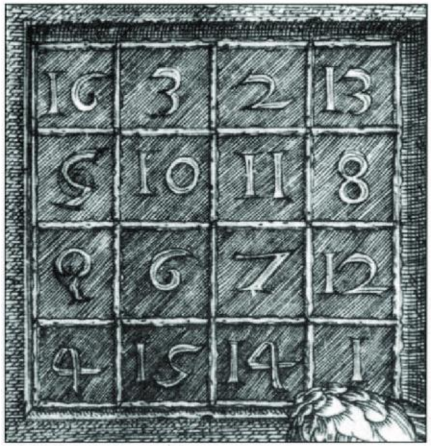
Logică și Structuri Discrete -LSD

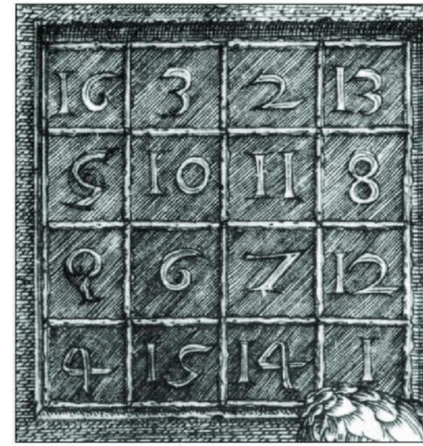


Cursul 7 – Grafuri

dr. ing. Cătălin Iapă

catalin.iapa@cs.upt.ro

Ce am parcurs până acum?



Funcții

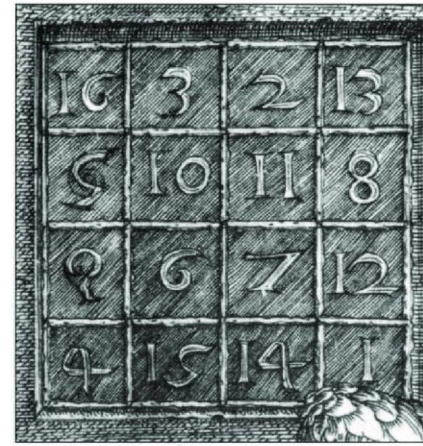
Funcții recursive

Liste

Mulțimi

Relații

Dicționare



Teoria grafurilor

Ce e un graf?

Drumuri și cicluri în graf

Reprezentarea și parcurgerea grafurilor

Grafuri în PYTHON

Exerciții cu grafuri în PYTHON

Teoria grafurilor

Teoria grafurilor este studiul matematic al grafurilor (reprezentând relații între obiecte).

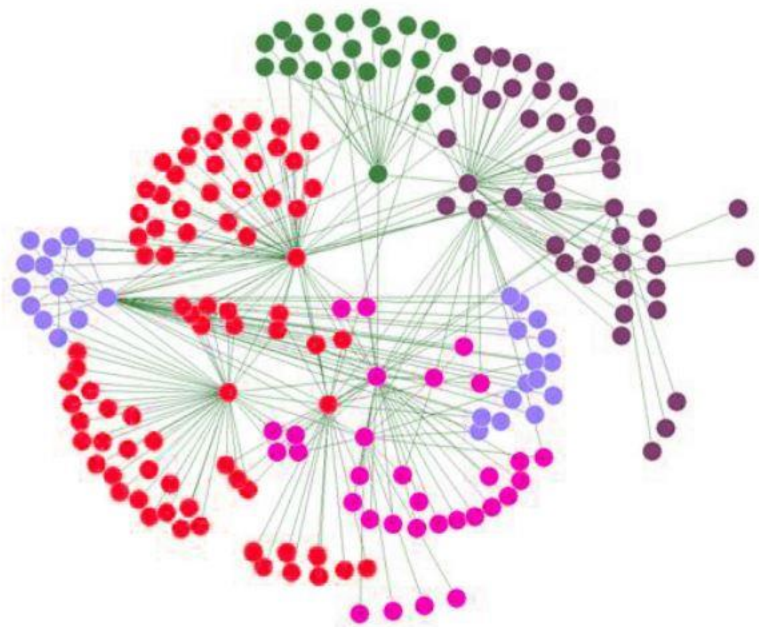
Grafurile reprezintă unul dintre obiectele de studiu în *matematica discretă*.

De aici a evoluat *știința rețelelor* (network science): studiul rețelelor complexe.

Exemple: rețele de calculatoare, în telecomunicații, energie, biologice, sociale etc.

Teoria grafurilor

“studiul reprezentărilor ca rețele a fenomenelor fizice, biologice și sociale, ducând la *modele predictive* ale acestor fenomene”.



[US National Research Council]

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

O provocare în sociologie

Una dintre cele mai discutate, cele mai studiate întrebări din toate timpurile din *sociologie* este:

În medie, pe parcursul vieții, cine au mai mulți parteneri de sex opus, bărbații sau femeile?

Voi ce credeți?

O provocare în sociologie

Vorbind generic, în *literatură* e considerat că bărbații au mai multe partenere de sex opus decât invers.

Asta și din cauză că sunt societăți în care e permisă poligamia, iar acolo, de regulă, bărbații au mai multe femei, nu invers.

O provocare în sociologie

Avem *2 studii* care au urmărit să răspundă la această întrebare:

1. *Universitatea din Chicago* a intervievat peste 2500 de oameni într-un studiu realizat în SUA.

Studiul relevă faptul că *barbații au în medie cu 74% mai multe partenere de sex opus decât femeile*.

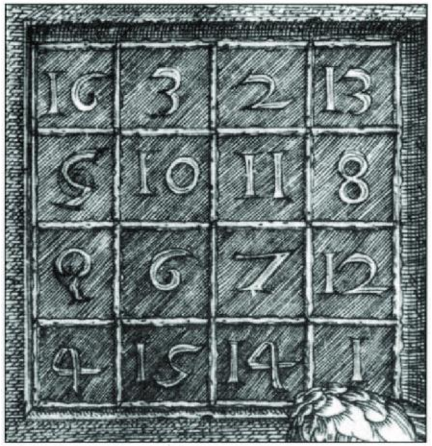
O provocare în sociologie

2. Un alt studiu a fost făcut tot în America de către *ABC News*.

Ei au sondat 1500 de oameni în 2004. Concluzia lor e că bărbații au în medie 20 de partenere de sex opus, în timp ce femeile au în medie doar 6, pe parcursul vieții.

