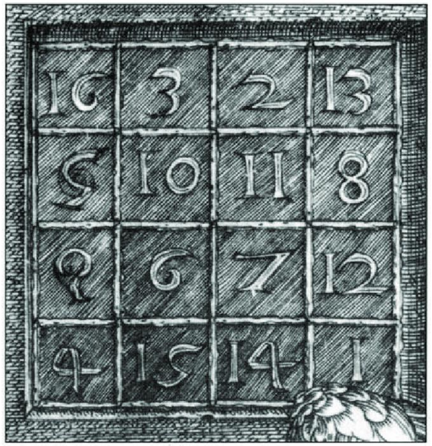
Logic and Discrete Structures -LDS



Course 7 – Graphs

Ș.l. dr. ing. Cătălin Iapă

catalin.iapa@cs.upt.ro

What have we covered so far?

Functions

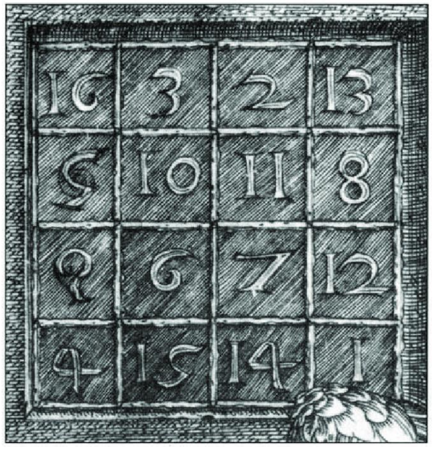
Recursive functions

Lists

Sets

Relations

Dictionaries



Graph theory

What is a graph?

Paths and cycles in graphs

Representing and traversing graphs

Graphs in PYTHON

Exercises with graphs in PYTHON

Graph theory

Graph theory is the mathematical study of graphs (representing relations between objects).

Graphs are one of the objects of study in discrete mathematics.

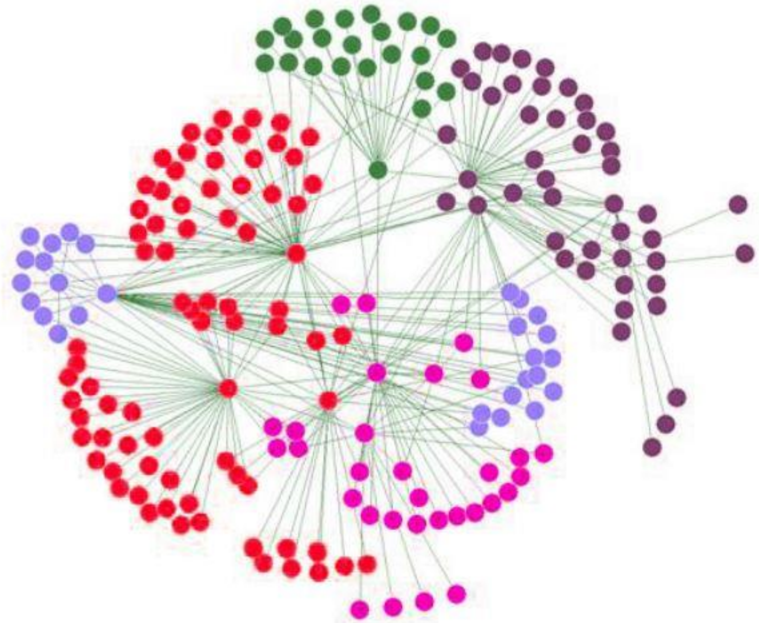
From this evolved network science: the study of complex networks.

Examples: computer, telecommunications, energy, biological, social networks, etc.

Graph theory

"the study of representations as networks of physical, biological and social phenomena, leading to predictive models of these phenomena".

[US National Research Council]



Imagine: https://en.wikipedia.org/wiki/Social_network

A challenge in sociology

One of the most discussed, most studied questions of all time in sociology is:

On average, over a lifetime, who has more partners of the opposite gender, men or women?

(Here, we only take into account relationships between partners of different genders to make it easier for us to approach the problem mathematically.)

What do you think?

A challenge in sociology

Generally speaking, it is considered in literature that men have more opposite-gender partners than vice versa.

This is also because there are societies where polygamy is allowed, and there, as a rule, men have more women, not vice versa.

A challenge in sociology

We have 2 studies that aimed to answer this question:

1. The University of Chicago interviewed over 2500 people in a study conducted in the US.

The study reveals that men have on average 74% more opposite gender partners than women.

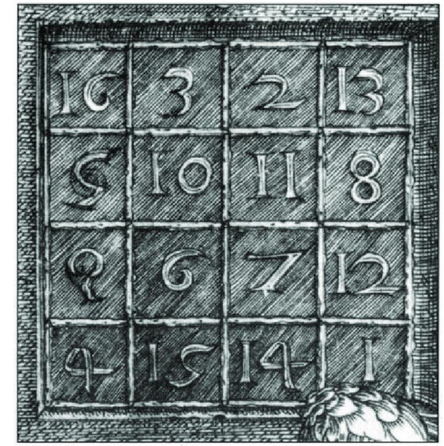
A challenge in sociology

2. Another study was also done in America by ABC News.

They questioned 1500 people in 2004. They concluded that men have an average of 20 partners of the opposite gender, while women have an average of only 6 over their lifetime.

