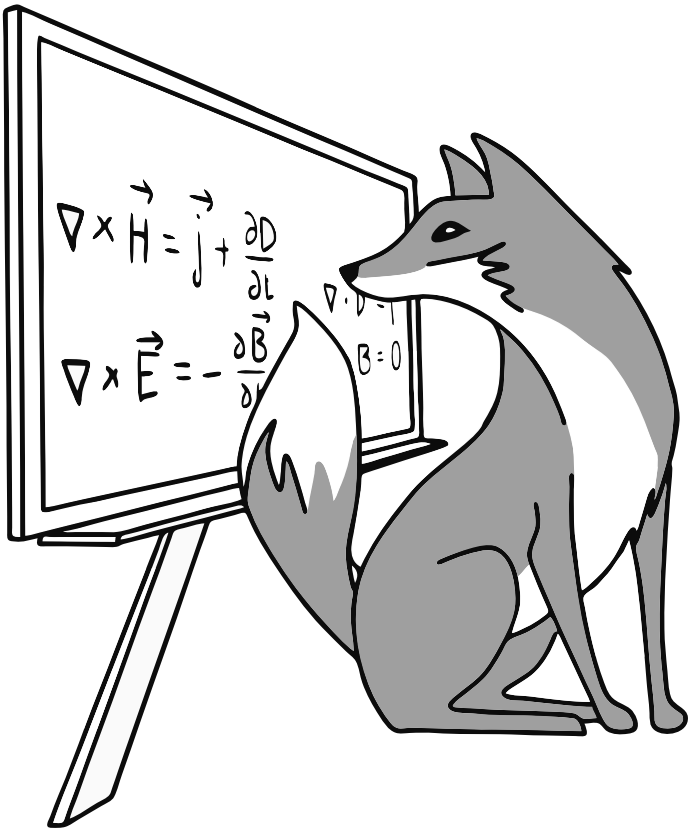
STUDENTSKÝ ČASOPIS A KORESPONDENČNÍ SEMINÁŘ

Ročník XXVI
Číslo 2

M & M

MATEMATIKA FYZIKA INFORMATIKA



Uvnitř najdete několik témat a s nimi souvisejících úloh. Zamyslete se nad nimi a pošlete nám svá řešení. My vám je opravíme, pošleme zpět s dalším číslem a ta nejzajímavější z nich otiskneme. Nejlepší řešitele zveme na podzim a na jaře na soustředění.

Milí řešitelé,

ve druhém čísle na vás čekají pokračování tématek z minulého dílu. V herním tématku se budete moct zamyslet nad situacemi, kde neexistuje jedna optimální strategie, ve Výpočetních modelech obohatíte své stroje o nová rozšíření ve formě různých druhů pamětí, ve třetím tématku o zpracování dat ze senzorů se budete zabývat odstraňováním šumu z výsledných obrazů a v rámci tématka o elektromagnetismu zjistíte, co se stane, když se náboje dají do pohybu.

Třetí týden v říjnu proběhlo podzimní soustředění v Moldavě. Fotky a jiné materiály k němu se objeví na webu v nadcházejících týdnech. Všechny řešitele taktéž zveme na tradiční vánoční víkendovku, která se bude konat od 6. do 8. prosince. Bližší informace naleznete na našem webu.

Spolu s novým číslem vám přišlo i zadání KoKoSu. Jedná se o matematický korespondenční seminář pro žáky 6. až 9. tříd a odpovídajících ročníků víceletých gymnázií, můžete ho tedy nabídnout mladším sourozencům nebo kamarádům se zájmem o matematiku.

Na závěr bychom vás chtěli upozornit na tři akce, které proběhnou v listopadu, jmenovitě na MathRace, Internetovou matematickou olympiádu (IMO) a Fyziklání online. Jedná se o týmové soutěže pro středoškoláky, které prověří nejen vaše znalosti z oblastí matematiky a fyziky, ale i vaši schopnost pracovat ve skupině. Přihlašování na tyto akce stále běží, máte tedy možnost se zúčastnit.

Hodně zábavy a mnoho úspěchů při řešení!

Vaši organizátoři

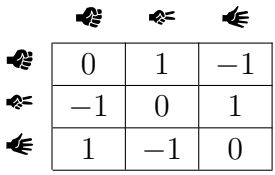
Zadání a řešení témat

Termín odeslání úloh 2. série: 17. 12. 2019

Téma 1 – Hry

Díl 2: Smíšené strategie

Naším tématem jsou stále maticové hry. Pokud má matice sedlový bod (viz minulé číslo), pak určuje pro oba hráče strategii, která je optimální. Znalost, jakou strategii použije soupeř, tedy není možné využít k výběru nějaké strategie výhodnější pro nás, jak jsme již viděli v minulém díle. Nemá-li však matice sedlový bod, je situace zajímavější. Názorným příkladem je hra kámen-nůžky-papír[1].



[1] Ikony „Rock icon“, „Paper icon“ a „Scissors icon“ od Johna Redmana pod licencí CC BY 3.0 jsou z `https://game-icons.net/`

Pokud by soupeř předem věděl, co uděláme, mohl by toho jednoduše využít a obohatit se s jistotou na náš úkor (s jistotou by získal 1 bod, takže my bychom vždy 1 bod ztratili, neboť hrajeme hru s nulovým součtem). V takovou chvíli je lepší být nepředvídatelným soupeřem. Proto místo otázky, jakou konkrétní strategii zvolit, si budeme klást otázku, jaké pravděpodobnostní rozdělení[2] na nabízených strategiích zvolit. Tomuto pravděpodobnostnímu rozdělení budeme říkat smíšená strategie. V případě hry kámen-nůžky-papír lze snadno dospět k závěru, že je optimální volit každou strategii ve třetině případů. Jakékoliv jiné (smíšené) strategie by mohl soupeř využít ve svůj prospěch

Když hráči aplikují smíšené strategie, začne být rozhodující otázka průměrného výsledku. Ukažme si to na následujícím příkladu.

