STUDENTSKÝ ČASOPIS A KORESPONDENČNÍ SEMINÁŘ

Ročník XXVI
Číslo 3

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,

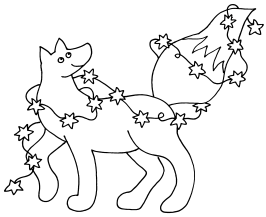
poslední barevné listy už opustily koruny stromů a přichází zima. Dny se krátí a světla ubývá, což přímo vybízí k trávení večerů u tepla krbu. Protože ale určitě nechcete jen tak zahálet, přinášíme vám třetí číslo.

V tématku o hrách si můžete přečíst o dalších typech her a zúčastnit se turnaje, ve kterém si zahrajete proti ostatním řešitelům. Ve Výpočetních modelech se podíváme na to, jak porovnávat rychlost strojů (odborněji časovou složitost) a jak je rozšířit o práci s celými čísly pomocí ALU. Třetí tématko si dá jeden díl pauzu a z elektromagnetismu tentokrát rozebereme chování kondenzátorů.

Na začátku prosince proběhla v Zábřehu na Moravě tradiční M&Mí vánoční víkendovka. Popovídali jsme si, zahráli deskové hry a zazpívali u kytary, po sobotním výletě si předali dárky, a příjemnou vánoční atmosféru dotvořila dokonce i trocha přítomného sněhu. Fotky již najdete na našem webu.

Přejeme vám všem veselé Vánoce a šťastný nový rok (nejen s M&M) a doufáme, že se v příštím roce s mnohými z vás setkáme na našich akcích.

Vaši organizátoři



Zadání a řešení témat

Termín odeslání úloh 3. série: 6. 2. 2020

Téma 1 – Hry

Díl 3: Matice, bimatice a turnaj

V tomto díle završíme část o maticových hrách, začneme studovat bimaticové hry a vyhlásíme první turnaj!

Maticové hry

Minule jsme si zkusili na malém příkladu, jak se hledá optimální (smíšená) strategie v maticové hře. Nyní bychom vám rádi představili obecný algoritmus na její hledání. Jedná se o pokročilou věc, ale přijde nám, že je to hezká konstrukce, kterou bychom vám rádi nabídli, byť ji pochopí jen ti s teoretickým zázemím, jež tu nemáme prostor vybudovat od základů. Její znalost nebude potřeba na pochopení zbytku tématka, takže tuto pasáž klidně přeskočte (a začněte číst rovnou

bimaticové hry, nebo skočte na Problém 1, protože k jeho vyřešení nejsou potřeba žádné znalosti z tohoto tématka).

Mějme hru, která je definovaná maticí $A \in \mathbb{Q}^{m\times n}$. Hledáme optimální strategii x^* pro prvního hráče, která je (formálně, ale nekonstruktivně) definovaná jako $x^* = \underset{x\in S_1}{\operatorname{argmax}}(\min_{y\in S_2}(x^T A y))$. Využijeme k tomu tzv. lineární programování[1].

Proměnné:	$v, x_1, \ldots, x_m$
Podmínky:	$x_1, \ldots, x_m \geq 0$
	$x_1 + \cdots + x_m = 1$
	$\forall j \in \{1, \ldots, n\} : a_{1,j}x_1 + a_{2,j}x_2 + \cdots + a_{m,j}x_m \geq v$
Cíl:	maximalizovat v

Obdobně sestavíme lineární program, který hledá optimální strategii pro druhého hráče.

Proměnné:	$w, y_1, \ldots, y_n$
Podmínky:	$y_1, \ldots, y_n \geq 0$
	$y_1 + \cdots + y_n = 1$
	$\forall i \in \{1, \ldots, m\} : a_{i,1}y_1 + a_{i,2}y_2 + \cdots + a_{i,n}y_n \leq w$
Cíl:	minimalizovat w

Slovní popis: První hráč hledá takovou konvexní kombinaci řádků matice A, aby měla co nejvyšší minimum výsledného řádku. Druhý hráč hledá takovou konvexní kombinaci sloupců matice A, aby měla co nejnižší maximum výsledného sloupce.

Uvedené lineární programy jsou navzájem duální[2]. Z toho plyne, že $v = w$. Jak máme tuto rovnost interpretovat? I kdyby musel první hráč předem oznámit, jakou smíšenou strategii bude volit (jaké budou pravděpodobnosti voleb jednotlivých řádků), tak na základě toho nemůže druhý hráč zvolit strategii, která by mu přinesla vyšší průměrný zisk, než by mohl bez této znalosti. To samé platí, kdyby druhý hráč předem oznámil svoji smíšenou strategii prvnímu hráči (jaké budou pravděpodobnosti voleb jednotlivých sloupců). Formálně to můžeme vyjádřit:

$$\max_{x\in S_1}(\min_{y\in S_2}(x^T A y)) = \min_{y\in S_2}(\max_{x\in S_1}(x^T A y))$$

Z toho plyne, že se jedná o sedlový bod na spojitém prostoru kartézského součinu všech smíšených strategií prvního a druhého hráče. Tento sedlový bod má každá maticová hra, i v případě, že matice definující hru žádný sedlový bod nemá (a pokud ho náhodou má, je optimální ryzí strategie rovna některé z optimálních smíšených strategií).

[1] `https://iti.mff.cuni.cz/series/2006/311.pdf` strana 4 až 15
[2] `https://iti.mff.cuni.cz/series/2006/311.pdf` strana 79 až 85

I když řešení úloh pomocí lineárního programování vypadá komplikovaně, je možnost řešit maticové hry pomocí lineárních programů skvělá. Máme jasný návod na hledání optimálních strategií a můžeme velmi dobře analyzovat jejich vlastnosti.

Zatímco nerovnost „$\max\min \leq \min\max$“ je snadné dokázat (viz úloha níže, ve které toto tvrzení zobecníme na jakékoliv funkce dvou proměnných), druhá nerovnost „$\max\min \geq \min\max$“ je podstatně hlubší a velice hodnotné zjištění. U složitějších typů her (například u bimaticových her, viz následující kapitola) už nebude existovat obecný algoritmus na hledání optimální strategie. Dokázat nějaké jejich vlastnosti bude podstatně náročnější.

Problém 1 [5b]: *Nechť X a Y jsou neprázdné množiny. Nechť f je funkce typu $X \times Y \to \mathbb{R}$. Dokažte, že:*

$$\max_{x\in X}(\min_{y\in Y} f(x,y)) \leq \min_{y\in Y}(\max_{x\in X} f(x,y))$$

Poznámka: Toto tvrzení dokažte zcela obecně a formálně. Nepoužívejte při tom analogie vztažené k teorii her. Opravdu zde mluvíme o jakékoliv funkci f, jejímž vstupem mohou být prvky konečné množiny, reálná čísla, pravděpodobnostní vektory, binární řetězce, štěňátka, koťátka, morčátka, prostě cokoliv.

