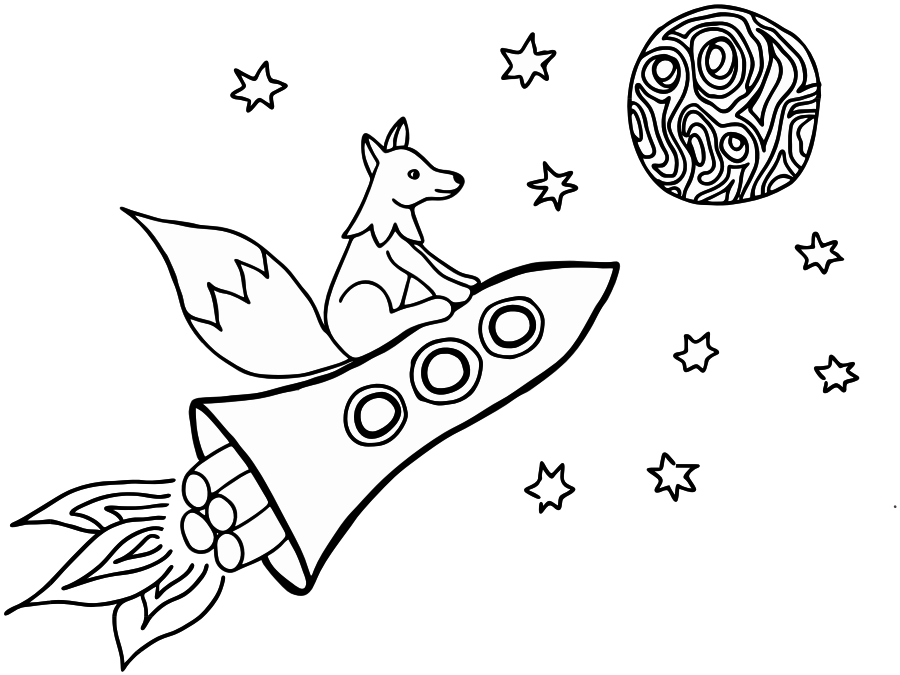
STUDENTSKÝ ČASOPIS A KORESPONDENČNÍ SEMINÁŘ

Ročník XXV
Číslo 5

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,

v rukou držíte letošní poslední číslo se zadáním. Další série bude obsahovat pouze řešení témátek a nejzajímavější řešitelské články. Zde máte poslední možnost přečíst si něco nového o kryptografii a elektronickém podepisování nebo o geometrických algoritmech či statistice počtu cestujících v tramvaji.

V sobotu 30. března začíná jarní soustředění, na které se už jistě těšíte. Pokud jste nebyli pozváni, nezoufejte a pilně řešte, na podzim budeme pořádat další.

Hodně úspěchů přejí

Organizátoři

Zadání a řešení témat

Termín odeslání 5. série: 14. 5. 2019

Téma 1 – Paradoxní výsledky

Toto témátko už nebude mít nové zadání, ale to neznamená, že nemůžete dál zkoušet testovat své vlastní teorie. Řešení předchozích sérií budou zveřejněna v příštím čísle, 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

Elektronický podpis

V tomto čísle se podrobněji podíváme na to, jak funguje elektronický podpis. Ukážeme si, jak funguje podepisování založené na asymetrické kryptografii, které je využíváno například v nových občanských průkazech nebo při použití klientských certifikátů na webu. Ale to nám nebude stačit, a tak se budeme zabývat i implementačně jednodušší formou ověření integrity zprávy, která je založena čistě na hashovacích funkcích.

Pokud jste v rámci tohoto tématu nečetli úvodní texty k hashovacím funkcím (vyšlo v 2. čísle) a k asymetrické kryptografii (najdete v 4. čísle), doporučujeme je před čtením tohoto textu alespoň zběžně prolistovat.

Co mám podepsat?

Asi už tušíte, že pokud v nějaké kryptografické operaci drobně upravím vstup, mívá to zásadní dopad na výsledek. V mnoha případech je takové chování dokonce velmi žádoucí. V případě elektronického podpisu to na nás ale klade velké nároky na přesnost: podpis bude vždy platný pouze pro přesně specifikovaný dokument. Pokud na jeho konec přidám navíc jeden prázdný řádek, podpis se stane neplatným. Než nějaká data podepíšu, musím přesně specifikovat, jak vypadají.

Pokud chci podepsat jenom nějakou skupinu hodnot, stačí mi definovat, jak je za sebe poskládám.

Problém 1 [1b]: *Představme si, že v elektronickém bankovnictví chceme podepsat příkaz k úhradě. Nejjednodušší přístup je vzít jednotlivé položky platby a naskládat je za sebe.*

Výsledný řetězec by mohl mít například tvar $UCET\_ODESILATELE \| UCET\_ADRESATA \| ZPRAVA\_PRO\_ADRESATA \| CASTKA \| KONSTANTNI\_SYMBOL \| VARIABILNI\_SYMBOL \| SPECIFICKY\_SYMBOL$*, kde* $\|$ *značí spojení řetězců.*

Je takový přístup pro reálné situace dostatečný, nebo přináší nějaké riziko? Dokážete navrhnout, jak vytvořit řetězec lépe?

Trochu komplikovanější je situace ve chvíli, kdy chceme podepsat celý dokument. Mohli bychom vzít binární soubor tak, jak je uložený na disku počítače, a k němu spočítat podpis. To se skutečně často dělá. Ale ve výsledku pak máme dva soubory – vlastní soubor s obsahem a podpis, což nemusí být praktické. Pravděpodobně jste někdy potkali podepsaný PDF dokument a žádný speciální soubor s podpisem jste nepotřebovali. To je možné díky tomu, že formát PDF (a mnohé další) počítá s možností uložení podpisu přímo do dokumentu. V rámci dokumentu je specifikovaná oblast, která je podepisovaná, a vedle je uložený podpis.

Napadá vás nějaké úskalí tohoto přístupu? Když máte podepsaný PDF dokument, není podepsaný celý dokument, ale pouze jeho část! Mnoho prohlížečů vám ale pouze zobrazí informaci o tom, že dokument je podepsaný, ale už nespecifikuje, co všechno podepsaná část obsahuje. Takže to může být třeba jenom nadpis.

Asymetrická kryptografie

Všimli jste si v minulém čísle zvláštní symetrie u RSA? Když šifruji pomocí veřejného exponentu (e), dělám úplně ty samé operace jako když pomocí privátního exponentu (d) dešifruji. Tato skutečnost nám umožňuje využít RSA „naopak“ pro elektronický podpis.

Pokud chci nějaký dokument podepsat, tak ho zašifruji pomocí privátního exponentu (d) a dokument i šifrový text, kterému nyní říkám *podpis*, zveřejním.

Každý, kdo zná můj veřejný klíč (n,e), si nyní může můj podpis ověřit. Pokud podpis dešifruje pomocí veřejného exponentu (e), dostane zveřejněný dokument.

Pro praktické použití je omezením skutečnost, že pomocí RSA nemohu šifrovat zprávy větší, než je modul (značili jsme n). Proto se typicky nepodepisuje přímo zpráva, ale její hash. Z dílu o hashovacích funkcích víme, že jsou navrženy tak, aby bylo výpočetně velmi obtížné najít dvě zprávy se stejným hashem. Pokud tedy dostaneme zprávu a podpis hashe, který zprávě odpovídá, je velmi pravděpodobné, že byla podepsána skutečně tato zpráva.

