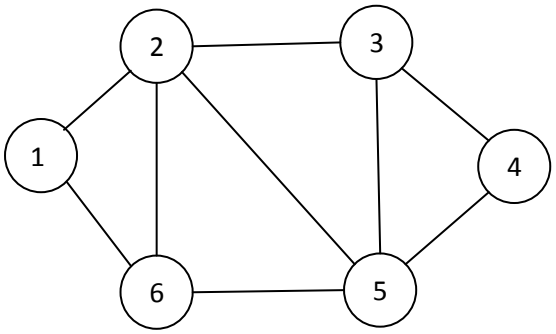
Grafy

Graf je sústava bodov (tzv. uzlov, vrcholov), medzi ktorými vedú spojnice (hrany grafu).
Uvažujeme len konečné grafy. Graf je určený počtom vrcholov grafu a dvojicami vrcholov, ktoré sú spojené hranou. Grafy môžu byť **neorientované** a **orientované**. V orientovaných grafoch majú hrany priradenú orientáciu, t.j. z ktorého do ktorého vrcholu vedú (v nakreslenom grafe vyznačujeme šípkou – vektorom). Grafy môžu byť **neohodnotené** a **ohodnotené**. V ohodnotenom grafe je každej hrane priradené tzv. ohodnotenie, čo je väčšinou jedno číslo, napr. dĺžka hrany. **Cestou** z vrcholu X do vrcholu Y rozumieme takú postupnosť hrán h_1, h_2, ..., h_n, že hrana h_1 vychádza z vrcholu X, vedie do vrcholu, z ktorého vychádza hrana h_2, ... až hrana h_n vedie do vrcholu Y. Cestami sa zaoberať nebudeme.

Reprezentácia grafu v programe

Máme graf s N vrcholmi a M hranami (N, M konštanty). Vrcholy grafu sú označené číslami od 1 po N, prípadne hrany číslami od 1 po M.

Príklad neorientovaného neohodnoteného grafu (počet vrcholov N = 6, počet hrán M = 9)



Matica susednosti
umožňuje reprezentovať neohodnotené neorientované aj orientované grafy. Priesečník riadka – jeden vrchol a stĺpca – druhý vrchol obsahuje informáciu, či sú vrcholy spojené hranou. Ide o štvorcovú maticu veľkosti N x N s hodnotami False/True alebo 0/1. Pri neorientovaných grafoch je matica susednosti symetrická.

Úloha G.1.1

Vytvorte reprezentáciu vyššie zobrazeného grafu pomocou matice susednosti a funkciu na zistenie, či medzi zadanými vrcholmi existuje hrana.

Analýza

Matica susednosti má tvar

0	1	0	0	0	1	(z 1 vedie hrana do 2 a 6)
1	0	1	0	1	1	(z 2 vedie hrana do 1, 3, 5 a 6)
0	1	0	1	1	0	(z 3 vedie hrana do 2, 4 a 5)
0	0	1	0	1	0	(zo 4 vedie hrana do 3 a 5)
0	1	1	1	0	1	(z 5 vedie hrana do 1, 3, 4 a 6)
1	1	0	0	1	0	(zo 6 vedie hrana do 1, 2, a 5)

Kódovanie hrán číslami 0 a 1 sme zvolili kvôli tomu, že hodnoty True a False nemožno priamo čítať z textového súboru, ak by bol graf uložený v textovom súbore vo forme tabuľky.

Ako sa matica susednosti zapísaná v textovom súbore dostane do dvojrozmerného poľa je uvedené v študijnom texte Dvojrozmerné pole. Zistiť, či medzi zadanými vrcholmi existuje hrana znamená zistiť, či v priesečníku zadaného riadka (odkiaľ) a stĺpca (kam) je hodnota 0 alebo 1.

Poznámka
Ak sú hodnoty matice uložené v textovom súbore, väčšina našich jednoduchých úloh by sa dala riešiť aj bez načítania hodnôt z textového súboru do dvojrozmerného poľa. Principiálne pri grafových algoritmoch budeme používať dvojrozmerné pole.

