem for trees and DAGs

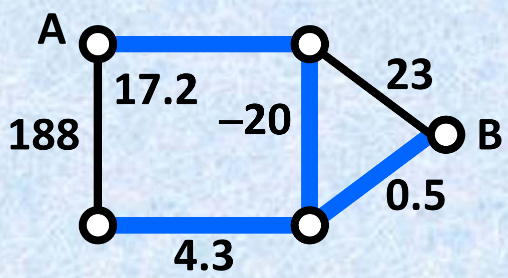
Algorithm: Dynamic programming
Compexity: $O(|V|+|E|)$

Minimum spanning tree

Minimum total cost (weight) of selected edges which connect all nodes in the graph. The selected edges form a tree.

Easy problem

Algorithms:	**Prim's**	$O(\ |V|^2\ )$	with matrix representation
		$O(\ |E| \cdot \log(|V|)\ )$	with linked list representation and with binary heap
	Kruskal's	$O(\ |E| \cdot \log(|V|)\ )$	
	Borůvka's	$O(\ |E| \cdot \log(|V|)\ )$	

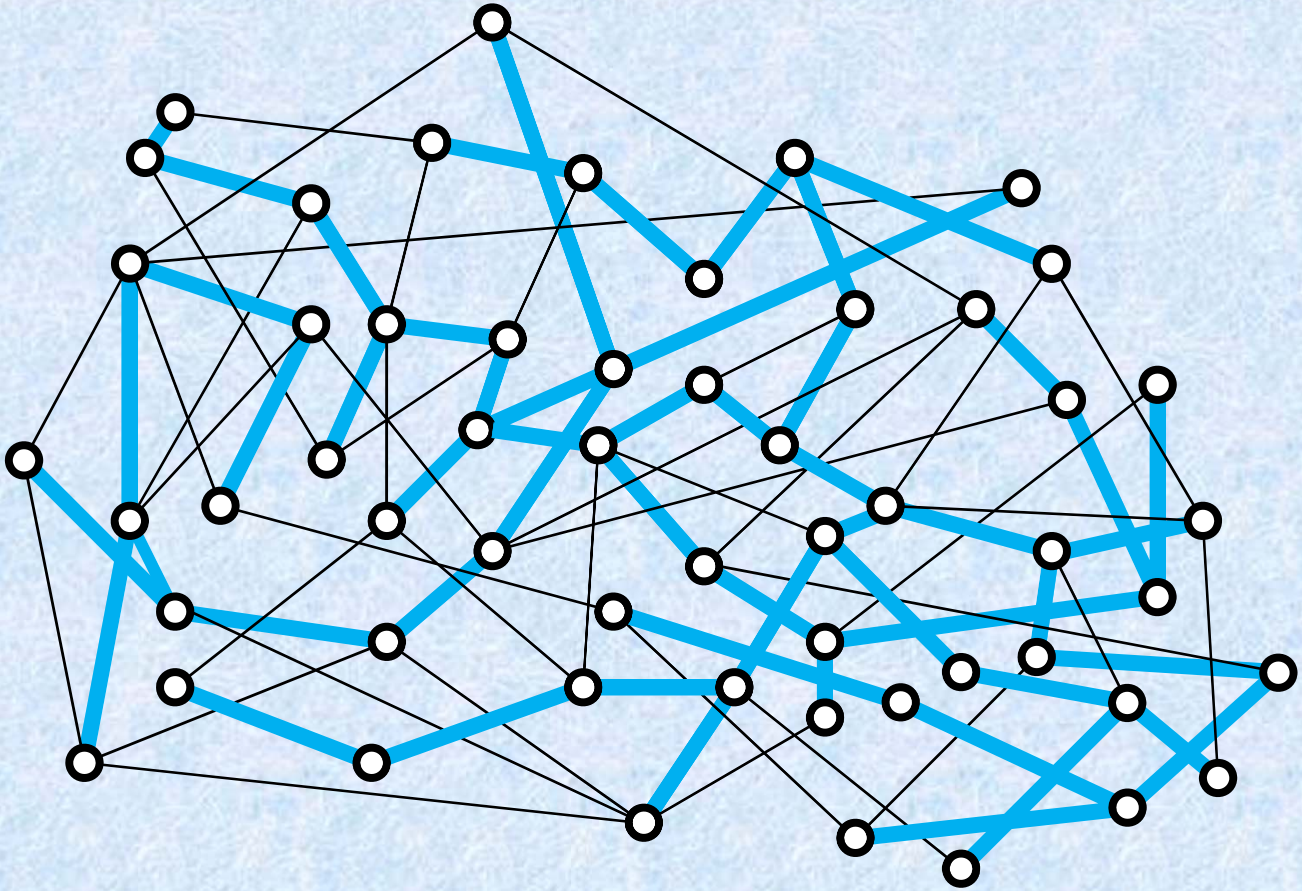


Minimum spanning tree

Minimum total cost (weight) of selected edges which connect all nodes in the graph. The selected edges form a tree.

Easy problem

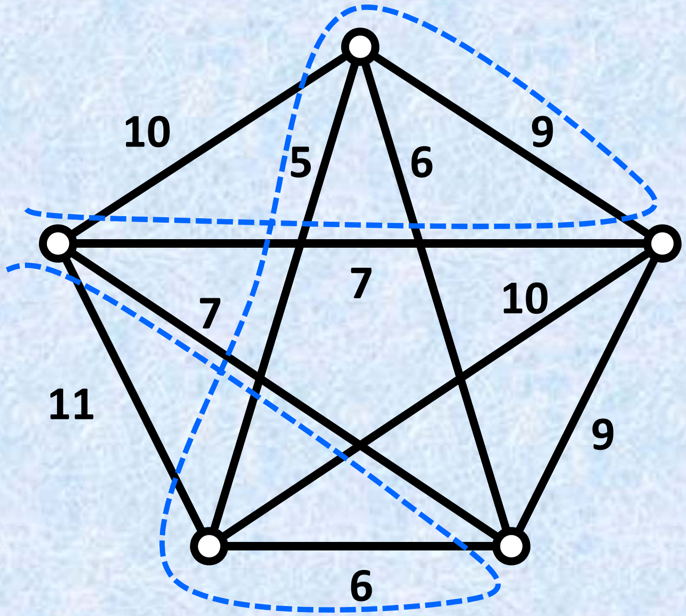