Bimaticové hry

Nyní se přesuneme k bimaticovým hrám. Zatímco u maticových her byl součet zisku obou hráčů fixní (typicky nula), u bimaticových her toto omezení není. Díky tomu mohou bimaticové hry umožnit spolupráci mezi hráči.

Nyní musíme zapisovat každou situaci dvěma hodnotami: zisk prvního hráče a zisk druhého hráče. Budeme je psát jako dvojici čísel oddělenou zpětným lomítkem, přičemž zde bude platit pro oba hráče, že vyšší hodnota je lepší. I nadále platí, že první hráč si vybírá řádek, druhý hráč si vybírá sloupec; a tyto dvě volby musí být provedeny „naráz“, tj. bez znalosti aktuální volby soupeře. Podívejme se na příklad hry, kde je snadné navázat spolupráci.

$0 \setminus 0$	$1 \setminus 3$
$2 \setminus 1$	$4 \setminus 4$

Pojem bimaticové hry vychází z toho, že se vlastně jedná o dvě matice zapsané do jedné tabulky. Výplatní matice prvního hráče je:

0	1
2	4

Výplatní matice druhého hráče je:

0	3
1	4

Zde je pro oba hráče jednoznačně lepší zvolit strategii 2. Takže ji oba zvolí, čímž každý získá 4 body. Navázání spolupráce mezi hráči tu nevyžaduje vzájemnou důvěru, protože sobecké optimum se zde shoduje se skupinovým optimem. Těžší to bude tady:

$3 \setminus 3$	$0 \setminus 5$
$5 \setminus 0$	$1 \setminus 1$

Zde je skupinové optimum $3 \setminus 3$, což nastává, pokud oba hráči zvolí strategii 1. Problém je, že když se na to podívá každý hráč individuálně (sobecky), volba strategie 2 vede vždy k lepšímu výsledku. Tato volba umožňuje získat 5 bodů místo 3 bodů (když druhý hráč volí strategii 1), případně 1 bod místo 0 bodů (když druhý hráč volí strategii 2). A když strategii 2 takto zvolí oba hráči, tak se společně dostanou do podstatně horší situace $1 \setminus 1$. Je zde nesoulad mezi individuálním a skupinovým zájmem.

Uvedený příklad je široce znám pod názvem vězňovo dilema[3]. Předpokládejme dál, že jsme sobečtí jedinci. Jaká strategie se vyplatí více? Pokud hru hrajeme jen jednou, tak už víme, že je to strategie 2. Ale co když budeme se stejným soupeřem hrát tuto hru každý den? Nedává větší naději, že se pokusíme navázat spolupráci pomocí strategie 1? Když soupeř taktéž zvolí strategii 1, získáme každý 3 body. V tu chvíli bychom mohli přepnutím na strategii 2 získat 5 bodů místo 3 bodů, ale to je krátkozraké řešení. Lze totiž očekávat, že soupeř ztratí ochotu s námi spolupracovat a nadále bude i on volit strategii 2. Pravděpodobně pak skončíme v situaci, kde každý hráč získává jen 1 bod. Tím, že hrajeme hru opakovaně, se i sobeckému hráči může vyplatit spolupracovat. Bohužel se však nejedná o absolutní pravdu, ale stojí to na předpokladech o soupeřově chování. Pokud by soupeř vybíral strategii podle předpřipraveného plánu bez zohlednění naší strategie, nebo kdyby soupeř strategii volil zcela náhodně, bylo by pro nás výhodnější volit vždy strategii 2.

Všimněte si, že se jedná o tzv. symetrickou hru. To znamená, že když je na pozici $[i,j]$ hodnota $a \setminus b$, tak na pozici $[j,i]$ musí být hodnota $b \setminus a$ (k čemuž samozřejmě musí být počet řádků roven počtu sloupců). Jinými slovy, oba hráči mají ve hře stejné podmínky.

[3] `https://sk.wikipedia.org/wiki/V%C3%A4z%C5%88ova_dilema`

Je zřejmé, že každou maticovou hru lze ekvivalentně zapsat jako bimaticovou hru. Uvažme například maticovou hru z minulého čísla:

1	−2
3	−4
−1	2

Tu můžeme přepsat do bimaticové podoby takto:

+1 \ −1	−2 \ +2
+3 \ −3	−4 \ +4
−1 \ +1	+2 \ −2

Definici symetrické hry teď můžeme aplikovat i na maticové hry čtvercového tvaru. V řeči maticových her můžeme symetrii vyjádřit takto:

$$\forall i \in \{1,2,\ldots,m\}\ \forall j \in \{1,2,\ldots,n\} : a_{j,i} = -a_{i,j}$$

Známá hra kámen-nůžky-papír je příkladem symetrické maticové hry, protože má nulový součet a zároveň mají oba hráči stejné možnosti a stejné výplaty. Z toho pro symetrické maticové hry mimo jiné plyne, že na diagonále musí být samé nuly[4].

			
	0	1	−1
	−1	0	1
	1	−1	0

K tématu bimaticových her se budou vázat následující dvě úlohy a taky turnaj.

Problém 2 [3b]: *Určete optimální strategii pro oba hráče v této bimaticové hře (bez opakování). Jinými slovy, pokuste se z pohledu každého hráče získat co nejvíce bodů za předpokladu, že druhý hráč je racionální a také se snaží maximalizovat svůj zisk.*

4 \ 3	5 \ 1	6 \ 2
2 \ 1	8 \ 4	3 \ 6
5 \ 9	9 \ 6	2 \ 8

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

Problém 3 [6b]: *Ve městě stojí dva stánky se zmrzlinou. Stánky si vybírají cenu zmrzliny mezi cenami 10 Kč, 20 Kč a 25 Kč. Předpokládejme, že za jinou cenu nelze zmrzlinu prodávat. Dále předpokládejme, že oba stánky mají nekonečnou zásobu zmrzliny (takže ji nemusejí nakupovat) a že cena chlazení a nabírání zmrzliny je zanedbatelná. Očekává se, že během jednoho letního měsíce si zmrzlinu v tomto městě koupí 6000 turistů a 4000 místních obyvatel. Turisté si vybírají stánek se zmrzlinou uniformně náhodně, takže můžeme předpokládat, že do každého stánku půjde přesně polovina turistů. Místní obyvatelé si vždy půjdou koupit zmrzlinu do stánku, kde je levnější; a pokud se ceny shodují, přesně polovina z nich půjde do každého stánku. Dále předpokládejme, že si každý návštěvník koupí jen jednu zmrzlinu (pokud by si chtěl koupit dvě, budeme ho počítat jako dva návštěvníky). Jakou cenu zmrzliny mají jednotlivé stánky zvolit, když chce každý maximalizovat svou tržbu?*

Nápověda na řešení:

Uvedené hry v Problému 2 a Problému 3 budeme hrát bez opakování, takže je potřeba se rozhodovat čistě sobecky, neboť tu není možnost navázat spolupráci.