Riešenie

```
import dvojpole

def nacitajZoSuboruDoDvojPola(nazov_suboru):
    with open(nazov_suboru,"r") as f:
        return [[int(riadok.split()[s]) for s in range(len(riadok.split()))] for riadok in f]

def existzAdoB(graf, odkial, kam):
    return graf[odkial][kam]

#=============================================
graf = nacitajZoSuboruDoDvojPola("graf.txt")

for i in range(len(graf)):                          # „skrášlenie“ výpisu uvedením čísla vrcholu
    print("{:3}.".format(i+1), end="")
print()
dvojpole.vypisDvojPole(graf)

A = int(input("Či existuje hrana z vrcholu: "))
B = int(input("Či existuje hrana do vrcholu: "))
if existzAdoB(graf, A-1, B-1):
    print("existuje")
else:
    print("neexistuje")
```

Úloha G.1.2

Úlohu G.1.1 doplňte o funkciu na zistenie, či v danej reprezentácii grafu existujú aspoň dva uzly, medzi ktorými je „jednosmerne“ orientovaná hrana (šípka na hrane je len jedným smerom, pri grafe znázorňujúcom cestnú sieť by to bola jednosmerná cesta medzi zadanými mestami - vrcholmi).

Analýza

Úloha vlastne vyžaduje zistiť, či, ak existuje hrana z vrcholu A do vrcholu B, existuje aj hrana z vrcholu B do vrcholu A, alebo konkrétne: všade v tabuľke, kde je na pozícii graf[i][j] hodnota 1, musí byť aj na pozícii graf[j][i] hodnota 1. Vtedy sú všetky hrany obojsmerné a jednosmerne orientovaná hrana v grafe neexistuje. Ak existuje v reprezentácii dvojica i,j pre ktorú graf[i][j] ≠ graf[j][i], odpoveď v úlohe G.1.2 je „existuje jednosmerne orientovaná hrana“.

Riešenie (ešte sa k problému vrátime v nasledujúcej úlohe)

```
def existJednosmernaHrana(graf):
    N = len(graf)
    for i in range(N):
        for j in range(N):
            if graf[i][j] != graf[j][i]:
                return True
    return False
```

Použitie

```
if existJednosmernaHrana(graf):
    print('V grafe existuje "jednosmerná" hrana!')
else:
    print('V grafe neexistuje "jednosmerná" hrana!')
```

Matica vzdialeností

je analógiou matice susednosti s tým rozdielom, že v matici sú uložené ohodnotenia jednotlivých hrán. Ak hrana medzi príslušnými vrcholmi neexistuje, jej ohodnotenie je 0 (nula). Matica vzdialeností reprezentuje ohodnotené grafy.

V nasledujúcich úlohách budeme predpokladať, že graf reprezentuje cestnú sieť, t.j., že vrcholy grafu sú mestá a hrany sú cesty medzi mestami. Pod cestou rozumieme hranu grafu (nie postupnosť hrán!).

Úloha G.2

V matici vzdialeností (označenej MV) reprezentovanej dvojrozmerným zoznamom csiet je zaznamenaná cestná sieť v km. Vytvorte funkciu, ktorá

a) skontroluje, či sú všetky cesty obojsmerné;
   v nasledujúcich úlohách b) až e) môžete predpokladať, že všetky cesty sú obojsmerné
b) vypíše celkovú dĺžku cestnej siete
c) vypíše dĺžku najdlhšej spojnice (hrany) a ktoré mestá spája
d) vypíše mesto, z ktorého vychádza najviac spojníc
e) vypíše mesto, z ktorého vychádza najviac spojníc, ak existuje viacej takýchto miest, vypíše to, pre ktoré je súčet vzdialeností vychádzajúcich ciest najväčší. Ak má úloha viac riešení, stačí vypísať ľubovoľné z nich.

```
csiet = [[0,15,27,0,0],
         [15,0,34,2,25],
         [27,34,0,9,12],
         [0,2,9,0,76],
         [0,25,12,76,0]]
```