Din asta reiese că *bărbații au, în medie, cu 233% mai multe partenere de sex opus decât femeile*. Cei de la ABC News spun că au o marjă de eroare de doar 2,5%.

Genul acesta de problemă se poate aborda foarte bine utilizând *grafuri*.



Teoria grafurilor

Ce e un graf?

Drumuri și cicluri în graf

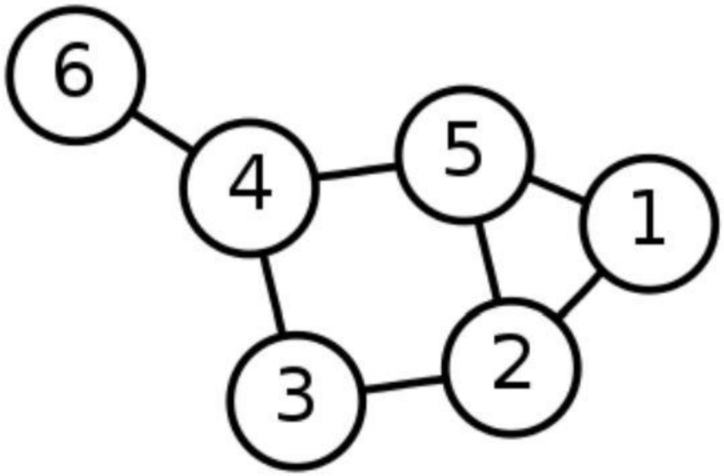
Reprezentarea și parcurgerea grafurilor

Grafuri în PYTHON

Exerciții cu grafuri în PYTHON

Ce e un graf?

Informal, un graf reprezintă o mulțime de *obiecte* (*noduri, vârfuri, puncte etc.*) între care există anumite *legături* (*linii*, *muchii*, *arce etc.*).



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

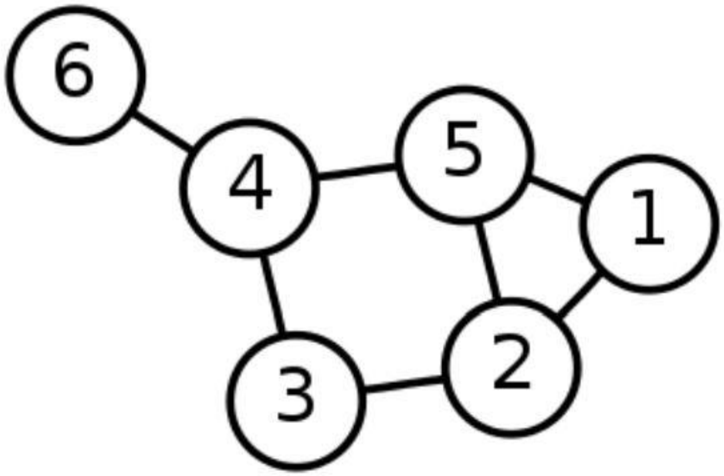
Ce e un graf?

Formal, un graf G e o pereche ordonată $G=(V, E)$

V - *mulțimea nodurilor (eng. Vertices)* și

E - *mulțimea muchiilor (eng. Edges)*

- o mulțime de perechi $(u, v) \in V \times V$



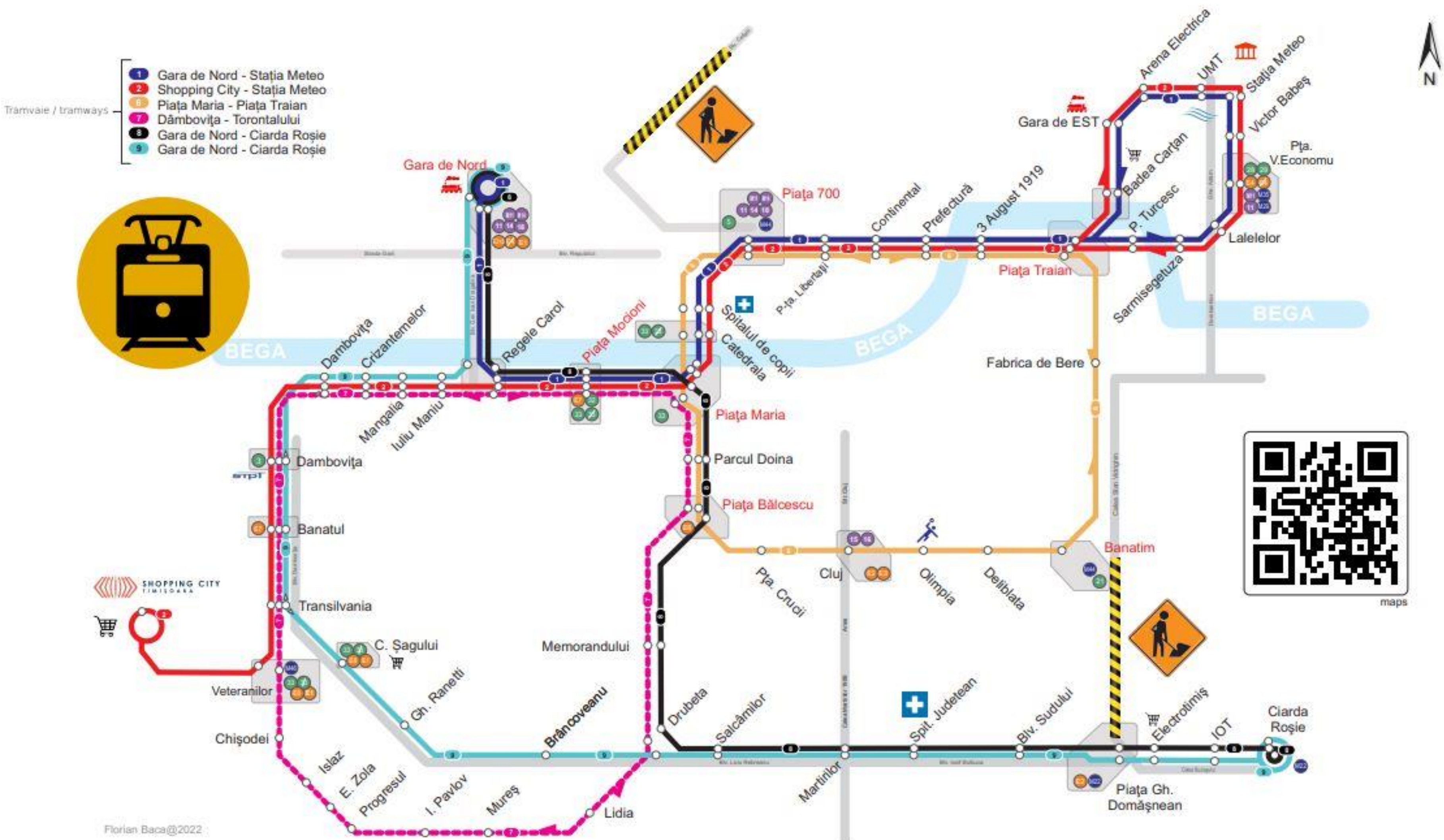
Imagine: http://en.wikipedia.org/wiki/File:6n_graf.svg