From this it follows that men have, on average, 233% more partners of the opposite gender than women. ABC News says they have a margin of error of just 2.5%.

This kind of problem can be addressed very well using graphs.



Graph theory

What is a graph?

Paths and cycles in graphs

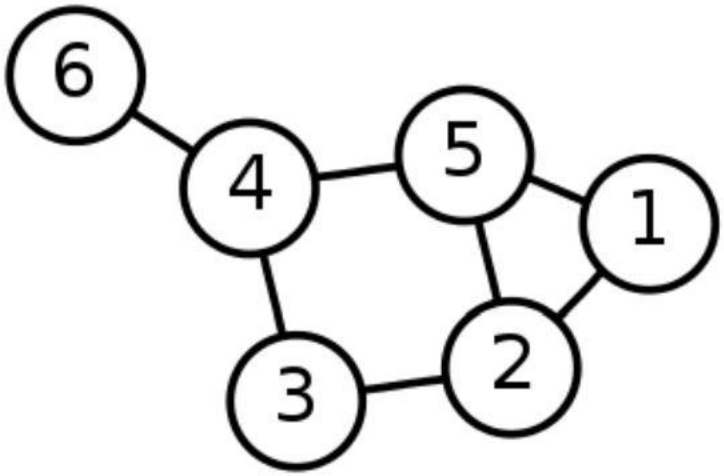
Representing and traversing graphs

Graphs in PYTHON

Exercises with graphs in PYTHON

What is a graph?

Informally, a graph represents a lot of objects (nodes, vertices, points, etc.) between which there are certain connections (lines, edges, arcs, etc.).



Imagine: http://en.wikipedia.org/wiki/File:6n_graf.svg

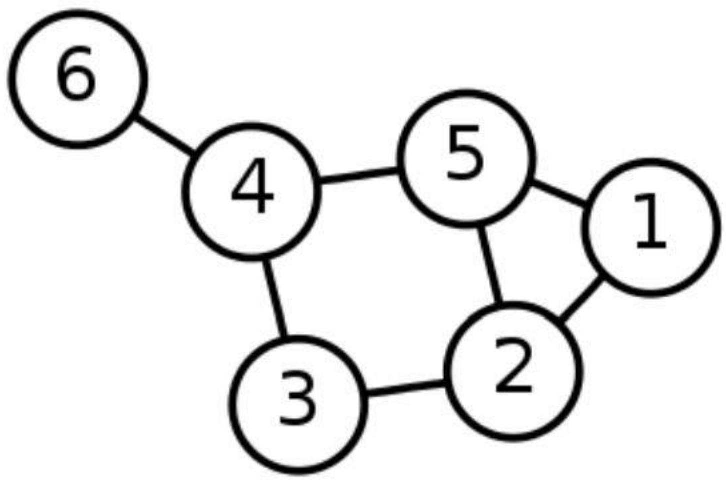
What is a graph?

Formally, a graph G is an ordered pair G=(V, E)