Pokud někdy uslyšíte o podepisování pomocí certifikátů (třeba i generovaných na různých čipových kartách a podobně) a bude vám to připadat hrozně složité, věřte, že za tím bývá často právě RSA, případně ne o moc složitější algoritmy nazývané DSA či ECDSA.

Problém 2 [3b]: *Pokud chceme uzavírat elektronicky smlouvy, objevuje se nám ještě jeden problém: často je důležité prokázat, kdy byla smlouva uzavřena. Napadá vás nějaké řešení, jak do podpisu přidat nezpochybnitelnou informaci o času, kdy k němu došlo?*

Problém 3 [1+10b]: *Ze stránek tématu* `https://mam.mff.cuni.cz/problem/2203/` *si můžete stáhnout PDF podepsané pomocí RSA. Zkuste zjistit co nejvíce o použitém klíči. Jeden bod je za modul a veřejný exponent. Za privátní klíč je speciální odměna 10 bodů. Pokud ho budete chtít získat, asi se budete muset porozhlédnout po nějakých známých útocích na RSA.*

Problém 4 [2b]: *Chcete poslat řešení úlohy obsahující libovolné nesmysly a získat plný počet bodů? Právě máte šanci. Za správné bude pokládáno každé řešení, které bude zasláno elektronicky v PDF podepsaném klíčem, který najdete na webových stránkách tématu.*

HMAC

Elektronický podpis je prostředkem, jak zaručit, že přenášená zpráva nebude po cestě nijak upravena. Pokud se spolu baví dvě strany a při komunikaci podepisují odesílané zprávy a ověřují podpisy přijatých zpráv, mají jistotu, že při přenosu nedošlo k žádné změně.

Ve chvíli, kdy si komunikující strany navzájem věří, je využití asymetrických podpisů zbytečně složité. Stačí využít vlastností nám již známých hashovacích

funkcí. Ty by měly fungovat tak, aby znalost hashe nedávala žádnou informaci o původním textu. Dokonce i když znám kus původního textu, tak by mi znalost hashe neměla pomoct poznat i zbylou část.

HMAC (Keyed-hash Message Authentication Code) v principu funguje tak, že si komunikující strany nejdříve vymění symetrický sdílený klíč (tedy obě strany mají stejný klíč) a potom počítají hash z klíče spojeného se zprávou.

Problém 5 [2 + 4b]: *Nejprimitivnějším přístupem pro tvorbu HMAC by bylo počítat Hash(klíč‖ zpráva), kde ‖ značí spojení textových řetězců.*

Takový přístup by ale při použití běžných hashovacích funkcí (SHA1, SHA256, RIKSHA, ...) umožňoval při odposlechu podepsaných zpráv podepisovat další zprávy i bez znalosti klíče. Za teoretický popis slabiny nabízíme 2 body.

Další 4 body lze získat za praktickou demonstraci. Zachytili jste zprávu „Mam rad jablka.“ a její podpis 50BA1E0A5660AC90. Víte, že podpis je spočítán jako RIKSHA(klíč ‖ zpráva). Dokážete získat podpis zprávy, která obsahuje mimo jiné text „Nemam rad hrusky“?

Popis hashovací funkce RIKSHA včetně její kompletní implementace lze najít ve 2. čísle.

Aby bylo zamezeno praktickým i spíše teoretickým útokům, počítá se HMAC dle standardu trochu složitěji, konkrétně:

$$\mathrm{HMAC}(K, m) = \mathrm{H}\left((K \oplus opad) \,\|\, \mathrm{H}\left((K \oplus ipad) \,\|\, m\right)\right),$$

kde H je hashovací funkce, K tajný klíč zarovnaný na velikost bloku, m podepisovaná zpráva a $opad = 5C5C\ldots5C$ a $ipad = 3636\ldots36$ dvě fixní konstanty dlouhé dle velikosti bloku hashovací funkce.

Ačkoli na první pohled může HMAC vypadat složitě, je jeho výpočet velmi rychlý a ve chvíli, kdy je potřeba zajistit integritu velkého množství dat, je velmi užitečný.

Problémy zadané v předchozích číslech

Až do termínu odeslání páté série můžete posílat také řešení všech problémů zadaných ve třetím a čtvrtém čísle a úloh 4, 5 a 6 z druhého čísla.

Pokud jste již nad těmito problémy strávili spoustu času a zajímalo by vás řešení, tak se nebojte, že byste se ho nedočkali. Vzorová řešení všech zadaných úloh otiskneme.

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

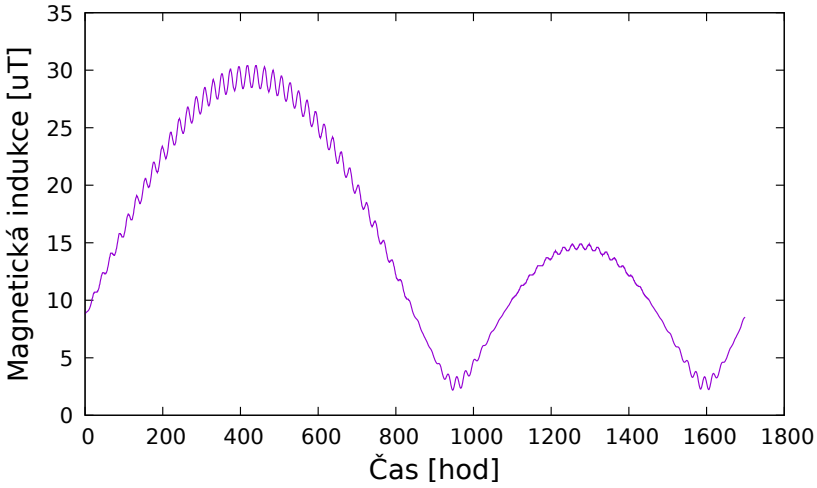
Téma 3 – Neznámý měsíc

Ke třetímu tématku můžete stále řešit třetí a čtvrtou sérii úloh. Řešení k nim najdete v příštím čísle.

Pro připomenutí přikládáme příslušná zadání.

Zadání 3. čísla

Při průzkumu měsíce jste nalezli další úlomky své lodi. Jeden obsahoval digitální magnetometr. Nechali jste ho na své základně, která je na rovníku měsíce, a zaznamenávali jste po dobu jednoho oběhu kolem planety amplitudu magnetické indukce (viz obrázek 1).



Obrázek 1: Měření magnetické indukce po dobu jednoho oběhu kolem planety

Vnější magnetické pole tělesa můžeme modelovat pomocí dvou pólů, z nichž každý leží v polovině úsečky mezi středem planety a jedním z jejích rotačních pólů. Jeden pól bude kladný, druhý záporný. Magnetickou indukci B v určitém bodě pak počítáme stejně, jako bychom počítali intenzitu elektrického pole, kdyby to byly dva elektrické náboje. Můžeme ale zanedbat konstanty, protože „magnetický náboj“ pólů Q nemá žádnou fyzikální reprezentaci. Počítáme tedy $B = \sum \frac{Q}{r^2}$, kde r je vzdálenost magnetického náboje od místa výpočtu indukce.

Co o planetě a měsíci víte předem: poloměr měsíce $R_\mathrm{m} = 5\,000\,\mathrm{km}$, poloměr planety $R_\mathrm{o} = 110\,000\,\mathrm{km}$, poloměr oběhu měsíce $a_\mathrm{m} = 400\,000\,\mathrm{km}$, sklon rotační osy planety vůči oběžné dráze měsíce $\theta = 25°$, perioda rotace měsíce 22 hod, perioda oběhu měsíce 1 700 hod.