Všimněte si toho, že pokud je nějaká strategie prvního hráče vždy horší než jiná konkrétní strategie prvního hráče, tak tu horší strategii může první hráč ignorovat. Druhý hráč může provést stejnou úvahu, že první hráč nikdy nepoužije tu horší strategii, což zase může pomoci zúžit volby druhého hráče. Ukažme si to na příkladu bimaticové hry 2×2:

$9 \setminus 4$	$0 \setminus 3$
$8 \setminus 7$	$20 \setminus 6$

Může první hráč říct, že nějaká z jeho strategií je vždy horší než ta druhá možnost? Zatím nevíme. Může druhý hráč říct, že nějaká z jeho strategií je vždy horší než ta zbývající? Ano! Druhý hráč může zavrhnout strategii 2, protože s ní v obou případech získá méně bodů než se strategií 1. Strategie druhého hráče je tedy zvolena. Když stejnou úvahu provede první hráč, situace se mu zredukuje na tuto:

$9 \setminus 4$
$8 \setminus 7$

Zde první hráč vidí, že má zvolit strategii 1. Takže první hráč zvolí strategii 1, přestože se mohla zpočátku zdát méně výhodnou. Výsledný "zobecněný sedlový bod" dává prvnímu hráči zisk 9 bodů a druhému hráči zisk 4 body.

Všimněte si, že vzniklý výsledek $9 \setminus 4$ je mnohem horší než výsledek $20 \setminus 6$, který se také v naší bimatici nachází. Bohužel je však pro dvojici racionálních hráčů nemožné k tomuto výsledku dojít, pokud se nehraje na více kol.

Zkuste aplikovat podobný postup na hry v úlohách výše.

Turnaj

Nyní bude na vás rozhodnout se, jak opakovanou hru hrát. Nezůstaneme však u vězňova dilematu, to už je moc ohrané[5]. Připravili jsme pro vás obdobu této hry. Naše hra bude také symetrická a opět v ní budou střety mezi sobeckými a skupinovými zájmy, ale tentokrát budou na výběr čtyři strategie. Tuto hru definuje následující matice:

\	A	B	C	D
A	6 \ 6	3 \ 1	3 \ 1	3 \ 1
B	1 \ 3	2 \ 2	8 \ 0	8 \ 0
C	1 \ 3	0 \ 8	10 \ 10	1 \ 18
D	1 \ 3	0 \ 8	18 \ 1	4 \ 4

V turnaji je vaším úkolem vytvoření skriptu v Pythonu[6] (potřebná verze Python 3.6), který bude umět hrát výše popsanou bimaticovou hru.

Pravidla pro jeho vytvoření jsou jednoduchá. Než si je popíšeme, stáhněte si prosím zdrojové kódy z repozitáře: `https://github.com/sauermar/Turnaj-M-M` a nahlížejte do nich současně se čtením tohoto návodu. Nyní se můžeme podívat na jednotlivá pravidla.

Zaprvé skript musí implementovat rozhraní ze souboru `player.py`, to znamená dodržovat podmínky, co musí vracet a co mají k dispozici jednotlivé metody popsané v rozhraní. Přičemž metoda je v Pythonu normální funkce, která se volá s objektem jako s prvním parametrem. Pokud si chcete práci ulehčit, můžete si zkopírovat vzorového hráče `mirror.py` (ten již rozhraní implementuje) a pouze jeho metody upravovat podle své strategie.

Rozhraní obsahuje inicializátor `__init__`, což je funkce, která se zavolá při vytvoření objektu vaší třídy. Tento objekt se bude vytvářet na začátku každého souboje během turnaje a tedy, bude se volat i váš inicializátor. Proto je inicializátor dobrý pro vytvoření atributů třídy, které jsou víceméně vlastní lokální proměnné objektu, vhodné například na zapamatování předchozího stavu hry. Metodu `author_name` pouze upravte, aby vracela string s vaším (celým) jménem.

Metoda `next_move` vrací následující hráčův tah prostřednictvím typu `Move`. Jedná se o datový typ, pomocí kterého popisujeme možné tahy, tedy řádky (příp. sloupce) matice uvedené výše.

[5] `https://axelrod.readthedocs.io/en/stable/reference/overview_of_strategies.html`

[6] Pokud jste se s Pythonem nesetkali, doporučujeme si projít tuto stránku: `https://naucse.python.cz/`

```
class Move(enum.Enum):
    a_safe_way = ("A", 0)
    betray = ("B", 1)
    cooperate = ("C", 2)
    deceive = ("D", 3)
```

Pro použití enumerace tahů je třeba typ `Move` z `player.py` naimportovat do vašeho skriptu. To jde udělat připojením `from player import Move` na začátek zdrojového kódu. Následně jde v kódu enumerace tahů použít dvěma způsoby `Move.a_safe_way`, odkazem na název, nebo `Move(("A", 0))`, odkazem na hodnotu.

A nakonec, skrz metodu `reward`, obdržíte instanci třídy `Result`, odkud je dostupný váš tah, oponentův tah a získané body. Jednotlivé metody použitelné na tomto objektu jsou definované v souboru `result.py`.

Jedná se o metodu `get_my_score`, která na základě zvolených tahů (strategií) vrátí vaše získané skóre, a `get_opp_score`, která stejným způsobem vrací soupeřovo získané skóre. Třída `Result` také obsahuje inicializátor, který k instanci objektu přiřadí atributy `my_move` a `opp_move`, dostupné přes tečku.

Nyní už znáte vše potřebné pro vytvoření vlastního skriptu, a jeho další vylepšení jsou tedy pouze na vás! Pokud by ale stále nebylo něco jasné, k dispozici jsou i „jednoduché“ vzorové skripty hráčů, ze kterých můžete vycházet při implementaci rozhraní. Jedná se o soubory `always_cooperate.py`, `always_deceive.py`, `unforgiving.py`, `score_counting.py` a `mirror.py`.

Sami si s dokončeným skriptem můžete zahrát pomocí souboru `testing.py`, kterému v konzoli zadáte název vašeho skriptu tečka název vaší třídy (která implementuje rozhraní `Player`) jako první argument a případně počet iterací (kol) hry jako druhý. Výchozí počet iterací je nastaven na 10. Nezapomeňte, že váš skript se musí nacházet ve stejné složce, kde je uložen soubor `testing.py`, jinak ho program nedokáže nalézt.

