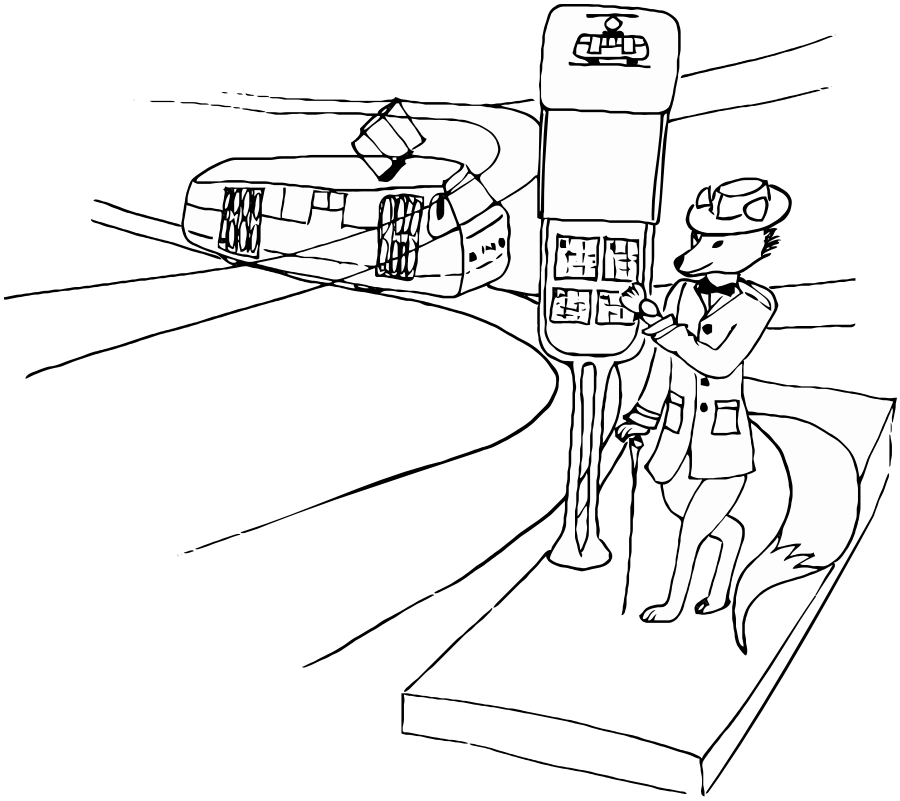
STUDENTSKÝ ČASOPIS A KORESPONDENČNÍ SEMINÁŘ

Ročník XXV
Číslo 4

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,
s novým pololetím přichází nová série časopisu M&M. Témátka budou v tomto čísle pojednávat o grafech, algoritmech, kryptografii, odkryjete další tajemné vlastnosti měsíce či prohloubíte znalosti o teoretickém modelu počtu cestujících v dopravních prostředcích.

Kromě témátek si můžete přečíst články z některých konfer, jež proběhly na podzimním soustředění v Borku. Konkrétně se jedná o balistickou konferu sepsanou Bc.^{MM} Kristýnou Kamenářovou a o konferu pojednávající o ovocném napětí, již sepsaly Mgr.^{MM} Marie Kalousková a Bc.^{MM} Adéla Foglarová.

Mnoho úspěchů a štěstí nejen do nadcházejícího pololetí přejí

Organizátoři

Zadání a řešení témat

Termín odeslání 4. série: 5. 3. 2019

Téma 1 – Paradoxní výsledky

Toto témátko už nebude mít nové zadání, ale to neznamená, že dál nemůžete zkoušet testovat své vlastní teorie. Řešení 2. a 3. série budou zveřejněna na konci ročníku, až do té doby můžete posílat své nápady. Pro připomenutí zde zopakujeme problémy zadané v minulých číslech, můžete si ale vymyslet i vlastní experimenty:

Problém 1: *Ověřte, zda i pro ostatní kapaliny (tzn. nejen pro vodu) platí, že na počátku teplejší kapalina mrzne rychleji. Vyzkoušejte např. mléko, vodu se solí a tak dále. Na čem si myslíte, že bude výsledek měření záviset? Zkuste navrhnout kapalinu, na které bude efekt nejsilnější. Zkuste změnit podmínky, za kterých voda a vámi zvolené kapaliny tuhnou, tím, že zamezíte pohybu molekul v objemu kapaliny či změníte odvod tepla tím, že do kapaliny něco vložíte. Diskutujte důsledky těchto vámi změněných podmínek.*

Problém 2: *Změřte závislost teploty kapaliny na čase pro dvě různé nádoby a poté i pro dvě různá podloží z různých materiálů. Jak materiál nádoby a podloží ovlivňuje průběh tuhnutí? Jak závisí doba tuhnutí na výkonu mrazáku? Odhadněte konkrétní hodnoty pro váš mrazák. Diskutujte vliv parametrů své nádoby a parametrů jejího okolí na velikost tepelného toku. Pokud jste použili nádobu, která nemá válcovitý tvar, vzorec pro výpočet tepelného toku si najděte např. v odborné literatuře či na internetu.*[1]

Pája a Matej; pavla.trembulakova@seznam.cz
e-mailová konference: paradoxni@mam.mff.cuni.cz

[1] např. studijní text Fyzikální olympiády: http://fyzikalniolympiada.cz/texty/texttz.pdf

Téma 2 – Principy kryptografie

Asymetrické šifrování

V předchozích dílech jsme si představili blokové šifry. Pro ně platí, že existuje jeden univerzální klíč a kdo ho zná, může šifrovat i dešifrovat zprávy. Dnes si ukážeme koncept asymetrické kryptografie, tedy takové šifrování, kde existují dva oddělené klíče. *Veřejný klíč* je možné využít pouze pro zašifrování textu. Pro dešifrování je potřebné znát *soukromý klíč*.

Jak to funguje

Asymetrickou šifru si můžeme představit jako dvě oddělené krabičky – jedna slouží pro šifrování a druhá pro dešifrování. Pro šifrování je nutná znalost veřejného klíče, pro dešifrování znalost soukromého klíče.

Jednoznačně nejrozšířenějším algoritmem pro asymetrické šifrování je RSA. Z dalších algoritmů můžeme jmenovat ještě třeba ElGamal.

Na rozdíl od zatím představených kryptografických komponent jsou algoritmy využívané v asymetrické kryptografii obvykle založeny na těžkých matematických problémech, jejichž obtížností se zabývala již celá řada matematiků.

Modulární aritmetika

Než se pustíme do přesného popisu RSA, zavedeme si značení, se kterým budou naše úvahy jednodušší. Říkáme, že čísla a a b jsou *kongruentní* modulo n, pokud dávají po dělení číslem n stejný zbytek. Tuto skutečnost zapisujeme $a \equiv b \pmod{n}$. Číslu n říkáme *modul*.

Například platí $37 \equiv 82 \pmod{15}$. Můžete si rozmyslet, že vztah $a \equiv b \pmod{n}$ je ekvivalentní s tvrzením $n|a-b$. V našem příkladě skutečně $15|37-82$.

Pro odůvodnění korektnosti RSA budeme ještě potřebovat jedno užitečné tvrzení z teorie čísel.

Pro přirozené číslo n označme jako *Eulerovu funkci* $\varphi(n)$ počet čísel menších nebo rovných n, která jsou s n nesoudělná.

Například čísla nesoudělná s devítkou jsou všechna, která nejsou dělitelná třemi (1,2,4,5,7,8), takže $\varphi(9) = 6$. Podobně můžeme spočítat, že $\varphi(11) = 10$ nebo $\varphi(12) = 4$.

Obecně platí, že pokud $n = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$ je prvočíselný rozklad čísla n, tak

$$\varphi(n) = (p_1 - 1)p_1^{\alpha_1 - 1}(p_2 - 1)p_2^{\alpha_2 - 1} \dots (p_k - 1)p_k^{\alpha_k - 1}.$$

Problém 1 [1b]: *Zdůvodněte, že vzorec pro výpočet Eulerovy funkce je správný.*

Nechť a a n jsou nesoudělná přirozená čísla. Potom platí

$$a^{\varphi(n)} \equiv 1 \pmod{n}.$$

Tomuto tvrzení se říká *Eulerova věta*.

Například již víme, že $\varphi(9) = 6$. Podle Eulerovy věty pro $a = 14$ a $n = 9$ tedy platí $14^{\varphi(9)} = 14^6 \equiv 1 \pmod 9$. Spočítejte si, že to je skutečně pravda.

Problém 2 [1b]: *Platí Eulerova věta i pro přirozená čísla a a n, jejichž největší společný dělitel je 2? Pokud ano, zdůvodněte. Pokud ne, najděte protipříklad.*

RSA

Šifra RSA je založena na obtížnosti problémů faktorizace a diskrétního logaritmu. *Faktorizací* matematici nazývají rozklad čísla na prvočísla. V RSA se využívá skutečnosti, že pokud vynásobíme dvě velká prvočísla, je obtížné takto vzniklé číslo faktorizovat.

Logaritmus je obecně funkce, která vrací exponent, na který musíme umocnit základ, abychom dostali zadané číslo[2]. Jako *diskrétní logaritmus* se označuje ta samá úloha, jen počítaná v celých číslech (místo reálných) modulo nějaké pevně stanovené číslo. Najít hodnotu diskrétního logaritmu je považováno za výpočetně obtížný problém.

Označme si $\log_2$ logaritmus o základu 2. Potom $\log_2 32 = 5$, protože $2^5 = 32$. Pokud budeme počítat diskrétní logaritmus modulo 11, tak nám vyjde $\log_2 10 \equiv 5$, protože $2^5 \equiv 10 \pmod{11}$.

Abychom mohli použít RSA, musíme si nejdříve vygenerovat veřejný a soukromý klíč. To můžeme udělat v následujících krocích:

1. Najdeme dvě dostatečně velká a dostatečně náhodná prvočísla p a q.
2. Spočítáme $n = pq$ a $\varphi(n) = (p-1)(q-1)$. Hodnota n bude využívána jako modul pro všechny operace. Hodnota $\varphi(n)$ musí zůstat utajená a po dopočítání klíčů ji můžeme zapomenout.
3. Zvolíme *veřejný exponent* e tak, aby $1 < e < \varphi(n)$ a e a $\varphi(n)$ byla nesoudělná.
4. Dopočítáme *soukromý exponent* d tak, aby $ed \equiv 1 \pmod{\varphi(n)}$.

Jako veřejný klíč použijeme dvojici (n, e), jako soukromý klíč dvojici (n, d).

Šifrování a dešifrování je nyní již jednoduché. Pokud chce kdokoli zašifrovat zprávu m, může to udělat pomocí veřejného klíče jako $c = m^e \pmod n$.

Dešifrovat zašifrovanou zprávu může pouze vlastník soukromého klíče umocněním zašifrované zprávy na d opět modulo n, tedy $m = c^d \pmod n$.

Z definice veřejného a soukromého exponentu musí existovat nezáporné celé h, pro které $ed = 1 + h\varphi(n)$. S využitím Eulerovy věty tak můžeme dopočítat:

$$c^d = m^{ed} = m^{1+h\varphi(n)} = m\left(m^{\varphi(n)}\right)^h \equiv m\,(1)^h = m \pmod n.$$

Tím jsme dokázali korektnost algoritmu.

[2] Více informací o logaritmech můžeš najít například na `https://matematika.cz/logaritmy`.

Problém 3 [2b]: *Mějme RSA s veřejným klíčem (493, 33) a soukromým klíčem (493, 353). Přišla vám zašifrovaná zpráva 93. Jaká je hodnota původního otevřeného textu?*

Znaky místo čísel můžeme šifrovat stejně jako v případě blokových šifer – můžeme využít například jejich ASCII reprezentaci.