Problém 1 [2b]: *Vysvětlete, čím je způsobený tvar grafu na obrázku 1.*

Problém 2 [5b]: *Určete amplitudu magnetické indukce na pólech měsíce, pokud zanedbáte pole způsobené planetou, a amplitudu magnetické indukce na pólech planety, pokud zanedbáte pole způsobené měsícem.*

Zadání 4. čísla

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? Můžete 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).

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

Téma 4 – Algoritmy od nuly (do n)

Geometrické algoritmy

Zadání 5. série

Vítáme vás u posledního dílu našeho tématka, ve kterém krátce zabrousíme do geometrických algoritmů. Jako obvykle ještě připomínám, že tématko je určeno pouze řešitelům, kteří algoritmy, které se snažíme vymyslet, ještě neznají.

Body a přímky

V našem krátkém úvodu do geometrických algoritmů budeme celou dobu pracovat v rovině s kartézskou soustavou souřadnic. Každý bod bude tedy reprezentován dvojicí čísel (v tomto díle tématka budeme předpokládat, že umíme pracovat s reálnými čísly stejně, jako s celými). Úsečka či přímka je potom jednoznačně určena dvojicí bodů. Jak pravděpodobně ze školy víte, každá přímka v rovině jde zapsat rovnicí $ax + by + c = 0$, kde a, b a c jsou konstanty (ke každé přímce existuje nekonečně mnoho takových rovnic – stačí celou rovnici přenásobit nenulovým číslem). Každou přímku tedy umíme zapsat dvěma způsoby (dvojicí bodů nebo rovnicí), které na sebe umíme v konstantním čase převádět. Pokud máme dané body $A[x_1, y_1]$ a $B[x_2, y_2]$, můžeme vyřešit soustavu dvou lineárních rovnic $ax_1 + by_1 + c = 0$ a $ax_2 + by_2 + c = 0$ a získat tím rovnicový tvar. Naopak dosazováním za jednu z proměnných do rovnicového tvaru můžeme získat libovolně mnoho bodů přímky. Průsečík dvou přímek tak můžeme jednoduše najít v konstantním čase – převedeme obě přímky na rovnicový tvar a vyřešíme soustavu

dvou lineárních rovnic. Podobně získáme i průsečík dvou úseček – najdeme průsečík příslušných dvou přímek a poté zkontrolujeme, že souřadnice leží ve správných intervalech.

Když máme daný bod a přímku, často se nám hodí zjistit, na které straně od přímky bod leží. Proto si pro každou přímku musíme určit, kterým směrem je orientovaná. Pro přímku danou body A, B tedy budeme vždy předpokládat, že je orientovaná směrem od A k B. Tak dokážeme o každém bodu říct, zda leží vpravo či vlevo od přímky (případně náleží přímce).

Problém 1 [1b]: *Máte zadanou orientovanou přímku $A[x_1, y_1]$, $B[x_2, y_2]$ a bod, který na ní neleží. Vymyslete, jak v konstantním čase zjistit, jestli bod leží vpravo nebo vlevo od přímky. Nezapomeňte na speciální případy, jako je přímka rovnoběžná s osou souřadnic.*

Když máme body a úsečky, můžeme si triviálně definovat n-úhelník. Určíme ho jednoznačně výčtem jeho vrcholů v pořadí, v jakém leží na jeho obvodu. Nyní už máme vše potřebné, abychom mohli algoritmicky ověřit, zda bod leží v daném n-úhelníku.

Problém 2 [1b]: *Máte konvexní n-úhelník zadaný výčtem bodů na obvodu a bod. Zjistěte, zda bod leží uvnitř n-úhelníku. Nezapomeňte na časovou složitost.*

Problém 3 [2b]: *Vyřešte předchozí problém i pro nekonvexní n-úhelníky.*

Zametání roviny

Nyní si předvedeme jednu obecnou techniku pro geometrické algoritmy – zametání roviny přímkou. Myšlenka je následující. Vezmeme si svislou přímku a posunujeme ji přes celou rovinu zleva doprava. Kdykoliv se přímka protne s nějakým objektem, zastavíme se a objekt nějakým způsobem zpracujeme. Takovéto situaci, ve které přímku zastavíme, říkáme *událost*. Ve skutečnosti samozřejmě přímkou nezametáme celou nekonečnou rovinu ani žádný spojitý interval, ale začneme od nejlevější události, skáčeme vždy na událost následující a skončíme u té poslední.

Zatím to zní velmi abstraktně, proto si zametání roviny předvedeme na konkrétním případu. Mějme zadaných n úseček v rovině. Pro jednoduchost nechť jsou úsečky v obecné poloze (tedy nenastávají žádné speciální případy jako rovnoběžnost s osou souřadnic, průsečík tří úseček v jednom bodě, dvě úsečky s více než jedním společným bodem, koncový bod úsečky ležící na jiné úsečce…). Naším cílem bude najít všechny průsečíky těchto úseček. Nejjednodušší způsob je samozřejmě projít všechny úsečky a pro každou z nich zkontrolovat průsečík se všemi zbylými úsečkami. Najít průsečík dvou úseček umíme v konstantním čase, časová složitost naivního algoritmu tedy bude $\mathcal{O}(n^2)$. Tato složitost je v obecném případě optimální, jelikož každé dvě úsečky se mohou protínat, což nám dává až $\mathcal{O}(n^2)$ průsečíků. Ve skutečnosti ale taková situace většinou nenastane. Budeme tedy předpokládat, že průsečíků úseček je přibližně $\mathcal{O}(n)$ a pokusíme se najít algoritmus s lepší časovou složitostí vzhledem k počtu průsečíků p.

Nyní tedy konečně využijeme slibované zametání roviny. Po celou dobu běhu algoritmu si budeme pamatovat dvě věci. První z nich je *průřez*, tedy seznam všech úseček, které přímka aktuálně protíná setříděný vzestupně podle souřadnic průniků s přímkou (v průřezu ale nemáme uloženy přímo souřadnice průsečíků s přímkou, které se průběžně mění, ale krajní body úseček, ze kterých umíme průsečík kdykoli v konstantním čase spočítat). Druhá věc je *kalendář*. To je seznam všech právě naplánovaných událostí, setříděný podle jejich x-ové souřadnice. V kalendáři máme uložené krajní body všech úseček, které leží napravo od zametací přímky a navíc pro každou úsečku v průřezu její případné průsečíky se dvěma úsečkami, se kterými v průřezu sousedí (jak se průsečíky do kalendáře dostanou se za chvíli dozvíte). Na začátku je tedy průřez prázdný a v kalendáři jsou pouze krajní body úseček. V každém kroku algoritmu vždy posuneme zametací přímku na následující událost (ta je hned na začátku kalendáře), následujícím způsobem upravíme průřez i kalendář a zpracovanou událost poté z kalendáře smažeme.

Pokud je událostí počáteční bod úsečky u, přidáme ji nejdříve na správné místo do průřezu (tedy mezi nějaké dvě úsečky t, v). Pokud jsme měli v kalendáři naplánován průsečík úseček t, v, odebereme ho (abychom dodrželi pravidlo z definice kalendáře, že jsou v něm uloženy jen průsečíky úseček, které v průřezu sousedí). Na závěr ověříme, zda se úsečka u protíná s úsečkami t a v a případné nalezené průsečíky přidáme do kalendáře.