Ce e un graf?

Mulțimea nodurilor trebuie să fie o mulțime *finită* și *nevidă*, deci nu e posibil să avem un graf fără noduri, dar e posibil să avem un graf fără muchii.

Așadar, un graf poate fi reprezentat sub forma unei *figuri geometrice* alcătuite din *puncte* (care corespund vârfurilor/nodurilor) și din *linii* drepte sau curbe care unesc aceste puncte (care corespund muchiilor sau arcelor).

Harta traseelor de tramvai din Timișoara



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

Grafuri – noțiuni generale

Se numește *ordin al unui graf* numărul de noduri al grafului.

Un nod *v* este *incident* cu o muchie *r* dacă muchia r atinge nodul v - $v \in r$.

Două noduri se numesc *adiacente* dacă există o muchie care le unește.

Două muchii sunt *adiacente* dacă există un nod care să fie incident cu ambele muchii.

Grafuri – noțiuni generale

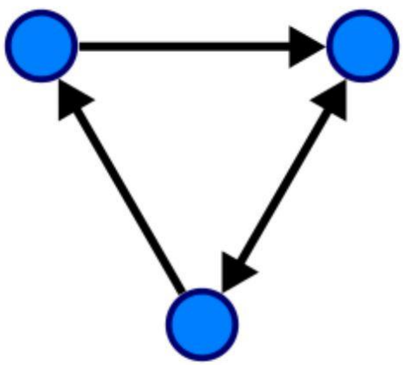
Se numește grad al unui nod, *n*umărul de muchii care sunt incidente la acel nod.

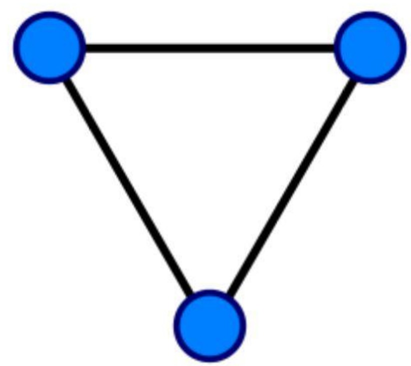
Dacă se adună *gradele tuturor nodurilor* din graful G se obține de două ori numărul de muchii.

Grafuri orientate și neorientate

Un graf e *orientat* dacă muchiile sale sunt perechi *ordonate*

Un graf e *neorientat* dacă muchiile sale sunt perechi *neordonate* (nu contează sensul parcurgerii)





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

Grafuri și relații

Mulțimea muchiilor unui graf formează o *relație* $E \in V \times V$ pe mulțimea nodurilor.

Un graf *neorientat* poate fi reprezentat printr-o relație *simetrică:*

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

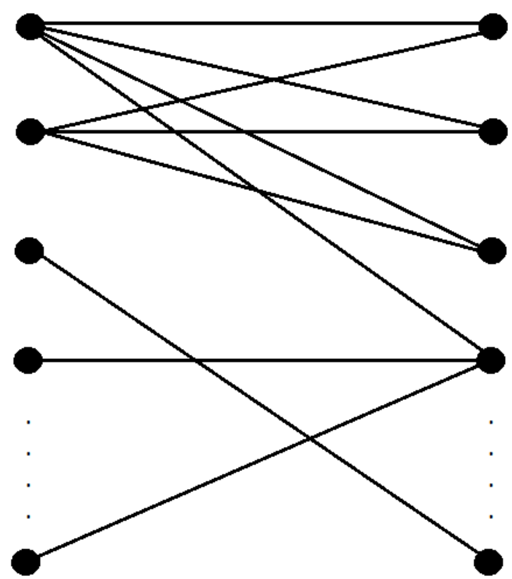
Într-un graf *orientat*, E e o relație oarecare (nu trebuie să fie simetrică, dar poate fi)

Reciproc, *orice relație binară* poate fi văzută ca un *graf orientat* pentru $(u, v) \in E$ introducem o muchie $u \rightarrow v$

O provocare în sociologie

Să revenim la *problema din sociologie*. Cum putem reprezenta cu grafuri problema partenerilor?

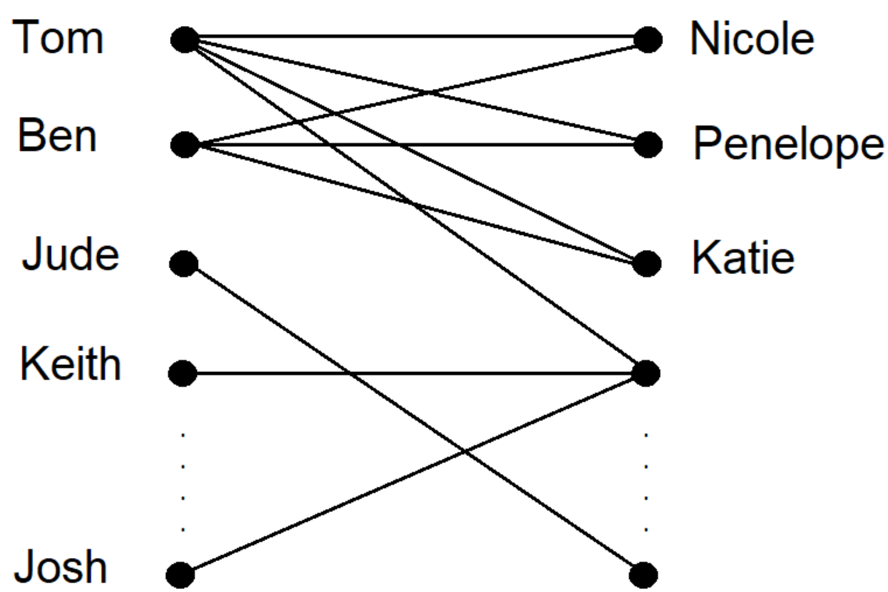
Dacă reprezentăm *mulțimea bărbaților* (mai jos în stânga) și *mulțimea femeilor* (mai jos în dreapta), putem reprezenta un astfel de graf:



O provocare în sociologie

Să revenim la *problema din sociologie*. Cum putem reprezenta cu grafuri problema partenerilor?