Analýza

Matica MV zodpovedajúca zoznamu csiet					Odpovede
0	15	27	0	0	a) všetky cesty sú obojsmerné
15	0	34	2	25	b) 200
27	34	0	9	12	c) 76, mestá 4 - 5
0	2	9	0	76	d) mesto 2 alebo 3, počet spojníc 4
0	25	12	76	0	e) mesto 3, počet spojníc 4, dĺžka 82

Zistiť, či sú všetky cesty obojsmerné (a) znamená zistiť, či je matica symetrická podľa hlavnej diagonály, t.j. či sa MV[r,s] = MV[s,r] pre všetky riadky a stĺpce matice. Zrejme stačí zobrať indexy z podfarbeného „trojuholníka“, t.j. od druhého po posledný riadok a v každom riadku hodnoty len k hlavnej diagonále (konfrontuj s riešením G.1.2).
Zistiť celkovú dĺžku cestnej siete (b) znamená sčítať všetky hodnoty v tabuľke a vydeliť ich dvoma, alebo opäť sčítať len hodnoty z podfarbeného „trojuholníka“ (ak sú všetky cesty obojsmerné!).
Zistiť najdlhšiu spojnicu a ktoré mestá spája (c) znamená nájsť najväčšiu hodnotu v matici a zistiť, v ktorom je riadku a stĺpci.
Zistiť mesto, z ktorého vychádza najviac spojníc (d), znamená nájsť riadok, v ktorom je najviac nenulových hodnôt. Ak existuje takýchto riadkov viacej (e), treba zistiť súčet týchto hodnôt a vypísať riadok, v ktorom je súčet najväčší.

Riešenie

a) sú všetky cesty obojsmerné?

```
def suVsetkyObojsmerne(graf):
    for i in range(1,len(graf)):
        for j in range(i):
            if graf[i][j] != graf[j][i]:
                return False
    return True

if suVsetkyObojsmerne(csiet):
    print("Všetky cesty sú obojsmerné!")
else:
    print("Všetky cesty nie sú obojsmerné!")
```

b) celková dĺžka cestnej siete (všetky cesty sú obojsmerné)

```
def vratDlzkuCestnejSiete(graf):
    ''' predpoklad: všetky cesty sú obojsmerné '''
    sucet = 0
    for i in range(1,len(graf)):
        for j in range(i):
            sucet += graf[i][j]
    return sucet
```

```
print("Celková dĺžka cestnej siete je {:.2f}
km".format(vratDlzkuCestnejSiete(csiet)))
```

*Úlohu b) riešte aj pre prípad, že nie sú všetky cesty obojsmerné (pozri riešenie úlohy G.7).

c) dĺžka najdlhšej cesty (spojnice) a ktoré mestá spája - jedno z možných riešení:

```
def vypisNajdlhsiuMedziMestami(graf):
    najmax = -1
    riad = 0
    for riadok in graf:
        riad += 1
        riad_max = max(riadok)
        if riad_max > najmax:
            najmax = riad_max
            z = riad
            do = riadok.index(najmax)+1

    print("Najdlhšia spojnica má {} km".format(najmax))
    print("a je medzi mestami {} a {}.".format(z,do))

vypisNajdlhsiuMedziMestami(csiet)
```

Výpis pre csiet:

```
Najdlhšia spojnica má 76 km
a je medzi mestami 4 a 5.
```

Poznámka
Riešenie nezisťuje, či neexistuje viac miest, medzi ktorými sa spojnica rovná najdlhšej. Môžete ošetriť aj tento prípad, t.j. nech vypíše všetky dvojice miest, medzi ktorými má spojnica dĺžku najdlhšej spojnice (pomôcka je na konci tohto študijného textu).

d) vypíše mesto, z ktorého vychádza najviac spojníc

```
def vypisMestoSnajviacSpojnicami(graf):
    najpocet = -1
    riad = 0
    for riadok in graf:
        riad += 1
        riad_pocet = len(riadok) - riadok.count(0)
        if riad_pocet > najpocet:
            najpocet = riad_pocet
            najmesto = riad

    print("Najviac ciest ({}) vychádza z mesta {}.".format(najpocet, najmesto))

vypisMestoSnajviacSpojnicami(csiet)
```