2	-7
-3	10

Řekněme, že první hráč (volící řádky) aplikuje strategii 1 v 1/4 případů a strategii 2 v 3/4 případů. Řekněme, že druhý hráč (volící sloupce) aplikuje každou strategii v 1/2 případů. K jakému průměrnému výsledku tato situace vede? S pravděpodobností 1/8 zvolí oba hráči strategii 1, což vede k zisku 2 bodů pro prvního hráče. S pravděpodobností 1/8 zvolí první hráč strategii 1 a druhý hráč strategii 2, což vede k zisku 7 bodů pro druhého hráče. S pravděpodobností 3/8 zvolí první hráč strategii 2 a druhý hráč strategii 1, což vede k zisku 3 bodů pro druhého hráče. A nakonec s pravděpodobností 3/8 zvolí oba hráči strategii 2, což vede k zisku 10 bodů pro prvního hráče. Průměrný výsledek zde je:

$$\frac{1}{8}\cdot 2+\frac{1}{8}\cdot(-7)+\frac{3}{8}\cdot(-3)+\frac{3}{8}\cdot 10=\frac{1}{4}-\frac{7}{8}-\frac{9}{8}+\frac{15}{4}=2$$

Průměrný výsledek není nic jiného než vážený průměr čísel v matici, kde každé číslo má váhu definovanou součinem pravděpodobnosti volby jeho řádku a pravděpodobnosti volby jeho sloupce. Formálně, pokud bychom matici určující hru označili A, vektor pravděpodobností prvního hráče označili x a vektor pravděpodobností druhého hráče označili y, je průměrný výsledek hry roven x^TAy. Toto je standardní formalismus pro maticové hry, ale my ho nebudeme příliš využívat, jelikož je neintuitivní pro ty, kteří nejsou zběhlí v lineární algebře.

Pojďme se nyní podívat, jak tato situace vypadá z pohledu prvního hráče. Pokud by první hráč věděl, že druhý hráč volí každou strategii v 1/2 případů, pak si první hráč okamžitě umí dopočítat, že volba strategie 1 mu dává v průměru zisk $-2{,}5$ bodu, zatímco volba strategie 2 mu dává v průměru zisk $+3{,}5$ bodu. Díky znalosti, jakou smíšenou strategii aplikuje druhý hráč, se první hráč snadno dokáže rozhodnout, že strategie 2 je pro něj lepší.

[2]Pravděpodobnostní rozdělení pro nás bude zobrazení, které každé strategii přiřadí pravděpodobnost její volby. Protože budeme hrát jen hry s konečnou množinou strategií, můžeme bez újmy na obecnosti zapisovat tato pravděpodobnostní rozdělení jako K-tici nezáporných čísel (vektor délky K), jejichž součet je roven jedné. Např: $(1/2; 1/6; 1/3)^T$

Rozveďme to ještě z pohledu druhého hráče. Je jeho smíšená strategie dobrá? Vskutku není! Jakmile první hráč pozná, že druhý hráč volí každou strategii v 1/2 případů, může hned druhého hráče „očesávat“ tím, že bude upřednostňovat strategii 2, což se mu z dlouhodobého hlediska musí vyplatit. Existuje ale vůbec pro druhého hráče nějaká dobrá strategie? Existuje taková smíšená strategie, kterou může druhý hráč dlouhodobě aplikovat tak, že i když ji první hráč pozná a adaptuje se na ni, bude situace stále příznivá pro druhého hráče?

Ano, existuje! Řekněme, že druhý hráč bude volit strategii 1 v 77 % procentech případů (oproti původním 50 %, se kterými „ostrouhal“). Když teď první hráč zvolí strategii 1, jeho průměrný zisk bude $0{,}77 \cdot 2 + 0{,}23 \cdot (-7) = -0{,}07$, tedy příznivý pro druhého hráče. A když první hráč zvolí strategii 2, jeho průměrný zisk bude $0{,}77 \cdot (-3) + 0{,}23 \cdot 10 = -0{,}01$, taktéž příznivé pro druhého hráče. Z toho plyne, že tato smíšená strategie druhého hráče je dobrá, protože se i při její znalosti první hráč nezvládne dostat do kladného zisku. Zda existuje nějaká ještě lepší smíšená strategie pro druhého hráče, zde už nebudeme rozebírat.

Optimální smíšenou strategii definujeme tak, že když pro každou možnou naši smíšenou strategii uvážíme nejnepříjemnější možnou strategii soupeře, tak je průměrný výsledek s naší optimální smíšenou strategií maximální mezi všemi našimi možnými smíšenými strategiemi.

Nyní si to zapíšeme formálně. Nechť S_1 je množina všech nezáporných vektorů délky m se součtem prvků rovným 1. Nechť S_2 je množina všech nezáporných vektorů délky n se součtem prvků rovným 1. Pak optimální smíšená strategie x^* prvního hráče je $x^* = \operatorname{argmax}_{x \in S_1}(\min_{y \in S_2}(x^T A y))$. Dále budeme optimální smíšenou strategii nazývat zkráceně jako optimální strategii.

Následujících pět úloh vychází ze hry určené níže uvedenou maticí. První hráč má na výběr ze tří strategií, druhý hráč má na výběr ze dvou strategií.

1	-2
3	-4
-1	2

Problém 1 [2b]: *Určete optimální strategii pro prvního hráče.*

Problém 2 [2b]: *Určete optimální strategii pro druhého hráče.*

Problém 3 [1b]: *K jakému průměrnému výsledku vede, pokud oba hráči použijí optimální strategii?*

Problém 4 [1b]: *Jaké jsou možné průměrné výsledky, pokud první hráč zvolí optimální strategii, ale druhý hráč se od optimální strategie odkloní?*

Problém 5 [1b]: *Jaké jsou možné průměrné výsledky, pokud naopak druhý hráč zvolí optimální strategii, ale první hráč se od optimální strategie odkloní?*

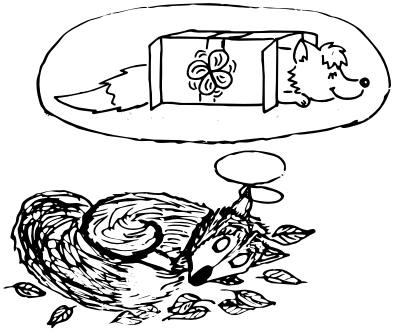
Markét, Martin; `martin.dvorak@matfyz.cz`
e-mailová konference: `hry@mam.mff.cuni.cz`

Téma 2 – Výpočetní modely

Pokud jste nečetli první díl[3], přečtěte si jej. Budeme dále používat notaci v něm zavedenou a přidávat našim strojům další schopnosti.

Díl 2: Paměť a proměnné

Na konci prvního dílu byly naše stroje pořád dost slabé – stavové registry nám dovolovaly si uchovávat jen konečné množství informací a tudíž existovala spousta problémů, které jsme neuměli řešit. Nejčastěji navrhovanými rozšířeními tudíž byly různé formy paměti, především zásobník a fronta.



Zásobník je rozšíření, do kterého je možné zapisovat řetězce, a které v každém kroku vypíše nejnovější znak v něm, čímž jej odstraní. Pokud v něm aktuálně žádné znaky nejsou, vypisuje „$“. Pokud do zásobníku zapisujeme delší řetězec, nejnovější znak je ten vpravo. Pokud ze zásobníku nechceme znak odebrat, zapíšeme jej hned zpátky.