Dacă reprezentăm *mulțimea bărbaților* (mai jos în stânga) și *mulțimea femeilor* (mai jos în dreapta), putem reprezenta un astfel de graf:



O provocare în sociologie

În *România*, numărul de noduri (persoane) ar fi 19.186.201 (la 1 ianuarie 2021) conform datelor de la Institutul Național de Statistică, dintre care aproximativ 9,34 milioane bărbați și 9,84 milioane femei.

Putem ști *numărul de muchii* al acestui graf? Nu, dar noi am avea de calculat raportul dintre media gradurilor nodurilor bărbați și media gradurilor nodurilor femei:

$$\mathrm{R} = \frac{M_{barbati}}{M_{femei}}$$

R = 1,74 conform studiului realizat de *Universitatea din Chicago*

R = 3,33 conform studiului realizat de *ABC News*

O provocare în sociologie

$$M_{barbati} = \frac{Nr.total\ Muchii}{Nr.noduri\ bărbați}, \qquad M_{femei} = \frac{Nr.total\ Muchii}{Nr.noduri\ femei}$$

O provocare în sociologie

$$M_{barbati} = \frac{Nr.total\ Muchii}{Nr.noduri\ bărbați}, \qquad M_{femei} = \frac{Nr.total\ Muchii}{Nr.noduri\ femei}$$

$$\mathrm{R} = \frac{M_{barbati}}{M_{femei}}$$

O provocare în sociologie

$$M_{barbati} = \frac{Nr.total\ Muchii}{Nr.noduri\ bărbați}, \qquad M_{femei} = \frac{Nr.total\ Muchii}{Nr.noduri\ femei}$$

$$\mathrm{R} = \frac{M_{barbati}}{M_{femei}} = \frac{Nr.total\ Muchii}{Nr.noduri\ bărbați} \bigg/ \frac{Nr.total\ Muchii}{Nr.noduri\ femei}$$

O provocare în sociologie

$$M_{barbati} = \frac{Nr.total\ Muchii}{Nr.noduri\ bărbați}, \qquad M_{femei} = \frac{Nr.total\ Muchii}{Nr.noduri\ femei}$$

$$\mathrm{R} = \frac{M_{barbati}}{M_{femei}} = \frac{Nr.total\ Muchii}{Nr.noduri\ bărbați} \Big/ \frac{Nr.total\ Muchii}{Nr.noduri\ femei}$$

$$= \frac{Nr.total\ Muchii}{Nr.noduri\ bărbați} * \frac{Nr.noduri\ femei}{Nr.total\ Muchii} = \frac{Nr.noduri\ femei}{Nr.noduri\ bărbați}$$

O provocare în sociologie

$$M_{barbati} = \frac{Nr.total\ Muchii}{Nr.noduri\ bărbați}, \qquad M_{femei} = \frac{Nr.total\ Muchii}{Nr.noduri\ femei}$$

$$\mathrm{R} = \frac{M_{barbati}}{M_{femei}} = \frac{Nr.total\ Muchii}{Nr.noduri\ bărbați} \Big/ \frac{Nr.total\ Muchii}{Nr.noduri\ femei}$$

$$= \frac{Nr.total\ Muchii}{Nr.noduri\ bărbați} * \frac{Nr.noduri\ femei}{Nr.total\ Muchii} = \frac{Nr.noduri\ femei}{Nr.noduri\ bărbați}$$

$$= \frac{9{,}84\ mil}{9{,}34\ mil} = 1{,}05$$

O provocare în sociologie

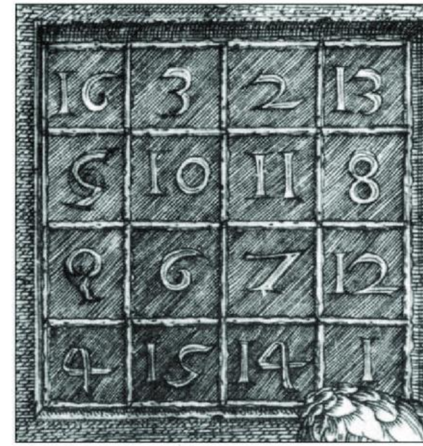
$$M_{barbati} = \frac{Nr.total\ Muchii}{Nr.noduri\ bărbați}, \qquad M_{femei} = \frac{Nr.total\ Muchii}{Nr.noduri\ femei}$$

$$\text{R} = \frac{M_{barbati}}{M_{femei}} = \frac{Nr.total\ Muchii}{Nr.noduri\ bărbați} \Big/ \frac{Nr.total\ Muchii}{Nr.noduri\ femei}$$

$$= \frac{Nr.total\ Muchii}{Nr.noduri\ bărbați} * \frac{Nr.noduri\ femei}{Nr.total\ Muchii} = \frac{Nr.noduri\ femei}{Nr.noduri\ bărbați}$$

$$= \frac{9{,}84\ mil}{9{,}34\ mil} = 1{,}05$$

Deci am demonstrat matematic , cu ajutorul teoriei grafurilor, că în România numărul de relații pe care îl au bărbații cu partenere de sex opus este *cu doar 5% mai mare* decât numărul de relații pe care îl au femeile cu parteneri de sex opus.



Teoria grafurilor

Ce e un graf?