Výpis pre csiet:

```
Najviac ciest (4) vychádza z mesta 2.
```

Riešenie študenta Patrika Šimanského:

```
def vypisMestoSnajviacSpojnicamiSiman(graf):
    N = len(graf)
    najpocet = -1
    for i in range(N):
        riad_pocet = N - graf[i].count(0)
        if riad_pocet > najpocet:
            najpocet = riad_pocet
            najmesto = i + 1

    print("Najviac ciest podľa Šimanského ({}) vychádza z mesta {}.".format(najpocet, najmesto))

vypisMestoSnajviacSpojnicamiSiman(csiet)
```

Poznámka
Štyri cesty vychádzajú aj z mesta 3. Upravte program tak, aby vypísal všetky mestá, z ktorých vychádza najviac ciest.

e) vypíše mesto, z ktorého vychádza najviac spojníc, ak existuje viacej takýchto miest, vypíše to, pre ktoré je súčet vzdialeností vychádzajúcich ciest najväčší.

```
def vypisMestoSnajPoctomAsuctom(graf):
    najpocet, najsucet = -1, -1
    riad = 0
    for riadok in graf:
        riad += 1
        riad_sucet = sum(riadok)
        riad_pocet = len(riadok) - riadok.count(0)
        if riad_pocet > najpocet:
            najpocet = riad_pocet
            najsucet = riad_sucet
            najmesto = riad

        elif riad_pocet == najpocet and riad_sucet > najsucet:
            najsucet = riad_sucet
            najmesto = riad

    print("Najviac ciest ({}) vychádza z mesta {}".format(najpocet, najmesto))
    print("so súčtom vzdialeností {} km.".format(najsucet))

vypisMestoSnajPoctomAsuctom(csiet)
```

Výpis pre csiet:

```
Najviac ciest (4) vychádza z mesta 3
so súčtom vzdialeností 82 km.
```

*Riešenie „poznámkovej“ úlohy c)

```
def vypisNajdlhsiuMedziMestamiHviezdicka(graf):
    najmax = -1
    najmesta = []
    riad = 0
    for riadok in graf:
        riad += 1
        riad_max = max(riadok)
        if riad_max > najmax:
            najmax = riad_max
            najmesta = [(riad, riadok.index(najmax)+1)]
        elif riad_max == najmax:
            najmesta.append((riad,riadok.index(najmax)+1))

    print("Najdlhšia spojnica má {} km".format(najmax))
    print("a je medzi mestami ")
    for z_do in najmesta:
        print(z_do[0]," - ",z_do[1])

vypisNajdlhsiuMedziMestamiHviezdicka(csiet)
```

Výpis pre csiet:

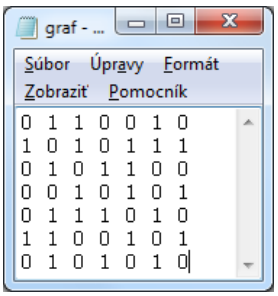
```
Najdlhšia spojnica má 76 km
a je medzi mestami
4  -  5
5  -  4
```

Prečo uvedené riešenie nie je dobré? (Pomôcka: v zozname csiet zmeňte nuly pre mestá 1 a 5 na 76 a skontrolujte výpis.)

Úloha G.3

V textovom súbore graf.txt je uložená reprezentácia cestnej siete v podobe matice susednosti. Vytvorte funkciu na načítanie údajov z textového súboru do dvojrozmerného zoznamu (poľa) a vytvorte a použite funkciu, ktorá vypíše, všetky dvojice miest, medzi ktorými je priame spojenie (hrana). Ak je spojenie obojsmerné, stačí, ak vypíše jeden smer. Pri jednosmernej ceste doplní výpis o text " - jednosmerná".

Ukážka výpisu pre textový súbor vpravo:
1 - 2
1 - 3 - jednosmerná
1 - 6
2 - 3
2 - 5
atď.