Here, the cost of an edge is proportional to its length (prefer shortest edges possible)



Travelling salesman problem (TSP)

Traverse a complete weighted graph, visit each node once and pay the minimum price for the journey = sum of costs of all visited edges.

Hard problem



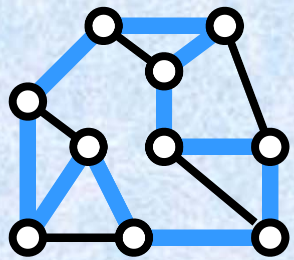
Hamilton path

Is there a path in the graph which contains each node (exactly once) ?

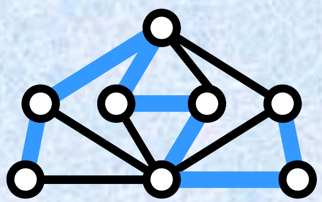
Hamilton cycle

Is there a cycle in the graph which contains each node

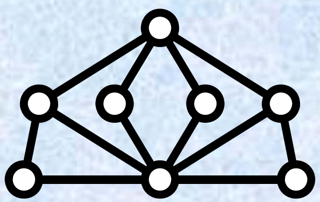
Hard problem



Both path and cycle exist.



Only pth exists, no cycle.



Neither a path nor a cycle exists.

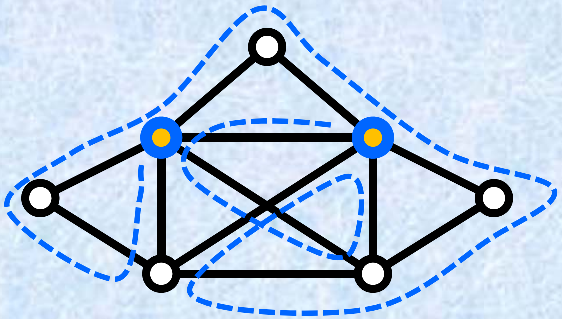
Euler trail

A trail that visits every edge exactly once (allowing for revisiting vertices). ?