Drumuri și cicluri în graf

Reprezentarea și parcurgerea grafurilor

Grafuri în PYTHON

Exerciții cu grafuri în PYTHON

Drumuri în graf

Un *drum* (o cale) într-un graf e o *secvență de muchii* care leagă o *secvență de noduri* $x_0, \ldots x_n$ cu $n \geq 0$ astfel ca $(x_i, x_{i+1}) \in E$ pentru orice $i < n$.

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

Putem defini un drum atât în grafuri orientate cât și neorientate.

Un drum are un *nod inițial* x_0 și un *nod final* x_n.

Lungimea unui drum e numărul de muchii parcurse.
în particular, poate fi zero (un nod x_0, fără niciun fel de muchii)

Cicluri în graf

Un *ciclu* e un drum de *lungime nenulă* în care nodurile de început și sfârșit sunt identice (aceleași).

Adeseori, lucrăm cu *cicluri simple*:

- cicluri în care muchiile și nodurile *nu apar de mai multe ori* (cu excepția nodului inițial care e și cel final).

Grafuri și componente conexe

Un graf e *conex* dacă are un drum *de la orice nod la orice nod*. (definiție generală, depinde de noțiunea de *drum* – în graf orientat sau neorientat)

Pentru grafuri *neorientate*:

O *componentă conexă* e un subgraf conex maximal.

- deci are un drum între oricare două noduri
- nu s-ar mai putea adăuga alte noduri păstrând-o conexă

Un graf cu n noduri și e muchii are un număr de componente conexe $\geq n - e$. Se poate demonstra prin inducție.

Grafuri orientate: slab conexe și tare conexe

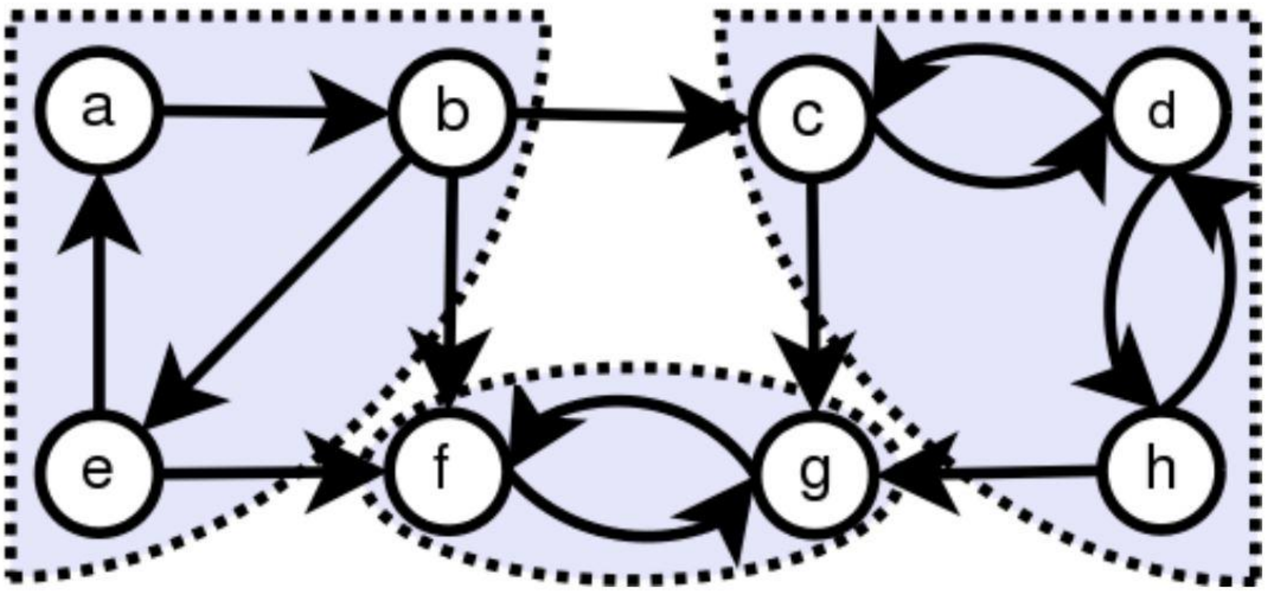
Un graf *orientat* e *slab conex* dacă are un drum *neorientat* de la orice nod la orice nod, și *tare conex* dacă are un drum *orientat* de la orice nod la orice nod.

O *componentă tare conexă* e un *subgraf* tare conex maximal. Componentele tare conexe sunt *disjuncte*:

- *R*(*u, v*) : *drum*(*u, v*) și *drum*(*v, u*) e o *relație de echivalență*, și componentele tare conexe sunt *clase de echivalență*

Grafuri orientate: slab conexe și tare conexe

Graful orientat din figură e *slab conex*. Are trei componente *tare conexe*.



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

Determinarea componentelor conexe (graf neorientat)

Componentele conexe sunt *clase de echivalență*

- orice nod e în componenta proprie - *reflexivitate*
- un drum de la *u* la *v* e și drum de la *v* la *u* - *simetrie*
- *drum*(*u*, *v*) ∧ *drum*(*v*, *w*) → *drum*(*u*, *w*) - *tranzitivitate*

Determinăm componentele conexe parcurgând muchiile grafului:

- inițial, fiecare nod e în propria componentă
- pentru o muchie (*u*, *v*) *unim* componentele lui *u* și *v*

Drumuri Euleriene (în grafuri neorientate)

Gradul unui nod (într-un graf neorientat) e numărul de muchii care ating nodul.

Un *drum eulerian* e un *drum* care conține *toate* muchiile unui graf exact o dată.

Un *ciclu eulerian* e un *ciclu* care conține *toate* muchiile unui graf exact o dată.

