# Domáca úloha č. 2

2-AIN-205, Leto 2020

Termín: 6.4.2020, 14:00, e-mailom vyučujúcemu
praktická úloha do 23:59

Skôr ako sa pustíte do riešenia domácej úlohy, oboznámte sa so všeobecnými pokynmi, ktoré sú priložené na konci tohto dokumentu. Riešenia, ktoré odovzdáte, musia byť vaše vlastné. Neopisujte a nesnažte sa nájsť riešenia v literatúre alebo na internete!

**1.** [*20 bodov*] **Super hra.** Máme daný orientovaný acyklický graf, pričom vrcholy sú očíslované od 1 po $n$ tak, že hrana vedie vždy od menšieho čísla k väčšiemu. Ďalej máme daných $k$ párov vrcholov $\{u_1, v_1\}$, $\{u_2, v_2\}$, $\ldots$, $\{u_k, v_k\}$.

V počítačovej hre štartujeme z vrcholu 1 a vyberáme si cestu, ktorá nás privedie do vrcholu $n$. Ak prejdeme na tejto ceste oba vrcholy $u_i$ a $v_i$, tak vyžerieme $i$-tu 100 bodovú prémiu a prémia zmizne z hry. Chceme nájsť cestu, ktorá vyžerie najväčší možný počet bodov.

**a)** Tento problém je NP-ťažký. Navrhnite, ako túto úlohu riešiť pomocou celočíselného lineárneho programovania. Riešením je návod, ako pre ľubovoľný vstup zostaviť príslušný celočíselný lineárny program.

**b)** Predstavte si, že namiesto $k$ dvojvrcholových prémií máme daných $k$ jednovrcholových prémií $w_1, w_2, \ldots, w_k$. Vždy keď prejdeme niektorým z týchto vrcholov, tak dostaneme 100 bodovú prémiu a úlohou je opäť zozbierať najväčší možný počet bodov. Nájdite čo najefektívnejší algoritmus, ktorý rieši takýto zjednodušený problém. (Dá sa to v polynomiálnom čase.)

**c)** **Bonus 10 bodov:** Predstavte si, že namiesto jedného pokusu máme $m$ pokusov. V každom pokuse môžeme zvoliť inú cestu, no vyžraté (dvojvrcholové) prémie sa pri ďalších pokusoch už znovu neobjavujú. Každá prémia sa teda cez všetky pokusy ráta len raz. Ako by ste riešili takto zmodifikovaný problém?

**2.** [*20 bodov*] **Vrcholové pokrytie, nezávislá množina a klika.** Nech $G = (V, E)$ je neorientovaný graf. Spomeňte si, že *vrcholové pokrytie* grafu je taká množina vrcholov, v ktorej má každá hrana aspoň jeden koniec, že *nezávislá množina* je taká množina vrcholov, že žiadne dva z nich nie sú spojené hranou a naopak *klika* v grafe je taká podmnožina vrcholov, z ktorých každé dva sú spojené hranou.

Množina vrholov $U \subseteq V$ je nezávislá množina práve vtedy, keď $V - U$ je vrcholové pokrytie grafu. $U$ je tiež nezávislá množina v grafe $G$ práve vtedy, keď $U$ je klika v doplnku grafu $G$.

Predpokladajme, že by sme poznali polynomiálny algoritmus $A$ na nájdenie najväčšej nezávislej množiny. V nasledujúcich príkladoch buď **dokážte** tvrdenie profesora Premúdrelého **alebo nájdite kontrapríklad**.

**a)** Najväčšiu kliku v grafe $G$ by sme mohli hľadať tak, že jednoducho spustíme algoritmus $A$ na doplnku grafu $G$. Výsledná množina vrcholov bude súčasne najväčšou klikou v grafe $G$. Profesor Premúdrelý tvrdí, že presne týmto istým spôsobom môžeme z polynomiálneho algoritmu s konštantným aproximačným faktorom pre hľadanie najväčšej nezávislej množiny vyrobiť algoritmus s konštantným aproximačným faktorom pre hľadanie najväčšej kliky.

**b)** Najmenšie vrcholové pokrytie grafu $G$ by sme mohli hľadať tak, že jednoducho spustíme na tom istom grafe $G$ algoritmus $A$, ktorý nám vráti množinu vrcholov $U$ ako najväčšiu nezávislú množinu, a potom najmenšie vrcholové pokrytie bude jednoducho množina vrcholov $V - U$. Profesor Premúdrelý tvrdí, že tak ako v prípade a), aj tu by sme mohli rovnakým spôsobom vytvoriť z polynomiálneho algoritmu s konštantným aproximačným faktorom pre hľadanie najväčšej nezávislej množiny algoritmus s konštantným aproximačným faktorom pre hľadanie najmenšieho vrcholového pokrytia.