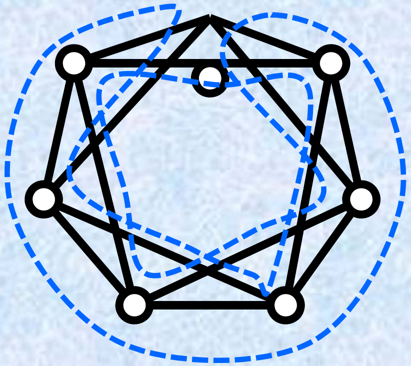
Ex: Can a postman walk through each street in their region exactly once?

Easy problem **Graph must be connected and it must contain at most 2 nodes of odd degree.**

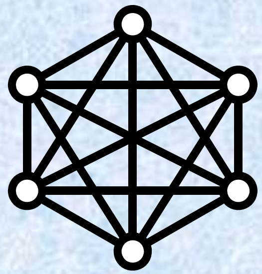
Algorithm: Hierholzer's O(|E|)



The trail starts and ends in the nodes with odd degree



The trail is closed, all node degrees are even



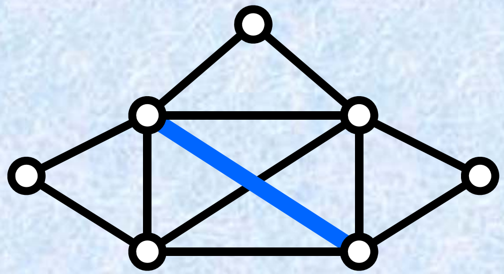
Euler trail does not exist, there are > 2 nodes with odd degree

Planar graph

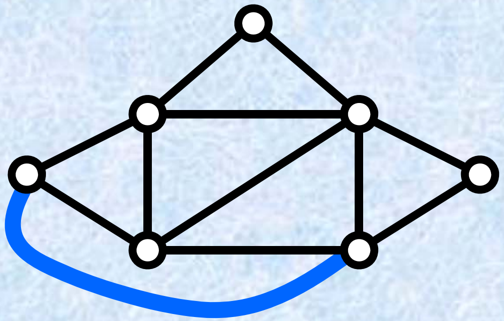
Can the graph be drawn in a plane wihout crossing its edges?

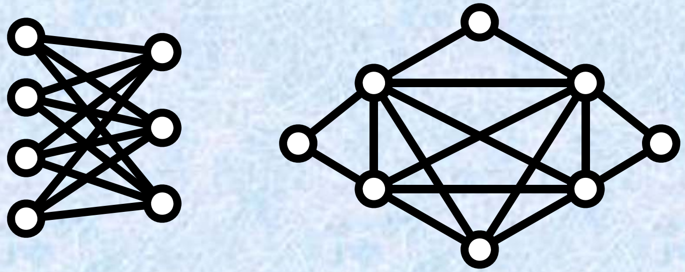
Easy question (however, little bit more advanced) 🙂

Algorithms: Hopcroft and Tarjan, O(|V|)
Boyer and Myrvold, O(|V|)



The graph is planar, the blue edge can be drawn differently:

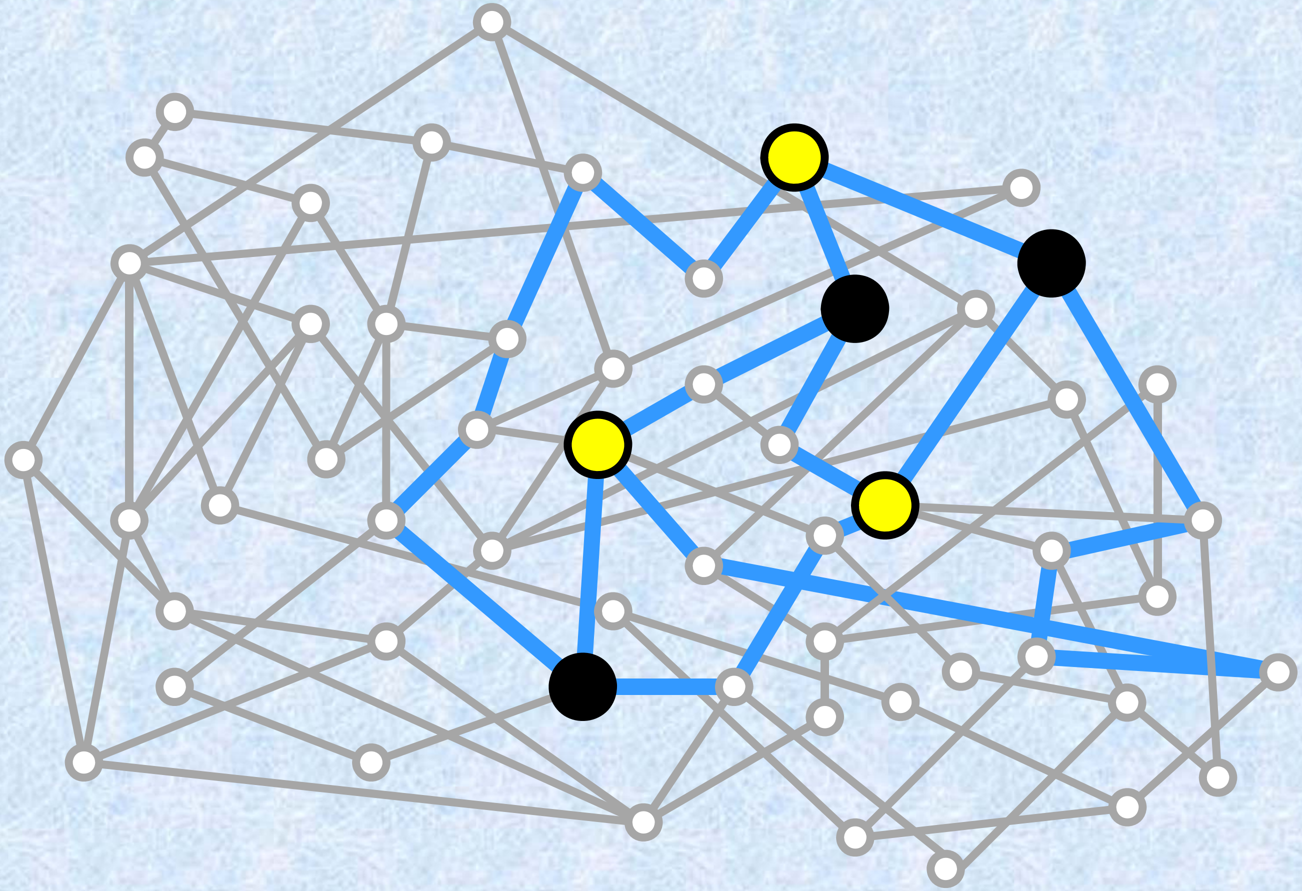




Not planar.
Non planar graphs "contain" either a complete graph on 5 nodes or a complete bipartite graph on 3 and 3 nodes.
The planar graphs do not.

Planar graph

Can the graph be drawn in a plane wihout crossing its edges?



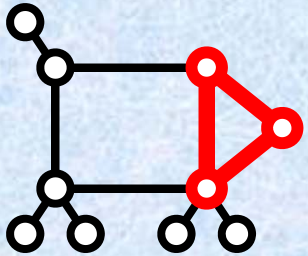
It is impossible here. Each black node is connected to each yellow node by a separate path and vice versa. It is the case of a complete bipartite graph with partitions of size 3 and 3. That graph cannot be drawn in the plane without edges crossing(s).

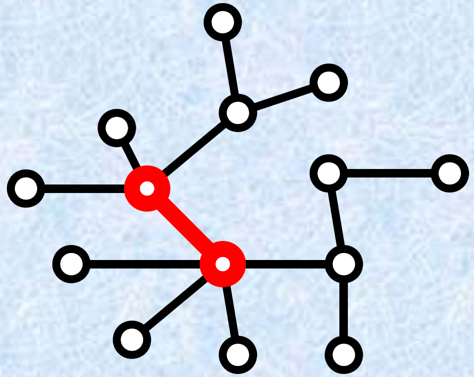
Clique number

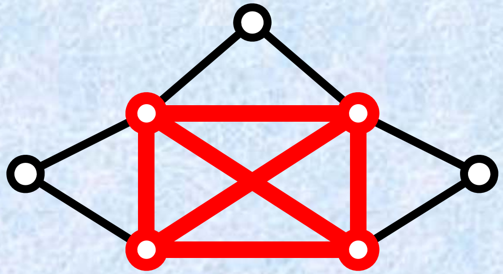
The size of the maximal clique, that is, of a subraph which is complete, that is, of the subgraph where each node is connected to each other node.

Ex. Choose a maximum group of your friends in which everybody knows each other.