Drumuri Euleriene (în grafuri neorientate)

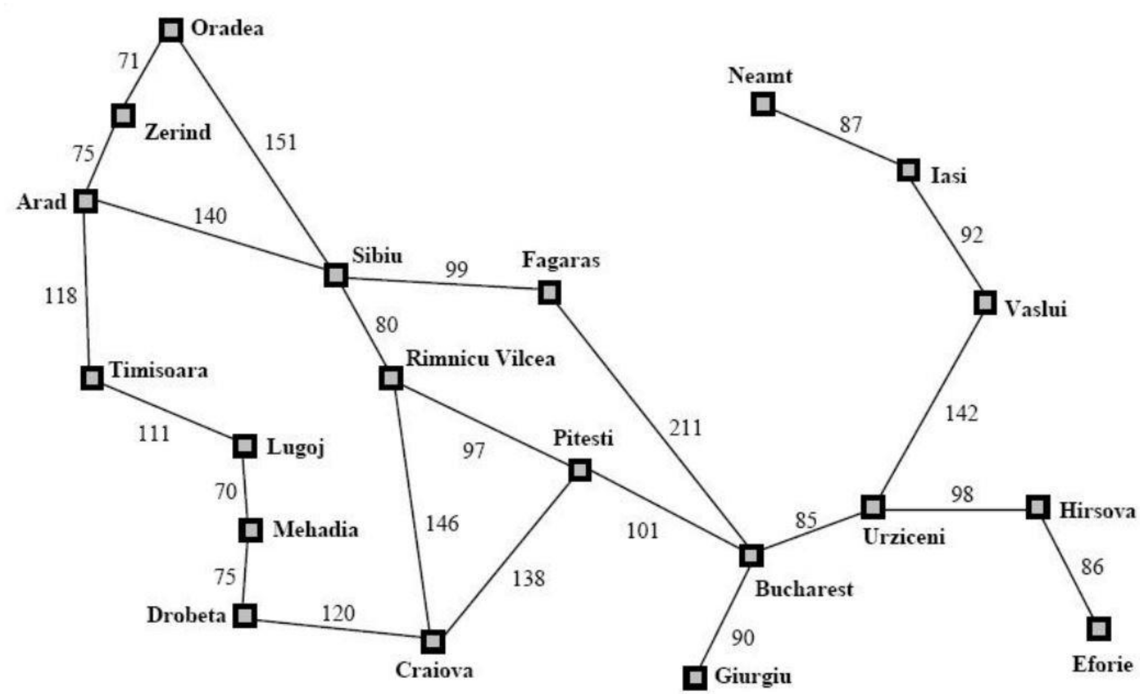
Un graf conex neorientat are un *ciclu eulerian* dacă și numai dacă *toate nodurile au grad par*.

Un graf conex neorientat are un *drum* (dar nu și un ciclu) *eulerian* dacă și numai dacă *exact două noduri au grad impar*.

(primul și ultimul nod din drum)

Exemple: hărțile ca și grafuri ponderate

Graf ponderat: fiecare muchie are asociată o valoare numerică numită *cost* (poate reprezenta lungime, capacitate, etc.)

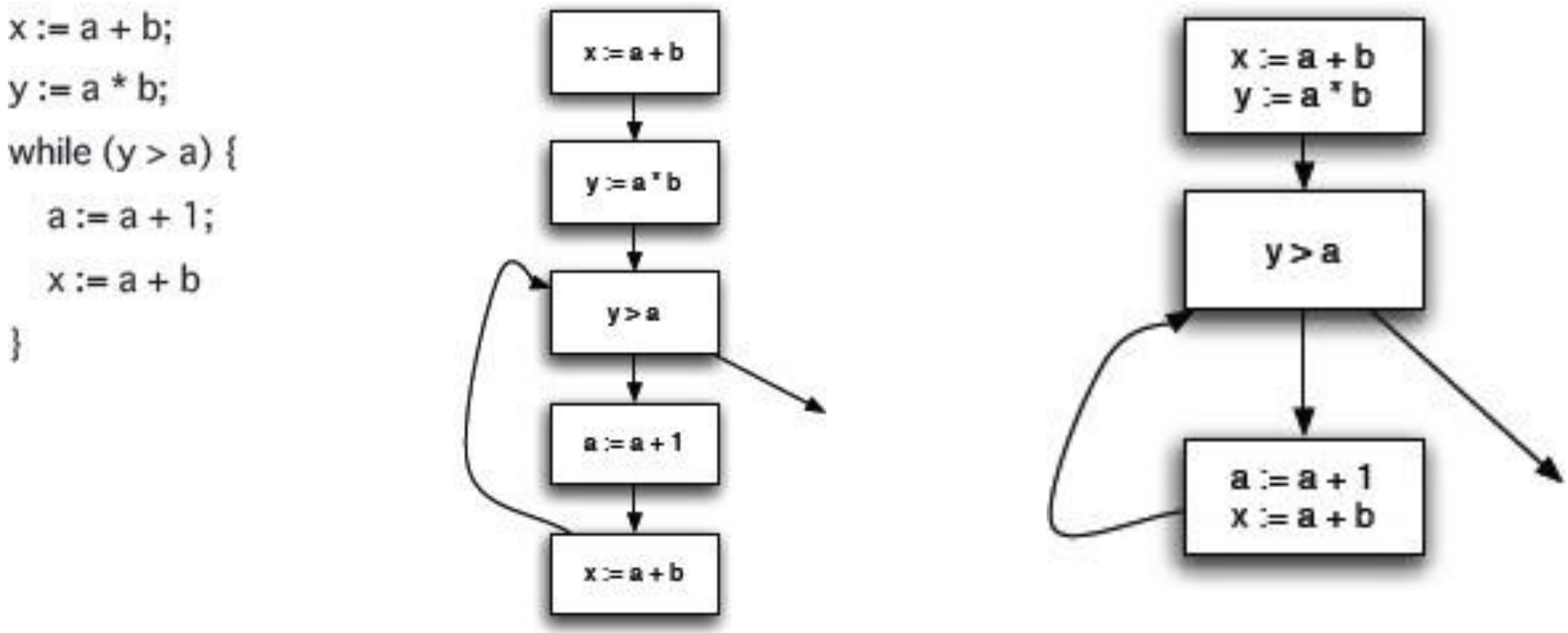


Hartă (inexactă) din Russell & Norvig, Introduction to AI

Exemple: Graful fluxului de control (control flow graph)

Reprezentarea programelor în compilatoare, analizoare de cod, etc.