V - the set of Vertices and

E - the set of Edges

- a set of pairs $(u, v) \in V \times V$



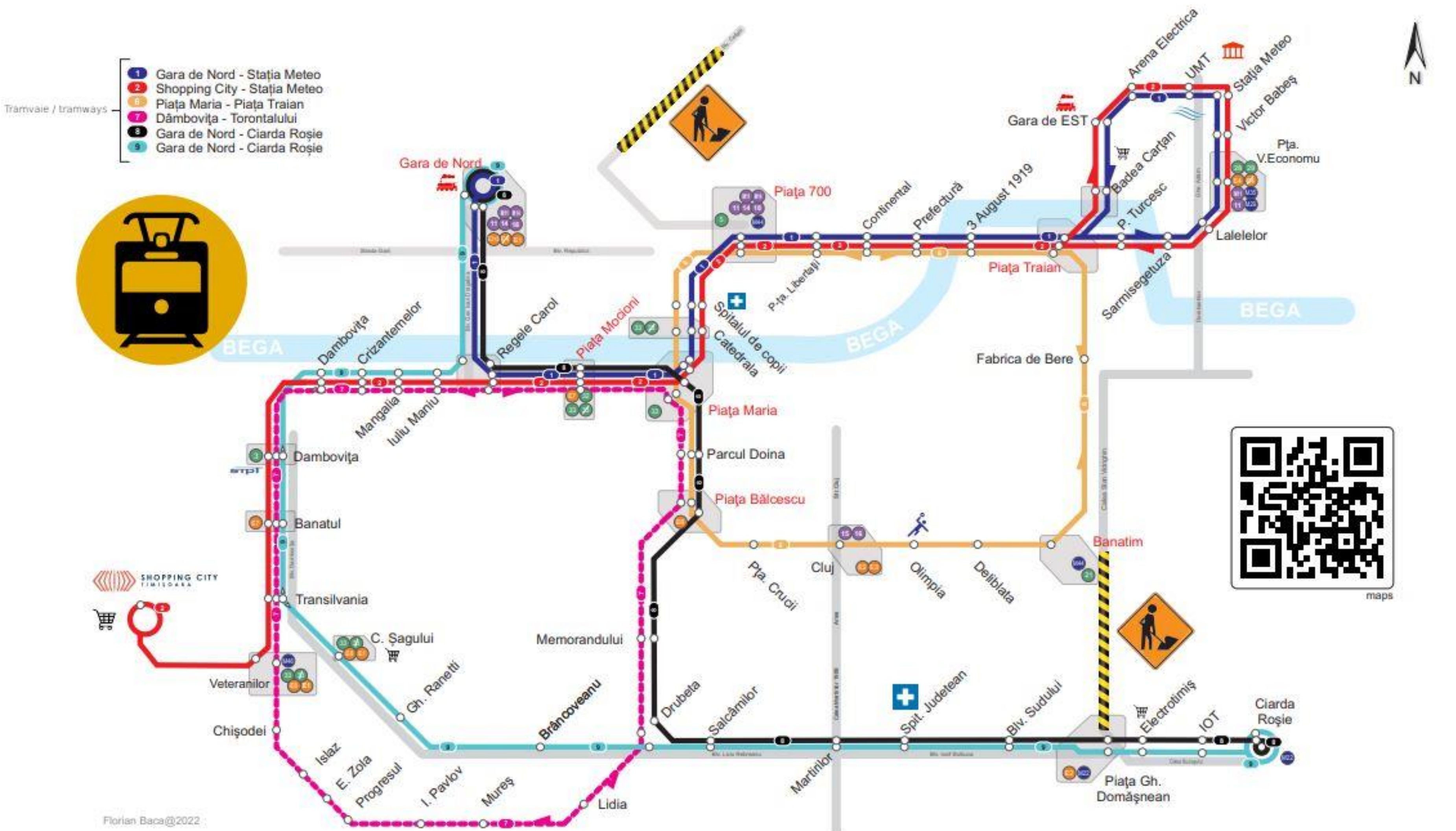
Imagine: http://en.wikipedia.org/wiki/File:6n_graf.svg

What is a graph?

The set of nodes must be a finite non-empty set, so it is not possible to have a graph without vertices, but it is possible to have a graph without edges.

So a graph can be represented as a geometric figure made up of points (corresponding to vertices/nodes) and straight or curved lines connecting these points (corresponding to edges or arcs).

Map of tram routes in Timisoara



Imagine: http://www.ratt.ro/reteaua.html

Graphs - general notions

The order of a graph is called the number of vertices of the graph.

A vertex v is incident to an edge r if edge r touches vertex v - $v \in r$.

Two vertices are called adjacent if there is an edge connecting them.

Two edges are adjacent if there is a vertex incident to both edges.

Graphs - general notions

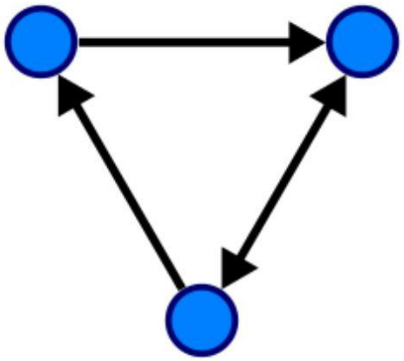
The degree of a vertex is the number of edges that are incident to that vertex.

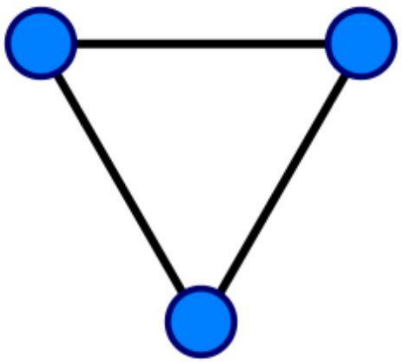
Adding the degrees of all the vertices in graph G gives twice the number of edges.

Directed and undirected graphs

A graph is directed if its edges are ordered pairs.

A graph is undirected if its edges are unordered pairs (no matter the direction of traversal).





Imagini: http://en.wikipedia.org/wiki/File:Directed.svg
http://en.wikipedia.org/wiki/File:Undirected.svg

Graphs and relations

The set of edges of a graph forms a relation E ∈ V x V over the set of nodes.