Hard problem



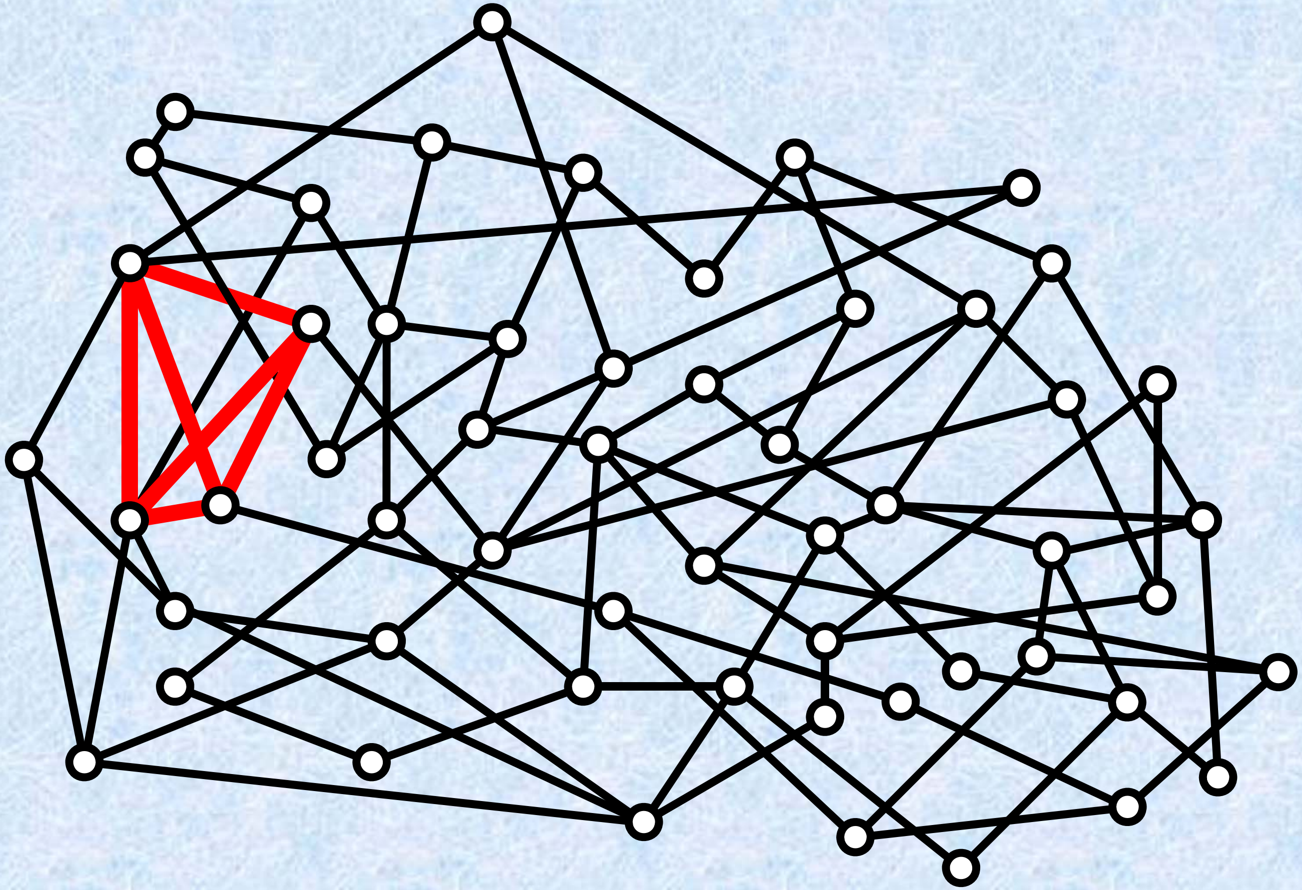




Clique number of all trees is 2.
(Rather obviously)

Clique number

The size of the maximal clique, that is, of a subraph which is complete,
that is, of the subgraph where each node is connected to each other node.