Pokud jde o koncový bod úsečky, zachováme se podobně. Smažeme úsečku z průřezu a přidáme do kalendáře případný průsečík dvou úseček, které spolu nyní nově sousedí.

Pokud je událostí průsečík úseček, vypíšeme průsečík na výstup a tyto dvě úsečky v průřezu přehodíme. Opět odebereme z kalendáře případné průsečíky úseček, které spolu nově nesousedí a přidáme případné průsečíky úseček, které spolu nově sousedí.

Tím je algoritmus hotov. Proč funguje? Stačí si uvědomit, že těsně před tím, než se dvě úsečky protnou, musejí spolu v průřezu sousedit. Když si tedy v každém okamžiku pamatujeme průsečíky sousedních úseček, na žádný z nich nezapomeneme a všechny vypíšeme na výstup.

Zbývá určit časovou složitost. Algoritmus evidentně udělá $\mathcal{O}(n+p)$ kroků, jelikož v každém kroku zpracuje krajní bod úsečky nebo průsečík. Jak dlouho trvá jeden krok algoritmu? V každém kroku provedeme konstantně mnoho operací s kalendářem a s průřezem. Záleží tedy na datové struktuře, do které si kalendář a průřez uložíme a na složitosti jejích operací. My pro obojí použijeme *vyvážený binární vyhledávací strom*[2] který dokáže udržovat setříděný seznam a provádět na něm operace vkládání, mazání a hledání následníka v logaritmickém čase vzhledem k počtu uložených prvků. Jelikož v kalendáři i v průřezu je vždy maximálně $\mathcal{O}(n)$ prvků, zabere nám každý krok algoritmu čas $\mathcal{O}(\log n)$ a časová složitost celého algoritmu je rovna $\mathcal{O}((n+p)\log n)$.

Vaším posledním úkolem je využít zametání roviny a dalších znalostí z tohoto dílu tématka k vytvoření algoritmu na nalezení nejmenšího konvexního n-úhelníku, ve kterém leží všechny zadané body. Slibujeme, že k řešení nebudete potřebovat žádné pokročilé datové struktury, které jsme si neukazovali.

Problém 4 [4b]: *Máte zadáno n bodů v rovině. Vaším úkolem je vymyslet co nejrychlejší algoritmus na nalezení nejmenšího konvexního n-úhelníku, který obsahuje všechny body. Můžete si to představit tak, že kolem hřebíků zatlučených v rovné desce natáhneme velkou kruhovou gumičku. Gumička tvoří hledaný konvexní n-úhelník. Výstupem vašeho algoritmu by měl být seznam vrcholů hledaného n-úhelníku v pořadí, v jakém leží na obvodu (každý z vrcholů bude evidentně roven některému z bodů na vstupu).*

Řešení 3. série

Nejdříve bychom se rádi omluvili za dvě chyby, které se nám vloudily do zadání 3. série. První z nich se nachází v zadání prvního problému – s využitím datových struktur, o kterých jsme v tématku zatím psali, se nedají splnit všechny podmínky. Konkrétně jsme do zadání omylem přidali požadavek na přidávání vrcholů (k jeho vyřešení bychom potřebovali rostoucí pole). Druhá chyba je ukryta v zadání problémů 2 a 3. Prohledávání do šířky, které jste měli v těchto problémech vymyslet se nedá jednoduše získat memoizací triviálního řešení. Za chyby se omlouváme a pokusíme se jich do budoucna vyvarovat.

[2] Vyvážené binární vyhledávací stromy si bohužel nestihneme ukázat, můžete si o nich přečíst třeba v kuchařce našich kamarádů z KSP: `https://ksp.mff.cuni.cz/kucharky/vyhledavaci-stromy/`

Problém 1

Zadání:
Vymyslete, jak do paměti uložit graf, abychom pro každý jeho vrchol mohli v čase $\mathcal{O}$(počet jeho sousedů) najít všechny jeho sousedy a abychom zároveň uměli do grafu v konstantním čase přidat hranu či vrchol (tím už dokážeme zároveň celý graf v čase $\mathcal{O}(n+m)$ vyrobit, pokud nám někdo postupně dává jeho vrcholy a hrany). Nemusíte ošetřovat případy nevalidních vstupů, jako je třeba přidání již existujícího vrcholu.

Řešení:
Jak již bylo zmíněno, podmínka na přidání vrcholu v konstantním čase byla v zadání nadbytečná, plný počet bodů tedy bylo možné získat i za řešení, které ji nesplňovalo.

Graf o n vrcholech a m hranách reprezentujeme polem délky n, ve kterém je na i-té pozici uložen spojový seznam všech sousedů i-tého vrcholu (v případě izolovaného vrcholu prázdný seznam). Když tedy chceme najít všechny sousedy vrcholu, najdeme v konstantním čase jeho pozici v poli a v čase lineárním vzhledem k počtu sousedů projdeme spojový seznam jeho sousedů. Pro přidání hrany mezi vrcholy u, v stačí oba vrcholy v konstantním čase najít v poli, na konec seznamu na u-té pozici přidat v a na konec seznamu na v-té pozici přidat u, což nám všechno zabere pouze konstantní čas. Pro přidávání vrcholů bychom potřebovali rostoucí pole, které jsme ale v tématku bohužel nestihli probrat.

Problémy 2 a 3

Zadání:
Vymyslete způsob, jak pro každý vrchol grafu spočítat jeho vzdálenost od daného vrcholu v.

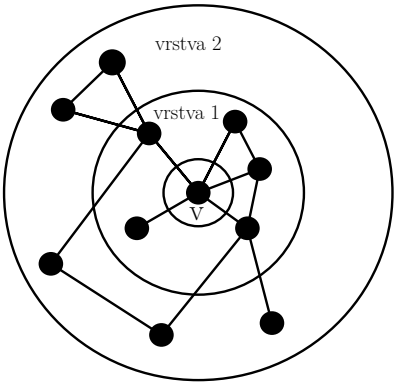
Zkuste vylepšit memoizací časovou složitost vašeho řešení problému 2. Nezapomeňte říct, jakou datovou strukturu použijete k ukládání již spočítaných výsledků funkce při memoizaci a zohlednit to v analýze časové složitosti.

Řešení:
Jak jsme psali již v úvodu, v řešení nakonec memoizaci nepoužijeme.

Vrcholy ve vzdálenosti 1 od v najdeme triviálně – jsou to jeho sousedi. Vrcholům ve vzdálenosti k dále budeme říkal vrcholy v k-té vrstvě. Všechny vrcholy ve druhé vrstvě jsou sousedy vrcholů z první vrstvy. Naopak to ale neplatí – ne každý soused vrcholu z první vrstvy leží ve druhé vrstvě. Vrcholy z první vrstvy totiž navíc mohou sousedit s jinými vrcholy z první vrstvy a s vrcholem v. V druhé vrstvě tedy leží právě ti sousedi vrcholů z první vrstvy, kteří neleží v první ani nulté vrstvě.

Analogicky můžeme zadefinovat vrcholy z třetí vrstvy – jsou to právě ti sousedi vrcholů z druhé vrstvy, kteří neleží v nulté, první ani ve druhé vrstvě[3]. Obecně

[3]Nultou vrstvu ve skutečnosti ani nemusíme zakazovat – vrchol k-té vrstvy totiž může sou-



Obrázek 2: Rozdělení vrcholů do vrstev

v k-té vrstvě leží právě ti sousedi vrcholů z $(k-1)$-ní vrstvy, kteří neleží ve vrstvách 0 až $k-1$. Pokud jsme tedy nalezli prvních $k-1$ vrstev, dokážeme k-tou vrstvu jednoduše najít v čase $\mathcal{O}(\textit{počet sousedů vrcholů z } (k-1)\textit{-ní vrstvy})$.