- nodurile: *instrucțiuni* sau *secvențe liniare* de instrucțiuni (*basic blocks*)
- muchiile: descriu secvențierea instrucțiunilor (*fluxul de control*)



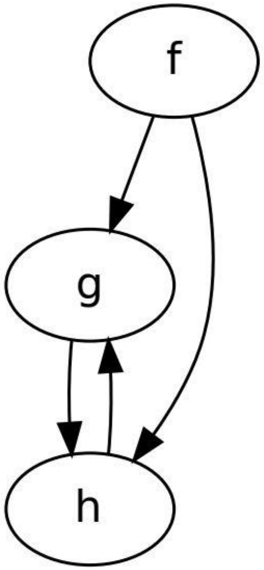
http://vinaytech.wordpress.com/2008/10/04/abstract-syntax-tree/

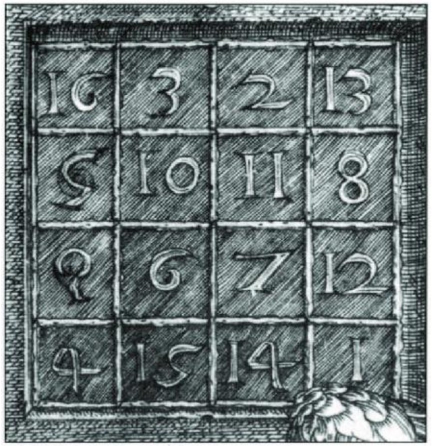
Exemple: Graful de apel al funcțiilor (call graph)

Introducem o muchie $f \rightarrow g$ dacă funcția f apelează pe g

Graful de apel e ciclic dacă există funcții (direct sau indirect) recursive

```
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)
```





Teoria grafurilor

Ce e un graf?

Drumuri și cicluri în graf

Reprezentarea și parcurgerea grafurilor

Grafuri în PYTHON

Exerciții cu grafuri în PYTHON

Reprezentarea grafurilor

Dacă identificăm nodurile prin numere (consecutive), putem reprezenta graful ca *matrice de adiacență* pătratică

M[i,j] = 1 dacă *există* muchie de la i la j

M[i,j] = 0 dacă *nu există* muchie de la i la j

sau M[i,j] poate conține lungimea/costul muchiei (graf ponderat)

Reprezentarea grafurilor

Reprezentarea prin *liste de adiacență*

- pentru fiecare nod *u*: lista/mulțimea nodurilor *v* cu muchii (*u, v*)

Putem păstra informația într-un *dicționar*:

- cheia din dicționar = nodul din graf
- valoarea din dicționar = lista/mulțimea nodurilor adiacente

Reprezentarea grafurilor

Reprezentarea prin *liste de perechi*

- pentru fiecare muchie de la u la v vom reține în listă/mulțime perechea (*u, v)*

Parcurgerea în adâncime (depth-first)

Parcurgerea grafului în adâncime e o traversare în *preordine*.

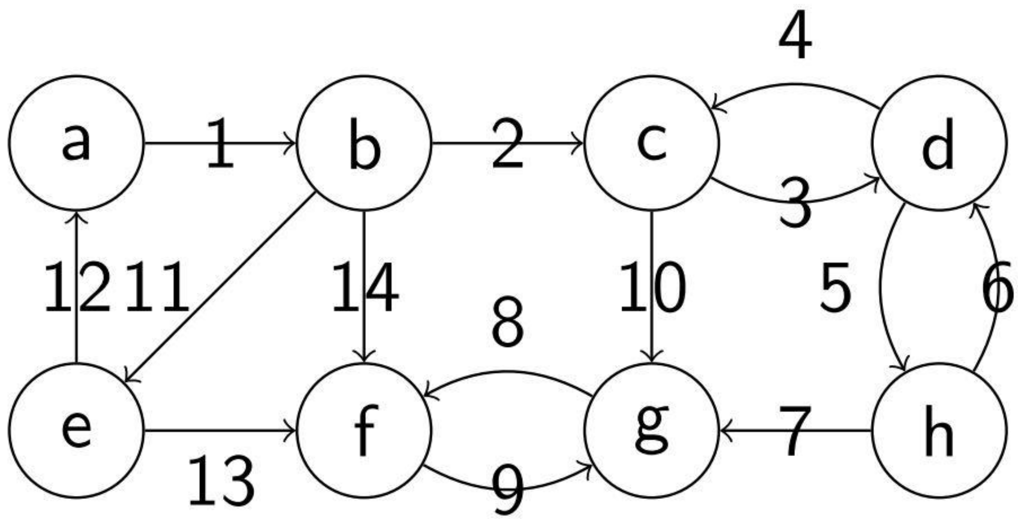
După vizitarea nodului se parcurg (recursiv) toți vecinii (dacă nu au fost vizitați încă)

Acționează ca și cum vecinii ar fi introduși într-o *stivă.*

Parcurgerea în adâncime (depth-first)

Fie graful de mai jos, cu listele de adiacență ordonate după litere.

Ordinea muchiilor parcurse de la *a* în adâncime e cea indicată:



Se poate programa: funcție recursivă, acumulând mulțimea nodurilor vizitate

Parcurgerea prin cuprindere (breadth-first)

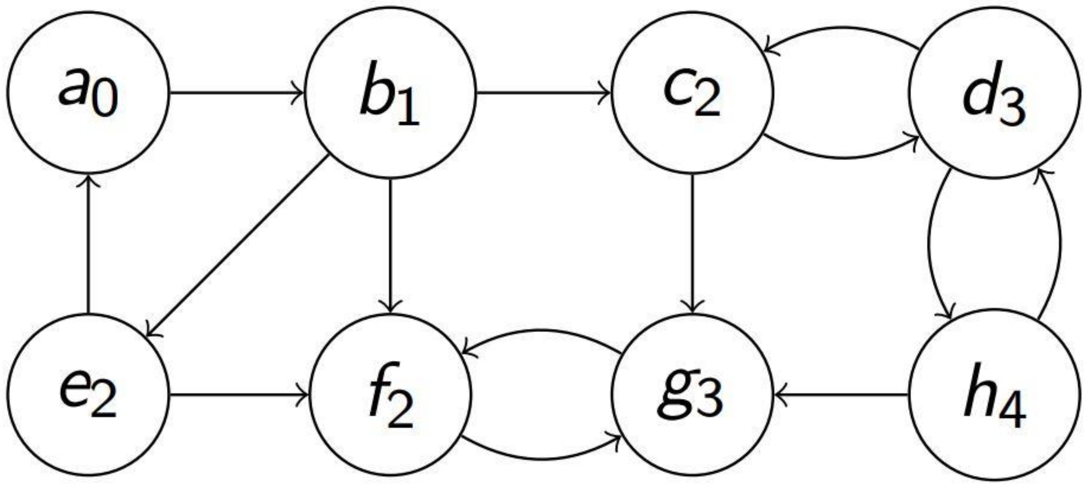
Parcurgerea prin cuprindere vizitează nodurile în ordinea *distanței minime* de nodul de plecare

(în “valuri” care se depărtează de la nodul de pornire)

Nodurile încă nevizitate se pun într-o *coadă*.