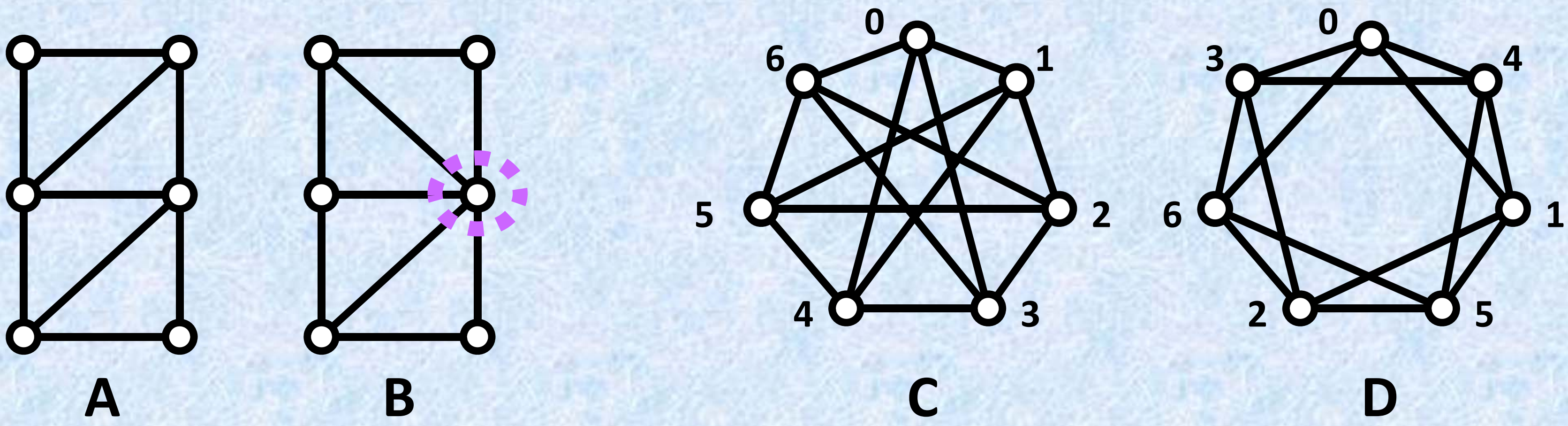


**Clique of size 5 (or bigger) is not in the graph.   To verify it mechanically, it is enough to check neighbour relations in all 5-element subsets of nodes.
The number of those subsets is   COMB(55, 5) = 3 478 761.**

Graph isomorphism

Is the structre of two graphs identical? In other words,
can one graph be drawn in such way that it looks exactly as the other one?

It is not know if this is a hard problem or an easy problem.



A and B are not isomorphic, right central node in B has degree 5, there is no analogous node anywhere in A. The structure of A and B must be different.

C and D are isomorphic, the nodes with the same labels correspond to each other, the edges in both C and D connect the nodes with the same labels

Partial recapitulation of the jungle of graph problems and their complexities

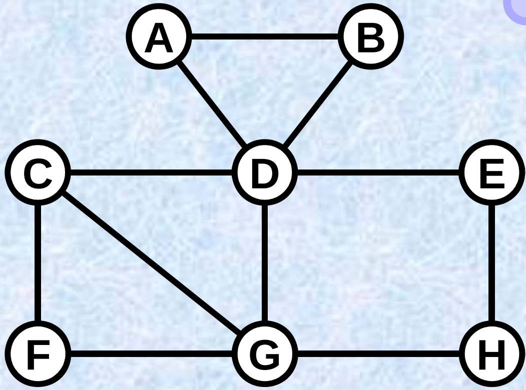
Easy problem	"It depends... "	Těžká otázka
Connectivity?	**Colorability?** 1,2 colors — easy; 3 or more colors — hard	Travelling salesman?
Shortest path?	**Isomorphism?** Trees, ciculants... — easy; regular graphs... — hard; etc...	Independence?
Min. spanning tree?	**Longest path?** DAG, tree — easy; general graph — hard	Dominancy?
Euler trail?		Hamiltonicity?
Planarity?		Clique number?

Many more questions ... ? Again, "it depends". There is no definite cookbook for determining the difficulty of a problem.