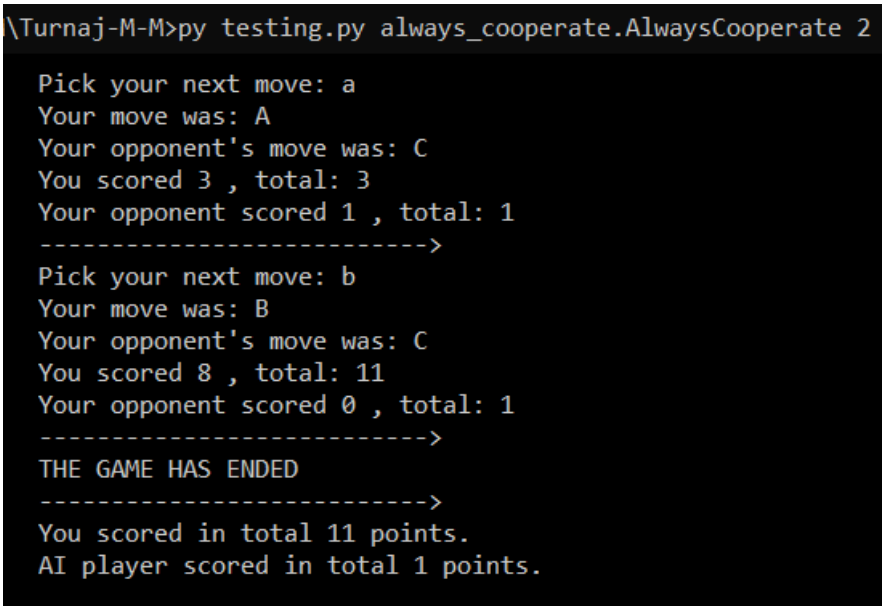
Ukázku hry v konzoli proti hráči always_cooperate můžete vidět na Obrázku 1.

Nakonec, až budete se svým hráčem naprosto spokojeni, stačí nám zdrojový kód skriptu odeslat e-mailem. My ho po termínu odevzdávání necháme zahrát se všemi ostatními skripty a následně sestavíme výsledkovou listinu podle součtu počtu bodů ze všech interakcí. Je tedy mnohem důležitější získat v každé interakci co nejvíce bodů, než abyste svým počtem bodů překonali soupeřův počet bodů. Až podle umístění ve výsledkové listině vám pak udělíme body do M&M. Můžeme vám však slíbit, že každý účastník turnaje získá alespoň 1 bod.

Všechny skripty budou mezi sebou hrát po dobu předem stanoveného počtu kol, jehož přesnou hodnotu vám nemůžeme prozradit, ale pro představu uvádíme, že to bude číslo mezi 100 a 200.

Pokud byste měli dotazy k tomu, jak psát skript hrající hru, napište nám prosím dotaz prostřednictvím e-mailové konference (viz níže). Do turnaje se kromě

vás, řešitelů M&M, zapojí i další hráči, ale tím se rozhodně nechte odradit od účasti!



Obrázek 1: Testování hry proti always_cooperate

Závěr

Dnes jsme zakončili část o maticových hrách tím, že jsme zformulovali algoritmus, který najde optimální strategii pro jakoukoliv maticovou hru. Poté jsme si zadefinovali nový typ her, bimaticové hry, které nabízí podstatně větší možnosti. Tyto hry se hodí k popisu mnoha situací v reálném světě, protože zisk pro jednoho člověka nemusí představovat ztrátu pro druhého člověka. Buďme vděční za tento svět!

Pokud vás něco z toho zaujalo a chtěli byste nám sdělit svoje postřehy, nebojte se napsat vlastní příspěvek k tomuto tématku. Rádi ho otiskneme a odměníme vás body.

Co nás čeká příště? Vyhlásíme výsledky turnaje, spustíme další turnaj a taky si povíme něco o využití teorie her v biologii.

Řešení 1. série

Problém 1

Zadání:
Nechť $a_{i,j}, a_{k,l}$ jsou sedlové body matice A. Dokažte, že $a_{i,j} = a_{k,l}$.

Řešení:
Zaměříme se na čtveřici hodnot $a_{i,j}$, $a_{i,l}$, $a_{k,j}$, $a_{k,l}$. Mohou to být čtyři prvky matice (tvořící vrcholy obdélníka), ale také může platit $i = k$ nebo $j = l$, to pro důkaz není důležité.

- Protože $a_{i,j}$ je sedlovým bodem, máme $a_{k,j} \leq a_{i,j} \leq a_{i,l}$.
- Protože $a_{k,l}$ je sedlovým bodem, máme $a_{i,l} \leq a_{k,l} \leq a_{k,j}$.

Když to poskládáme dohromady, získáme sérii nerovností:

$$a_{i,j} \leq a_{i,l} \leq a_{k,l} \leq a_{k,j} \leq a_{i,j}$$

Všimneme si, že první a poslední prvek je totožný. Tudíž platí:

$$a_{i,j} = a_{i,l} = a_{k,l} = a_{k,j} = a_{i,j}$$

Tím pádem jsou všechny uvedené prvky matice sedlovými body se stejnou hodnotou. ■

Problém 2

Zadání:
Vymyslete matici bez sedlového bodu. Stačí, když napíšete jeden příklad takové matice (bez vysvětlení).

Řešení:
Jako příklad matice bez sedlového bodu může sloužit kterákoliv matice z druhého dílu, třeba zmíněná hra kámen-nůžky-papír.

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

Téma 2 – Výpočetní modely

Pokud jste nečetli předchozí díly, doporučuji si je přečíst – najdete je na adrese `https://mam.mff.cuni.cz/rocnik/26` ve formě PDF. Alternativně je na webu tématka uvedená poněkud stručnější verze.

Díl 3: Čas a čísla

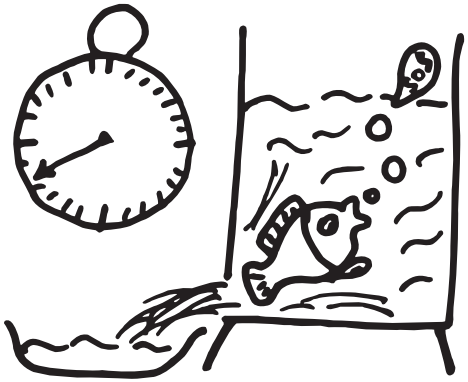
Časová složitost

Nejdříve se pojďme zamyslet nad tím, jak by se dala srovnávat rychlost námi navržených strojů. Jednotkou času pro nás bude krok. Budeme tedy zkoumat, kolik kroků pro daný vstup stroj provede. Všimněte si, že se může stát, že stroj poběží nekonečně dlouho. Těmto strojům se budeme snažit vyhnout.