V zájmu toho, aby nám vycházely jednotlivé části pokynů i nadále do sloupců, domluvme se, že mezery v pokynech budeme ignorovat. Pokud budeme potřebovat s mezerami pracovat, budeme je psát jako „␣“.

```
IO
zásobník
0,$ →  ,0
1,$ →  ,1
0,0 →  ,00
1,0 →  ,01
0,1 →  ,10
1,1 →  ,11
$,0 → 0,
$,1 → 1,
```

Stroj 1: Příklad stroje, který obrátí pořadí bitů ze vstupu

[3] https://mam.mff.cuni.cz/media/cislo/pdf/26/26-1.pdf

Fronta je poněkud sofistikovanější rozšíření. Opět nám umožňuje do ní zapisovat řetězce, ale v každém kroku vypisuje nejstarší v ní přítomný znak, který navíc automaticky neodstraňuje. Fronta totiž má i druhou položku v pokynu, kde za každý znak „>“ zahodí nejstarší znak, který obsahuje.

Můžeme si tedy frontu představit jako řadu znaků, na jejíž (pravý) konec můžeme připisovat další znaky a kterou čteme od začátku (zleva).

```
IO
fronta
stavový registr (počáteční stav S)

0,$,S → ,0, ,L
1,$,S → ,1, ,L
0,0,S → ,0, ,L
1,0,S → ,1, ,L
0,1,S → ,0, ,L
1,1,S → ,1, ,L
0,0,L → ,0, ,S
1,0,L → ,1, ,S
0,1,L → ,0, ,S
1,1,L → ,1, ,S
$,0,S → , ,>,L
$,1,S → , ,>,L
$,0,L → 0, ,>,S
$,1,L → 1, ,>,S
```

Stroj 2: Příklad stroje, který vypíše pouze znaky s lichou vzdáleností od konce. Nejdříve zkopíruje vstup do fronty a při tom (podle toho, zda skončí ve stavu L nebo S) určí, zda je liché nebo sudé délky. Poté čte z fronty a vypisuje pouze příslušné znaky.

Můžete si všimnout, že popisy strojů poměrně rychle rostou do délky. Provedeme tedy malou odbočku a ukážeme si pár triků, jak je zase zkrátit.

Nejdříve si rozmysleme, jak přechodová funkce vybírá krok, který provede. Bude to dělat tak, že bude koukat na svůj popis řádek po řádku a vždy vybere první odpovídající řádek. Malá písmena budeme používat jako proměnné. Pokud přechodová funkce narazí na řádek obsahující malá písmena, pokusí se každému malému písmenu přiřadit nějakou hodnotu tak, aby výpis odpovídal tomuto řádku. Pokud se jí to podaří, provede tento krok. Pokud ne, zkusí další řádky.

```
vstup
vstup
výstup

a,a → 1
a,b → 0
```

Stroj 3: Stroj, který pro každou pozici vypíše jedničku pokud se oba vstupy shodly a nulu pokud nikoliv.

Speciální chování bude mít znak podtržítka („_“). Jeho význam je „tato hodnota mě nezajímá“ a tak se chová, jako kdyby každé podtržítko bylo nahrazeno jiným (jinde nepoužitým) písmenkem.

Podtržítka se mohou vyskytnout pouze před šipkou a každé písmenko za šipkou se musí vyskytnout i před ní.

Stručnější verze předchozího stroje tudíž může vypadat třeba takhle:

```
IO
fronta
stavový registr (počáteční stav S)

0,_,S →  ,0, ,L
1,_,S →  ,1, ,L
0,_,L →  ,0, ,S
1,_,L →  ,1, ,S
$,0,S →  , ,>,L
$,1,S →  , ,>,L
$,a,L → a, ,>,S
```

Stroj 4: Kompaktnější verze předchozího stroje. Rozmyslete si, že výpis $, $, L nemůže nastat.

Páska je třetí (a pro tento díl poslední) nové rozšíření. Představte si nekonečnou pásku, nad kterou se posouvá čtecí hlava. Páska je rozdělená na políčka a do každého políčka je možné zapsat jeden z konstantní množiny znaků. V každém kroku páska vypíše do výpisu znak, který se aktuálně nachází pod hlavou. V pokynu následně dostane znak, který na tutéž pozici má zapsat, a směr, kterým se má posunout. Není-li uvedeno jinak, předpokládáme, že všechna políčka pásky na začátku obsahují znak mezery („␣“) a pokud hlava v pokynu nedostane jiný znak, tak po sobě vždy zanechá opět mezeru. Hlava se může posouvat doleva (značíme „<“), doprava („>“), nebo zůstat na místě („ “). V prvních dvou případech se vždy posune o právě jedno políčko.

```
stavový registr (počáteční stav A)
IO
páska

A,1,␣ → A, ,1,>
A,#,␣ → R, ,␣,<
R,#,␣ → S, ,␣,>
R,#,a → R, ,a,<
S,#,0 → S, ,0,>
S,#,1 → L, ,0,>
S,#,␣ → R,0,␣,<
L,#,0 → L, ,0,>
```

```
L,#,1 → S, ,1,>
L,#,␣ → R,1,␣,<
```

Stroj 5: Stroj, který vypíše binární zápis počtu jedniček, které dostal na vstupu. Vstup zkopíruje na pásku a pak vždy smaže polovinu jedniček, čímž postupně zjišťuje bity od nejméně významného. Když smaže poslední jedničku, dále vypisuje nekonečnou řadu nul.

Problém 1: *Sestrojte stroj, který pozná správně uzávorkované řetězce.*

Problém 2: *Sestrojte stroj, který pozná palindromy.*

Problém 3: *Nalezněte stroj, který dělá totéž co stroj 3 výše, ale*
a) má co nejkratší popis.
b) potřebuje méně kroků.

Problém 4: *Sestrojte násobičku. (Násobička na dvou vstupech dostane dvě přirozená čísla ve dvojkové soustavě a na výstup vypíše jejich součin.)*

Problém 5: *Představte si, že nemáte k dispozici pásku. Sestavte stroj, který ji simuluje – na jednom vstupu čte znak k zápisu, na druhém směr posunu a na výstup píše aktuálně čtený znak.*

Problém 6: *Jaká další rozšíření bychom mohli chtít? (Zamyslete se nad tím, jak s nimi bude interagovat přechodová funkce.)*

Problém 7: *Zamyslete se nad tím, jak jinak bychom mohli dostávat vstupní řetězce. Které problémy by to usnadnilo? Co by to zkomplikovalo? Chtěli byste změnit i něco jiného? Proč a jak?*

Problém 8: *Někteří z vás již možná tuší, co je to nedeterminizmus. (Pokud ne, tak se o něm dozvíte v příštím čísle.) Jak by se dal formulovat v řeči našich modelů?*

Matej; `lieskovsky.matej+stroje@gmail.com`
e-mailová konference: `stroje@mam.mff.cuni.cz`

Téma 3 – Zpracování obrazových dat ze senzorů

Díl 2: Redukce šumu

V minulém díle jsme si ukázali, jak funguje obyčejný radar a jak zhruba vypadají naměřená data. Prozkoumali jsme postupně první tři dimenze – vzdálenost, úhel natočení antény a intenzitu signálu. Data jsme si zobrazili pomocí 2D obrázku, kde třetí dimenzi symbolizovala barva.