Graph most ususal representations

Graph most ususal representations

Undirected graph



Linked list representation

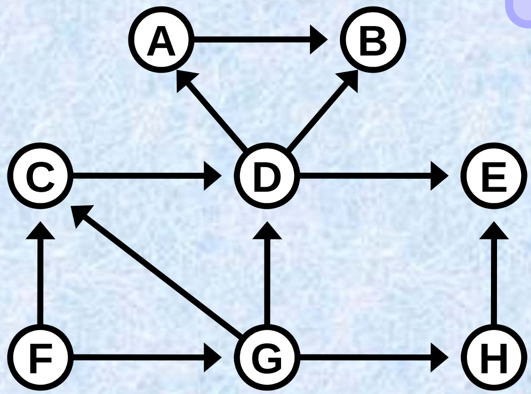
A → B → D
B → D → A
C → D → F → G
D → C → G → E → B → A
E → H → D
F → C → G
G → C → H → D → F
H → G → E

Adjacency matrix

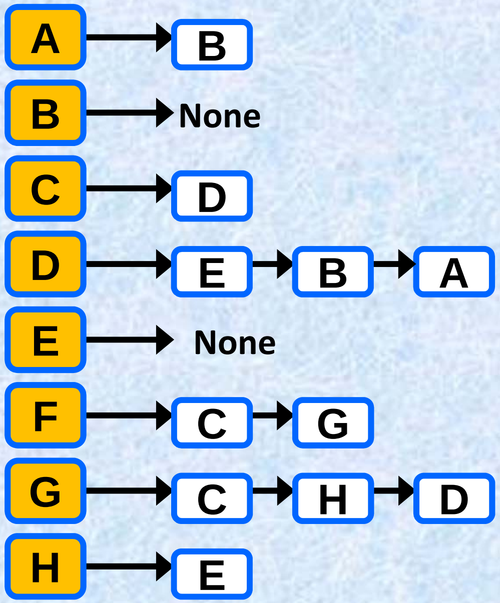
	A	B	C	D	E	F	G	H
A	0	1	0	1	0	0	0	0
B	1	0	0	1	0	0	0	0
C	0	0	0	1	0	1	1	0
D	1	1	1	0	1	0	1	0
E	0	0	0	1	0	0	0	1
F	0	0	1	0	0	0	1	0
G	0	0	1	1	0	1	0	1
H	0	0	0	0	1	0	1	0

Graph most ususal representations

Directed graph



Linked list representation

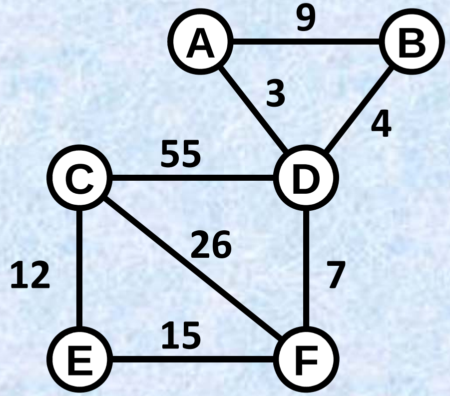


Adjacency matrix

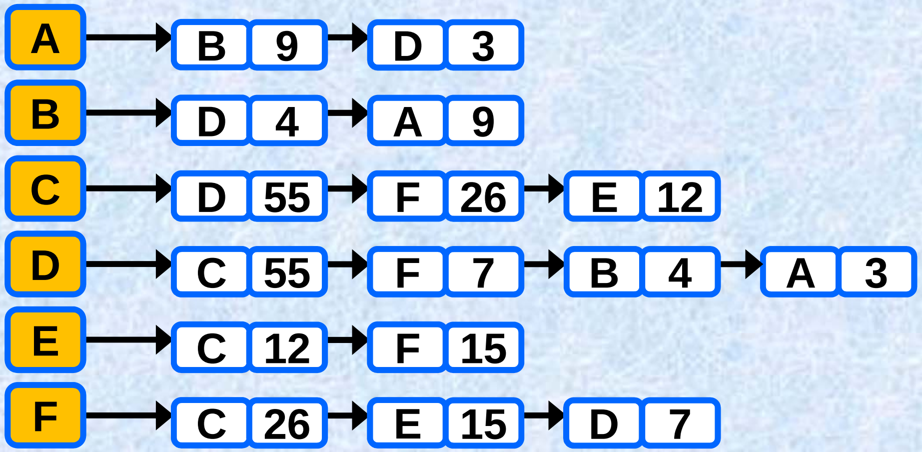
	A	B	C	D	E	F	G	H
A	0	1	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0
C	0	0	0	1	0	0	0	0
D	1	1	0	0	1	0	0	0
E	0	0	0	0	0	0	0	0
F	0	0	1	0	0	0	1	0
G	0	0	1	1	0	0	0	1
H	0	0	0	0	1	0	0	0