Problém 4 [2b+]: *Jaké jsou zvyklosti pro volby velikosti modulu a veřejného a privátního exponentu? Můžete vycházet z veřejně dostupných doporučení. Bonusové body jsou za vlastní průzkum. Například téměř každý server umožňující šifrovaný přístup přes HTTPS využívá certifikát s RSA klíčem. Podívat se, jaké hodnoty jsou využívány v praxi, tedy není vůbec obtížné.*

Problém 5 [3b]: *Protože problém faktorizace je sice obtížný, ale nikoli neřešitelný, doporučuje se RSA klíče po nějaké době (třeba po několika letech) vyměnit. To vyžaduje vygenerovat dvě nová velká prvočísla. To je výpočetně náročné, a tak jsme se rozhodli vždy generovat jenom jedno velké prvočíslo a druhé nechat z minulého klíče. Je takový přístup bezpečný?*

Vlastnosti RSA

Problém 6 [3b+]: *Představte si, že znáte dvě zašifrované zprávy c_1 a c_2, které vznikly zašifrováním neznámých textů m_1 a m_2. Dokážete z nich spočítat zašifrovanou zprávu, která odpovídá $m_1 \cdot m_2$ nebo $m_1^2 \cdot m_2^3$? Jak by tomu bylo možné zamezit?*

Řešení problémů z 2. čísla

Úloha 1

Zadání:
V kryptografii je obecně za úspěšný považován libovolný útok, který má alespoň 50% šanci na úspěch. Představme si ideální hashovací funkci, která má výstup o velikosti 64 bitů. Kolik výpočtů hashovací funkce budeme muset provést, abychom s pravděpodobností alespoň 50 % našli dvě hodnoty se stejným hashem?

Řešení:
Na úvod se musíme omluvit, že úloha byla mnohem obtížnější, než jsme chtěli zadat. Každý kryptograf ví, že podle narozeninového paradoxu pro blok velikosti n nastává kolize přibližně po $2^{n/2}$ pokusech. V našem případě tedy budeme muset provést přibližně 2^{32} výpočtů. V praxi žádný přesnější výpočet potřeba není. Pokud ale chceme najít přesné řešení, musíme se pustit do složitých výpočtů.

Celkem existuje 2^{64} různých hashů. Předpokládejme, že pravděpodobnost, že výpočtem získáme konkrétní hash, je rozdělená rovnoměrně. První hash se určitě nebude shodovat s žádným již spočítaným. Pravděpodobnost, že ani druhý hash se nebude shodovat s prvním, je $(2^{64} - 1)/2^{64}$. Pravděpodobnost, že i-tý vypočítaný

hash se nebude shodovat s žádným předchozím, vyjadřuje vzorec $(2^{64} - i)/2^{64}$. Takže pravděpodobnost, že mezi prvními k hashi nebudou žádné dva stejné, můžeme vyjádřit jako

$$\mathrm{P}(k) = 1 \cdot \left(\frac{2^{64}-1}{2^{64}}\right) \cdots \left(\frac{2^{64}-(k-1)}{2^{64}}\right) = \frac{2^{64}!}{2^{64k}(2^{64}-k)!}.$$

Nás zajímá jev opačný, tedy potřebujeme najít nejmenší k takové, aby platilo

$$1 - \mathrm{P}(k) \geq \frac{1}{2}.$$

Analyticky vyřešit tuto nerovnici je nad naše matematické schopnosti. Dokonce jenom vyčíslit hodnotu pro $k = 2^{32}$ je velmi obtížné.

Vzpomeneme si proto, že umíme programovat, a necháme počítač, aby úlohu spočítal za nás. Následující skript vychází ze vzorce uvedeného výše. Pro jednotlivá k postupně počítá pravděpodobnost kolize a testuje, zda je již dostatečně velká.

```
#/usr/bin/env python3
pst = 1
k = 1
h2na64 = pow(2,64)
while True:
    pst *= (h2na64-(k-1))/h2na64
    if pst <= 0.5:
        print(k, pst)
        break
    k += 1
```

Na mém nepříliš výkonném počítači jsem se výsledku dočkal přibližně za půl hodiny. Hledané k vyšlo 5056937540. Tato hodnota je ovlivněna skutečností, že Python standardně zaokrouhluje na 16 desetinných míst (počítat s větší přesností by bylo možné, ale výpočetně náročnější). Skutečnému řešení se ale bude velmi blížit.

Platí $\log_2(5056937540) \doteq 32{,}2$, takže vidíme, že původní odhad 2^{32} byl poměrně dobrý.

Problém 2

Zadání:
Hashování hesel je určitě vhodným postupem. Pokud ale jednoduše spočítáme hash z hesla, stále můžeme při odcizení hashů čelit určitým rizikům. Napadá vás, kde jsou nedostatky takového postupu? Dokážete najít a popsat, jak ukládat hesla lépe? Napovíme, že možností na vylepšení existuje celá řada.

Řešení:
Správnému řešení tohoto problému byli blízko Dr.$^{\mathsf{MM}}$ Beáta Hroncová i Mgr.$^{\mathsf{MM}}$ Tomáš Sourada.

Hlavním problémem prostého hashování hesel je skutečnost, že pokud mají dva uživatelé stejné heslo, mají i stejný hash hesla. To výrazně usnadňuje potenciálnímu útočníkovi, který hash či hashe získá, nalezení původního hesla (tomuto postupu se říká *crackování*). Pokud použijeme dobrou hashovací funkci, je nejefektivnější možností, jak najít původní heslo, zkoušet zahashovat různá hesla a porovnávat výsledek se získanými hashi. Když má útočník k dispozici mnoho hashů, může výsledek svého snažení zkusit hledat najednou v celém seznamu získaných hashů. Navíc hashe častých hesel zahashovaných pomocí obvyklých hashovacích funkcí je možné dokonce i vygooglit, což se týká dokonce i českých hesel. Zkuste třeba najít, jaké heslo má hash (spočítaný neznámou funkcí) 29d847ffce86b63c39a756a25b198751.

Řešením tohoto problému je takzvané *solení* hesel. To spočívá v tom, že pro každého uživatele vygenerujeme unikátní náhodný řetězec nazývaný *sůl* a uložíme si dvě hodnoty: sůl a hash hesla spojeného se solí. Pokud nám nějaký útočník ukradne data z databáze, získá hashe i soli a může tedy opět crackovat hesla, ale pro každý hash bude muset dělat výpočty zvlášť a hlavně nebude moct využít žádné předpočítané hodnoty.

Toto řešení je možné i dále vylepšovat. Například k heslu nemusíme před hashováním přidávat pouze sůl, ale můžeme připojit i nějakou konstantu specifickou pro celou aplikaci, která není uložená nikde v databázi. Té se říká pro změnu *pepř*.

Ještě si můžeme říct něco o crackování hesel. Obvyklým postupem je, že si útočník vezme nějaký seznam častých hesel a ty zkouší postupně hashovat. Tento seznam se ještě pokouší nějak sám doplnit, například k němu přidá všechna jména a sportovní kluby, které jsou pro oblast, ze které očekává uživatele, relevantní. Když mu to nestačí, může si vzít třeba i seznam všech slov v předpokládaném jazyce. Pokud ani to nepomůže, obvykle přijdou na řadu pravidla, jak hesla ze seznamu modifikovat. Může se například pokusit za každé slovo z jazyka přidat několik čísel nebo měnit velikost počátečních písmen. Tato znalost může být užitečná, až si budete vymýšlet nějaké své heslo. To by obecně mělo být dostatečně dlouhé (za naprosté minimum lze považovat 8 znaků, ale delší bude lepší) a pro útočníka by mělo být obtížné ho pomocí pravidel vytvořit. Pokud použijete čtyři slova a vložíte mezi ně nějaké číslo, pravděpodobně útočník s crackováním neuspěje.

Úloha 3

Zadání:

Všimněte si funkce `pad`*, která připojuje za zprávu před doplnění nulami vždy stejnou konstantu. Tento postup jsme si vypůjčili od běžně používaných hashovacích funkcí. Napadá vás, k čemu je tato konstrukce dobrá?*

Řešení:

Konstanta ve funkci `pad` slouží jako alespoň částečná obrana před výpočtem hashe z prodlouženého textu bez znalosti původního (*length extension attack*).

Pokud bychom padding nepoužívali, mohl by útočník na základě znalosti hashe spočítat hash původní jemu neznámé zprávy doplněné o nějaký další text. Takový útok nemá příliš mnoho praktických uplatnění, ale je to něco, co je obecně spíše nežádoucí.

Úlohy 4, 5 a 6

Zadání úlohy 4 [2b]:
Dokážete najít dvě vstupní hodnoty, pro které je výsledný hash stejný?

Zadání úlohy 5 [3b]:
Víte, že pomocí funkce RIKSHA byl zahashován text, který není delší než 8 bytů. Výsledný hash je `02365E1E061E62BA`*. Dokážete najít nějaký (ne nutně stejný) vstup, který má stejný hash?*

Zadání úlohy 6 [5b]:
Existuje nějaká hodnota hashe, kterou funkce RIKSHA nevrátí pro žádný vstup? Nezapomeňte své tvrzení důkladně zdůvodnit.

K těmto úlohám zatím nepřišlo žádné řešení, a tak jsme se je rozhodli ponechat minimálně do dalšího čísla otevřené a budeme se nadále těšit na vaše řešení.

Ke čtvrté úloze napovíme, že k vyřešení stačí pochopit, jak funguje funkce RIKSHA. Měli byste ji zvládnout zcela bez programování.

Naopak u páté úlohy se bez programování neobejdete. Doplníme informaci, že text se skládal pouze z malých a velkých písmen anglické abecedy.

Náročnost šesté úlohy odpovídá jejímu bodovému hodnocení. Pokud ale chcete pouze získat pět bodů, tak existují i snazší cesty.

Kuba, Káťa a Lenka; `jakub.topfer@matfyz.cz`
e-mailová konference: `krypto@mam.mff.cuni.cz`

Téma 3 – Neznámý měsíc

Agentura, která vás posílala, vás nechala zabalit do krabičky cca $20 \times 20 \times 20$ cm libovolné vybavení. Tuto krabičku jste nyní našli. Co všechno můžete o planetě zjistit pomocí zařízení či materiálů dovezených v takto omezeném prostoru? Můžeme zjistit, zda je na planetě bakteriální život? Nebo určit složení půdy? Nebo něco jiného? Neberte toto prostorové omezení úplně přesně, stačí nám odhady, co by se tam mohlo vejít. Můžete používat jakékoliv současné technologie a předpokládat i trochu vývoje (např. integrování různých senzorů k jednomu počítači). Výstupem by mělo být složení krabičky, které považujete za optimální. Jako bonus můžete přiložit výkresy rozložení krabičky (pokud chcete 3D, Autodesk Inventor a Fusion 360 jsou pro studenty zdarma).

Vzorové řešení

Problém 1: Odklon rotačnej osi

Zadání:
Vedeli by ste určiť odklon rotačnej osi mesiaca od ekliptiky mesiaca s bežne dostupnými vecami? Ak nie, čo by ste minimálne potrebovali?