Ve druhém díle se znovu podíváme na tyto 2D obrázky a pokusíme se z nich odstranit šum.

Pravděpodobně se vám nepodaří odstranit 100% šumu. Zkuste ale slovně popsat, jaký typ šumu se vám podařilo odstranit. Na plný počet bodů není nutné odstranit úplně všechen šum.

Jak odstranit šum

Uvedu několik způsobů (zdaleka ne všechny), jak je možné redukovat šum v černobílém obrázku. Nenechte se ale následujícími odrážkami vázat, kreativita je zde vítána.

- *Threshold:* Vybereme si odstín šedé a všechny světlejší barvy prohlásíme za bílou, naopak všechny tmavší za černou.

- *Dynamický threshold:* Odstín šedé nevolíme my, ale necháme jej vypočítat podle vlastností obrázku. Threshold určí grafický editor nebo náš program za nás.

- *Rozmazání:* Existuje několik způsobů, jak rozmazat obrázek, a každý z nich může poskytnout jiné výsledky. K rozmazání budeme pravděpodobně využívat konvoluci (`https://cs.wikipedia.org/wiki/Konvoluce`), aniž bychom o tom věděli.

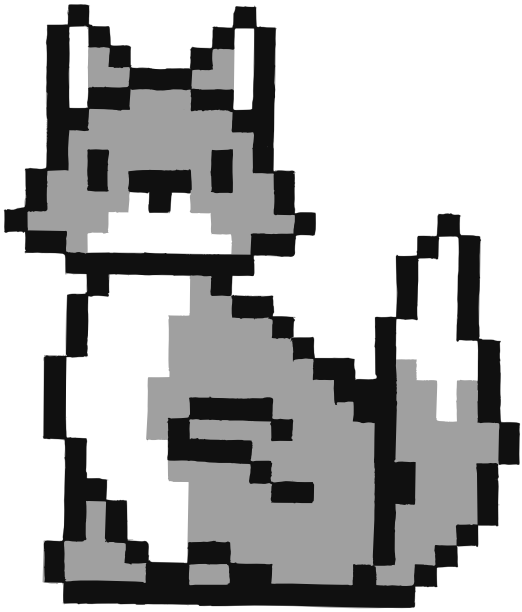
- *Ignorování části obrázku:* Může se stát, že senzor není dokonalý a některé pixely vždy zobrazují falešná data. Tyto pixely nebo celé oblasti můžeme ignorovat. Je ale vhodné mít na paměti, že náš senzor se v tomto místě stává slepým.

- *Filtr „odstranit šum“:* Většina grafických editorů tuto položku nabízí. Zamýšleli jste se už nad tím, jak funguje? (K pochopení funkce tohoto filtru nám pravděpodobně nebude stačit učivo středoškolské matematiky. V rámci tohoto tématka se nemusíte zabývat analýzou tohoto filtru.)

- *Umělá inteligence:* Metody založené na umělé inteligenci jsou v oboru zpracování obrazu stále populárnější. V rámci tohoto tématka se jimi zabývat nebudeme. To vás ale nijak neomezuje v používání těchto metod, pokud víte jak.

- *Předzpracování (pre-processing):* U některých obrázků je vhodné například upravit jas nebo aplikovat jiný filtr ještě před tím, než spustíme samotné odstraňování šumu.
- *Kombinace výše uvedeného:* K dosažení lepších výsledků často kombinujeme více filtrů, které na obrázek postupně aplikujeme. Jeden obrázek může obsahovat více typů šumu, které se nám pomocí jediného filtru nepodaří zredukovat.

Problém 1: *V minulém díle jste dostali celkem šest obrázků různých typů terénu, u kterých jste hádali, co asi zobrazují. Najdete je ve formátu PNG v odkazu na konci tohoto článku. Zkuste si tyto obrázky otevřít v nějakém grafickém editoru (doporučuji například Gimp) a najít jeden postup aplikovatelný na všechny obrázky (ne úprava obrázku štětcem ručně pomocí myši), který z těchto obrázků co nejlépe odstraní šum a zvýrazní detekované objekty (zem, strom, skála, létající objekt, ...). Ve svých řešeních popište, jaké problémy jste museli řešit a jaké postupy jste zkoušeli aplikovat. Odevzdejte také postupy a vámi upravené obrázky.*



Problém 2: *Šum v obrázku můžeme redukovat i bez pomoci grafického editoru. Pokusíme se naprogramovat vlastní filtr redukující šum v jazyce Python.*

V datech k tomuto článku najdete soubory s příponou `*.npy`*, které obsahují data ke stejným obrázkům jako v minulém díle. Tento soubor opět přečtete stejným způsobem jako v minulém díle, viz následující kód:*

```
import numpy as np
import matplotlib.pyplot as plt

frame = np.load("1.npy")
plt.imshow(frame, interpolation='nearest', cmap='binary')
plt.show()
```

Projděte postupně všechny pixely načteného obrázku a aplikujte na každý pixel vámi vytvořený filtr:

```
# Muzete upravit libovolnou cast kodu,
# staci ale, kdyz upravite jen radek 17.
import numpy as np
import matplotlib.pyplot as plt

frame = np.load("1.npy")

fig = plt.figure()
fig.add_subplot(1, 2, 1)
plt.imshow(frame, interpolation='nearest',
           cmap='binary', aspect='auto')

for y in range(frame.shape[0]):
    for x in range(frame.shape[1]):
        # zde vlozte svuj filtr, i jeden radek staci
        # nasledujici radek je priklad threshold filtru
        frame[y][x] = frame[y][x] if frame[y][x]>500 else 0

fig.add_subplot(1, 2, 2)
plt.imshow(frame, interpolation='nearest',
           cmap='binary', aspect='auto')
plt.show()
```

Vyzkoušejte si, jestli váš filtr funguje na všechny obrázky (nebo aspoň na většinu), které máte k dispozici. Upravte výše uvedený kód v jazyce Python.