Pokud je možných vstupů konečně mnoho, tak lze sestrojit stroj, který si pamatuje pro každý možný vstup správný výstup a místo výpočtu jen přečte vstup a najde správný uložený výsledek. Proto budeme většinou chtít, aby naše stroje byly schopné zpracovávat libovolně dlouhé vstupy.

Obecně nám půjde o složitost v nejhorším případě vzhledem k délce vstupu. Délku vstupu budeme značit n a pokud používáme standardní vstup nebo IO, je to číslo, které udává, po kolika krocích začne vypisovat „$“. Složitost v nejhorším případě znamená, že nás bude zajímat maximální počet kroků, které může stroj pro vstup dané délky potřebovat.

Abychom nemuseli počet kroků počítat přesně, zavádíme takzvanou „óčkovou notaci“. Óčková notace je horní odhad na počet kroků, které stroj provede, pokud zanedbáme multiplikativní konstanty a spustíme ho pro dostatečně dlouhý vstup. Pokud má tedy stroj časovou složitost $\mathcal{O}(n^2)$, znamená to, že pro dostatečně velká n existuje konstanta k taková, že stroj provede maximálně kn^2 kroků (například pro každé n větší než 100 provede stroj maximálně $42n^2$ kroků). Také si můžeme všimnout, že například algoritmus, který vždy provede $3n^4 + 2n + 5$ kroků, běží v čase $\mathcal{O}(n^4)$, jelikož $3n^4 + 2n + 5 < 5n^4$ pro každé $n > 1$ (ze stejného důvodu bychom mohli tvrdit, že tentýž algoritmus běží v čase $\mathcal{O}(2^n)$, tato informace by ale pro nás nebyla příliš cenná – vždy se snažíme o co nejtěsnější odhad.)



Práce s čísly

V mnoha případech by se nám hodilo umět pracovat s celými čísly. Všimněme si, že pracovat s konstantně velkými čísly můžeme už teď – využívali jsme toho při kontrolování dělitelnosti. Problém nastává, pouze pokud velikost čísla začne nějak záviset na vstupu. V takovém případě již není možné popsat aritmetické operace v popisu přechodové funkce. Jak v tomto tématku začíná býti zvykem, vyřešíme tento problém dalším rozšířením.

Aritmetická jednotka (tradičně označovaná ALU) je rozšíření, které v pokynu dostane dvě čísla a operaci, kterou s nimi má provést. V příštím kroku pak na výpis vypíše výsledek. Podobně jako u registru, také u ALU si můžeme vybrat, co vypisuje v prvním kroku. Pro začátek si povolíme následující operace, ale někdy bude zajímavé používat i jinou sadu.

- `+` (sčítání)
- `-` (odčítání)
- `*` (násobení)
- `/` (dělení, zaokrouhleno dolů)
- `%` (zbytek po dělení, takzvané modulo)

Čísla si dovolíme ukládat do stavových registrů i zapisovat je na pásku. Do fronty a zásobníku čísla raději ukládat nebudeme, protože z popisu by nebylo jasné, zda vkládáme vícecifernou konstantu, nebo několik znaků. Z podobného důvodu je na místě jistá opatrnost při používání čísel s IO. Pro jednoznačnost budeme vždy uvádět, zda na vstupu dostaneme bity, desítkové číslice, nebo zda v jednom kroku načteme celé číslo – občas na tom závisí složitost algoritmu.

```
vstup (čte čísla)
vstup (čte čísla)
stavový registr (počáteční stav A)
ALU
výstup (tiskne čísla)

$,$,a,0 → 0,0,0,/,a
$,$,a,b → b,a,b,%,
a,b,A,_ → b,a,b,%,
```

Stroj 1: Stroj implementující Eukleidův algoritmus na hledání největšího společného dělitele, který na vstupech dostane dvě celá čísla. Předpokládáme, že pokus o dělení nulou stroj zastaví.

Práce s čísly nám nově dovolí používat ještě jedno další rozšíření, a to je pole. Pole v pokynu vždy dostane dvě položky. První je vždy číslo a udává číslo buňky, jejíž dosavadní obsah má v dalším kroku vypsat na výpis, a pak jej nahradit druhou položkou. Pro jednoduchost si přidáme možnost dát poli jako druhou

položku podtržítko (_), v kterémžto případě zůstane obsah buňky nezměněn. Pokud není řečeno jinak, opět budeme předpokládat, že všechny buňky na začátku obsahují samé mezery.

```
vstup (čte čísla)
stavový registr (počáteční stav 0)
ALU (v prvním kroku vypisuje 0)
pole
výstup (tiskne čísla)

$,0,i,_ → 1,i,1,-,i,_,
a,0,i,_ → 0,i,1,+,i,a,
$,1,i,_ → 2,0,0,+,i,_,
$,2,_,k → 3,0,0,+,k,_,
$,3,_,x → 4,0,0,+,k,_,x
```

Stroj 2: Stroj, který pošle na výstup k-té číslo vstupu, kde k je poslední číslo, které dostal na vstupu. (Indexujeme od nuly.)

Problém 1 [2b]: *Popište, jak by šlo implementovat nedestruktivní čtení z pole bez použití podtržítka v pokynu. Úkolem tedy je popsat, jak by se dal upravit libovolný stroj používající pole na ekvivalentní (a podobně rychlý) stroj který poli nikdy podtržítko nepošle.*

Problém 2 [3b za všechny stroje]: *Určete časovou složitost strojů z předchozího čísla.*

Problém 3 [2b]: *Jak by se podle vás měla chovat aritmetická jednotka v případě dělení nulou? Stroj 1 se v takovém případě zastaví. Mohlo by být k něčemu užitečné, kdyby se stalo něco jiného? Jak a proč?*

Problém 4 [3b]: *Představte si, že nemáte k dispozici pole. Sestavte stroj, který jej simuluje – na jednom vstupu čte, kterou buňku má přečíst, na druhém, co tam má zapsat, a na výstup píše to, co přečetl. Váš stroj bude téměř jistě pomalejší než skutečné pole. Předpokládejme tedy, že se vstupy změní vždy až po vypsání přečteného obsahu buňky.*

Problém 5 [2b]: *Sestrojte stroj, který na vstupu dostane nejdříve posloupnost čísel $a_1 \dots a_n$, pak oddělovač (třeba „X“), a pak druhou posloupnost čísel $b_1 ... b_m \in \{1 \dots n\}$, a který má postupně vypsat b_i-té číslo z první posloupnosti. (Tedy vypíše $a_{b_1} a_{b_2} \dots a_{b_m}$.)*

Problém 6 [3b]: *Zkuste předchozí úkol vyřešit bez použití pole. Jaké jsou časové složitosti obou verzí?*

Problém 7 [3b]: *Časová složitost stroje 1 není omezitelná pomocí délky vstupu. Navrhněte podobný stroj, který ale na vstupech dostane binární zápis čísel, a zkuste omezit jeho složitost.*

Problém 8 [10b]: *Nyní už máme skutečně mocné stroje. Zkuste sestavit stroj, který dokáže simulovat nějaký jednoduchý mikroprocesor. Tohle je rozsáhlý úkol, nebojte se na něm spolupracovat.*

Poznámka: Na slibovaný nedeterminismus v tomto díle nakonec nezbylo místo. Nezoufejte, dočkáte se jej!