An undirected graph can be represented by a symmetric relation:

$$\forall u, v \in V \,.\, (u, v) \in E \rightarrow (v, u) \in E$$

In a directed graph, E is any relation (it doesn't have to be symmetric, but it can be)

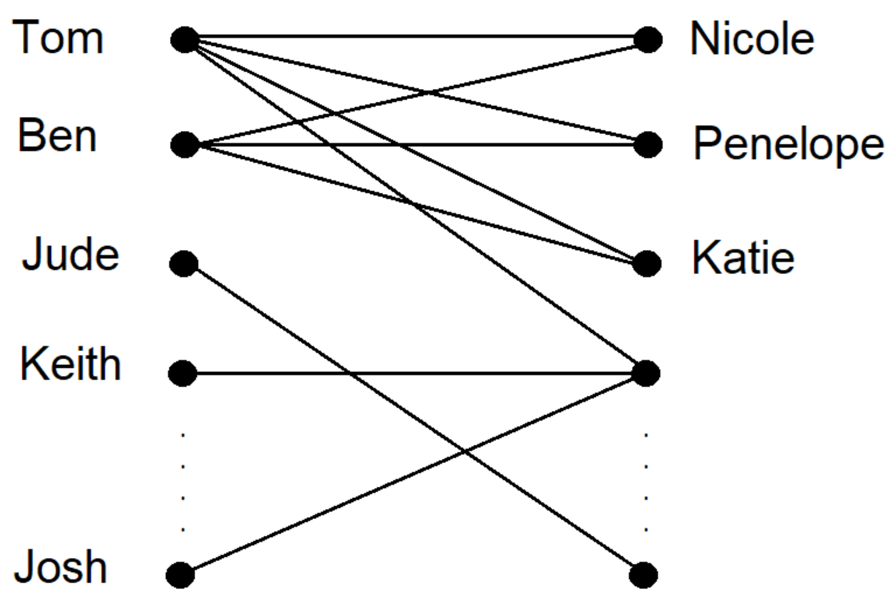
Reciprocally, any binary relation can be seen as a directed graph for (u, v) ∈ E we introduce an edge

u → v

A challenge in sociology

Let's go back to the problem in sociology. How can we represent the partner problem with graphs?

If we represent the set of men (below left) and the set of women (below right), we can plot such a graph:



A challenge in sociology

In Romania, the number of nodes (persons) is 19,186,201 (on 1 January 2021) according to data from the National Institute of Statistics, of which approximately 9.34 million men and 9.84 million women.

Can we know the number of edges of this graph?

No, but we would have to calculate the ratio of average male node degrees to average female node degrees:

$$\mathrm{R} = \frac{M_{men}}{M_{women}}$$

R = 1,74 according to University of Chicago study

R = 3,33 according to ABC News study

A challenge in sociology

$$M_{men} = \frac{Total\ no.of\ edges}{No.of\ men's\ nodes},$$

$$M_{women} = \frac{Total\ no.of\ edges}{No.of\ women's\ nodes}$$

$$\mathrm{R} = \frac{M_{men}}{M_{women}} = \frac{Total\ no.of\ edges}{No.of\ men's\ nodes} \Big/ \frac{Total\ no.of\ edges}{No.of\ women's\ nodes}$$

$$= \frac{Total\ no.of\ edges}{No.of\ men's\ nodes} * \frac{No.of\ women's\ nodes}{Total\ no.of\ edges}$$

$$= \frac{No.of\ women's\ nodes}{No.of\ men's\ nodes} = \frac{9{,}84\ mil}{9{,}34\ mil} = 1{,}05$$

So we have shown mathematically, using graph theory, that in Romania the number of relationships that men have with opposite gender partners is only 5% higher than the number of relationships that women have.


16 3 2 13
5 10 11 8
9 6 7 12
4 15 14 1

Paths in graphs

A path in a graph is a sequence of edges connecting a sequence of vertices $x_0, \ldots x_n$ with $n \geq 0$ such that $(x_i, x_{i+1}) \in E$ for any $i < n$.

$$x_0 \rightarrow x_1 \rightarrow \ldots \rightarrow x_{n-1} \rightarrow x_n$$

We can define a path in both directed and undirected graphs.

A path has an initial vertex x_0 and a final vertex x_n.

The length of a path is the number of edges traversed.in particular, it can be zero(a vertex x_0, with no edges)

Cycles in graphs

A cycle is a non-zero length path in which the start and end vertices are identical (the same).

We often work with simple cycles:

Cycles in which edges and vertices do not occur more than once (except for the start node which is also the end node).

Complete graphs and connected components

A graph is connected if it has a path from any vertex to any vertex. (general definition, depends on the notion of path - in directed or undirected graph)

For undirected graphs: A connected component is a maximal connected subgraph.

- so it has a path between any two vertices
- no more vertices could be added by keeping it connected

A graph with n vertices and e edges has a number of connected components ≥ n - e. It can be proved by induction.

Directed graphs: weakly connected and strongly connected

A directed graph is weakly connected if it has an undirected path from any vertex to any vertex, and strongly connected if it has a directed path from any vertex to any vertex.