Data ke stažení

Data k tomuto dílu:
`https://mam.mff.cuni.cz/media/prilohy/26-2-3-2D.zip`

Béda; `bedrich.said@gmail.com`
e-mailová konference: `radary@mam.mff.cuni.cz`

Téma 4 – Vybrané kapitoly z elektromagnetismu

Díl 2: Biot-Savartův zákon

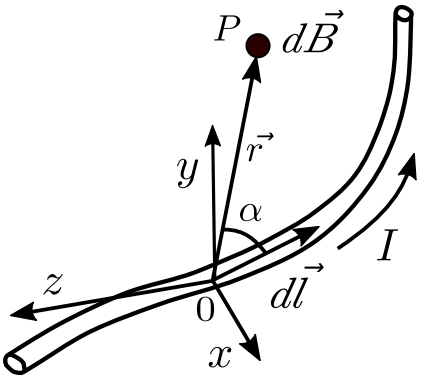
V minulém díle jsme se zabývali převážně elektrostatikou, nyní si ukážeme, co se děje, když se náboje dají do pohybu. Kde je pohyb náboje, tam vzniká magnetické pole, a magnetické pole působí silou na pohybující se náboj.

My se budeme zabývat prvním případem, kdy pohybující se náboj generuje magnetické pole. Tento jev popisuje Biot-Savartův zákon, který určuje velikost a směr magnetické indukce $\mathrm{d}\boldsymbol{B}$ vyvolané malým kouskem vodiče $\mathrm{d}\boldsymbol{l}$, kterým teče proud I v bodě označeného polohovým vektorem $\boldsymbol{r}$. Pro magnetickou indukci platí princip superpozice, proto je výsledné pole $\boldsymbol{B}$ určeno součtem všech příspěvků $\mathrm{d}\boldsymbol{B}$ přes jednotlivé kousky vodiče:

$$\boldsymbol{B} = \int \mathrm{d}\boldsymbol{B} = \frac{\mu_0}{4\pi} \int_l \frac{\mathrm{d}\boldsymbol{l} \times \boldsymbol{r}}{r^3} \implies B = \frac{\mu_0}{4\pi} \int_l \frac{\mathrm{d}l r \sin\alpha}{r^3}$$

Jak uvidíme, $\mathrm{d}\boldsymbol{l}$, $\boldsymbol{r}$ a $\mathrm{d}\boldsymbol{B}$ jsou vektorové veličiny, které mají směr jako na obrázku 1. Element délky $\mathrm{d}\boldsymbol{l}$[4] míří ve směru toku proudu, proud sám není vektorová veličina. Polohu vektoru $\mathrm{d}\boldsymbol{B}$ určuje v souřadném systému polohový vektor $\boldsymbol{r}$, jehož počátek je obvykle totožný s počátkem soustavy souřadnic. Konstanta μ_0 se nazývá permeabilita vakua a má hodnotu $\mu_0 = 1{,}26 \cdot 10^{-6}\,\mathrm{N \cdot A^{-2}}$. Jednotka magnetické indukce je T (Tesla). Dá se zapsat jako:

$$\mathrm{T} = \mathrm{N \cdot A^{-1} \cdot m^{-1}} = \mathrm{kg \cdot s^{-1} \cdot C^{-1}}$$



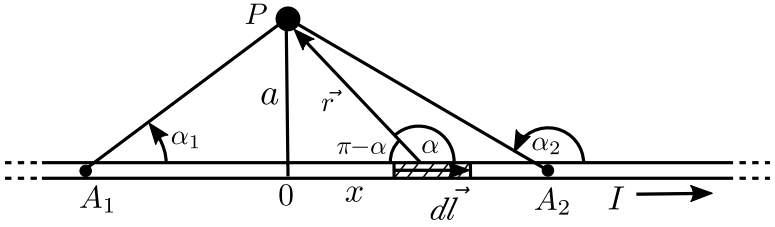
Obrázek 1: Ilustrace Biot-Savartova zákona pro obecný vodič protékaný proudem

Vektorový součin, který v zákoně figuruje, se dá zjednodušit podle vzorce $|\boldsymbol{a} \times \boldsymbol{b}| = |a||b| \sin\alpha$, kde α je úhel mezi vektory $\boldsymbol{a}$ a $\boldsymbol{b}$. Výsledný vektor bude

[4] element je zde nekonečně malá část celku

kolmý na oba vektory a směr určíme podle pravidla pravé ruky (jelikož násobíme vektory v pravotočivé souřadné soustavě). Pravidlo pravé ruky říká: pokud položíme pravou ruku tak, že máme prsty ve směru prvního vektoru a druhý nám vystupuje z dlaně, tak vztyčený palec ukazuje směr třetího vektoru vzniklého vektorovým součinem. Geometricky lze velikost vektoru $\boldsymbol{c}$ prohlásit za obsah rovnoběžníku vymezeného vektory $\boldsymbol{a}$ a $\boldsymbol{b}$ s počátky v jednom bodě. Vektorový součin není komutativní, po prohození vektorů $\boldsymbol{a}$ a $\boldsymbol{b}$ by vektor $\boldsymbol{c}$ vzniklý jejich násobením měl opačný směr.

Opět si ukážeme, jak se s tímto zákonem pracuje a odvodíme si vzorec pro výpočet magnetické indukce nekonečně dlouhého rovného vodiče[5] v místě vzdáleném od vodiče kolmo 3 cm, kterým protéká konstantní proud 2 mA. Rozkreslíme si situaci:



Obrázek 2: Nákres nekonečně dlouhého přímého vodiče, kterým protéká konstantní proud I. Nula značí počátek souřadného systému os x, y, z, v níž je umístěn vektor $\mathbf{r}$; samotný souřadný systém zde pro přehlednost není zakreslen.

Značení jsme zvolili $a = 3\,\mathrm{cm}$, $I = 2\,\mathrm{mA}$. Osa x je totožná s vodičem, takže platí, že $\mathrm{d}l = \mathrm{d}x$. Biot-Savartův zákon bude mít tvar:

$$\boldsymbol{B} = \int \mathrm{d}\boldsymbol{B} = \frac{\mu_0}{4\pi} \int_l \frac{I\mathrm{d}\boldsymbol{l} \times \boldsymbol{r}}{r^3}$$

$$B = \frac{\mu_0}{4\pi} \int_{-\infty}^{\infty} \frac{Ir \sin\alpha}{r^3} \mathrm{d}l$$

Použili jsme přepis vektorového součinu pomocí funkce sinus a vytkli konstanty před integrál. Druhý řádek je skalární forma téhož zákona. Integrujeme, tedy sčítáme nekonečně malé části délky nekonečně dlouhého vodiče, což může být trochu problém. Abychom neintegrovali od mínus do plus nekonečna, pomůžeme si trochou úprav.