Řešení:
Najjednoduchší spôsob je podobný nášmu riešeniu pre zistenie polomeru mesiaca. Využijeme dlhú tyč zapichnutú do zeme, aby so zemou zvierala pravý uhol. Počas celého dňa budeme zaznamenávať veľkosť tieňu a z najmenšieho nameraného tieňu zistíme uhol odklonu hviezdy od kolmice k povrchu pomocou:

$$\alpha = \operatorname{arctg}\left(\frac{\textit{dĺžka tieňu}}{\textit{dĺžka tyče}}\right)$$

Tento najmenší tieň by mal smerovať na sever alebo juh. Ak budeme zaznamenávať veľkosť uhlov počas obehov mesiaca s planétou okolo hviezdy, mali by sme dostať funkciu závislú na dvoch *sin* o rôznych periódach a amplitúdach. Periódy budú závislé na čase obehu okolo planéty a hviezdy. Amplitúdy budú závislé na sklone orbity mesiaca okolo planéty a sklone orbity planéty okolo hviezdy. Zavedieme si konvenciu, že uhly namerané v smere na sever budú kladné a na juh záporné. Ak sa nenachádzame na rovníku, odmeriame rozdielnu veľkosť maxím pri smere na sever a juh. To vieme jednoducho opraviť tým, že všetky hodnoty posunieme tak, aby boli obidve maximá rovnaké. Ak vieme odmerať zemepisnú šírku, odklon osi je rovnaký ako zemepisná šírka polárneho kruhu, ktorý je definovaný ako miesto, kde aspoň jeden deň v roku hviezda nevyjde nad obzor a aspoň jeden deň v roku hviezda nezapadne za obzor.

Problém 2: Magnetické pole

Zadání:
Potvrďte prítomnosť magnetického poľa na povrchu mesiaca. Vedeli by ste určiť, či toto magnetické pole generuje mesiac, planéta alebo aj obe zároveň?

Řešení:
Potrebujeme zostrojiť primitívny kompas. To je v podstate jednoduché, musíme nájsť malí zmagnetizovaný objekt alebo si ho vyrobiť pomocou elektromagnetu. Loď určite v sebe mala medené káble, z ktorých môžeme vytvoriť cievku pre vytvorenie magnetu. Taktiež môžeme nájsť permanentné magnety v lodi alebo na planéte vo forme rúd, ako je napr. magnetit. Zmagnetizovaný objekt môžeme umiestniť na vodnú hladinu alebo na tenkú tyčku, aby sme mali malý výkyv aj vo vertikálnom smere a nie iba točenie v horizontálnom smere. Môžeme predpokladať, že ak budeme pozorovať rovnaké správanie kompasu ako na Zemi, mesiac má magnetické pole. Ak bude mať magnetické pole planéta, pri obehu okolo planéty budeme môcť pozorovať zmenu výchylky kompasu, alebo pri pohybe na mesiaci

nebudeme pozorovať chovanie ako na Zemi. Pri prítomnosti oboch polí bude závisieť na ich intenzite a smere. Môže aj nemusí byť tento stav možné zistiť bez použitia moderných prístrojov.

Zaujímavé riešenie odovzdal Mgr.[MM] Marco Souza de Joode, kde napísal, že niektoré zvieratá využívajú magnetické pole Zeme. Napr. kačky pristávajú v bezvetrí hlavne v smere siločiar. Ak sme mali na lodi nejaké zviera alebo život na planéte využíva smer magnetického poľa, vedeli by sme určiť zmeny magnetického poľa podľa zmeny ich zvyklostí.

Kuba a Kubo; kusnir.jk@gmail.com
e-mailová konference: mesic@mam.mff.cuni.cz

Téma 4 – Algoritmy od nuly (do n)

Intervalové datové struktury

Zadání čtvrté série

Vítáme vás u čtvrtého (tentokrát netradičně kratšího) dílu tématka pro začínající informatiky (a zvídavé matematiky, fyziky a biology). Úvodem bych rád poznamenal, že dva problémy z druhého čísla zůstávají stále částečně nevyřešené, dočtete se o nich v sekci *Řešení druhé série*.

Dnes se budeme zabývat datovými strukturami, které nám pomáhají rychle odpovídat na otázky ohledně nějaké uložené posloupnosti čísel. Konkrétněji nám půjde o takzvané *intervalové dotazy* – zeptáme se datové struktury na nějaký *interval* (třeba na prvky 35 až 42) a struktura nám o něm dá nějakou informaci (třeba součet všech prvků posloupnosti v daném intervalu). Nejdříve si ale stručně připomeneme, co jsou to datové struktury.

Datové struktury

Datovými strukturami jsme se zabývali už v minulém díle, proto si je nyní jen stručně připomeneme (vy, kteří jste ho nečetli, nalistujte prosím v minulém čísle u našeho tématka sekci „Datové struktury“). *Datová struktura* je způsob uložení informací v paměti, který nám umožňuje s nimi lépe pracovat. U datových struktur rozlišujeme jejich *rozhraní* a *implementaci*. Rozhraní datové struktury nám říká, jaký typ dat do ní můžeme ukládat a jaké operace s nimi můžeme provádět. Implementací datové struktury rozumíme to, jak jsou data uložena a jak konkrétně jednotlivé operace fungují.

Z toho, jakou implementaci použijeme, potom plyne časová a prostorová složitost. *Prostorová složitost* datové struktury je to, kolik místa v paměti zabírá. Používáme pro ni óčkovou notaci, stejně jako pro složitost časovou. U datových struktur rozlišujeme časovou složitost jejich *inicializace* (to je počáteční „stavba“ struktury a případné uložení dat do paměti) a složitost jednotlivých operací. Naše implementace fronty pomocí spojového seznamu z minulého dílu by tedy měla časovou složitost inicializace $\mathcal{O}(1)$, časovou složitost operací *enqueue* i *dequeue*

rovněž $\mathcal{O}(1)$ a prostorovou složitost $\mathcal{O}(n)$, kde n je maximální počet prvků ve frontě.

Problém 1 [1b]: *Spojový seznam se v praxi používá hlavně v případech, kdy potřebujeme vkládat prvky doprostřed seznamu. Skutečné procesory totiž typicky načítají naráz velký úsek paměti, takže s daty, která jsou v paměti uložena blízko sebe, umí pracovat výrazně rychleji než se spojovým seznamem, který má data roztroušená různě po paměti. Navrhněte způsob implementace fronty pomocí pole při zachování časové i prostorové složitosti. Můžete předpokládat, že předem znáte maximální počet prvků ve frontě.*

Prefixové součty

Mějme posloupnost celých čísel. Rádi bychom si vytvořili datovou strukturu, která nám pro každý *interval* (tedy souvislý úsek posloupnosti) umí co nejrychleji říct, jaký je součet všech čísel ležících v tomto intervalu. Pokud je tedy naše posloupnost $11, 3, 27, 30, 54, 6, 42$, a zeptáme se na interval $[4, 6]$, datová struktura odpoví 60, protože $54 + 6 = 60$ (intervaly budeme vždy brát zleva uzavřené a zprava otevřené, nyní jsme se tedy ptali na pozice 4 a 5). Mohli bychom prostě posloupnost uložit do pole a při každém dotazu sečíst příslušná čísla. Tím bychom dostali paměťovou složitost $\mathcal{O}(n)$, inicializaci v $\mathcal{O}(n)$ a dotazy rovněž v $\mathcal{O}(n)$. Jak to udělat lépe?

Datové struktury často fungují tak, že si předpočítáme něco, co potom využijeme pro rychlejší vyhodnocování dotazů. Předpočítat všechny dotazy si nechceme (všech možných intervalů je totiž n^2), předpočítáme si ale součty všech intervalů, které začínají nultým prvkem posloupnosti. Takových intervalů je pouze n, takže si je prostě uložíme do pole délky n (na pozici i si budeme pamatovat součet intervalu $[0, i + 1]$). Těmto intervalům se říká *prefixy*[3], proto těmto předpočítaným součtům říkáme *prefixové součty*. Pro zjištění součtu intervalu $[i, j]$ využijeme součty intervalů $[0, i]$ a $[0, j]$, které máme předpočítané. Interval $[i, j]$ obsahuje právě ta čísla, která leží v $[0, j]$, ale neleží v $[0, i]$, stačí tedy odečíst součet intervalu $[0, i]$ od součtu intervalu $[0, j]$ a dostáváme součet intervalu $[i, j]$. Když tedy dostaneme dotaz $[i, j]$, odečteme číslo na pozici $i - 1$ od čísla na pozici $j - 1$ v našem poli prefixových součtů a výsledek vrátíme.

Tím jsme získali konstantní časovou složitost dotazu. Paměťová složitost zůstala zřejmě lineární, zbývá nám časová složitost inicializace. Na první pohled by se mohlo zdát, že bude kvadratická, protože jeden prefixový součet má až n sčítanců a prefixových součtů je n. My ale nemusíme jednotlivé součty počítat nezávisle. Začneme od nultého součtu (to je prostě nultý prvek posloupnosti) a každý další prefixový součet spočítáme tak, že k tomu předchozímu přičteme další prvek posloupnosti. Každý ze součtů tedy získáme v konstantním čase, čímž dostáváme inicializaci v $\mathcal{O}(n)$.

[3] prefix by byl česky *předpona*, český překlad se ale v informatice v tomto kontextu nepoužívá

Problém 2 [2b]: *Co kdybychom místo posloupnosti měli tabulku celých čísel o rozměrech $n \times m$? Navrhněte co nejefektivnější datovou strukturu, které zadáme „podtabulku“ (souvislý obdélník v naší tabulce) a vrátí nám součet všech čísel v této podtabulce. Podtabulku budeme zadávat jako pozici levého horního a pravého dolního rohu. Nezapomeňte na časovou a prostorovou složitost.*

Rozklad na bloky

Nyní bychom po naší datové struktuře chtěli další operaci – změnit jeden prvek posloupnosti na jiný. Prefixové součty na tomto úkolu zcela selžou – po změně jednoho prvku bychom museli přepočítat až n prefixových součtů, za což by si naše nová operace vysloužila lineární časovou složitost (pokud bychom tedy změny prováděli podobně často jako dotazy na intervalové součty, bylo by stejně rychlé vůbec žádnou datovou strukturu nestavět a součet pokaždé znovu spočítat). Prefixové součty je tedy vhodné použít pouze tehdy, když prvky posloupnosti neměníme. Říkáme, že se jedná o *statickou* datovou strukturu.

Naše nová datová struktura tedy bude *dynamická*. Rozdělíme si posloupnost na *bloky* (souvislé úseky) velikosti b (hodnotu b zvolíme později tak, aby pro nás byla co nejvýhodnější). Tím získáme n/b bloků (pokud n není dělitelné b, doplníme posloupnost nulami). Nyní spočítáme součet čísel v každém bloku a uložíme si posloupnost těchto součtů (bude mít délku n/b). Nakonec si pro každý blok i pro posloupnost součtů spočítáme prefixové součty. Počítání prefixových i celkových součtů bloků nám zabralo čas $\mathcal{O}(n)$, počítání prefixových součtů posloupnosti součtů trvalo $\mathcal{O}(b)$, a jelikož $b \leq n$, zabrala nám celá inicializace lineární čas. Prostorová složitost je také lineární, opět protože posloupnost součtů bloků není delší než původní posloupnost.

Každý interval můžeme nyní rozdělit na počáteční část, která začíná uvnitř bloku a končí na jeho konci, dále středovou část, která se skládá z několika celých bloků, a koncovou část, která začíná na začátku bloku a končí uvnitř něj. Kterékoliv z těchto částí mohou být prázdné(pokud by byl například intervalem jeden blok, pak byla prázdná počáteční a koncová část). Součet libovolného intervalu tedy spočítáme jako součet těchto tří částí. Součet počáteční i koncové části zjistíme z prefixových součtů pro bloky, ve kterých tyto části leží – to nám zabere čas $\mathcal{O}(1)$ – a součet středové části získáme jako součet součtů bloků, ze kterých je středová část tvořena. Protože pro součty bloků máme také předpočítané prefixové součty, zabere nám to rovněž čas $\mathcal{O}(1)$. Součet daného intervalu tedy získáme v konstantním čase.