**3.** [*20 bodov*] **Programátorská úloha** (viď všeobecné pokyny). Na vstupe máte ohodnotený orientovaný kompletný graf s $n$ vrcholmi. Vašou úlohou je v ňom nájsť najlacnejšiu Hamiltonovskú cestu (t.j. cestu, ktorá každý vrchol navštívi práve raz a nemusí skončiť tam, kde začína). Túto úlohu budeme riešiť pomocou celočíselného lineárneho

programovania. Vašou úlohou bude napísať program, ktorý dostane na vstupe graf a na výstup vypíše celočíselný lineárny program vo formáte LP (tak ako na cvičeniach).

Od lineárneho programu požadujeme, aby obsahoval premenné $x1\_1, x1\_2, \ldots, x1\_n, x2\_1, \ldots, xn\_n$, kde $xi\_j = 1$ práve vtedy, keď v Hamiltonovskej ceste je hrana z vrchola $i$ do vrchola $j$. Iné premenné môžete používať ako sa vám zachce. Navyše musí obsahovať maximálne 1000 premenných a 1000 podmienok a musí sa dať vyriešiť za maximálne 5 sekúnd.

**Formát vstupu:** V prvom riadku vstupu je počet vrcholov $n$. V ďalších $n$ riadkoch sa nachádza matica susednosti, t.j. v každom riadku je $n$ čísel, kde $j$-te číslo v $i$-tom riadku je $c_{ij}$: cena hrany, ktorá ide z vrchola $i$ do vrchola $j$.

**Formát výstupu:** Viď: `http://lpsolve.sourceforge.net/5.0/CPLEX-format.htm`.

**Obmedzenia a bodovanie:** Na zisk plného počtu bodov je potrebné, aby váš program dal korektný výsledok pre vstupy, kde $n \leq 20$.

**Príklad vstupu:**

```
3
0 1 4
1 0 1
4 1 0
```

**Príklad výstupu:**

```
Minimize
obj: 1 x1_2 +4 x1_3 +1 x2_1 +1 x2_3 +4 x3_1 +1 x3_2
Subject To
ci1: x1_2 + x1_3 <= 1
ci2: x2_1 + x2_3 <= 1
ci3: x3_1 + x3_2 <= 1
co1: x2_1 + x3_1 <= 1
co2: x1_2 + x3_2 <= 1
co3: x1_3 + x2_3 <= 1
cx1: x1_2 + x1_3 + x2_1 + x3_1 >= 1
cx2: x2_1 + x2_3 + x1_2 + x3_2 >= 1
cx3: x3_2 + x2_3 + x3_1 + x1_3 >= 1
Binary
x1_2 x1_3 x2_1 x2_3 x3_1 x3_2
End
```

# General Instructions

**Theoretical tasks.** The answer to theoretical tasks submite handwritten or printed on a paper under the door of the office M-163. No late submission are allowed. Do not forget to write your full name legibly on each sheet and also staple your solutions.

Write your solutions so that they contain all information necessary to easily understand them, but at the same time try to aim for brevity. Prove all claims, including in the cases when it is not explicitly written in the problem statement.

If the task is about solving an algorithmic problem, submmit the best algorithm you can design. The first criterion is that the algorithm *is correct*, the second criterion is the *running time and memory complexity*. Correct and slow algorithm is worth more points than fast incorrect algorithm. Inefficient but correct algorithms will always receive at least 50% of points. In your solution, you should:

- Describe the main idea of the algorithm.
- Write a pseudocode for your algorithm.
- Prove the correctness of your algorithm.
- Analyse its running time and memory complexity.

**Programming tasks.** You need to submit a functional program, no written part is required. Submit the solution through the web interface `https://testovac.ksp.sk/tasks/`. Your solutions will be evaluated immediately on several inputs and you will learn number of points (there are several sets of inputs and you only gain points if you solve correctly all inputs in a particular set). You can submit a solution several times, we will only take into account the last submitted solution before the deadline. You need to register before you can submit solutions (see the panel on the left side of the web page). The details about how to write and submit your programs (including supported programming languages) are in the section "Čo odovzdávať?".