Vidíme, že bod P je vzdálen od vodiče $a = 3\,\mathrm{cm}$, a že proud je konstantní. Bohužel, pro každý bod vodiče je vektor $\boldsymbol{r}$ proměnný. Navíc nevíme téměř nic o nekonečně krátkém kousku vodiče $\mathrm{d}\boldsymbol{l}$, pouze můžeme určit x-ovou souřadnici každého nekonečně malého kousku vodiče od kolmé vzdálenosti bodu P po délce

[5] Tento příklad lze aplikovat i na velmi dlouhý vodič

vodiče. Budeme si proto muset Biot-Savartův zákon vyjádřit trochu jinak a využít k tomu úhel α mezi vektorem $\boldsymbol{r}$ a vodičem, který reprezentuje osu x:

$$r = \frac{a}{\sin\alpha}$$

Všimněme si, že se jedná pouze o velikost vektoru $\boldsymbol{r}$, tedy jeho délku. Proto zde nevystupuje jako vektor.

$$x = -a\,\mathrm{cotg}\,\alpha$$

Derivací druhé rovnice podle alfy získáme diferenciál $\mathrm{d}x$, který jsme potřebovali nahradit ve výpočtu velikosti magnetické indukce:

$$\frac{\mathrm{d}x}{\mathrm{d}\alpha} = \frac{a}{\sin^2\alpha}$$

Celou rovnici vynásobme $\mathrm{d}\alpha$:

$$\mathrm{d}x = \frac{a}{\sin^2\alpha}\mathrm{d}\alpha$$

Substituujeme integrál pro výpočet magnetické indukce B s tím, že bodu vodiče blížícímu se k mínus nekonečnu odpovídá úhel 0° a plus nekonečnu odpovídá úhel 180°.

$$B = \int \mathrm{d}B = \frac{\mu_0}{4\pi}\int_l \frac{I\mathrm{d}x r\sin\alpha}{r^3} = \frac{\mu_0}{4\pi}\int_{\alpha_1}^{\alpha_2} I\frac{\sin^2\alpha}{a^2}\cdot\frac{a}{\sin^2\alpha}\sin\alpha\mathrm{d}\alpha =$$

$$= \frac{\mu_0 I}{4\pi a}\int_{\alpha_1}^{\alpha_2}\sin\alpha\mathrm{d}\alpha$$

Nyní integrujeme přes konkrétní meze - integrál přes úhel, který svírá vektor $\boldsymbol{r}$ s vodorovně nakresleným vodičem, tedy osou x. Jeho hodnoty tedy nabývají od 0° do 180°. Po integraci podle úhlu alfa dostaneme vztah, který hledáme:

$$B = \frac{\mu_0}{4\pi}I\int_{\alpha_1}^{\alpha_2}\frac{a}{r^2\sin\alpha}\mathrm{d}\alpha = \frac{\mu_0 I}{4\pi a}\int_{\alpha_1}^{\alpha_2}\sin\alpha\mathrm{d}\alpha = \frac{\mu_0 I}{4\pi a}(\cos\alpha_1 - \cos\alpha_2) =$$

$$= \frac{\mu_0 I}{4\pi a}[1-(-1)] = \frac{\mu_0 I}{2\pi a}$$

Vytkli jsme i proud před integrál, což nám umožnil náš předpoklad, že proud je všude ve vodiči konstantní. Po dosazení všech hodnot dostaneme $B = 13{,}34\cdot 10^{-9}\,\mathrm{T}$.

Problém 1 [3b]: *Určete směr vektoru magnetické indukce v nekonečně dlouhém vodiči v příkladu výše. Jaký tvar budou mít magnetické siločáry?*

Problém 2 [4b]: *Napište Biot-Savartův zákon pro magnetickou indukci nacházející se v ose cívky, která má jeden závit a jejíž osa je kolmá na plochu závitu. Odhadněte, jak budou vypadat magnetické siločáry v ose a v rovině ploch cívky.*

(Nápověda: Rozložte si náhodně orientovaný vektor $\boldsymbol{B}$ *do dvou kolmých směrů, z nichž jeden bude rovnoběžný s plochou cívky. Integrujte přes* $\mathrm{d}\boldsymbol{l}$ *po uzavřené křivce, tedy smyčce cívky.)*

Problém 3 [4b]: *Pokud dáme vedle sebe dva nekonečně dlouhé rovnoběžné vodiče, kterými protéká proud, začnou spolu interagovat díky generované magnetické indukci. Popište, jak bude vypadat výsledná magnetická indukce v prostoru kolem vodičů, pokud proudy ve vodičích potečou stejným směrem a pokud potečou opačným.*

Pája, Fanda; fandazajic@gmail.com
e-mailová konference: elmag@mam.mff.cuni.cz

Konference Domašov 2019

Fibonacciho soustava 6,7 b

Mgr.MM Tomáš Flídr a Mgr.MM Klára Hloušková

Úvod

V rámci práce na naší konfeře jsme se zabývali číselnou soustavou, která místo mocnin nějakých čísel (jako desítková má mocniny o základu deset nebo binární základ dvě) používá jako řády členy Fibonacciho posloupnosti. Tu jsme definovali začátkem 1, 2, protože kdyby začínala na dvě jedničky, zápis by ani nemohl být jednoznačný. Zkoumali jsme vlastnosti takto zapsaných čísel, operace s nimi i možnosti využití v kódování.

Zápis

Abychom dosáhli jednoznačného a co nejjednoduššího zápisu a vůbec mohli počítat, zapisovali jsme čísla ve Fibonacciho soustavě pouze ciframi 1 a 0 (Fibonacciho čísla rostou pomaleji než mocniny dvojky, takže by nemělo být potřeba více cifer) a navíc tak, aby v čísle nebyly žádné dvě jedničky za sebou (první takováto dvojice by se sečetla na jedničku o řád výše). Pro používání ale zápis musí být bijekcí na přirozená čísla a nulu, takže se do něj všechna tato čísla musí dát převést a musí být jednoznačný.

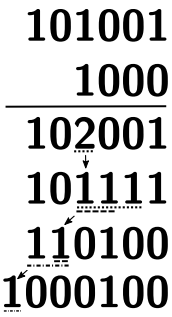
První podmínku můžeme dokázat ukázáním algoritmu, který takto zapíše libovolné přirozené číslo. Je jím tzv. Hladový algoritmus, který číslo rozloží na součet Fibonacciho čísel tak, že vždy odečte největší možné. Všimněme si, že nikdy neodečte dvě po sobě jdoucí Fibonacciho čísla – pokud by je mohl odečíst, odečetl by jejich součet o řád výše.