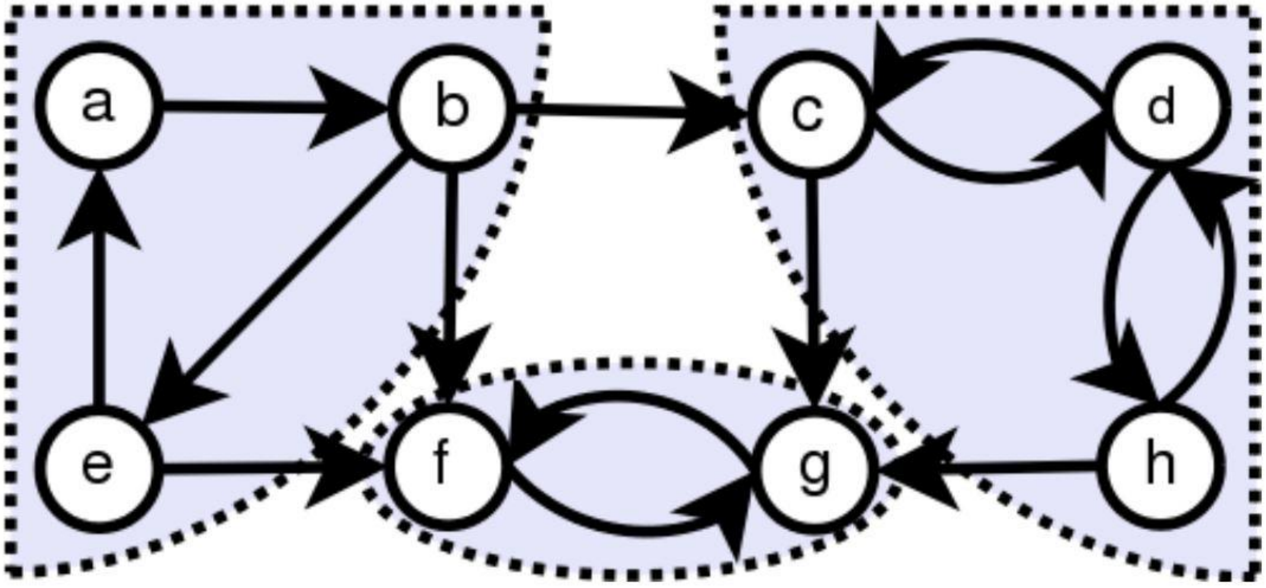
A strongly connected component is a maximal strongly connected subgraph.

Strong connected components are disjoint:

R(u, v) : path(u, v) and path(v, u) is an equivalence relation, and strongly connected components are equivalence classes

Directed graphs: weakly connected and strongly connected

The oriented graph in the figure is weakly connected. It has three strongly connected components.



Imagine: http://en.wikipedia.org/wiki/File:Scc.png

Determination of connected components (undirected graph)

Connected components are equivalence classes

- any node is in its own component - reflectivity
- a path from u to v is also a path from v to u – symmetry
- path(u, v) $\wedge$ path(v, w) $\rightarrow$ path(u, w) – transitivity

We determine the connected components by traversing the edges of the graph:

- initially, each node is in its own component
- for an edge (u, v) we join the components of u and v

Eulerian paths (in undirected graphs)

The degree of a vertex (in an undirected graph) is the number of edges touching the vertex.

An Eulerian path is a path that contains all edges of a graph exactly once.

An Eulerian cycle is a cycle that contains all edges of a graph exactly once.

Eulerian paths (in undirected graphs)

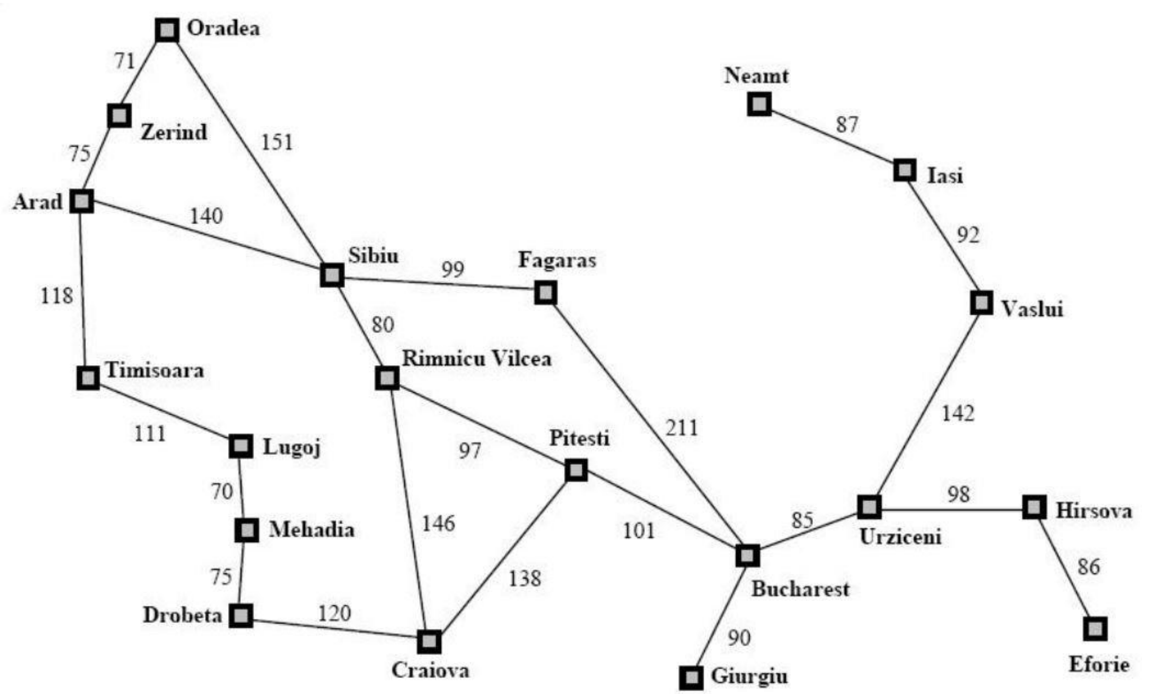
An undirected connected graph has an Eulerian cycle if and only if all vertices have even degree.