Takovým to způsobem tedy můžeme postupně nalézt všechny vrstvy (tedy pro každý vrchol spočítat vzdálenost od v). Vytvoříme si pole délky n, ve kterém je na i-té pozici uloženo, v jaké vrstvě leží vrchol i (u zatím nezařazených vrcholů může být třeba -1). Na začátku nastavíme vrstvu v na 0 a vložíme ho do fronty. Ve frontě tedy nyní máme celou nultou vrstvu. Poté vždy odebereme vrchol z fronty, zjistíme jeho číslo vrstvy k, projdeme všechny jeho sousedy a těm, kteří ještě v žádné vrstvě neleží, nastavíme vrstvu $k+1$ a přidáme je do fronty. Je dobré si uvědomit, že ve frontě máme vždy vrcholy dvou vrstev – z jedné z nich odebíráme vrcholy a do druhé přidáváme. Nemusíme si ale pamatovat, kde v danou chvíli leží rozhraní mezi vrstvami, když při odebrání vrcholu z fronty vždy zkontrolujeme číslo jeho vrstvy.

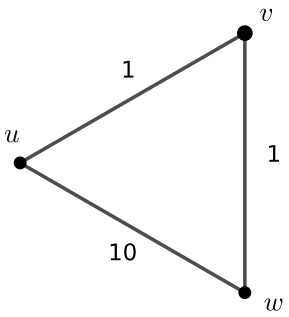
Jaká je časová složitost algoritmu? Každý vrchol do fronty jedou přidáme a jedou odebereme, to nám bude trvat $\mathcal{O}(n)$. Navíc ještě během odebírání procházíme všechny sousedy odebíraného vrcholu. To nám bude trvat tolik času, kolik má vrchol sousedů, což je, sečteno přes všechny vrcholy, dvojnásobek počtu hran grafu, tedy $\mathcal{O}(m)$. Celé prohledání grafu do šířky tedy stihneme v čase $\mathcal{O}(n+m)$.

Problém 4

Zadání:

Najde váš algoritmus nejkratší cestu i v ohodnoceném grafu? (V takovém případě myslíme délkou cesty součet ohodnocení jejích hran, viz výše.) Pokud ano, vysvětlete proč. Pokud ne, najděte protipříklad.

sedit pouze s vrcholy ve vrstvách $k-1$, k a $k+1$, protože vrstvy udávají vzdálenosti od v a není možné, aby spolu sousedily vrcholy, jejichž rozdíl vzdáleností od v je větší než 1.



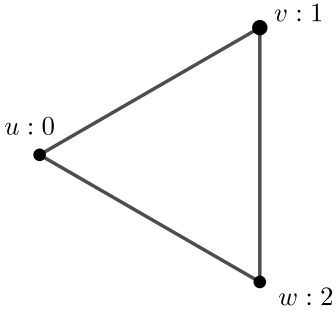
Obrázek 3: Protipříklad pro problém 4

Řešení:
Prohledávání do šířky v ohodnoceném grafu nejkratší cestu najít nemusí. Jako protipříklad nám stačí trojúhelník (úplný graf na třech vrcholech) u, v, w. Hrany z počátečního vrcholu u ohodnotíme čísly 1 a 10 a zbylou hranu (v, w) číslem 1. Prohledávání do šířky z vrcholu u potom najde pro vrcholy u, v, w vzdálenosti od u 0, 1, 10, přestože skutečné vzdálenosti jsou 0, 1, 2.

Problém 5

Zadání:
Ukažte příklad, kdy DFS (prohledávání do hloubky) najde cestu, která nebude nejkratší.

Řešení:
Stačilo nají libovolný graf obsahující cyklus. Nejjednodušším protipříkladem je tedy opět trojúhelník u, v, w. DFS spuštěné z vrcholu u najde do jednoho ze zbylých vrcholů cestu délky 1 a do druhého cestu délky 2, i když do obou vrcholů vede zřejmě cesta délky 1.



Obrázek 4: Protipříklad pro problém 5

Problém 6

Zadání:
Ukažte, že když je graf souvislý, navštíví DFS všechny vrcholy.

Řešení:
Pro spor předpokládejme, že v nějakém souvislém grafu existuje alespoň jeden vrchol, který DFS nenajde. Ze všech takových vrcholů vezmeme ten, jehož vzdálenost od počátečního vrcholu u je nejmenší a označíme si ho v. Protože je graf souvislý, existuje nejkratší cesta z u do v. Podívejme se podrobněji na předposlední vrchol w této cesty. Jeho vzdálenost od u je o jedna menší, než vzdálenost v od u, DFS ho tedy určitě našlo (protože v jsme volili jako nejbližší vrchol, který nenajde). Když DFS došlo do w, muselo z něj navštívit všechny nenavštívené vrcholy, tedy i vrchol v, což je spor s tím, že v je nenavštívený.

Problém 7

Zadání:
Mějme graf, říkejme mu G, a proveďme na něm DFS. Vezměme všechny hrany G, po kterých projdeme v průběhu DFS, a ostatní vymažme. Vysvětlete, proč bude výsledný graf strom.

Řešení:
Strom je souvislý graf, který neobsahuje kružnice. Náš graf hran, po kterých prošlo DFS (říkejme mu H) je určitě souvislý (protože každé dva vrcholy jsou spojeny cestou přes počáteční vrchol). Stačí tedy dokázat, že graf H neobsahuje kružnice. Každá kružnice v grafu H musela být kružnicí i v původním grafu G. Stačí nám tedy dokázat, že pro každou kružnici v G platí, že alespoň po jedné její hraně DFS neprojde.

Vezměme si tedy nějakou kružnici k v grafu G. Pokud do žádného z vrcholů k DFS nevstoupilo, platí pro každou hranu kružnice, že neleží v grafu H. V opačném případě si vrchol, do kterého DFS vstoupilo jako první, označíme v_1. Toho ze sousedů v_1 na kružnici k, do kterého DFS vstoupilo dříve, si dále označíme v_2 a zbylé vrcholy si označíme dále po řadě podél obvodu kružnice $v_3, v_4 \dots v_t$. DFS tedy vstoupilo do v_1, a nějaký čas poté vstoupilo do v_2. To udělalo určitě předtím, než uzavřelo v_1, jelikož v_1 a v_2 jsou sousedi. Platí tedy, že v_1 a v_2 jsou otevřené a v_t je stále nenavštívený. Nyní si uvědomíme, že než DFS uzavře v_2 a vrátí se z něj (možná přes nějaké další vrcholy) do v_2, musí nejdříve navštívit v_3. Než ale uzavře v_3, musí navštívit v_4. Stejnou úvahu můžeme dále opakovat, proto platí, že než DFS uzavře v_2 a vrátí se do v_1, musí nejdříve navštívit všechny zbylé vrcholy kružnice, tedy i v_t. Vrchol v_t tedy DFS nenavštíví z vrcholu v_1. Zároveň DFS určitě vrchol v_1 nenavštíví z vrcholu v_t, jelikož v_1 je první navštívený vrchol na kružnici k. Tím jsme dokázali, že DFS neprojde po hraně (v_1, v_t), což je hrana kružnice k. Úvaha evidentně funguje pro každou kružnici, čímž je dokázáno, že pro každou kružnici v G platí, že alespoň po jedné její hraně DFS neprojde.