Splnění druhé podmínky (jednoznačnost zápisu) je dáno tím, že jednička na dané pozici nemůže být zapsána jinak. Je zřejmé, že ji není možné zapsat pomocí vyšších řádů, protože všechna čísla ve vyšších řádech jsou větší. Obdobně jsou všechna čísla vzniklá použitím menších řádů menší: Největší povolené číslo s daným počtem řádů je takové, ve kterém se střídají cifry 1 a 0 a na prvním místě je jednička. Když k tomuto číslo přičteme 1, poslední dvě cifry se sečtou na 1 na místě 0 o řád výše, což se bude opakovat zezadu až na největší řád, kde vytvoří 1 na o jedna vyšším řádu než nejvyšší řád původního čísla s tím, že ostatní řády budou 0. Máme tedy požadovaný (takzvaně pěkný) zápis a můžeme se pustit do početních operací.

Základní operace

Když už jsme zjistili, že dokážeme zapsat číslo skutečně jednoznačně, můžeme zkoumat základní operace. Budeme používat pouze čísla v pěkném zápisu, který jsme si na začátku definovali. V tomto zápise např. 7 zapíšeme jako 1010, protože $7 = 0 \cdot 1 + 1 \cdot 2 + 0 \cdot 3 + 1 \cdot 5$.

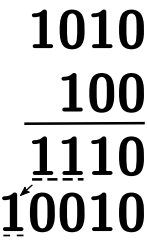
Začneme sčítáním, je takové nejpřímočařejší. Kdybychom čísla sčítali stejně jako ve dvojkové nebo desítkové soustavě, vznikaly by (vysčítáním se o řád výš) nám dvojky a vyšší číslice, kterých bychom se obtížně zbavovali. Jak to tedy zajistíme, abychom se byli schopni zbavit dvojky všude a aby nám nevznikaly trojky a víc? Pomocí jednoduchého postupu ilustrovaného obrázkem 3. Nejdříve si vybereme jedno číslo, které budeme přičítat k druhému. Náš vybraný sčítanec si rozdělíme na čísla začínající jedničkou, za kterou jsou samé nuly. Tím si zajistíme, že se nám vysčítá pouze dvojka a pouze na jednom místě. Poté sečteme standardním způsobem, tedy přičítáme čísla na odpovídajících řádech.



Obrázek 3: Příklad sčítání ve Fibonacciho soustavě

Můžou nám nastat různé situace, a sice že máme nad sebou právě dvě jedničky, tady nám vychází v součtu dvojka. Jak se této dvojky zbavíme? Vznikla jako 1+1, tedy jednu z těchto jedniček tam necháme a druhou rozdělíme jako součet dvou předchozích řádů s hodnotou jedna, tedy máme alespoň tři jedničky za sebou. Tyto jedničky si rozdělíme do dvojic odpředu a vysčítáme o řád výš. Jak víme,

že nám nevznikne další dvojka vepředu? Protože nemůžeme mít dvě jedničky za sebou (protože bychom je mohli vysčítat dopředu). A pokud nám vznikne dvojka o dva řády níže než ta původní, tak se jí zbavíme stejným způsobem jako té předchozí. Pomocí tohoto postupu buď získáme číslo v našem pěkném zápisu, nebo dojdeme na konec, kde existuje jen konečný počet možností. Některé z nich mohou způsobit vznik dvou jedniček za sebou, což může zahájit sérii sčítání po sobě jdoucích jedniček, která možná ovlivní i nejvyšší řád čísla.



Obrázek 4: Příklad sčítání ve Fibonacciho soustavě

Pokud nám dvojka nevznikne? Tak vysčítáme případnou dvojici jedniček (v případě trojice sčítáme od nejvyššího řádu, jako na obrázku 4). Přitom nám dvojka nevznikne (protože nemůžeme mít dvě jedničky za sebou), takže nanejvýš zase vysčítáme jedničky o řád výš, případně znovu posuneme o řád. Jednu část čísla jsme tedy přičetli. Toto opakujeme pro každou jedničku vybraného sčítance.

Jak by vypadaly ostatní operace? Úpravou sčítání můžeme snadno získat násobení, protože to je vlastně opakované sčítání. A odčítání je operace opačná ke sčítání. Nejprve si rozdělíme číslo, které budeme odečítat od toho druhého, stejně jako při normálním sčítání. Poté vezmeme jednu část a tu odečteme. Pokud bychom odečítali od nuly jedničku, tak musíme některou z jedniček na vyšším řádu (řekněme nejbližší vyšší) rozdělit na dvě na nižších řádech. Toto budeme opakovat, dokud nedostaneme jedničku právě na místě, kde ji máme odečíst, a pak už jen zapíšeme na příslušném řádu nulu a nové číslo sepíšeme. Pěkný zápis bude vždy výsledkem, protože jediné místo, kde bychom měli dvě jedničky za sebou je tam, kde jsme odečítali, tedy jedna zmizí. Pokud odečítáme jedničku od jedničky, tak pouze bude mít příslušný řád hodnotu nula a číslo se na jiném místě nezmění. Tyto postupy opakujeme v příslušných situacích pro každou část rozděleného čísla.

Nevýhodou této soustavy je velká časová složitost sčítání – v každém kroku hrozí, že budeme muset projít celé číslo, takže přičtení jedničky v nějakém řádu trvá $\mathcal{O}(n)$, takže obecně sčítání má časovou složitost $\mathcal{O}(n^2)$, takže násobení dokonce $\mathcal{O}(n^3)$.

Kódování

Fibonacciho čísla by mohla mít praktické užití právě v kódování dlouhých čísel. Představme si, že chceme zapsat řadu velkých čísel pouze pomocí nul a jedniček (tedy aby byla čísla např. snadno čitelná pro počítač) bez přítomnosti dělících znaků. Jak bychom to mohli udělat, abychom byli schopni poznat, kde číslo končí a kde začíná?

První možnost pouze pomocí Fibonacciho soustavy je docela prostá. Víme, že pokud máme Fibonacciho číslo v pěkném tvaru, nenastane nám, že by v jeho zápisu byly dvě jedničky za sebou. Tedy zapíšeme první číslo a za něj přidáme jedničku. Víme, že další číslo musí začínat jedničkou (nula před číslem nemá význam), tedy se vyskytnou ve výsledném řetězci dvě jedničky za sebou. Při dekódování na takovém místě zastavíme, oddělíme před dvojicí jedniček, pak tu první zahodíme a čteme další číslo. Odhadem pro růst Fibonacciho čísel je zlatý řez, takže počet cifer je odhadnutelný logaritmem o základu zlatého řezu. Například číslo 101001010, neboli 83 v desítkové soustavě, zapíšeme jako 1 101001010.