An undirected connected graph has an Eulerian path (but not a cycle) if and only if exactly two vertices have odd degree.

(the first and last vertices in the path)

Examples: maps as weighted graphs

Weighted graph: each edge has an associated numerical value called cost (can represent length, capacity, etc.)



Hartă (inexactă) din Russell & Norvig, Introduction to AI

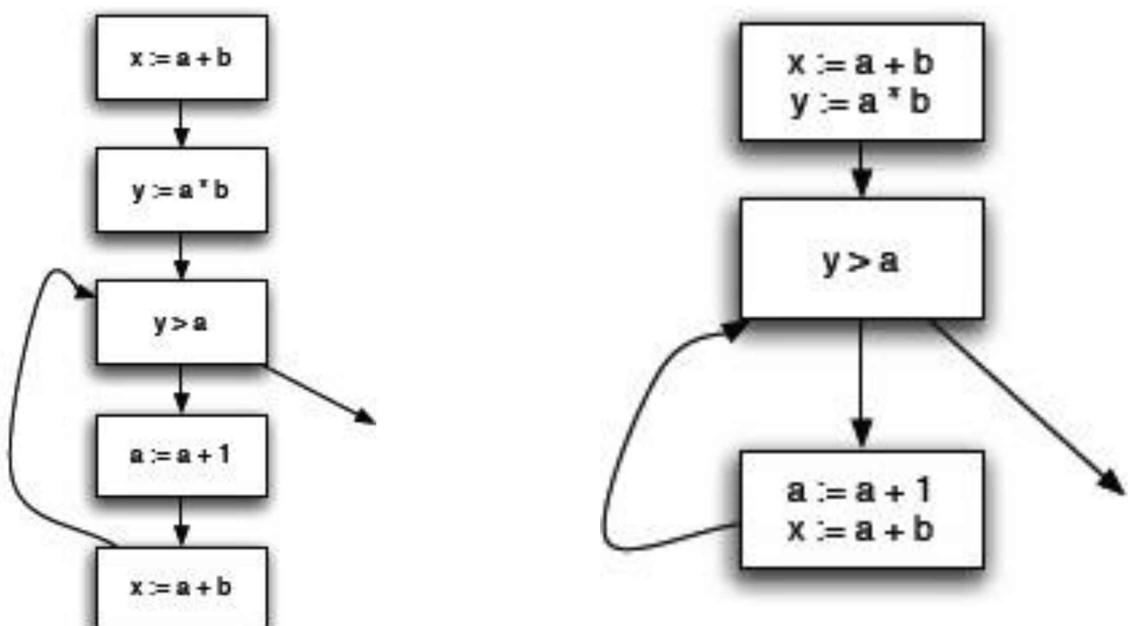
Exemples: Control flow graph

Representation of programs in compilers, code analyzers

- nodes: instructions or linear sequences of instructions (basic blocks)

- edges: describe the sequencing of instructions (control flow)

```
x := a + b;
y := a * b;
while (y > a) {
   a := a + 1;
   x := a + b
}
```



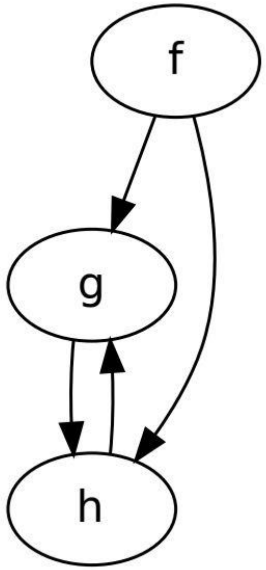
http://vinaytech.wordpress.com/2008/10/04/abstract-syntax-tree/

Exemples: Call graph

To represent a call graph we introduce an edge f → g if the function f calls g

The call graph is cyclic if there are (directly or indirectly) recursive functions

```
def g(x):
        return 0 if x==0 else 1+h(x-1)
def h(x):
        return 1 if x==0 else 2*g(x-1)
def f(x):
        return h(x) + g(x)
```



Graph theory

What is a graph?