Parcurgerea prin cuprindere (breadth-first)

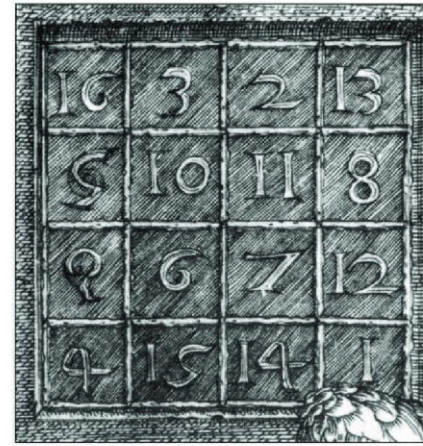
In figura de mai jos, se indică distanța minimă de la nodul a (nodurile cu distanță mai mare sunt parcurse mai târziu)



O implementare: funcție recursivă,

acumulând: mulțimea tuturor nodurilor vizitate

frontiera: mulțimea nodurilor noi atinse în runda curentă



Teoria grafurilor

Ce e un graf?

Drumuri și cicluri în graf

Reprezentarea și parcurgerea grafurilor

Grafuri în PYTHON

Exerciții cu grafuri în PYTHON

Grafuri în PYTHON

În PYTHON putem reprezenta un graf cu ajutorul unui dicționar

Dacă avem graful G = (V, E), V = {a, b, c, d, e}, E = {ab, ac, bd, cd, de}

```
graf = {
   "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'}}
```

Afișarea nodurilor unui graf

Pentru a afișa nodurile unui graf reținut cu un dicționar e necesar să afișăm cheile dicționaru lui.

```
graf = {
   "a" : {"b","c"},
   "b" : {"a", "d"},
   "c" : {"a", "d"},
   "d" : {"e"},
   "e" : {"d"}
}
```

Afișarea nodurilor unui graf

Pentru a afișa nodurile unui graf reținut cu un dicționar e necesar să afișăm cheile dicționaru lui.

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

Afișarea muchiilor unui graf

print(afisare_muchii(graf))

 # {('a', 'c'), ('d', 'e'), ('a', 'b'), ('e', 'd'), ('b', 'a'), ('b', 'd'), ('c', 'a'), ('c', 'd')}

Afișarea muchiilor unui graf

```
import functools

def afisare_muchii(graf, muchii = set()):
   def functie(acc,elem):
      cheie, valoare = elem
      def f_multime(acc2,elem2):
         muchii.add((cheie,elem2))
      functools.reduce(f_multime, valoare, 0)
   functools.reduce(functie, graf.items(), 0)
   return muchii

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

Adaugarea unui nod nou

```
graf = {
   "a" : {"b","c"},
   "b" : {"a", "d"},
   "c" : {"a", "d"},
   "d" : {"e"},
   "e" : {"d"}
}
```

```
print(adaugare_nod(graf, "f"))                    # {'a': {'c', 'b'}, 'b': {'d', 'a'}, 'c': {'d', 'a'}, 'd': {'e'}, 'e': {'d'}, 'f': set()}
```

Adaugarea unui nod nou

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

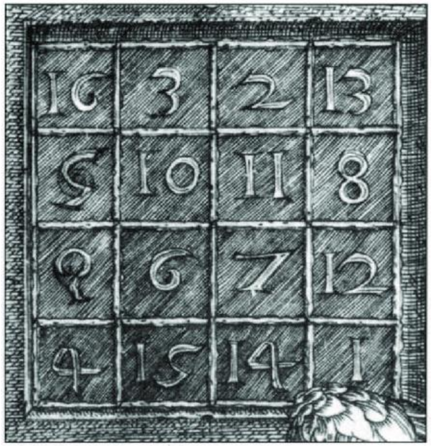
Adaugarea unei muchii noi

```
print(adaugare_muchie_orientat(graf,("a","d")))
print(adaugare_muchie_orientat(graf,("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()}
```

Adaugarea unei muchii noi

```
def adaugare_muchie_orientat(graf, muchie):
    if (muchie[0] in graf):
        graf[muchie[0]].add(muchie[1])
    else:
        graf[muchie[0]]={muchie[1]}
    if (not muchie[1] in graf):
        graf[muchie[1]] = set()
    return graf

print(adaugare_muchie_orientat(graf,("a","d")))
print(adaugare_muchie_orientat(graf,("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()}
```



Teoria grafurilor

Ce e un graf?

Drumuri și cicluri în graf

Reprezentarea și parcurgerea grafurilor

Grafuri în PYTHON

Exerciții cu grafuri în PYTHON

Exerciții

1. Fie un graf reprezentat de mulțimea perechilor de noduri adiacente. Să se creeze structura de date care reține informațiile despre graf într-un dicționar.

Exemplu:

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

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

Exerciții

```python
import functools
def constructie_graf(multime, dictionar = {}):
    def functie(acc, elem):
        if (elem[0] in dictionar):
            dictionar[elem[0]].add(elem[1])
        else:
            dictionar[elem[0]] = set({elem[1]})
        if(not elem[1] in dictionar):
            dictionar[elem[1]] = set()
    functools.reduce(functie, multime, 0)
    return dictionar
print(constructie_graf({(1, 3), (1, 2), (2, 4), (4, 1)}))
```

Vă mulțumesc!

Bibliografie

- Problema raportului dintre numărul de relații pe care bărbații îl au cu partenere de sex și numărul de relații pe care femeile îl au cu parteneri de sex opus a fost preluată din cursul ***Mathematics for Computer Science*** de la Massachusetts Institute of Technology (de pe https://ocw.mit.edu/)
- Conținutul cursului se bazează preponderent pe materialele de anii trecuți de la cursul de LSD, predat de conf. dr. ing.Marius Minea și ș.l. dr. ing. Casandra Holotescu (http://staff.cs.upt.ro/~marius/curs/lsd/index.html)