Problém 8

Zadání:
Máme na vstupu zakořeněný strom reprezentující matematický výraz následujícím způsobem. Kořen odpovídá celému výrazu. V každém vnitřním vrcholu (tedy „nelistu“ – kořen je také vnitřním vrcholem) je operace $+$, nebo $*$. V listech jsou čísla. Strom pak můžeme převést na výraz tak, že rekurzivně převedeme na výraz všechny syny aktuálního vrcholu, dáme je do závorky a na těchto podvýrazech provedeme operaci, která je v aktuálním vrcholu. Proto budeme navíc předpokládat, že každý vnitřní vrchol má alespoň 2 syny (operace $+$ a $*$ totiž vyžadují alespoň dva operandy). Popište algoritmus, který vyhodnotí zadaný strom výrazu. Nezapomeňte na časovou složitost.

Řešení:
Využijeme přímočaře návod na stromové algoritmy. Strom již máme zakořeněný a každý list už má na začátku určenou svou hodnotu. Vytvoříme si tedy rekurzivní funkci, která bere jako parametr vrchol, pokud je v něm číslo, vrátí ho, a pokud je v něm operace, zavolá se rekurzivně na všechny syny vrcholu, aplikuje operaci na výsledky rekurzivních volání a výsledek vrátí.

Zbývá nám určit časovou složitost. Funkci voláme jednou na každý vrchol, máme tedy $\mathcal{O}(n)$ volání. Jedno volání sice trvá lineárně vzhledem počtu synů, můžeme ale tento čas „naúčtovat“ těmto synům, čímž dostaneme konstantní čas na jedno volání a celý algoritmus tedy poběží v $\mathcal{O}(n)$.

Problém 9

Zadání:
Vymyslete lineární algoritmus, který najde všechny mosty v souvislém grafu. Poradíme, že se vám bude hodit prezentovaný způsob návrhu algoritmů pro stromy (s mírnou úpravou pro nalezení všech mostů) a klasifikace hran z DFS. Tři body dostanete za lineární řešení, které zjistí, zda nějaký most v grafu existuje. Část bodů vám přidělíme i za pomalejší algoritmus.

Řešení:
Jak již bylo naznačeno v zadání, prohledáme graf nejdříve do hloubky, čímž získáme strom (tvořený vrcholy původního grafu a stromovými hranami). Žádná ze zpětných hran jistě netvoří most, protože po jejím odstranění nám stále zbývají všechny stromové hrany, po kterých jde přejít mezi libovolnými dvěma vrcholy. Stačí nám tedy zjistit, které ze stromových hran jsou mosty. Zvolme si tedy některý vrchol v a ptejme se, kdy je hrana e vedoucí z v nahoru (směrem do kořene) mostem. Pokud z podstromu pod v (patří do něj i samotný vrchol v) vede nějaká zpětná hrana ven (tedy do nějakého vrcholu nad v), potom e nemůže být most, protože po jejím odstranění graf zůstane souvislý – podstrom pod v je spojen se zbytkem grafu touto zpětnou hranou. Naopak, pokud z podstromu pod v žádná zpětná hrana ven nevede, potom po odstranění e nebude existovat žádná cesta mezi kořenem stromu a v a e je tedy most.

Z toho plyne, že nám stačí pro každý vrchol v spočítat, kam nejvýše vede zpětná hrana z podstromu zavěšeného pod ním. Vrcholům stromu tedy přiřadíme čísla podle toho v jaké leží *úrovni* – kořen leží v úrovni 0, jeho synové v úrovni 1 a tak dále. *Dosah* vrcholu v nám bude říkat, do jaké nejvyšší úrovně vede hrana z podstromu pod v. Dosah v nám jednoznačně určuje, zda je hrana vedoucí z v nahoru most, stačí nám tedy pro každý vrchol spočítat jeho dosah. To je ale velmi podobné problému 8. Když totiž známe dosah všech synů vrcholu v, dokážeme jednoduše spočítat dosah v – bude to prostě minimum z dosahů jeho synů a z úrovní konců všech zpětných hran, které z něj vedou.

Vytvoříme si tedy rekurzivní funkci, která dostane jako parametr vrchol v, rekurzivně spočítá dosahy jeho synů, projde úrovně všech jeho sousedů a z těchto čísel vybere minimum. Pokud je minimum větší nebo rovno úrovni v, označí hranu vedoucí z v nahoru jako most. Tuto funkci stačí zavolat na kořen stromu a máme všechny mosty označeny.

Poslední věc, na kterou jsme zapomněli, je počítání úrovní vrcholů. Ty můžeme ale jednoduše zjistit při počátečním prohledáváním do hloubky – počátečnímu vrcholu nastavíme úroveň 0 a vždy, když vstoupíme do vrcholu, nastavíme mu úroveň o jedna větší, než má vrchol, ze kterého do něj vstupujeme (podobně jako když jsme u prohledávání do šířky počítali vzdálenosti).

Nyní nám zbývá určit časovou složitost. Prohledávání do hloubky nám zabere čas $\mathcal{O}(m+n)$. Volání funkce na jeden vrchol trvá $\mathcal{O}(\textit{počet synů vrcholu})$ a na každý vrchol ji voláme právě jednou, každou hranou i vrcholem tedy opět strávíme konstantní čas, což nám dává na celý graf $\mathcal{O}(m+n)$. Protože je graf souvislý, je hran alespoň tolik, jako vrcholů a celý algoritmus tudíž běží v čase $\mathcal{O}(m)$.

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

Téma 5 – Přeplněná tramvaj

Úvod

Připomínám z předchozích sérií, že toto téma se zabývá vývojem počtu pasažérů uvnitř dopravního prostředku v průběhu jeho cesty linkou, značíme $S(m)$ střední hodnotu počtu pasažérů sedících v prostředku po projetí m stanic po výjezdu z depa. Dále připomínám, že toto je poslední zadání tohoto ročníku, čili pro vás všechny poslední příležitost k nasbírání bodů a zlepšení vaší pozice v žebříčku. Můžete se inspirovat např. Mgr.MM Ondřejem Chlubnou, který ve třetí sérii jako jediný vyřešil všechny úlohy zabývající se teoretickým popisem křivky $S(m)$, Bc.MM Kristýnou Kamenářovou, která tuto křivku experimentálně změřila na lince A pražského metra v obou směrech, nebo Mgr.MM Klárou Hlouškovou a jejími úvahami nad prováděním experimentů tohoto typu.

Přeji hodně štěstí při řešení úloh této série!

Zadání

Při řešení všech úloh můžete využívat výsledky nalezené při řešení úloh zadaných v předchozích sériích. Zde uvádím nejdůležitější dva z nich. Postup, jak k nim dospět, naleznete ve vzorových řešeních v tomto a následujícím čísle.

Na ideální lince s rovnoměrně vytíženými stanicemi je střední počet pasažérů v tramvaji mezi stanicemi m a $m+1$ roven:

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

kde N je celkový počet stanic na lince a C je jistá konstanta závislá na čase a na volbě linky.