Další způsob, jak takhle čísla zapsat, je např. pomocí binární soustavy, ale nemůžeme použít trik s jedničkou, protože v binární soustavě můžou být dvě jedničky za sebou. Tedy jak bychom mohli čísla oddělit? Nabízí se prosté řešení, které využívá faktu, že číslo musí vždy začínat jedničkou. Připíšeme-li před něj nějaký počet nul, hodnotu nezměníme. Tak co kdybychom před něj předepsali tolik nul, kolik cifer má následující zapisované číslo, ty spočítali a potom přečetli právě tolik cifer? Pokud bychom chtěli zápis zkrátit tím, že zapíšeme nul o konstantu méně, měli bychom problém zapsat malá čísla jako třeba nulu a jedničku, které mají pouze jednu cifru. Kontrola správnosti (ale nikoli jistá) by mohla být, že pokud máme více než jednu nulu (jednou nulou může být zapsáno i samotné číslo nula),

musí další čtené číslo začínat jedničkou. Hlavní nevýhoda tohoto zápisu je, že na rozdíl od předchozího nám případná chyba ovlivní celou řadu čísel a ne jen jedno (pokud bychom měli chybu v předchozím zápisu, můžeme prostě navázat za další dvojicí). Přestože odhad počtu cifer je velký, dvojnásobek dvojkového logaritmu, výhodou tohoto zápisu je čtení bez převádění z Fibonacciho soustavy. 83 je takto 0000000 101001010.

Toto je poněkud zdlouhavé, dalo by se to zapsat i kratším zápisem, a sice třeba pokud bychom samotné číslo zapsali binární soustavou a před něj napíšeme ve Fibonacciho soustavě pozpátku počet cifer. To nám zkrátí zápis oproti předchozímu docela značně. Dělící znak jsou znovu dvě jedničky, protože každé (i binární i ve Fibonacciho soustavě zapsané) číslo musí začínat jedničkou. Ve Fibonacciho soustavě nemáme dvě jedničky za sebou, tedy víme, že jakmile narazíme na dvě jedničky, musíme přestat číst délku, obrátit ji a pak přečíst takto dlouhé číslo. Tento zápis má délku pouze dvojkový logaritmus plus logaritmus o základu zlatého řezu z tohoto dvojkového logaritmu. Zápis čísla 83 je 0101 1010010, protože má 7 (tedy 1010) cifer.

Teoreticky je pro velká čísla ještě výhodnější takzvaný rekurzivní binární zápis. Zde se před číslo zapíše jeho délka minus jedna (alespoň jedna cifra je vždy), před ni její délka a tak dále, až než se dostaneme k dvojcifernému číslu 10 nebo 11 (čísla 1 a 0 jako taková je potřeba zapsat s jedničkou před nimi). Aby systém poznal, že daná hodnota není opět počet cifer, za číslo se napíše nula – každý počet cifer totiž začíná na jedničku. V tomto případě se 83 napíše 10 110 1010010 0.

Výsledková listina 2. čísla

				Úlohy						
Poř.	**Jméno**	**R.**	$\sum_{-1}$	**t1**	**t2**	**t3**	**t4**	**k**	$\sum_{0}$	$\sum_{1}$
1.	Doc.MM J. Havelka	4	129,4	5,0	11,0		9,0		25,0	25,0
2.	Mgr.MM K. Vokálová	4	33,6	5,0	8,7	1,5	6,0		21,2	21,2
3.	Bc.MM J. Knillová	1	19,1	5,0	5,1	3,0	6,0		19,1	19,1
4.	Mgr.MM K. Hloušková	4	48,4	4,9	5,0	3,0		5,0	17,9	17,9
5.	Mgr.MM L. Kunčarová	4	20,6	5,0	5,0		7,0		17,0	17,0
6.	Bc.MM J. Kaifer	4	19,8	3,5	12,5				16,0	16,0
7.	Mgr.MM E. Vítková	4	45,6	5,0	0,8	3,0	5,5		14,3	14,3
8.	Mgr.MM T. Flídr	2	39,5	5,0	3,5			5,0	13,5	13,5
9.–10.	Mgr.MM O. Gonzor	3	42,7	5,0	7,8				12,8	12,8
	Bc.MM D. Perout	3	12,8	5,0	7,8				12,8	12,8
11.	Bc.MM K. Pernicová	3	12,4	4,9	7,5				12,4	12,4
12.	Bc.MM J. Kvapil	2	15,7	5,0	7,0				12,0	12,0
13.	Bc.MM A. Opl	2	11,5	5,0	6,5				11,5	11,5
14.	Mgr.MM B. Kopčák	4	28,6	5,0	4,1	2,0			11,1	11,1
15.	Bc.MM V. Janáček	3	10,5	5,0	5,5				10,5	10,5

Poř.	Jméno	R.	$\sum_{-1}$	Úlohy t1	t2	t3	t4	k	$\sum_0$	$\sum_1$
16.	Bc.[MM] O. Piroutek	2	10,0	5,0			5,0		10,0	10,0
17.	M. Fof	2	9,7	5,0	4,7				9,7	9,7
18.	A. Žáčková	3	9,5	5,0	4,5				9,5	9,5
19.	Bc.[MM] L. Vomelová	4	15,3	4,7	3,5				8,2	8,2
20.–22.	Mgr.[MM] O. Chlubna	3	42,8	5,0		3,0			8,0	8,0
	Dr.[MM] M. Kalousková	4	58,3	5,0	3,0				8,0	8,0
	Bc.[MM] V. Žák	4	11,6	5,0	3,0				8,0	8,0
23.	Mgr.[MM] J. Piroutek	4	21,1	4,4	3,5				7,9	7,9
24.	Mgr.[MM] M. Boček	1	29,2	5,0	2,8				7,8	7,8
25.–26.	J. Bláhová	3	6,5	4,0	2,5				6,5	6,5
	Mgr.[MM] M. Holubička	4	44,3	4,5		2,0			6,5	6,5
27.	J. Jedlička	2	5,7	5,0	0,7				5,7	5,7
28.	Bc.[MM] E. Neumannová	3	15,3	1,2	4,0				5,2	5,2
29.–30.	M. Bučková	3	9,9	5,0					5,0	5,0
	J. Kalvoda	3	5,0	5,0					5,0	5,0
31.	Bc.[MM] F. Bujnovský	2	12,2	3,4					3,4	3,4
32.	M. Turinská	3	3,0	2,0					2,0	2,0
33.	Bc.[MM] M. Vícha	1	12,4		1,4				1,4	1,4

Sloupeček $\sum_{-1}$ je součet všech bodů získaných v našem semináři, $\sum_0$ je součet bodů v aktuální sérii a $\sum_1$ součet všech bodů v tomto ročníku. Tituly uvedené v předchozím textu slouží pouze pro účely M&M.

Kontakty:

M&M, OPMK, MFF UK
Ke Karlovu 3
121 16 Praha 2

E-mail: mam@matfyz.cz
Web: mam.matfyz.cz
FB: casopis.MaM

matfyz
M&M