Když změníme jeden prvek posloupnosti, budeme muset v čase $\mathcal{O}(b)$ znovu spočítat prefixové součty bloku, ve kterém prvek leží, a v čase $\mathcal{O}(n/b)$ pro posloupnost součtů spočítat prefixové součty (poté, co v ní změníme jeden prvek). Tato operace nám tedy zabere čas $\mathcal{O}(b + n/b)$.

Nyní nám zbývá určit hodnotu b. Naším cílem samozřejmě je, aby byl výraz $\mathcal{O}(b + n/b)$ co nejmenší. Součet se asymptoticky chová stejně jako maximum a s rostoucím b první sčítanec roste a druhý klesá, výraz tedy bude nejmenší pro

$b = n/b$, což nám dává $b = \sqrt{n}$. Naše bloková datová struktura tedy potřebuje lineární prostor, lineární čas na inicializaci, změna prvku má časovou složitost $\mathcal{O}(\sqrt{n})$ a dotaz na intervalový součet stihneme v $\mathcal{O}(1)$.

Intervalová minima

Vytvořili jsme dynamickou datovou strukturu, která umí v čase $\mathcal{O}(\sqrt{n})$ měnit prvky a v konstantním čase počítat intervalové součty. Ve skutečnosti je ale naše struktura obecnější a po drobné úpravě může místo intervalových součtů počítat něco jiného – třeba intervalová minima. Naopak myšlenka prefixových součtů se na intervalová minima ani ve statické verzi použít nedá – z prefixových minim totiž nedokážeme nijak získat minimum požadovaného úseku (můžete si zkusit vymyslet příklad, kde by takový postup selhal). Pro počítání minim si budeme pamatovat pouze minima jednotlivých bloků. Pro každou ze tří částí intervalu (počáteční, středovou a koncovou) dokážeme minimum spočítat v čase $\mathcal{O}(\sqrt{n})$ – najdeme primitivním způsobem minimum počáteční i koncové části a pro středovou část vezmeme minimum z minim bloků. Z těchto tří minim potom jen v konstantním čase spočteme celkové minimum. Při změně prvku prostě jen v čase $\mathcal{O}(\sqrt{n})$ znovu spočítáme minimum daného bloku. Intervalové minimum i změnu prvku tak umíme spočítat v čase $\mathcal{O}(\sqrt{n})$, inicializace i paměťová složitost zůstala lineární.

Problém 3 [2b]: *Naši datovou strukturu pro počítání intervalových minim jde ve skutečnosti ještě vylepšit. Navrhněte obdobnou blokovou datovou strukturu na počítání minim, která využívá tří úrovní místo dvou a získejte tak časovou složitost operací $\mathcal{O}(\sqrt[3]{n})$ (nezapomeňte všechny složitosti řádně zdůvodnit).*

Problém 4 [5b]: *Proč bychom měli zůstat u tří úrovní? Spočítejte, jaký počet úrovní je optimální, a určete časovou složitost inicializace i jednotlivých operací a složitost prostorovou. Získáte tak velmi zajímavou a užitečnou datovou strukturu.*

Řešení druhé série

Problém 1

Zadání:
Zkuste na našem formálním modelu, který jsme si vytvořili na konci minulého dílu, vytvořit funkce. Nezapomeňte, že po svém ukončení se funkce musí vrátit hned za místo, ze kterého byla zavolána.

Řešení:
K tomuto problému se sešla dvě řešení, ani jedno z nich ale nevyřešilo úlohu pro obecný případ. Konkrétněji – nikomu z vás se nepovedlo vytvořit implementaci funkcí, která by umožňovala rekurzivní volání. Necháváme proto úlohu zatím otevřenou a otiskujeme zde korektní implementaci funkce na sečtení dvou čísel od Mgr.MM Ondřeje Chlubny, abyste na ni ve svých řešeních mohli navázat.

Mgr.MM Chlubna správně využil toho, že může dát funkci jako parametr řádek, na který má skočit po svém dokončení (kdybychom chtěli být úplně formální, označili bychom každý řádek jeho číslem a jako parametr bychom funkci dali toto číslo). Pokud by ale funkce volala sama sebe, přepsala by si v buňce 14 návratovou adresu a z prvního volání už by se nikdy nevrátila. Zkuste tedy implementaci vylepšit, aby funkce mohly volat samy sebe. Poradíme vám, že se vám k tomu bude hodit zásobník (psali jsme o něm v minulém čísle v sekci o datových strukturách).

Řešení podle Mgr.MM Ondřeje Chlubny:
Funkce by se dala interpretovat jako skok na určitou část kódu. Příklad funkce pro sečtení dvou čísel může být:

```
1. [0] ← 0                          (načtení prvního čísla)
2. [1] ← 0                          (načtení druhého čísla)
3.
4. [12] ← [0]                       (načtení vstupu funkce)
5. [13] ← [1]                       (načtení vstupu funkce)
6. [14] ← tam1
7. Skok na funkce_soucet
8. tam1
9. [2] ← [15]                       (načtení výstupu funkce)
10. print [2]
11.
12. [12] ← [0]                      (načtení vstupu funkce)
13. [13] ← [2]                      (načtení vstupu funkce)
14. [14] ← tam2
15. Skok na funkce_soucet
16. tam2
17. [3] ← [15]                      (načtení výstupu funkce)
18. print [3]
19.
20. funkce_soucet                   (začátek funkce)
21. [15] ← [12] + [13]              (výpočet ve funkci)
22. skok na [14]
```

Místo toho, aby byly hodnoty poslány přímo do funkce, mají své vlastní proměnné. Vždy, když se skáče na `funkce_soucet`, jsou v [12] a [13] sčítance, v [14] je napsáno, kam má kód pokračovat, a výstup funkce je v [15].

Problém 2

Zadání:
Spočítejte faktoriál pomocí rekurze bez použití cyklu. Otázka, kterou je dobré si položit při řešení tohoto problému, je: „Kdybych měl spočítané $(n-1)!$*, jak z toho*

spočítám $n!$?“. Toto je ostatně otázka, kterou je dobré si položit, kdykoliv řešíte nějaký algoritmický problém, a která vede na rekurzivní algoritmus (rozmyslete si proč).

Řešení:
Využijeme toho, že $n! = n \cdot (n-1)!$. Naše funkce se tedy může rekurzivně zavolat na $n-1$ a vrácený výsledek vynásobit číslem n:

```
1. Funkce faktoriál(n):
2.     Pokud n = 0:
3.         Vrať 1
4.     Vrať n·faktoriál(n − 1)
```

Problém 3

Zadání:
Náš algoritmus na házení vajíček lze pro některé velikosti vstupů ještě o trochu zlepšit (jen o konstantu). Zkuste si rozmyslet jak a napište nám to.

Řešení:
Zlepšení ve skutečnosti spočívá pouze v tom, že k paneláku nepřidáváme žádná imaginární patra – kód bez jakýchkoliv úprav prostě spustíme pro libovolné n. Tím si pro některé vstupy ušetříme jedno volání funkce. Zlepšení je tedy skutečně nepatrné, ale je dobré si uvědomit, že v praxi vstup nemusíme doplňovat na mocninu dvojky – potřebovali jsme ji jen proto, aby se nám lépe počítala časová složitost.

Problém 4

Zadání:
Vymyslete algoritmus, který najde v setříděném poli délky n dané číslo v čase $\mathcal{O}(\log n)$. Nezapomeňte na slovní popis algoritmu a zdůvodnění časové složitosti.

Řešení:
Úlohu vyřešíme stejně jako úlohu s panelákem. Místo pater nyní budeme mít čísla v poli a rozbití vajíčka v i-tém patře bude reprezentováno tím, že číslo na i-té pozici je větší než hledané číslo x. Pole je setříděné, tyto pozice tedy tvoří souvislý úsek na konci pole a úloha je tedy skutečně analogická. Obecněji – pokud máme pole n čísel, z nichž prvních k splňuje nějakou vlastnost a zbylých $n-k$ tuto vlastnost nesplňuje, můžeme pro nalezení k použít náš „vajíčkový algoritmus“ (běžně se nazývá *binární vyhledávání*). V tomto případě se jedná o vlastnost „je menší nebo rovno x“. Až najdeme největší číslo s touto vlastností, jen zkontrolujeme, zda je rovno x, a podle toho buď vrátíme příslušnou pozici, nebo vrátíme -1 (což značí, že číslo v poli není).

Od „vajíčkového algoritmu“ se naše binární vyhledávání bude lišit pouze tím, že po nalezení příslušné pozice v poli zkontroluje, zda na ní leží číslo x. To se

ale stane pouze jednou a zabere to konstantní čas, časová složitost tedy bude samozřejmě stejná. Proto jen stručně připomeneme, že jedno volání funkce trvá konstantní čas a pokaždé ji voláme na dvakrát menší vstup, počet volání tedy bude roven $\log n$, z čehož dostáváme složitost $\mathcal{O}(\log n)$. Následuje pseudokód:

Vstup: pole *vstup*, délka pole n
1. Funkce `binarni_vyhledavani`(*posloupnost*, n, *start*):
2. Pokud $n = 1$:
3. Pokud *posloupnost*[*start*] $= x$:
4. Vrať *start*
5. Jinak:
6. Vrať -1
7. Pokud $n > 1$:
8. Pokud *posloupnost*[*start* $+ n/2$] $> x$:
9. Vrať `binarni_vyhledavani`(*posloupnost*, $n/2$, *start*)
10. Jinak:
11. Vrať `binarni_vyhledavani`(*posloupnost*, $n/2$, *start* $+ n/2$)

Výstup: `binarni_vyhledavani`(*vstup*, n, 0)

Problém 5

Zadání:
Předpokládejme, že máme dvě setříděné posloupnosti čísel (mohou být různě dlouhé). Vymyslete, jak s co nejlepší časovou složitostí z těchto dvou posloupností udělat jednu setříděnou posloupnost obsahující čísla z obou. Při vyjadřování časové složitosti použijte n jakožto celkový počet čísel. Nezapomeňte určit časovou složitost svého algoritmu.

Řešení:
Tomuto problému se anglicky říká *merge*, do češtiny se překládá jako *slévání*. Samotný algoritmus je velmi jednoduchý. Porovnáme vždy nejmenší čísla v obou slévaných polích a menší z nich z příslušného pole odstraníme a přidáme ho na konec vytvářené posloupnosti. Až čísla v jednom z polí dojdou, přidáme na konec posloupnosti i všechna ta, která ve druhém z polí zbyla.

Proč algoritmus funguje? Představme si, jaký bude jeho první krok. Vytvářená posloupnost je v tu chvíli prázdná. Který prvek do ní přidáme jako první? Protože vytváříme setříděnou posloupnost, musí to být samozřejmě ten nejmenší, tedy ten menší z minimálních prvků dvou slévaných polí. První krok algoritmu – odebrat globálně nejmenší prvek a přidat ho na konec vytvářené posloupnosti – je tedy správný. Nyní si stačí uvědomit, že stojíme znovu před stejným problémem. Výsledná posloupnost bez prvního prvku je totiž pořád setříděná posloupnost, takže si můžeme představit, že dotyčný prvek v žádném z polí nikdy nebyl a že začínáme znovu od začátku, jen vytváříme posloupnost o jedna kratší. Můžeme tedy pokaždé znovu použít stejný argument, dokud se jedno pole nevyprázdní. Poté už

je jasné, že zbylé prvky patří na konec vytvářené posloupnosti. Algoritmus si tedy můžete představit i jako rekurzivní – po každém přidání prvku řešíme stejný problém o jedna menší velikosti. My ho ale raději napíšeme pomocí cyklů (v praxi jsou cykly efektivnější než rekurze, takže pokud algoritmus umíme jednoduše napsat oběma způsoby, volíme ten bez rekurze).