Řešení úloh z prvního dílu

Problém 1

Zadání:
Lze sestrojit stroj, který zjistí, zda je počet 0 sudý a počet 1 dělitelný třemi?

Řešení:
Ano, lze. Například tento:

```
vstup
stavový registr (počáteční stav 0)   (Zbytek po dělení počtu nul dvěma)
stavový registr (počáteční stav 0)
                                     (Zbytek po dělení počtu jedniček třemi)
```

```
0,0,0 → 1,0
0,0,1 → 1,1
0,0,2 → 1,2
0,1,0 → 0,0
0,1,1 → 0,1
0,1,2 → 0,2
1,0,0 → 0,1
1,0,1 → 0,2
1,0,2 → 0,0
1,1,0 → 1,1
1,1,1 → 1,2
1,1,2 → 1,0
```

Pokud nechceme mít dva stavové registry, tak je můžeme zkombinovat a mít jeden stavový registr se šesti stavy. Ty můžeme značit třeba „00, 01, 02, 10, 11, 12“, nebo „1,2,3,4,5,6“ – na tom nám zase tolik nesejde.

Problém 2

Zadání:
Lze sestrojit převodník, který provádí bitovou negaci?

Řešení:
Ano. Například tento:

```
IO

0 → 1
1 → 0
```

Problém 3

Zadání:
Lze sestrojit sčítačku? (Sčítačka na dvou vstupech dostane dvě přirozená čísla ve dvojkové soustavě a na výstup vypíše jejich součet.)

Řešení:
Ano. Potřebujeme pracovat s čísly od nejméně významného bitu. Je dobré nezapomenout, že čísla můžou být různě dlouhá a výsledek může být o bit delší než vstup.

```
vstup
vstup
výstup
stavový registr (počáteční stav 0)
```

(Přenášíme jedničku o řád výše, nebo ne.)

```
0,0,0 → 0,0
0,0,1 → 1,0
0,1,0 → 1,0
0,1,1 → 0,1
1,0,0 → 1,0
1,0,1 → 0,1
1,1,0 → 0,1
1,1,1 → 1,1
$,0,0 → 0,0
$,0,1 → 1,0
$,1,0 → 1,0
$,1,1 → 0,1
0,$,0 → 0,0
0,$,1 → 1,0
1,$,0 → 1,0
1,$,1 → 0,1
$,$,1 → 1,0
```

Problém 4

Zadání:
Lze sestavit stroj, který pozná dělitelnost jedenácti? Sedmi? Číslem zadaným na druhém vstupu?

Řešení:
Pro libovolné předem dané číslo k to jde. Budeme potřebovat vstup, který bude číst od nejméně významného bitu, a stavový registr, který bude mít stavy 0 až $k-1$ a začne ve stavu 0.

Přechodová funkce bude vypadat následovně: Pokud jsem ve stavu i a vstup vypsal bit b, přejdu do stavu s číslem odpovídajícím zbytku po dělení $2i+b$ číslem k. Všimněme si, že těchto $2i$ přechodů stačí spočítat „ručně“ při konstrukci stroje. Jak můžete vidět níže, není to moc složité :-)

Příklad stroje pro dělitelnost sedmi:

```
stavový registr (počáteční stav 0)
vstup

0,0 → 0
0,1 → 1
1,0 → 2
1,1 → 3
2,0 → 4
2,1 → 5
3,0 → 6
3,1 → 0
4,0 → 1
4,1 → 2
5,0 → 3
5,1 → 4
6,0 → 5
6,1 → 6
```

Pro dělitelnost číslem zadaným na vstupu to nejde – dejme tomu, že by takový stroj existoval a používal x stavů. Tudíž určitě neumí rozeznávat dělitelnost číslem $x+1$, což je spor a proto takový stroj nemůže existovat.

(Pokud by stroj měl vícero stavových registrů, tak je nejdříve zkombinujeme jako v prvním problému.)

Problém 5

Zadání:
Lze sestavit stroj, který pozná palindromy? (Palindrom je řetězec, který se čte zepředu stejně jako zezadu, např. 1101011.)

Řešení:
Nelze.

Dejme tomu, že takový stroj existuje a má x stavů. Všimněme si, že existuje více než x „půlvstupů“ délky x, a proto alespoň dva z nich způsobí, že stroj na jejich konci bude ve stejném stavu. (Tomuto se říká Dirichletův princip.)

Pokud tedy vezmeme dva takové půlvstupy a za oba připojíme ten první z nich napsaný pozpátku, tak dostaneme dva vstupy, z nichž jeden je palindrom, druhý není, a náš stroj je není schopen odlišit.

(Zde musím uznat, že mě překvapil nápad některých z vás, kteří si vyrobili stroj se dvěma vstupy a následně do jednoho z nich poslali původní vstup popředu

a do druhého pozpátku, čímž tento problém zvládli vyřešit. Striktně vzato to náš model asi spíše nedovoluje, bonusové body za to ale byly.)

Problém 6

Zadání:
Existují problémy, který naše stroje zatím nemohou vyřešit?

Řešení:
Ano. Třeba dělitelnost zadaným číslem a palindromy, jak je uvedeno výše. Takových problémů je spousta a většinou zahrnují potřebu pamatovat si proměnlivě mnoho informací. Pokud chcete zjistit více, tak doporučuji začít čtením o konečných stavových automatech. Pěkný úvod je třeba na Wikipedii:
`https://cs.wikipedia.org/wiki/Kone%C4%8Dn%C3%BD_automat`.

Problém 7

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

Řešení:
Viz další díly tématka. :-)

Problém 8

Zadání:
V čem by se lišily naše stroje, kdyby vstup po posledním znaku místo vypisování $ rovnou ukončil výpočet?

Řešení:
Mimo jiné by výstup nemohl být delší než vstup, což by nám dost vadilo například u sčítání.