Graph most ususal representations

Undirected weighted graph



Linked list representation

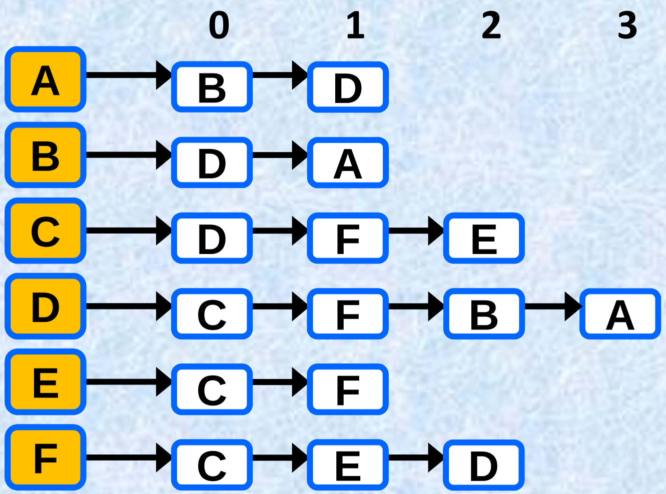


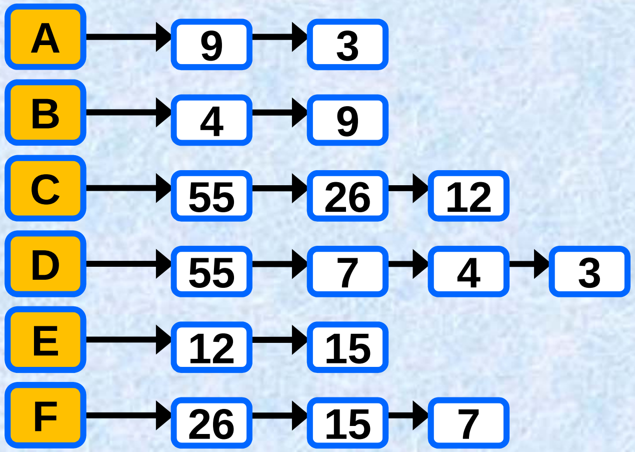
Weight (cost) matrix

	A	B	C	D	E	F
A	0	9	0	3	0	0
B	9	0	0	4	0	0
C	0	0	0	55	12	26
D	3	4	55	0	0	7
E	0	0	12	0	0	15
F	0	0	26	7	15	0

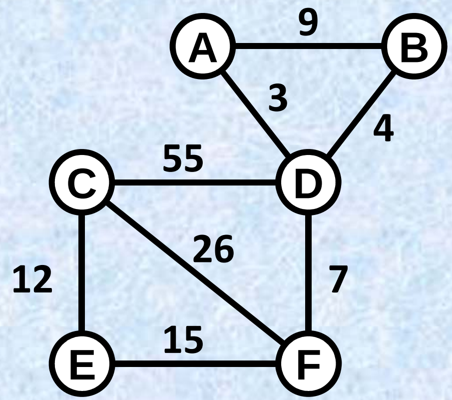
Graph most ususal representations

Linked list/ array representation





Undirected weighted graph



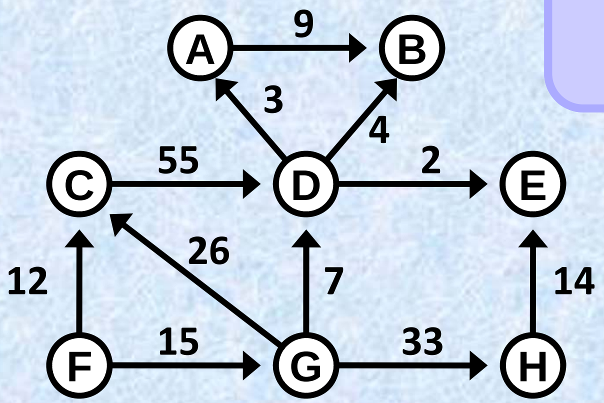
The weights of edges are at the same index in the second list.

\+ Pro: Simpler object or even no object at all in the arrays.

\- Con: Keepeing lists in sync needs more care and caution in the code.

Graph most ususal representations

Directed weighted graph



The representation is usually a more or less obvious combination of the methods in the previous cases -- Weight matrix or linked list.