Zbývá nám určit časovou složitost algoritmu. V každém kroku algoritmus porovná dvě čísla a jedno z nich přemístí do výsledné posloupnosti. Jeden krok tedy zabere konstantní čas. V každém kroku se posloupnost zvětší o 1, kroků je tedy celkem n, což nám dává složitost $\mathcal{O}(n)$. Následuje pseudokód:

Vstup: pole a a b a jejich délky len_a a len_b
1. $n \leftarrow len\_a + len\_b$
2. $final \leftarrow$ Pole délky n
3. $i = 0$; $j = 0$; $k = 0$;
4. Dokud $i \neq len\_a$ a zároveň $j \neq len\_b$: (dokud v obou polích zbývají prvky)
5. Pokud $a[i] < b[j]$: (pokud je minimum v prvním poli menší)
6. $final[k] \leftarrow a[i]$ (přidáme ho na výstup)
7. $i \leftarrow i + 1$
8. Jinak: (jinak přidáme na výstup minimum druhého pole)
9. $final[k] \leftarrow b[j]$
10. $j \leftarrow j + 1$
11. $k \leftarrow k + 1$
12. Dokud $i \neq len\_a$: (pokud zbyla čísla v prvním poli, přemístíme je na výstup)
13. $final[k] \leftarrow a[i]$
14. $k \leftarrow k + 1$
15. $i \leftarrow i + 1$
16. Dokud $j \neq len\_b$: (pokud zbyla čísla v druhém poli, přemístíme je na výstup)
17. $final[k] \leftarrow b[j]$
18. $k \leftarrow k + 1$
19. $j \leftarrow j + 1$

Výstup: pole $final$

Problém 6

Zadání:

Vymyslete třídící algoritmus běžící v čase lepším než kvadratickém (jinými slovy lepším než $\mathcal{O}(n^2)$). Nezapomeňte udat s odůvodněním jeho časovou složitost. Prozradíme vám, že se vám bude hodit nejen rekurze, ale i kapitola o logaritmech.

Řešení:

Úlohu se povedlo vyřešit jako jedinému Mgr.MM Tomáši Souradovi, jehož řešení zde otiskujeme. Mgr.MM Sourada ale udělal dvě chyby v analýze časové složitosti, takže jeho algoritmus má sice lepší složitost než kvadratickou, ale hned ze dvou důvodů tato složitost není $\mathcal{O}(n \log n)$ (jak Mgr.MM Sourada ve svém řešení

píše). Vybízíme vás proto, abyste přišli na to, kde udělal Mgr.[MM] Sourada chybu, jaká je skutečná složitost jeho algoritmu a hlavně jak dosáhnout časové složitosti $\mathcal{O}(n \log n)$. Opět připomínáme, že k řešení se vám kromě problému 5 bude hodit i rekurze a logaritmy.

Řešení podle Mgr.[MM] Tomáše Sourady:
Nejprve si pole rozdělíme na menší podpole délky $\lceil \sqrt{n} \rceil$ (horní celá část odmocniny z n – pokud $n \neq a^2$, pak poslední podpole bude o něco kratší). Těchto podpolí bude zhruba $\sqrt{n}$. Každé toto podpole setřídíme (algoritmem dodaným v řešení k prvnímu dílu, jehož časová složitost pro vstup délky n je n^2). To bude mít časovou složitost $\sqrt{n}^2 = n$. Pro všechna pole dohromady $n\sqrt{n}$. Pak si tato podpole vezmeme vedle sebe, vezmeme nejmenší prvky z těchto polí a tyto prvky si setřídíme (prvků je $\sqrt{n}$, časová složitost je tedy $\sqrt{n}^2 = n$). Vezmeme nejmenší z těchto nejmenších prvků (časová složitost 1), škrtneme ho a zapíšeme na nultou pozici nového (později výsledného) pole délky n.

Zbude nám setříděná posloupnost délky $\sqrt{n} - 1$ nejmenších prvků našich podpolí. Z podpole, z něhož byl ten škrtnutý prvek, vezmeme další nejmenší prvek v pořadí a se složitostí $\mathcal{O}(\log n)$ dle problému 4 najdeme, kam patří v naší posloupnosti nejmenších prvků podpolí a zařadíme ho tam. Teď znovu vezmeme nejmenší prvek z nejmenších, škrtneme ho a zapíšeme do výsledného pole na nejmenší neobsazenou pozici. Pak do posloupnosti nejmenších opět se složitostí $\mathcal{O}(\log n)$ zařadíme nejmenší prvek z podpole, ve kterém byl škrtnutý prvek. Takto pokračujeme dál. Pokud z nějakého podpole škrtneme poslední prvek, (a už tedy není odkud brát prvek do naší posloupnosti nejmenších), prostě se délka posloupnosti nejmenších zkrátí o jedna.

Celkem do posloupnosti nejmenších musíme zařadit $\sqrt{n}(\sqrt{n} - 1)$ prvků, každý se složitostí $\mathcal{O}(\log n)$, celková časová složitost je tedy $\mathcal{O}(n \log n)$.

Jakmile skončíme, budeme mít seřazenou výslednou posloupnost. Celková časová složitost celého algoritmu je ta z těch, které se sčítaly ($\mathcal{O}(n\sqrt{n}), \mathcal{O}(n)$, $\mathcal{O}(n \log n)$), která je nejhorší. To jest $\mathcal{O}(n \log n)$. Celková časová složitost algoritmu je tedy $\mathcal{O}(n \log n)$.

Tom; `domestomas+mam@gmail.com`
e-mailová konference: `algoritmy@mam.mff.cuni.cz`

Téma 5 – Přeplněná tramvaj

Úvod

K tomuto tématku přišlo několik prvních řešení a já se k nim nyní stručně vyjádřím. Vynechám-li jednoduchou úlohu s kostkou, tak jmenovitě Mgr.[MM] Klára Hloušková, Mgr.[MM] Marco Souza de Joode a autorská dvojice Mgr.[MM] Martin Boček & Bc.[MM] Michal Vícha přispěli ke zkoumání vývoje počtu pasažérů v dopravním prostředku experimentálním způsobem. Všichni zmínění pracovali s autobusovými linkami.

Mnozí na základě svých výsledků vynášeli pochybnosti o existenci obecného tvaru křivky[4] $S(m)$ a podkládali je pozorovanou nerovnoměrností vytíženosti jednotlivých stanic. Výzkum Mgr.MM Bočka & Bc.MM Víchy věnující se opakovanému měření linky z Jančí do Opavy ukázal, že tato nerovnoměrnost v rámci jediné linky není vždy dílem statistické odchylky a že jde tudíž skutečně o neměnnou vlastnost linky. Mgr.MM Hloušková se ji pokouší nikoli neúspěšně popsat uvážením počtu obyvatel v oblastech, jimiž její autobus projíždí, čímž předchází zčásti úlohu, která bude zadána teprve v tomto čísle.

Nalezená nerovnocennost stanic na reálných linkách je poněkud nepřívětivý objev pro věrohodnost mnou navrženého modelu z předchozího čísla, který mimo jiné předpokládá právě rovnocennost stanic. Nicméně stále zůstává naděje, že i ona experimentální nerovnocennost je v globálním pohledu existujících linek rovnoměrně rozloženým jevem, tudíž při proměření dostatečně vysokého počtu nezávislých linek by mohla vymizet ve prospěch důsledků modelu, jímž jste se zabývali v předchozí sérii. Za tímto účelem vás vyzývám k upřednostnění střídání linek v experimentech před podrobným proměřováním jediné linky.

Dále se hledání alternativy k navrženému modelu chování cestujících věnuje Mgr.MM Souza de Joode. V jeho modelu si nastoupivší člověk volí za cíl cesty střed trasy, která mu ve chvíli nástupu zbývá do konečné stanice. Jedná se o jisté zjednodušení mého modelu, sám autor označuje své pasažéry za prostoduché a ještě v témž příspěvku zároveň vyvozuje závěr, že v jeho modelu má křivka $S(m)$ tvar stříšky. Toto ne zcela odpovídá experimentům, ovšem ani jim v zásadě neodporuje, proto navrhuji jeho model při zpracovávání výsledků měření rovněž uvážit.

V závěru úvodu bych vám chtěl připomenout, abyste nadále pokračovali v experimentálním výzkumu křivky $S(m)$. Jde o vykonání cesty libovolným dopravním prostředkem přes co nejdelší část linky (ideálně přes celou) a zaznamenání počtu pasažérů uvnitř prostředku mezi každými dvěma stanicemi. Třebaže je experimentální práce občas nepohodlná, je dobře hodnocená, neboť velké množství experimentálních výsledků je nutnou podmínkou pro možnost pozdějšího posouzení správnosti jakékoli naší teorie.

Zadání

K vyřešení úloh v této sérii budete potřebovat výsledek získaný řešením předchozí série, což zde ze začátku znamená jistou komplikaci, neboť v tomto časopise není zvykem zveřejňovat vzorová řešení dříve než dvě série po zadání. Proto vám nyní prozradím pouze výsledek s tím, že na jeho odvození si budete muset počkat do další série.

Na lineární lince při předpokladu rovnocenných stanic a pasažérů cestujících za předem daným cílem je výsledkem[5]:

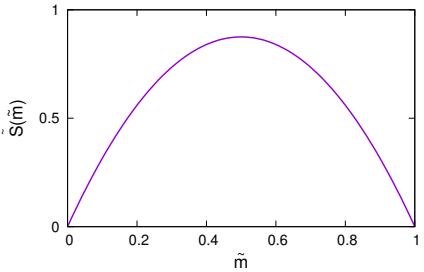
[4] Připomínám, že v tomto tématku se zkoumá závislost počtu lidí v dopravním prostředku na fázi jeho cesty linkou. Konkrétně S značí počet lidí a m počet projetých stanic. Pro detaily viz `https://mam.mff.cuni.cz/problem/2207/`.

[5] Viz poznámka pod čarou č. 4.

$$S(m) = Cm(N - m),$$

tj. parabola s parametrem C charakterizujícím danou linku a závislým na konstantě D (má význam očekávaného počtu pasažérů čekajících na každé stanici) a na počtu stanic N.

Na cyklické lince se stejnými předpoklady je řešením totéž během prvních $N/2$ stanic a konstanta po zbytek cesty. Výsledek za předpokladu náhodně cestujících pasažérů v této sérii zatím nepotřebujete.



Obrázek 1: Graf funkce $S(m)$, veličiny $\tilde{S}$,$\tilde{m}$ odpovídají veličinám S, m přenásobeným takovými konstantami, aby nabývaly pouze hodnot z intervalu $[0, 1]$, viz níže.

Výše uvedený tvar $S(m)$ očividně nesouhlasí s výsledky všech vašich experimentů. To může být způsobeno statistickou odchylkou naměřených hodnot (měření bylo provedeno stále příliš málo, než abychom mohli toto vyloučit), ale také selháním naší teorie. Vaším cílem v této sérii proto bude naši teorii zpřesnit a přiblížit pozorované realitě. Toho docílíme upuštěním od některých zjednodušujících předpokladů.