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

Téma 3 – Zpracování obrazových dat ze senzorů

V tomto čísle další díl nenajdete, můžete nám ale dále posílat řešení úloh z předchozích dvou dílů, nebo též své vlastní rozšiřující nápady a postřehy. Předchozí díly najdete v archivu na `https://mam.mff.cuni.cz/rocnik/26/`. Úlohy z druhého dílu zde ve zkrácené verzi zopakujeme, plnou verzi včetně kódu v Pythonu najdete ve 2. čísle. Na pokračování tématka se můžete těšit v dalším čísle po novém roce.

Problém 1: *V prvním 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 prvním díle. Tento soubor opět přečtete stejným způsobem jako v prvním díle, projděte postupně všechny pixely načteného obrázku a aplikujte na každý pixel vámi vytvořený filtr.

Vyzkoušejte si, jestli váš filtr funguje na všechny obrázky (nebo aspoň na většinu), které máte k dispozici. K řešení využijte kód v jazyce Python zveřejněný ve 2. díle.

Data ke stažení:
`https://mam.mff.cuni.cz/media/prilohy/26-2-3-2D.zip`

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

Téma 4 – Vybrané kapitoly z elektromagnetismu

Díl 3: Kondenzátory

V tomto díle tématka se budeme věnovat kondenzátorům a výpočtu jejich kapacity. Kondenzátor je elektrotechnická součástka, která je charakterizována kapacitou. Skládá se ze dvou vodivých desek (elektrod), na které je přiveden opačný náboj. Mezi elektrodami je vrstva nevodiče (dielektrikum), která nedovolí nabitým elektrickým částicím přecházet z jedné elektrody na druhou. Kapacita je fyzikální veličina, která vyjadřuje schopnost kondenzátoru uchovat elektrický náboj. Vypočítá se

$$C = \frac{Q}{U}, \tag{1}$$

kde C je kapacita kondenzátoru, Q náboj na elektrodách a U napětí mezi elektrodami.

Velikost napětí mezi deskami A, B kondenzátoru můžeme vyjádřit pomocí elektrické intenzity E jako:

$$U = \frac{W}{Q} = \frac{\int_A^B F \mathrm{d}x}{Q} = \int_A^B \frac{F}{Q} dx = \int_A^B E \mathrm{d}x, \tag{2}$$

kde W je práce pro přenesení náboje Q z jedné elektrody na druhou a F síla působící na tento náboj.

Z těchto dvou vztahů jsme schopni vyřešit spousty úloh, kde máme zadaný tvar kondenzátoru, náboje na elektrodách a naším cílem je spočítat jeho kapacitu. Obecný postup je následující:

1. Uvědomíme si, že elektrody kondenzátoru jsou vodivé, a proto jsou všechny body jedné elektrody na stejném (el.) potenciálu, tedy můžeme uvažovat napětí mezi libovolnými dvěma body elektrod. Zvolíme tedy tyto dva body podle symetrie úlohy (typicky nejkratší vzdálenost).

2. Z Gaussova zákona (viz první díl tématka) si spočteme intenzitu elektrického pole E v závislosti na vzdálenosti x od jedné elektrody.

3. Spočteme napětí podle rovnice (2).

4. Dosadíme do vzorce (1) pro kapacitu kondenzátoru.

Problém 1 [3b]: *Zdůvodněte, proč je vždy po přiložení napětí na kondenzátor na obou elektrodách stejně velký (opačný) náboj. Který fyzikální jev to způsobuje a jak?*

Problém 2 [3b]: *Kulový kondenzátor tvoří dvě soustředné vodivé kulové slupky o poloměrech 5 cm a 10 cm. Na elektrody je přiveden náboj o velikosti 1 C. Jaká je jeho kapacita? Může takový kondenzátor existovat?*

Problém 3 [3b]: *Válcový kondenzátor tvoří dva dlouhé soustředné válce o poloměrech 20 cm a 26 cm. Na elektrody je přiveden náboj o velikosti 500 nC. Vypočítejte kapacitu kondenzátoru.*

Problém 4 [3b]: *Jaká je elektrická intenzita vně kondenzátorů v problémech 2 a 3?*

Problém 5: *Popište co nejpřesněji jevy na koncích válcového/deskového kondenzátoru a uveďte, jak ovlivňují kapacitu kondenzátoru.*

Vzorové řešení problémů z prvního dílu

V tomto čísle otiskujeme dvě řešení problémů z prvního dílu. Najdete zde postupně řešení různě nabitých vodivých koulí od Mgr.MM Lucie Kunčarové a řešení elektrického pole v okolí nekonečně velké nabité desky od Mgr.MM Jolany Knillové.

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

Elektrické pole v okolí vodivé koule nabité homogenně a na povrchu (7 b)

Mgr.MM Lucie Kunčarová

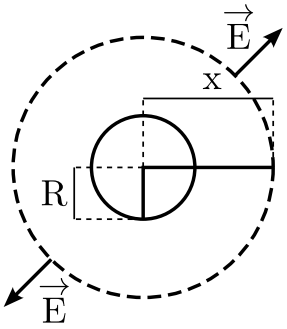
Gaussova plocha

- Stejně jako v okolí bodového náboje bude mít v okolí obou koulí elektrické pole sférickou symetrii.
- Zvolíme si tedy jako Gaussovu plochu kouli s poloměrem x, $x > R$, pro vypočtení elektrického pole vně nabitých koulí a kouli s poloměrem x, $x < R$, pro vypočtení elektrického pole uvnitř nabitých koulí.

Koule s homogenně rozmístěným nábojem

Jako první se zaměřím na kouli s homogenně rozmístěným nábojem.

Pole vně koule



Obrázek 2: Nákres zjišťování elektrického pole vně koule

Vektor E bude směřovat od středu koule. Bude ve všech směrech kolmý na povrch Gaussovy koule. Gaussův zákon tedy můžeme zjednodušit na

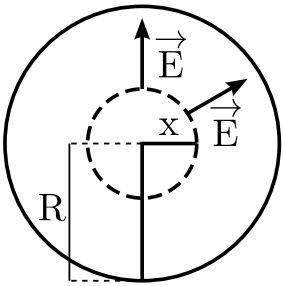
$$ES = \frac{Q}{\varepsilon_0},$$

kde S je povrch Gaussovy koule $S = 4\pi x^2$. Po dosazení dostáváme

$$E = \frac{Q}{4\pi\varepsilon_0 x^2}.$$

Tedy pole v okolí rovnoměrně nabité koule je stejné jako pole v okolí bodového náboje.