Riešenie

```
def nacitajZoSuboru(nazov_suboru):
    with open(nazov_suboru,"r") as f:
        return [[int(riadok.split()[s]) for s in range(len(riadok.split()))] for riadok in f]

graf = nacitajZoSuboru("graf.txt")
dvojpole.vypisDvojPole(graf)

def vypisSpojenia(graf):
    print("Cesty medzi mestami:")
    N = len(graf)
    for i in range(N-1):                                    # "prejdeme horný trojuholník" v matici
        for j in range(i+1,N):
            if graf[i][j] + graf[j][i] > 0:
                if graf[i][j] == 0:
                    print(j+1," - ",i+1," - jednosmerná")
                elif graf[j][i] == 0:
                    print(i+1," - ",j+1," - jednosmerná")
                else:
                    print(i+1," - ",j+1)

vypisSpojenia(graf)
```

Úloha *G.4

Vytvorte funkciu, ktorá vygeneruje maticu vzdialeností (všetky cesty nech sú obojsmerné).

Riešenie

```
import random, dvojpole, math

def vytvorObojsmernuMaticuVzdialenosti(N):
    #vytvorenie "nulovej" matice NxN
    tab = [[0 for j in range(N)] for i in range(N)]
    for pocet in range(int(math.sqrt(N))*N):
        r = random.randint(1,N-1)
        s = random.randint(0,r-1)
        tab[r][s] = random.randint(1,100)
        tab[s][r] = tab[r][s]
    return tab

MV = vytvorObojsmernuMaticuVzdialenosti(5)
dvojpole.vypisDvojPole(MV)
```

Úloha *G.5
Vytvorte funkciu, ktorá vygeneruje maticu vzdialeností aj s jednosmernými cestami.

Riešenie

```
import random, dvojpole, math

def vytvorMaticuVzdialenosti(N = 3, obojsmerne = True):
    #vytvorenie "nulovej" matice NxN
    tab = [[0 for j in range(N)] for i in range(N)]
    for pocet in range(int(math.sqrt(N))*N):
        r = random.randint(1,N-1)
        s = random.randint(0,r-1)
        tab[r][s] = random.randint(1,100)
        tab[s][r] = tab[r][s]
        if not obojsmerne:
            if random.randint(0,N) <= N/2:
                if random.randint(0,1):
                    tab[r][s] = 0
                else:
                    tab[s][r] = 0
    return tab

pocet_miest = 3
len_obojsmerne = False
MV = vytvorMaticuVzdialenosti(pocet_miest, len_obojsmerne)
dvojpole.vypisDvojPole(MV)
```

Úloha G.6
Vytvorte funkciu, ktorá vypíše počet jednosmerných ciest z matice vzdialeností.

Riešenie študenta Martina Gajdoša:

```
def vratPocetJednosmernych(graf):
    pocetJednosmernych = 0
    for i in range(len(graf)):
        for j in range(i):
            if graf[i][j] != graf[j][i]:
                pocetJednosmernych += 1
    return pocetJednosmernych

print("Počet jednosmerných ciest: {}.".format(vratPocetJednosmernych(MV)))
```

Úloha G.7
Vytvorte funkciu, ktorá vypíše súčet ciest „zo všeobecnej“ matice vzdialeností, t.j. matica vzdialeností môže obsahovať aj jednosmerné cesty. Dĺžky obojsmerných ciest sa započítavajú len raz!

Riešenie

```
def vypisSucetCiest(graf):
    N = len(graf)
    sucet_jednosmernych = 0
    sucet_obojsmernych = 0
    for r in range(N):
        for s in range(r):
            if graf[r][s] + graf[s][r] > 0:
                if graf[r][s] == graf[s][r]:
                    sucet_obojsmernych += graf[r][s]
                else:
                    sucet_jednosmernych += (graf[r][s] + graf[s][r])
    sucet_ciest = sucet_jednosmernych + sucet_obojsmernych
    print("Dĺžka ciest:", sucet_ciest)

vypisSucetCiest(MV)
```