Na reálné lince vytíženost jednotlivých stanic popisujeme koeficienty $\alpha_1, \ldots, \alpha_N$ tak, aby $\alpha_1 + \cdots + \alpha_N = N$ a zároveň zůstal zachován poměr $\alpha_1 : \ldots : \alpha_N = D_1 : \ldots : D_N = p_1 : \ldots : p_N$, kde D_k je střední počet pasažérů vygenerovaných na k-té stanici a p_k je pravděpodobnost, že si kterýkoli z těchto pasažérů vybere za cíl stanici[4] k. Na takové lince lze výše uvedený vztah zobecnit do tvaru:

$$S(m) = C(\alpha_1 + \cdots + \alpha_m)(\alpha_{m+1} + \cdots + \alpha_N)$$

kde C, N mají stejný význam jako v předchozí verzi (ověřte sami, že pro $\alpha_1 = \cdots = \alpha_N$ se oba tvary shodují). Tohoto tvaru lze intuitivně dosáhnout vykreslením paraboly popisující rovnocenné stanice a jejím *rozsekáním* na segmenty, jejichž délka odpovídá velikosti koeficientů α_k příslušných stanic. Pro detaily viz vzorové řešení v dalším čísle.

Problém 1: *Linka B pražského metra má 24 stanic. Evžen nastoupil v první stanici, ve které se zaplnilo K procent z kapacity míst k sezení, a jede až na konečnou. Evžen nechce sedět, pokud by to mělo znamenat, že na někoho jiného nezbude místo k sezení, zároveň mu je ale kontakt s cizími lidmi natolik nepříjemný, že bude raději stát celou cestu, než aby si sedl a někomu pak případně své místo nabídl.*

Při jakých hodnotách C, K si může Evžen hned po svém nástupu sednout? Předpokládejme, že nám není známa vytíženost jednotlivých stanic, proto uvažujeme stanice rovnocenné.

Problém 2: *Náš model předpokládá, že je na každé stanici vygenerován nějaký náhodný počet pasažérů. Dosud jsme uvažovali, že je nám známa střední hodnota tohoto počtu D, konkrétní povahou generování pasažérů jsme se však nezabývali. Cílem této úlohy je zamyslet se nad rozložením pravděpodobnosti pro počet vygenerovaných pasažérů.*

Předpokládejme, že každá stanice linky poskytuje jediné dopravní spojení právě jednomu sídlišti, kde bydlí M lidí (M je v řádech stovek až tisíců), a každý

[4] podrobnější popis našeho modelu naleznete v zadání 3. čísla: `https://mam.mff.cuni.cz/media/cislo/pdf/25/25-3_OOYcqIX.pdf`, str. 31

člověk zde bydlící se může rozhodnout s pravděpodobností p někam jet (tj. být „vygenerován“ jako pasažér) nebo $1-p$ zůstat na sídlišti.

a) *Vypočítejte v takovém modelu pravděpodobnost $P_{M,p}(n)$, že bude v dané stanici vygenerováno n pasažérů při pevně zadaných parametrech M,p. Nedaří-li se vám nalézt přesný výsledek, stačí alespoň dobrá aproximace*[5]*. [2b]*

b) *Vypočítejte při známých parametrech M,p pro počet vygenerovaných pasažérů střední hodnotu D a střední kvadratickou odchylku σ. [1b]*

c) *Pokuste se pro proceduru generování pasažérů nalézt lepší model, než je ten uvedený výše.*

Stanice	m	$S(m)$				
(depo)	0	0	0	0	0	0
I. P. Pavlova	1	6	8	8	5	4
Dětská nemocnice Karlov	2	14	7	7	7	16
Apolinářská	3	18	11	9	12	23
Větrov	4	18	9	11	12	26
U nemocnice	5	17	6	5	6	22
Palackého náměstí	6	11	5	3	4	7
Karlovo náměstí	7	16	9	6	10	8
U nemocnice	8	22	10	11	13	13
Větrov	9	22	9	11	11	10
Apolinářská	10	14	6	13	8	13
Na bojišti	11	12	4	10	9	14
Kateřinská	12	12	4	9	10	14
I. P. Pavlova	13	0	0	0	0	0

Tabulka 1: Naměřené počty cestujících v autobusech linky 148 vyjíždějících v 13:00, 13:30, 14:00, 14:30 a 15:00

Problém 3: *Na autobusové lince 148 Pražské MHD byly v uvedených časech naměřeny hodnoty uvedené v tabulce 1. Tato linka má zajímavou vlastnost, že ačkoli se sama prezentuje jako cyklická (výchozí stanice je totožná s cílovou), některými stanicemi projíždí dvakrát, pokaždé v jiném směru, a představuje proto jistý přechod mezi cyklickou a lineární linkou. Proveďte pro ni následující úkoly.*

a) *Vypočítejte pro tuto linku křivku $S(m)$ za předpokladu rovnocenných stanic. Berte při tom v potaz, že pro pasažéry jedoucí z některých stanic nepřipadají některé jiné stanice v úvahu jakožto cíl cesty. Při výpočtu se můžete inspirovat vzorovým řešením Úlohy 3.1a), viz níže. [2b]*

[5] může vám přijít vhod náš článek o základech teorie pravděpodobnosti ze druhého čísla: `https://mam.mff.cuni.cz/media/cislo/pdf/25/25-2.pdf`, str. 18

b) Proveďte tentýž úkol s předpokladem stanic, jejichž vytíženost je popsána koeficienty α_k. [2b]

c) Srovnáním s experimentálními hodnotami z tabulky níže odhadněte časový vývoj parametru $C = C(t)$ z úlohy a) a parametrů $C(t), \alpha_1(t), \ldots, \alpha_N(t)$ z úlohy b). [2b]

d) Vymyslete způsob, jak číselně kvantifikovat odchýlení experimentálních výsledků od vašich teoretických a na základě toho rozhodněte, zda je přesnější metoda a) nebo b) a jak moc. [2b]

Problém 4: *Bez nutnosti výpočtu se zamyslete nad následujícími problémy:*

a) Může se stát, že by měla křivka $S(m)$ více než jedno lokální maximum? Za jakých podmínek?

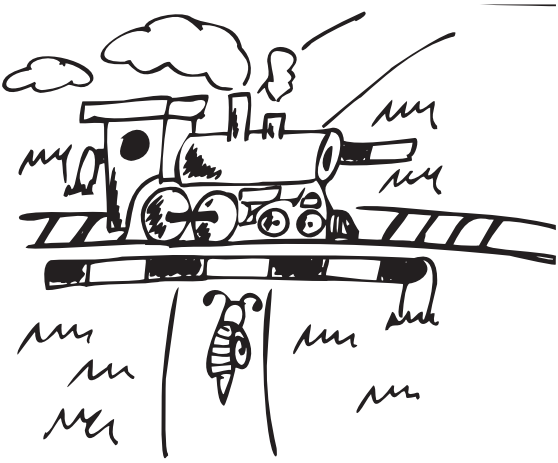
b) Co se stane, když jsou na lince dvě stanice jistým způsobem propojené a lidé mezi nimi jezdí častěji, ačkoli se taková dvojice z hlediska ostatních stanic nechová nijak výrazně (např. u jedné stanice stojí Matfyz a u druhé koleje). Dala by se tomuto případu nějak přizpůsobit naše teorie?

c) Lze užívaný popis křivky $S(m)$ při velmi vysokých hodnotách N nějak upravit/zjednodušit?

Problém 5: *Poslední příležitost k dokončení úkolů setrvávajících po celý ročník.*

a) Změřte vývoj počtu pasažérů na vybrané lince autobusu, vlaku, tramvaje nebo metra.

b) Přineste vlastní návrhy na zlepšení teorie vybudované v předchozích sériích tohoto témátka.