Pole uvnitř koule



Obrázek 3: Nákres zjišťování elektrického pole uvnitř koule

Na začátku budeme postupovat podobně.

$$E \cdot 4\pi x^2 = \frac{Q'}{\varepsilon_0},$$

kde Q' je náboj uvnitř Gaussovy plochy. Náboj je rozdělen rovnoměrně, část náboje v Gaussově kouli je úměrná jejímu objemu.

$$\frac{Q'}{Q} = \frac{4\pi x^3 \rho}{4\pi R^3 \rho} = \frac{x^3}{R^3} \implies Q' = \frac{Q x^3}{R^3}$$

Po dosazení vychází

$$E = \frac{Qx}{4\pi R^2 \varepsilon_0}.$$

Koule s nábojem rozmístěným pouze na povrchu

Pole vně koule $x > R$

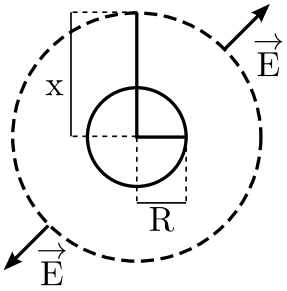
Náboj na sféře je rozmístěn symetricky, elektrické pole sféry je symetrické. Vektor E bude směřovat od středu koule, $E \perp S_G$, kde S_G je plocha Gaussovy koule. Gaussovu větu si tedy můžeme zjednodušit

$$ES = \frac{Q}{\varepsilon_0},$$

kde $S = 4\pi x^2$, po dosazení

$$E = \frac{Q}{4\pi \varepsilon_0 x^2}.$$

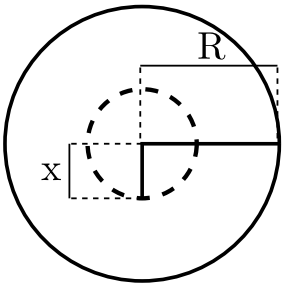
Pole v okolí nabité sféry je stejné jako pole bodového náboje i rovnoměrně nabité koule.



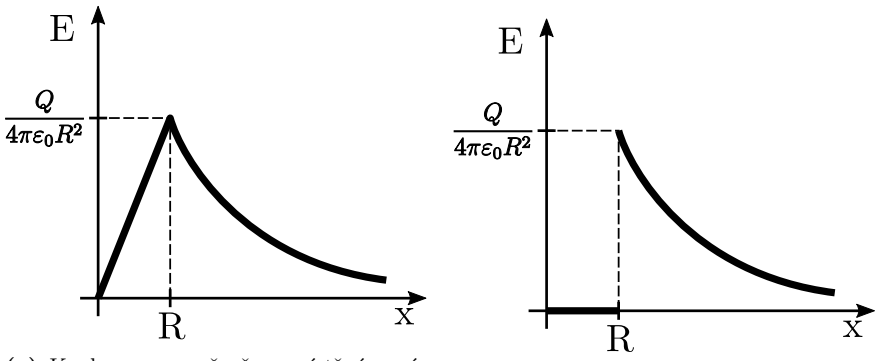
Obrázek 4: Nákres zjišťování elektrického pole vně koule

Pole uvnitř koule $x < R$

Pro výpočet intenzity použijeme Gaussovu větu, Q je náboj uzavřený uvnitř Gaussovy plochy. Plocha je ale zvolena tak, že uvnitř ní není žádný náboj, intenzita elektrického pole uvnitř koule musí být tedy taky rovna 0, protože v integrálu není nulová plocha, musí být tedy všude nulová elektrická intenzita.



Obrázek 5: Nákres zjišťování elektrického pole uvnitř koule



(a) Koule s rovnoměrně rozmístěným nábojem

(b) Koule s nábojem na povrchu

Obrázek 6: Průběh elektrické intenzity E v závislosti na vzdálenosti od středu koule x

Elektrické pole v okolí nekonečně dlouhé nabité desky (7 b)

Mgr.MM Jolana Knillová

Abychom mohli vyřešit, jak se bude chovat elektrické pole v okolí nekonečně dlouhé nabité desky, musíme správně určit Gaussovu plochu. V tomto případě to bude povrch válce, jehož středem prochází rovnoběžně s podstavami nabitá deska. Válec se skládá ze dvou podstav a pláště. Jelikož jsou obě podstavy stejně velké, bude tok jimi procházející stejný, a proto je do vztahu napočítáme dvakrát:

$$\oint_{\text{plášť}} \vec{E} \cdot \mathrm{d}\vec{S} + 2 \oint_{\text{podstava}} \vec{E} \cdot \mathrm{d}\vec{S} = \frac{Q}{\varepsilon_0}$$

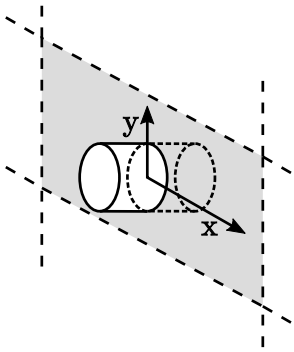
Plášť válce je na Gaussovu plochu kolmý, proto je integrál roven nule. Podstava naopak s plochou svírá úhel 0°, proto můžeme říct, že:

$$2ES_p = \frac{Q}{\varepsilon_0},$$

kde náboj můžeme pomocí plošné hustoty vyjádřit jako $Q = \omega S_p$. Potom

$$2ES_p = \frac{\omega S_p}{\varepsilon_0} \implies E = \frac{\omega}{2\varepsilon_0}$$

Z tohoto vztahu můžeme vyčíst, že elektrická intenzita v okolí nabité roviny nezávisí na vzdálenosti od ní – nabitá deska tudíž kolem sebe vytváří homogenní pole.



Obrázek 7: Nákres zjišťování elektrického pole v okolí desky

Výsledková listina 1. čísla

V této sérii vám nebudou přičteny žádné body, neboť body za 1. sérii jsme vám přičetli do výsledkové listiny, kterou najdete na konci 2. čísla. Body na řešení 2. čísla vám přičteme po Vánocích ve čtvrtém čísle. Přikládáme proto výsledky minulého čísla.

Poř.	**Jméno**	**R.**	$\sum_{-1}$	$\sum_{1}$
1.	Doc.^{MM} J. Havelka	4	129,4	25,0
2.	Mgr.^{MM} K. Vokálová	4	33,6	21,2
3.	Mgr.^{MM} J. Knillová	1	20,1	20,1
4.	Mgr.^{MM} K. Hloušková	4	48,4	17,9
5.	Mgr.^{MM} L. Kunčarová	4	20,6	17,0
6.	Bc.^{MM} J. Kaifer	4	19,8	16,0
7.	Mgr.^{MM} E. Vítková	4	45,6	14,3
8.	Mgr.^{MM} T. Flídr	2	39,5	13,5
9.–10.	Mgr.^{MM} O. Gonzor	3	42,7	12,8
	Bc.^{MM} D. Perout	3	12,8	12,8

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

Sloupeček $\sum_{-1}$ je součet všech bodů získaných v našem semináři 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