Paths and cycles in graphs

Representing and traversing graphs

Graphs in PYTHON

Exercises with graphs in PYTHON

Graph representation

If we identify the nodes by (consecutive) numbers, we can represent the graph as a square adjacency matrix

M[i,j] = 1 if there is edge from i to j

M[i,j] = 0 if there is no edge from i to j

or M[i,j] can contain the length/cost of the edge (weighted graph)

Graph representation

Representation by adjacency lists

- for each vertex u: list/set of vertices v with edges (u, v)

We can store the information in a dictionary:

- key in dictionary = node in graph
- value in the dictionary = list/set of adjacent vertices

Graph representation

Representation by lists of pairs:

- for each edge from u to v we retain in the list/set - the pair (u, v)

Depth-first search

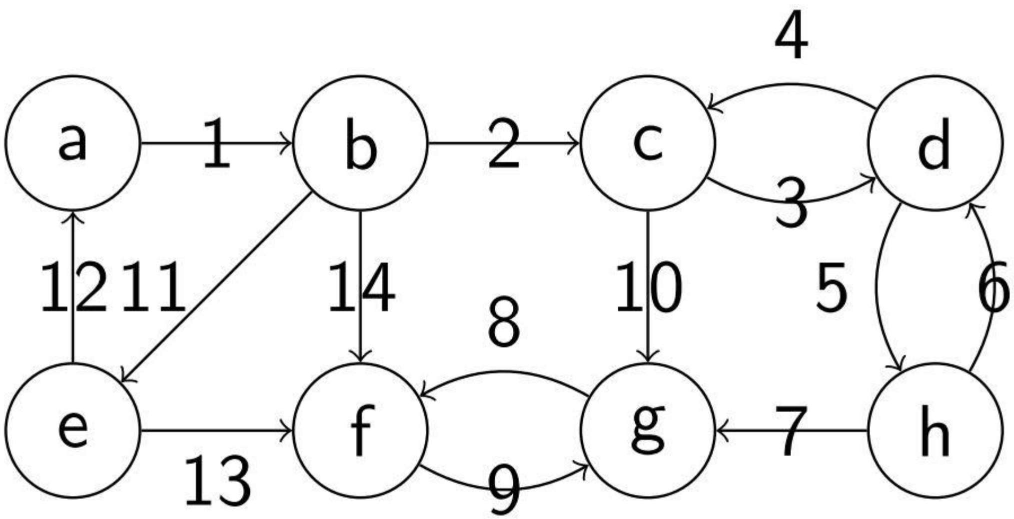
The depth traverse of the graph is a pre-order traverse.

After visiting the node, all directly adjacent nodes are traversed (recursively) (if not already visited)

Act as if directly adjacent nodes were inserted into a stack.

Depth-first search

Let the graph below, with the adjacency lists ordered by letters.The order of the lines from a to depth is as shown:



We can program: recursive function, accumulating the set of visited nodes

Breadth-first search

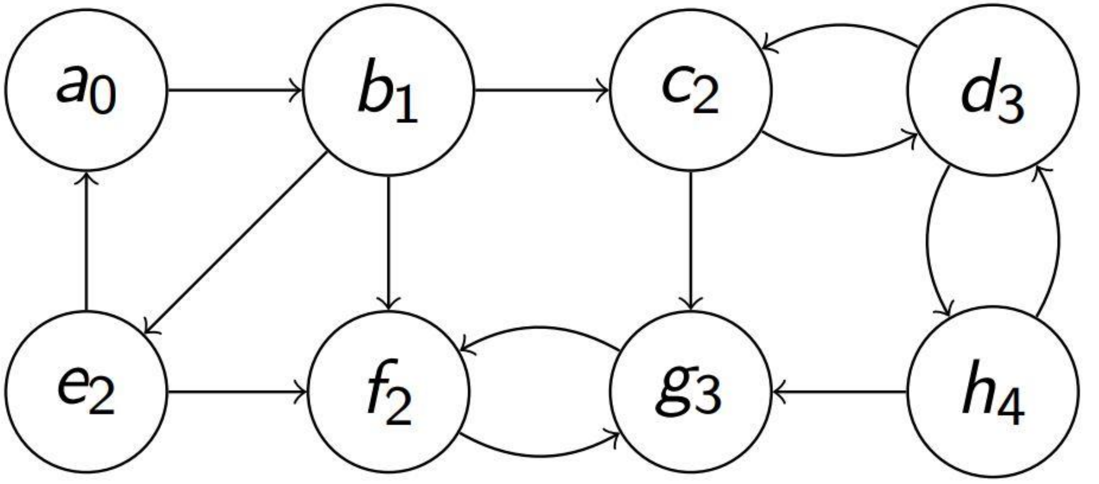
Breadth-first traversal visits vertices in order of minimum distance from the starting vertex

(in "waves" moving away from the starting node)

Nodes not yet visited are put in a queue.

Breadth-first search

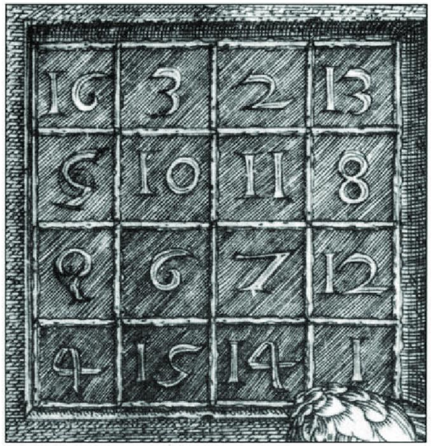
In the figure below, the minimum distance from vertex a is indicated (vertices with longer distances are covered later)



An implementation: recursive function,

accumulating: the set of all visited nodes

frontier: the set of new vertices reached in the current round



Graph theory

What is a graph?