Jedním a pravděpodobně nejzávažnějším takovým zjednodušením byl předpoklad rovnocennosti všech stanic jakožto zdrojů i cílů cestujících. V tomto směru naši teorii zobecníme tím, že k-té stanici přiřadíme jistý koeficient α_k, který bude popisovat vytíženost stanice. Na k-té stanici bude tedy vygenerováno $D\alpha_k$ pasažérů a každý z nich si zvolí za svůj cíl stanici j s pravděpodobností[6] $\frac{\alpha_j}{\sum_{i\neq k}\alpha_i}$.

Vytíženost se zde stále vztahuje k funkci stanic coby zdrojů i cílů cestujících (není pak např. možné, aby v některé stanici pokaždé nastupovala spousta lidí, nikdo by v ní však nevystupoval).

Problém 1 [4b]: *Zobecněte výše uvedený tvar křivky $S(m)$ na situace s nerovnocennými stanicemi popsanými jediným systémem koeficientů $\alpha_1, \ldots, \alpha_N$.*
Nápověda: *Zkuste pro začátek předpokládat*[7] $\forall k : \alpha_k \in \mathbb{N}$.

[6] Matematický zápis sumy, chápeme ho jako: $\sum_{i\neq k} \alpha_i = \alpha_1 + \cdots + \alpha_{k-1} + \alpha_{k+1} + \cdots + \alpha_N$

[7] Tento matematický zápis říká, že všechny koeficienty α_k jsou přirozené, tj. např. $1, 2, 3, \ldots$

Další ne zcela oprávněné zjednodušení představoval způsob, kterým si pasažér vybírá cíl. Je poměrně naivní si myslet, že bude pasažér vybírat blízké stanice stejně často jako stanice vzdálené, neboť v některých případech pro něho může být výhodnější bližší stanice dojít pěšky nebo se do vzdálenějších dopravit jiným spojem. Proto by bylo vhodnější uniformní rozdělení volby cíle nahradit jiným.

Předem si uvědomme, že vyřešení této úlohy pro obecné rozdělení by nám pak ve spojení s výsledkem předchozí úlohy umožňovalo popsat již zcela libovolnou linku (včetně neomezených generátorů pasažérů, viz výše). My se prozatím smíříme s náhražkou tohoto pravděpodobně obtížně dosažitelného cíle. Efekt nechuti cestujících k dalekým cestám se pokusíme popsat tvrzením, že žádný pasažér nechce v dopravním prostředku projet méně než n_0 stanic ani více než n_1 stanic.

Problém 2 [3b]: *Každá stanice vygeneruje v průměru D pasažérů, z nich si každý uniformně vybírá za svůj cíl některou ze zbývajících stanic s tím, že vzdálenost jím vybrané cílové stanice od jeho stanice výchozí musí být v intervalu $[n_0, n_1]$. Vypočítejte křivku $S(m)$.*

Jakákoli další zobecnění a přiblížení naší teorie experimentálním výsledkům jsou velmi vítána a hodnocena dle správnosti a přínosnosti. Berte však vždy v úvahu, že teorie je tím lepší, čím méně má volných parametrů (tedy křivka získaná řešením Problému 1 nám pravděpodobně bude méně užitečná než křivka nalezená v předchozí sérii, protože má volných parametrů $N + 1$, zatímco stará křivka si vystačila s jediným). Stejně tak uvažte, že za zobecnění se nepočítá jen to, že navrhnete odstranit některý předpoklad, zároveň musíte také navrhnout, jak dosáhnout výsledku bez tohoto předpokladu.

Problém 3: *Nalezněte další slabá místa našeho modelu z předchozího čísla a navrhněte pro současnou teorii vylepšení.*

A na konec série bude zadána nejdůležitější část naší práce, záměr všech doposud prováděných experimentů a nutná podmínka smysluplnosti naší teorie. Bude jí srovnání teoretických výsledků s experimentem.

Problém 4: *Pokuste se proložit naměřené výsledky (viz níže) křivkou $S(m)$ uvedenou výše a odhadněte chybu, s jakou teorie odporuje experimentům.*

Nápověda 1: Připomínám dříve zmíněnou nápovědu ke srovnávání výsledků z různých linek. Přejděte k redukované proměnné $\tilde{m} = m/N$ a k redukovanému počtu cestujících $\tilde{S} = S/C'$, kde C' je jistá konstanta (ne nutně tatáž, jako C v rovnici výše) popisující danou linku. Cílem je zajistit, aby v rámci našeho měření bylo $\tilde{m} \in [0{,}1]$ a $\tilde{S} \in [0{,}1]$ pro libovolnou linku. Ověřte sami, že teoretický tvar křivky $S(m)$ výše tato transformace nezmění.

Nápověda 2: Jak už bylo zmíněno dříve v tomto čísle, vychýlení experimentu oproti teorii může být stále způsobeno statistickou chybou, neuvažujeme-li zde jako náhodný prvek povahu chování cestujících, nýbrž náhodný výběr měřené linky. To znamená, že máme-li dojít k výsledku souhlasícímu s naší teorií, je třeba

brát linky jako rovnocenné, tj. měřit přednostně více různých linek a nepřikládat větší váhu dříve víckrát měřeným linkám.

Nápověda 3: K prokládání měřených výsledků křivkou je nejlepší využít program Gnuplot. Jestli s ním neumíte, můžete si k němu najít manuál na stránkách Fykosu: `https://fykos.cz/_media/sex/gp-zaklady.pdf`

m	Trasa A					Trasa A opačným směrem		Trasa B	Trasa C
1	30	38	29	28	39	4	4	19	44
2	45	57	40	37	56	41	33	30	46
3	45	57	40	36	57	48	40	31	54
4	24	24	16	15	28	48	40	31	54
5	31	28	27	25	30	48	40	32	54
6	38	39	35	36	39	44	37	40	52
7	38	39	35	36	38	40	32	27	53
8	38	39	35	36	38	39	33	27	29
9	49	39	35	36	38	34	31	18	23
10	42	47	47	45	49	34	24	16	10
11	41	46	42	41	47	29	23	11	2
12	41	46	43	42	47	29	23	9	–
13	41	45	43	42	47	28	19	7	–
14	41	44	43	42	40	42	28	0	–
15	30	27	28	18	19	43	33	–	–
16	10	12	5	3	5	50	28	–	–

Tabulka 1: Došlá měření tvaru křivky $S(m)$. Trasa A je autobusová linka 900242 z Jančí do Opavy, trasa B linka 230026 11 z Kolína do Žiželic, a trasa C linka 340 z Dejvické do Levého Hradce. U trasy C chybí měření ze dvou stanic linky.

Vzorové řešení

Problém 1

Zadání:

Proveďte co nejvíce měření křivky $S(m)$ na jakékoli dopravní lince autobusu, tramvaje, nebo jakéhokoli podobného dopravního prostředku. Nezapomeňte u provedených měření zhodnotit jejich věrohodnost a diskutovat, kde mohla případná nepřesnost vzniknout. Pro přehlednost také uveďte, které linky a kdy jste zkoumali.

Řešení:

Pro tento problém jednak vzorové řešení neexistuje (resp. lze nejvýše říct, že očekávaným řešením je parabola s maximem v bodě $m = N/2$), jednak by stejně nebylo zatím uvedeno, protože problém zůstává stále otevřen a zůstane tak až do konce ročníku. Experimentálních hodnot, s nimiž můžeme srovnat naši teorii, není nikdy dost a vědecká práce jejich znalost vyžaduje!

Problém 2

Zadání:
Vymyslete jeden nebo více zjednodušujících modelů, z nichž každý nějakým způsobem popíše, dle jakých pravidel reální lidé v dopravním prostředku nastupují a vystupují, aby se na tato pravidla dalo dále navázat.

Řešení:
Řešení tohoto problému bylo ve skutečnosti přineseno už v zadání předchozí série. Nejlepším modelem pro popis cestujících stále zůstává představa, že každá stanice v okamžiku příjezdu vygeneruje v průměru D pasažérů, z nichž každý si s uniformní pravděpodobností vybere svůj cíl mezi stanicemi na lince.

Další možný model popisuje bezcílné cestující; na každé stanici jich je opět vygenerováno D a každý pasažér ve stanici se s pravděpodobností p rozhodne nastoupit, zatímco každý pasažér ve voze s pravděpodobností q vystoupit.

Třetí nezávislý model přináší Mgr.MM Marco Souza de Joode, který předpokládá, že v každé stanici nastoupí konstantní počet pasažérů, z nichž každý vystoupí právě v polovině cesty, která mu v okamžiku nástupu zbývá do konečné stanice.

Ani tento problém tedy nemá jednoznačné vzorové řešení. Žádný model totiž nevystihuje plně chování reálných pasažérů, kteří se řídí velmi mnoha různými parametry, proto lze správnost modelů pouze srovnávat tím, jak dobře toto chování aproximují, tj. jak dobře odpovídají experimentům.

Problém 3

Zadání:
Náhodná veličina X odpovídá součtu tří nezávislých hodů šestistěnnou kostkou. Zamyslete se, co zde představuje množinu Ω, zakreslete do grafu pravděpodobnostní rozdělení a spočítejte jeho střední hodnotu a varianci.

Řešení:
Vygenerováno může být cokoli od 3 do 18, tedy $\Omega = \{3, \ldots, 18\}$.

Střední hodnotu a varianci bylo možné počítat složitě z definice, nebo si také uvědomit, že pro dvě nezávislé náhodné veličiny A,B platí $\mu(A+B) = \mu(A)+\mu(B)$ a $V(A+B) = V(A)+V(B)$. U střední hodnoty je toto tvrzení zřejmé, u variance lze dokázat (s notací $\mu(A) = a, \mu(B) = b$ a užitím vlastností střední hodnoty uvedených v předchozí sérii) následovně:

$$V(A+B) = \mu((A+B-a-b)^2) =$$

$$= \mu(A^2 - 2aA + a^2 + B^2 - 2bB + b^2 + 2AB + 2ab - 2aB - 2Ab)) =$$

$$= \mu((A-a)^2) + \mu((B-b)^2) + 2(\mu(AB) + \mu(ab) - \mu(aB) - \mu(Ab)) =$$

$$= V(A) + V(B) + 2(ab + ab - ab - ab) = V(A) + V(B)$$

Stačí nám tedy pro jediný hod kostkou A vypočítat střední hodnotu $\mu(A) = 3{,}5$ a varianci $V(A) = \frac{35}{12}$, čímž snadno dostaneme $\mu(X) = 3\mu(A) = 10{,}5$ a $V(X) = 3V(A) = \frac{35}{4}$.

Graf pravděpodobnostního rozložení bylo potřeba vytvořit ručně. Bylo na něm vidět, že ačkoli jde o součet tří uniformních veličin, jeho tvar se již blíží Gaussovskému rozložení, což odpovídá centrální limitní větě[8].

Evžen; JanSkvara@email.cz
e-mailová konference: tramvaj@mam.mff.cuni.cz

Téma 6 – Vrcholové pokrytí

Možná jste se při čtení předchozí série kladli otázku, k čemu je taková schopnost řešit problém minimálního vrcholového pokrytí. Samotná definice možná zní dosti abstraktně, ale ať už se snažíte rozmístit bezpečnostní kamery na křižovatky ve městě nebo se snažíte sledovat veškerý pohyb dat po lokální počítačové síti, minimální vrcholové pokrytí a jeho případná rozšíření jsou užitečné nástroje. Hledání vrcholového pokrytí není moc častý problém, ale dovolí nám vyzkoušet mnohem zajímavější techniky než běžné grafové problémy.

Ani jeden z prvních pěti grafů není moc velký. S trochou snahy by každý z nich měl být řešitelný na papíře. Doufám ale, že jste se zamysleli nad tím, jaké vlastnosti grafu by mohly být užitečné. Možná někteří z vás už začali pracovat na jednoduchých programech, které vám pomůžou řešit větší problémy.

V druhé sérii jsme zveřejnili větší grafy. Některé ještě jdou rozumně vyřešit ručně, některé jsou triviální s trochou pomoci od počítače a některé budou vyžadovat trochu inovace. Využijte toho, že se jedná o tématko – sdílejte svá pozorování a dost pravděpodobně do konce roku budete umět řešit problémy, které by nikdo z nás sám řešit neuměl.

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

Konference Borek 2018

Terminální balistika (8 b)

Bc.[MM] *Kristýna Kamenářová*

Abstrakt

V naší práci jsme se zabývali takzvanou terminální balistikou, která zkoumá chování střely ve chvíli, kdy zasáhne cíl. Konkrétně jsme stříleli ze vzduchovky různými diabolkami do připraveného želatinového gelu a sledovali jsme, v jaké vzdálenosti se diabolka zastaví, což podle nás závisí na její hmotnosti a tvaru. Problém jsme řešili nejprve teoreticky a následně jsme naše závěry ověřili experimentálně střelbou do gelu.

[8] viz https://mam.mff.cuni.cz/media/cislo/pdf/25/25-2.pdf, strana 19

Co je terminální balistika?

Balistika je aplikovaná věda zabývající se teoretickým a experimentálním studiem pohybu střel vystřelovaných z hlavňových zbraní a raket od počátku jejich pohybu v hlavni až po zasažení a zničení cíle. Uplatňuje se především ve vojenství, lovectví, sportu, kosmonautice a kriminalistice[1].

Terminální balistika je ještě společně s prenatální, vnitřní, přechodovou, vnější a postterminální balistikou jedním z odvětví kriminalistické balistiky. Zabývá se chováním a působením střely od okamžiku, kdy dopadne na cíl, do okamžiku, kdy se střela a veškeré její fragmenty přestanou pohybovat. Je tedy uplatňována především při lovu zvěře nebo při práci policejních či vojenských odstřelovačů. Částí terminální balistiky je i tzv. balistika ranivá, která studuje účinky zbraní a střeliva na biologické cíle (např. na člověka či při lovu zvěře).

Prenatální balistika studuje a popisuje děje probíhající před samotným výstřelem, při kterých vznikají na zbrani nebo na náboji stopy, například úmyslně vytvořené změny na zbrani, vzpříčení náboje v nábojišti nebo stopy po jiných závadách při nabíjení nebo mezi dvěma výstřely[2].

Vnitřní balistika se zabývá jevy, které se odehrávají uvnitř zbraně předtím, než střela opustí ústí hlavně. Patří k nim například složení náboje, hoření střelného prachu, rychlost úderu zápalníku, tlaky v nábojové komoře, rychlost střely procházející vývrtem a zpětný ráz.

Přechodová balistika se zabývá ději od okamžiku, kdy střela opustí ústí hlavně, do okamžiku, kdy na střelu ještě působí plyny vytékající z hlavně ven[1].

Vnější popisuje stabilizovaný nebo nestabilizovaný let střely prostorem od okamžiku, kdy na ni přestanou působit plyny vytékající z hlavně ven, do okamžiku, kdy střela dopadne na cíl.

Postterminální balistika zkoumá děje probíhající po prostřelení cíle (překážky). Zabývá se otázkami o událostech probíhajících poté, co střela nebo její fragment (úlomek pláště, olověné nebo ocelové jádro) či fragment cíle opustí cíl. Uplatnění má především v kriminalistice (například účinky střely po prostřelení okna, karosérie vozidla atd.)[2].

Teoretické úvahy

Jakmile jsme se seznámili s ovládáním a zacházením se zbraní, přesunuli jsme se ke zkoumání diabolek, se kterými budeme střílet. K dispozici jsme měli 4 druhy diabolek různých tvarů – dvě s ostrou, jednu s kulatou a jednu s rovnou (zploštělou) špičkou – a rozdílných hmotností od 0,48 g do 1 g. Jejich parametry jsou uvedeny v Tabulce 2.

(a) diabolka 1 **(b)** diabolka 2 **(c)** diabolka 3 **(d)** diabolka 4

Obrázek 2: Tvary zkoumaných diabolek

Diabolka (špička)	Délka [mm]	Hmotnost [g]
1 (ostrá)	7,7	1
2 (ostrá)	7,2	0,64
3 (kulatá)	5,25	0,48
4 (zploštělá)	5,35	0,49

Tabulka 2: Parametry diabolek

Dle našeho mínění nejdále doletí první diabolka, protože má ostrou špičku a je z nich nejtěžší. Naopak v nejmenší vzdálenosti se vyskytne čtvrtá diabolka, poněvadž její zploštělá přední část bude mít větší odpor nejen ve vzduchu, ale i v želatinovém gelu.

Měření

Pro experimentální měření jsme museli nejprve vyrobit balistický gel, ve kterém se budou měřit a porovnávat vzdálenosti vstřelených diabolek. Na jeho vytvoření jsme použili průhlednou želatinu, kterou bylo třeba nejprve uvařit. Aby náš gel neměl příliš řídkou konzistenci pro naše účely, bylo třeba jej vyrobit hustší. Do 4 l vody by se podle návodu na obale mělo dát 8 sáčků želatiny. My jsme uvařili gel 3× hustší (tzn. na 4 l bylo použito 24 sáčků). Ještě teplý byl gel z hrnce přelit do plastového boxu, ve kterém tuhnul do dalšího dne.

Po uvaření želatinového gelu a jeho následném ztuhnutí bylo již možné naše úsudky a teoretické závěry uvést do praxe. Stříleli jsme vzduchovkou s přibližovací zaměřovací optikou ze vzdálenosti 20 m a diabolkami o průměru 4,5 mm do gelu o výšce asi 4,5 cm a délce 55 cm. Bylo provedeno několik výstřelů s každým typem diabolky, z nichž některé nezůstaly v gelu a vyletěly ven. Opuštění gelu bylo způsobeno vyletěním diabolky vrchní a boční částí nejčastěji z důvodu špatného náklonu zbraně. Pouze ve dvou případech diabolky proletěly skrz gel.

Vzdálenosti byly měřeny od okraje gelu po místo, kde se diabolka při svém průletu zastavila, záhyb trajektorie diabolky byl zanedbán. Výsledky měření ukazuje Tabulka 3. V tabulce vidíte naměřené vzdálenosti diabolek, které v gelu zůstaly, jejich průměr a odchylku vypočítanou pomocí vzorce

$$\sigma_x = \sqrt{\frac{1}{n \cdot (n-1)} \cdot \sum_{i=1}^{n} (x_i - \bar{x})^2}$$

Ačkoli tato odchylka může být zavádějící, jelikož jsme provedli příliš málo měření, můžeme říci, že diabolky jednoho druhu doletěly do přibližně stejné vzdálenosti, jak je patrné z její naměřené hodnoty.

Nejtěžší diabolka (č. 1) doletěla nejdál díky své podstatně větší hmotnosti oproti ostatním a ostřejší špičce ve srovnání s diabolkami 3 a 4. Ačkoli diabolky 3 a 4 mají stejné hmotnosti, tak se jejich prostupnost materiálem dost liší – přibližně o 50 mm – kvůli tvaru jejich přední části. Vzdálenosti špičatých diabolek č. 1 a 2

Diabolka	Vzdálenosti [mm]	Průměr [mm]	Odchylka [mm]
1	201 200 209	203,3	3
2	170 179	174,5	5
3	160 158	159	1
4	121 110	115,5	5,5

Tabulka 3: Výsledky měření

byly rozdílné průměrně o 29 mm, zatímco vzdálenosti diabolek č. 2 a 3 se od sebe lišily jen o asi 16 mm, což je způsobeno menším rozdílem jejich hmotností.

Kromě střílení do gelu bez nějaké překážky před ním (když pomineme vzduch) jsme ještě plánovali zkoumat chování střely při průchodu různými typy látky, např. elastickou, lesklou, pevnější či tenčí, umělou i bavlněnou, ale bohužel z časových důvodů to již nebylo možné.

Závěr

Prakticky se nám podařilo dokázat, jak jsme se domnívali, že vzdálenost průstřelu gelu závisí na tvaru a hmotnosti diabolky, tzn. pokud je diabolka špičatá, tak čím je těžší, tím má větší průraznost a doletí dál. Z námi používaných diabolek byly špičaté diabolky 1 a 2, přičemž ta první byla o 0,36 g těžší a urazila v gelu průměrně asi o 30 mm delší vzdálenost v porovnání s diabolkou 2. Naopak, když diabolka měla pouze zploštělou přední část (diabolka 4), tak doletěla do mnohem menší vzdálenosti nejen ve srovnání s diabolkami 1 a 2, ale i diabolkou 3, která měla špičku zakulacenou.

Na této konfeře jsem pracovala společně s Mgr.[MM] Klárou Hlouškovou a Jakubem Kováčem pod vedením Kateřiny Čížkové na podzimním soustředění M&M v Borku ve východních Čechách. Tímto bych zároveň chtěla poděkovat Kateřině za přípravu a vedení této konfery a Kláře s Jakubem za skvělou spolupráci. Také bych chtěla poděkovat všem, kterým jsem zaslala nedokončenou podobu tohoto článku a pomohli mi svými připomínkami zvýšit jeho kvalitu.

Zdroje

[1] ONDRÁČEK, Jan a kol. *Střelecká příprava: Základy balistiky.* Masarykova univerzita [online]. Brno: Fakulta sportovních studií Masarykovy univerzity, 2011, 25. 1. 2013 [cit. 2018-12-02]. Dostupné z: `http://www.fsps.muni.cz/inovace-SEBS-ASEBS/elearning/strelba/balistika`

[2] PLANKA, Bohumil. *Kriminalistická balistika.* Plzeň: Vydavatelství a nakladatelství Aleš Čeněk, 2010. ISBN 978-80-7380-036-9.

Ovocné napětí 9 b

Mgr.MM Marie Kalousková a Bc.MM Adéla Foglarová

Všichni jistě znáte citronovou baterii, která generuje napětí a, pokud se to podaří, dokáže rozsvítit třeba i světelnou diodu. Nás zajímalo, jestli lze vytvořit podobný generátor napětí z jakéhokoliv ovoce a na čem vlastně závisí velikost generovaného napětí. Nakonec jsme se neomezily jen na ovoce, ale zkoumaly jsme i zeleninu[9].

Hypotéza

Napětí závisí nejen na použitých elektrodách, ale také na vzdálenosti, do které umístíme elektrody od sebe – to však nebylo to, na co jsme se při našem zkoumání zaměřily. Chtěly jsme zjistit, jak velké napětí budou produkovat různé druhy ovoce a ověřit si tak náš předpoklad: Čím kyselejší ovoce, tím vyšší napětí. Naše hypotéza vycházela ze znalostí míry disociace kyselin. Totiž: čím je kyselina kyselejší, tím snadněji disociuje. To znamená, že snadněji uvolňuje ionty do elektrolytu a napomáhá tak k tvorbě napětí.