Paths and cycles in graphs

Representing and traversing graphs

Graphs in PYTHON

Exercises with graphs in PYTHON

Graphs in PYTHON

In PYTHON we can represent a graph using a dictionary

We have the graph G = (V, E), V = {a, b, c, d, e}, E = {ab, ac, bd, cd, de}

```
graph = {
  "a" : {"b","c"},
  "b" : {"a", "d"},
  "c" : {"a", "d"},
  "d" : {"e"},
  "e" : {"d"}
}# {'a': {'b', 'c'}, 'b': {'d', 'a'}, 'c': {'d', 'a'}, 'd': {'e'}, 'e': {'d'}}
```

Displaying the vertices of a graph

To display the vertices of a graph held with a dictionary it is required to display the keys of the dictionary.

```
graph = {
   "a" : {"b","c"},
   "b" : {"a", "d"},
   "c" : {"a", "d"},
   "d" : {"e"},
   "e" : {"d"}
}
def displayV(graf):
   return list(graf.keys())
print(displayV(graf))              # ['a', 'b', 'c', 'd', 'e']
```

Displaying the edges of a graph

```
import functools

def dispalyE(graph, edges = set()):
    def f(acc,elem):
        k, v = elem
        def f_set(acc2,elem2):
            edges.add((k,elem2))
        functools.reduce(f_set, v, 0)
    functools.reduce(f, graph.items(), 0)
    return edges

print(dispalyE(graf))
# {('a', 'c'), ('d', 'e'), ('a', 'b'), ('e', 'd'), ('b', 'a'), ('b', 'd'), ('c', 'a'),
('c', 'd')}
```

Adding a new vertex

```
graph = {
  "a" : {"b","c"},
  "b" : {"a", "d"},
  "c" : {"a", "d"},
  "d" : {"e"},
  "e" : {"d"}
}
def addV(graph, vertex):
  if(not vertex in graph):
    graph[vertex] = set()
  return graph
print(addV(graf, "f"))                    # {'a': {'c', 'b'}, 'b': {'d', 'a'}, 'c': {'d', 'a'}, 'd': {'e'}, 'e': {'d'}, 'f': set()}
```

Adding a new edge

```
def addE_directed(graph, edge):
   if (edge[0] in graph):
      graph[edge[0]].add(edge[1])
   else:
      graph[edge[0]]={edge[1]}
   if (not edge[1] in graph):
      graph[edge[1]] = set()
   return graph

print(addE_directed(grafph,("a","d")))
print(addE_directed(grafph,("f","g")))
# {'a': {'b', 'c', 'd'}, 'b': {'d', 'a'}, 'c': {'d', 'a'}, 'd': {'e'}, 'e': {'d'}}
# {'a': {'b', 'c', 'd'}, 'b': {'d', 'a'}, 'c': {'d', 'a'}, 'd': {'e'}, 'e': {'d'}, 'f': {'g'}, 'g': set()}
```

Graph theory

What is a graph?

Paths and cycles in graphs

Representing and traversing graphs

Graphs in PYTHON

Exercises with graphs in PYTHON

Exercises

1. Let a graph be represented by the set of pairs of adjacent vertices. Create the data structure that holds information about the graph in a dictionary.

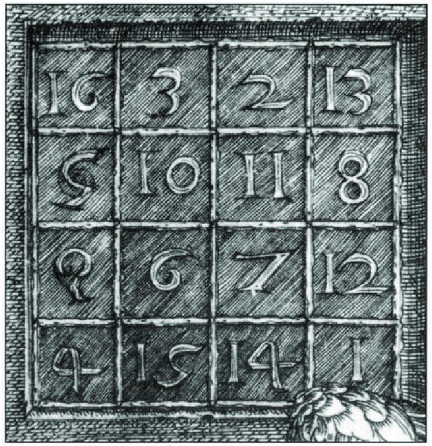
Exemple:

Input: {(1, 3), (1, 2), (2, 4), (4, 1)}

Output: {2: {4}, 4: {1}, 1: {2, 3}, 3: set()}

Exercises

```
import functools
def construct_graph(pairs, dictionary = {}):
    def f(acc, elem):
        if (elem[0] in dictionary):
            dictionary[elem[0]].add(elem[1])
        else:
            dictionary[elem[0]] = set({elem[1]})
        if(not elem[1] in dictionary):
            dictionary[elem[1]] = set()
    functools.reduce(f, pairs, 0)
    return dictionary
print(construct_graph({(1, 3), (1, 2), (2, 4), (4, 1)}))
```



Thank you!

Bibliografie

- The issue of the ratio of the number of relationships men have with partners of the opposite gender to the number of relationships women have with partners of the opposite gender was taken from the Mathematics for Computer Science course at the Massachusetts Institute of Technology (from https://ocw.mit.edu/)
- The content of the course is mainly based on material from previous years' LSD course, taught by Prof. Marius Minea, Ph.D., Ph. Casandra Holotescu (http://staff.cs.upt.ro/~marius/curs/lsd/index.html)