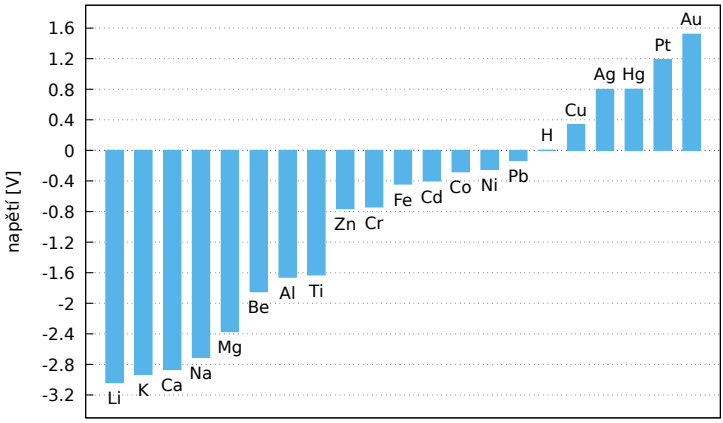
Měření

Jako způsob měření napětí jsme zvolily tradičně voltmetr, pro měření kyselosti jsme použily lakmusové papírky. U každého ovoce jsme obě veličiny změřily dvakrát — jednou pro pevný stav, podruhé pro tekutý — z ovoce jsme získaly šťávu, nebo alespoň pyré (třeba v případě brambor). Lakmusové papírky jsme přikládaly na ovoce, a pak jsme podle stupnice kyselosti zjišťovaly, jaké hodnoty pH indikovaly[10]. Platí, že čím menší je hodnota pH, tím je vzorek kyselejší. K voltmetru jsme připojovaly dvě elektrody – měděnou katodu a zinkovou anodu. Elektrody jsme vždy ponořily do stejné hloubky i vzdálenosti, abychom se co nejvíce vyvarovaly chyb měření. V případě pevného stavu byla vzdálenost 1,5 cm, v případě tekutého 3 cm. To proto, že v pevném stavu jinak neprobíhaly chemické reakce dostatečně rychle a naopak v tekutém stavu probíhaly při přiblížení elektrod příliš rychle, vzdálenost tudíž nemohla být stejná. Hodnoty pH a napětí jsme tedy pak porovnávaly vždy pro stejnou konzistenci ovoce.

Jak to funguje

Katoda, která se nabije kladným nábojem (tzn. odevzdá elektrony) při disociaci, přitahuje elektrony přicházející z anody. Mezi elektrodami probíhají chemické reakce a vzniká elektrochemické napětí. Ovoce funguje jako primární článek, tedy článek, který přímo generuje napětí. Zinek a měď jsme si vybraly kvůli jejich pozici v Beketovově řadě kovů, která udává velikost standardních redoxních potenciálů některých kovů. Čím větší je rozdíl v této veličině, tím větší napětí ovoce generuje (oba kovy by se měly nacházet na opačné straně řady vzhledem k vodíku). Se zinkovými a měděnými elektrodami se pracuje velmi často a je tedy jisté, že fungují,

[9] I když se dále v článku hovoří o ovoci, jsou tyto věty platné i pro zeleninu (pozn. red.).
[10] pH je záporný dekadický logaritmus koncentrace oxoniových kationtů



Obrázek 3: Beketovova řada elektrochemického napětí kovů

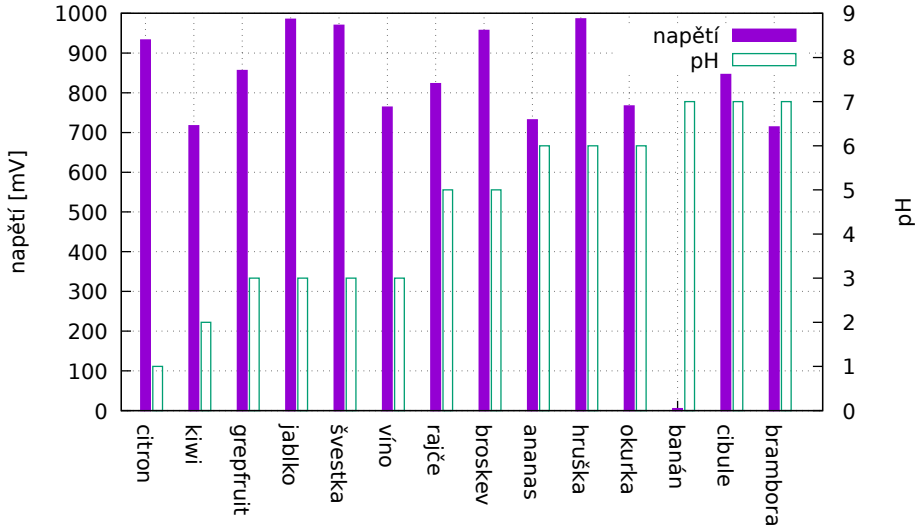
což byl další důvod k jejich výběru. Rozdíl standardního redoxního potenciálu mezi námi využitými kovy (zinkem a mědí) je 1,1 V, to bylo maximální napětí, které jsme teoreticky mohly naměřit.

	Tekutý stav			Pevný stav		
ovoce	pH	U [mV]	ΔU [mV]	pH	U [mV]	ΔU [mV]
ananas	2	845	7	6	732	6
banán	5	915	7	7	5	0
brambora	7	896	7	7	714	6
broskev	5	900	7	5	957	8
cibule	6	892	7	7	846	7
citron	1	944	8	1	933	7
grepfruit	3	810	6	3	856	7
hruška	4	913	7	6	986	8
jablko	4	910	7	3	985	8
kiwi	3	717	6	2	717	6
okurka	7	755	6	6	767	6
rajče	5	847	7	5	823	7
švestky	3	930	7	3	970	8
víno	5	727	6	3	764	6

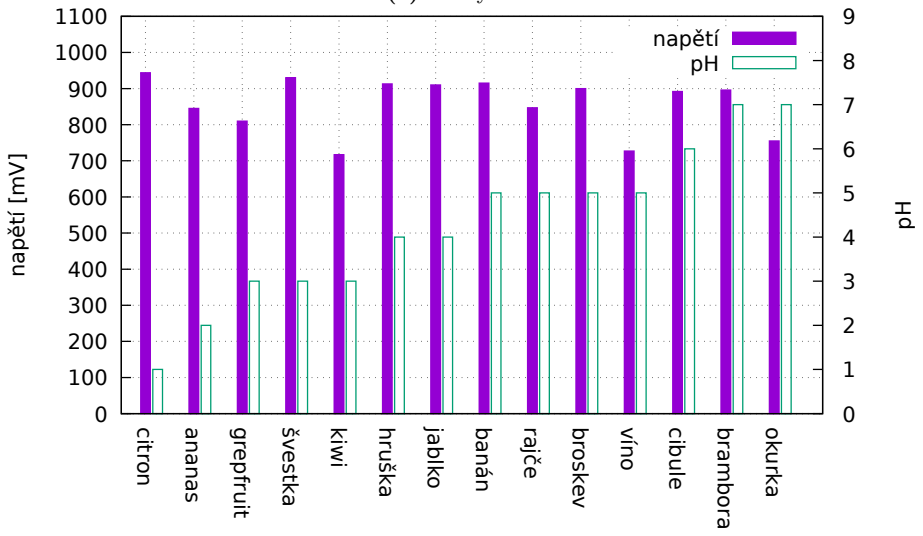
Tabulka 4: Naměřené hodnoty pro pevný a tekutý stav, U značí naměřené napětí, ΔU odchylku napětí, ovoce je seřazeno abecedně

Výsledky

Na Obrázcích 4a a 4b můžete vidět na vodorovné ose ovoce, které je vzestupně seřazené podle pH, které jsme naměřily. Pokud by bylo napětí závislé na pH, pak by plné sloupce musely stoupat přibližně stejně jako prázdné sloupce pH, tak to však není. Odchylka zmíněná v Tabulce 4 byla uvedena výrobcem multimetru.



(a) Pevný stav



(b) Tekutý stav

Obrázek 4: Srovnání naměřeného pH a napětí

Závěr

Naše hypotéza se nepotvrdila, i když to mohlo být způsobeno nepřesností měření, které jsme kvůli nedostatku času neopakovaly vícekrát. Také různá konzistence druhů ovoce mohla ovlivnit napětí. Ale myslíme si, že to bylo způsobeno hlavně tím, že se v ovoci nachází organické kyseliny, jež jsou oproti anorganickým slabší. Důvodem je disociační konstanta kyselin, takže nepůsobí tolik na velikost generovaného napětí.

Chtěly bychom poděkovat Pavle Trembulakové za přípravu a vedení této konfery na podzimním soustředění M&M v Borku 2018.

Bc.MM Adéla Foglarová a Mgr.MM Marie Kalousková

Zdroje a doporučené články

Beketovova řada kovů a elektrodové potenciály:
`https://cs.wikipedia.org/wiki/Beketovova_řada_kovů`
`https://www.wikiskripta.eu/w/Elektrodový_potenciál`
Zajímavé stránky, které nás inspirovaly:
`https://www.ceskatelevize.cz/porady/10121359557-port/201-elektrina-z-ovoce-a-zeleniny/video/`
`http://fyzmatik.pise.cz/64-elektrina-z-citronu.html`

Výsledková listina 2. čísla

Poř.	Jméno	R.	$\sum_{-1}$	Úlohy t1	t2	t3	t4	t5	t6	k	$\sum_0$	$\sum_1$
1.	Mgr.MM T. Sourada	4	34,8		9,0		14,0	3,0	1,0		27,0	34,8
2.	Mgr.MM O. Chlubna	2	24,3		1,0	9,0	7,5		2,0		19,5	24,3
3.	Mgr.MM M. Souza de Joode	2	40,3			2,5	1,0	9,0			12,5	20,5
4.–5.	Mgr.MM K. Hloušková	3	21,5					7,0			7,0	17,0
	Bc.MM V. Materna	3	17,0								0	17,0
6.	Bc.MM T. Flídr	1	13,0								0	13,0
7.	Mgr.MM M. Kalousková	3	37,8					0,5		7,0	7,5	12,3
8.–9.	Mgr.MM M. Boček	Z9	21,4					11,0			11,0	11,0
	Bc.MM M. Vícha	Z9	11,0					11,0			11,0	11,0
10.	J. Štěpo	Z9	9,5		1,5			3,0			4,5	9,5
11.	Dr.MM L. Kundratová	4	69,7								0	9,0
12.	F. Bujnovský	1	8,8								0	8,8
13.–14.	Bc.MM K. Kamenářová	4	10,7							8,0	8,0	8,0
	Mgr.MM J. Pallová	4	36,8								0	8,0
15.–17.	Bc.MM A. Foglarová	4	12,6							7,0	7,0	7,0
	Dr.MM J. Havelka	3	98,4	2,0	1,0			3,0	1,0		7,0	7,0
	Mgr.MM L. Kopfová	4	38,7						7,0		7,0	7,0

Poř.	Jméno	R.	$\sum_{-1}$	Úlohy t1	t2	t3	t4	t5	t6	k	$\sum_0$	$\sum_1$
18.	Doc.^{MM} K. Rosická	4	122,8					3,0			3,0	6,0
19.	Dr.^{MM} B. Hroncová	4	80,8		5,6						5,6	5,6
20.	V. Jůzková	2	5,5								0	5,5
21.	Mgr.^{MM} O. Gonzor	2	26,2								0	4,5
22.	J. Kvapil	1	3,7								0	3,7
23.	L. Kunčarová	3	3,6								0	3,6
24.	J. Kováč	4	3,5								0	3,5
25.	Dr.^{MM} K. Balej	4	83,4								0	3,0
26.	R. Zavřel	3	2,3								0	2,3
27.	N. Koscelanská	3	2,0					2,0			2,0	2,0
28.	Mgr.^{MM} E. Vítková	3	27,0								0	1,5
29.	J. Přerovská	3	1,0						1,0		1,0	1,0

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