Vzorové řešení

Problém 1

Zadání:
Spočítejte vývoj střední hodnoty počtu lidí v prostředku za předpokladu, že se lidé chovají, jak je zde popsáno, a zakreslete křivku závislosti $S(m)$. A to:

a) Na lineární lince, tj. takové, která má začátek a konec a žádná stanice na ní není zastoupena více než jednou. [3b]

b) Na cyklické lince, tj. takové, kde se po projetí n různých stanic prostředek vrátí do první stanice a svou jízdu opakuje. Předpokládejme, že po lince jezdí vozy v obou směrech. [3b]

c) Zamyslete se, jaký vliv na platnost vašich řešení úloh a) a b) může mít počet vozů zajišťujících dopravu po lince.

Řešení:
Část a) Nechť tramvaj právě vyjela ze stanice m. Kterýkoli vygenerovaný člověk sedí uvnitř právě tehdy, nastoupil-li do ní v k-té stanici, přičemž $k \leq m$, a zároveň je jeho cílem některá z $N - m$ ještě neprojetých stanic. Pasažér nastoupivší v k-té stanici měl na výběr z $N - 1$ možných cílů, některý z $N - m$ neprojetých si tedy zvolil s pravděpodobností $p_k = \frac{N-m}{N-1}$. Dle zákona velkých čísel z k-té stanice do bodu mezi stanicemi m a $m + 1$ dorazí v průměru Dp_k pasažérů. Ze všech možných výchozích stanic $k = 1, \ldots, m$ do tohoto bodu tedy dorazí v průměru pasažérů:

$$S(m) = \sum_{k=1}^{m} Dp_k = D \sum_{k=1}^{m} \frac{N-m}{N-1} = D\frac{m(N-m)}{N-1} = Cm(N-m)$$

Hledaná křivka $S(m)$ v předpokládaném modelu klasických pasažérů a rovnocenných stanic má tedy tvar paraboly s vrcholem uprostřed linky a s kořeny v koncových bodech linky. Při dané délce linky je zde jediným volným parametrem jakási *míra vytíženosti* C, ve které musí být jistým způsobem zahrnut typ dopravního prostředku, denní hodina i celková vytíženost linky. Navíc lze tento parametr z křivky vytknout, jak bylo dříve předpokládáno[6], tudíž lze skutečně přejít k redukované křivce $\tilde{S} = S \cdot N/C$ v redukované proměnné $\tilde{m} = m/N$, která by měla být stejná pro všechny linky splňující naše předpoklady.

Část b) Předpokládejme lichý[7] počet stanic na lince, nechť je to $2n + 1$. Dále předpokládejme, že cestující vždy volí nejkratší cestu mezi stanicemi (a že je

[6] viz Nápověda k řešení; Téma 5; Číslo 2: `https://mam.mff.cuni.cz/media/cislo/pdf/25/25-2.pdf`

[7] Při sudém počtu $2n$ stanic se bude vše chovat stejně, jen bude problematickou stanice ve vzdálenosti $n-1$, u které nebude jednoznačně dáno, kterým spojem se k ní dopravit. Každopádně tento případ je pro naše pozdější účely nezajímavý.

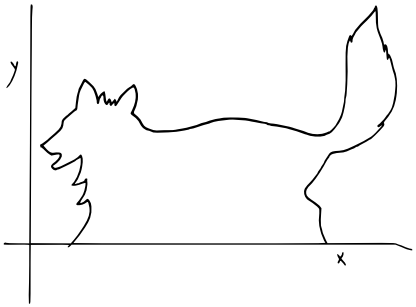
linka natolik frekventovaná, aby menší počet mezistanic skutečně vždy odpovídal časově kratší cestě).

Pasažér vygenerovaný v k-té stanici má na výběr z $2n$ možných cílů, ve směru právě projíždějící tramvaje jich ale leží jenom n, tudíž nastoupí s pravděpodobností $1/2$. Nastoupil-li takový pasažér, do stanice $k+j$ tímto spojem dorazí s pravděpodobností $p_j = \frac{n-j}{n}$, je-li $j \leq n$, a $p_j = 0$ pro $j \geq n$. Využijeme Bayesův vzorec a opět vysčítáme pravděpodobnosti p_j přes všechny možné výchozí body k, dostaneme dva možné výsledky. Je-li možných výchozích bodů méně než n, tj. tramvaj od zahájení provozu projela méně než půl linky, tj. $m \leq n$, je výsledkem:

$$S(m) = D \sum_{k=0}^{m-1} \frac{N-k}{2N} = D\frac{(2N-m+1)m}{2N}$$

a pro všechna $m \geq n$:

$$S(m) = D \sum_{k=0}^{N-1} \frac{N-k}{2N} = D\frac{N(N+1)}{2}\frac{1}{2N} = \frac{D(N+1)}{4}$$



Část c) Cílem bylo všimnout si, že na méně frekventované lince, pro demonstraci řekněme lince s jediným vozem, se projeví skutečnost, že vygenerovaní cestující, kteří při prvním průjezdu nenastoupili, čekají na stanici, dokud se vůz nevrací v jejich směru. To se projeví tím, že při druhém průjezdu linkou v k-té stanici nečeká D pasažérů, nýbrž $D + \frac{N-k}{N-1}D$, což je jednak výrazně odlišný výsledek a jednak znamená jistou nerovnocennost stanic z hlediska nástupů, která tím činí celý náš předchozí postup neplatným.

Oproti tomu na velmi frekventované lince, řekněme lince, kde se v každé stanici vždy setkají vozy z obou směrů, bychom tento efekt nepozorovali, protože by po průjezdu prostředku ve stanici nikdy žádný pasažér nezbyl.

Na reálné lince se sice stává málokdy, že by se dva protijedoucí spoje ve stanici setkaly, určitá fyzikální intuice nám však říká, že dokud jsou jejich časy příjezdů posunuty jen jistou o malou konstantu (řekněme v řádu minut), naše aproximace využívající rovnoměrné generování cestujících by měla být stále vcelku platná.

Problém 2

Zadání:
Spočítejte vývoj střední hodnoty počtu lidí v prostředku za předpokladu, že lidé cestují bez cíle, tj. na každé stanici čekající pasažér s pravděpodobností p nastoupí a jedoucí pasažér s pravděpodobností q vystoupí, a zakreslete křivku závislosti $S(m)$. Předpokládejme velmi dlouhou lineární linku.

Řešení:
Dle zákona velkých čísel v m-té stanici pD cestujících nastoupí a $qS(m-1)$ cestujících vystoupí. Definujeme-li tedy $S(0) = 0$ (stanicí 0 je depo), vyplývá z toho rekurzivní vztah:

$$\begin{aligned} S(m) &= pD + (1-q)S(m-1) \\ S(m-1) &= pD + (1-q)S(m-2) \\ &\vdots \\ S(1) &= pD + (1-q)S(0) = pD \end{aligned}$$

který lze rozepsat na přímý výpočet:

$$S(m) = pD \sum_{k=0}^{m-1} (1-q)^k = pD \frac{1-(1-q)^{m-1}}{1-(1-q)} = \frac{pD}{q}\left[1-(1-q)^{m-1}\right]$$

což je funkce, která očividně pro velmi vysoké m (mnoho projetých stanic) konverguje ke konstantě $\frac{p}{q}D$.

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

Téma 6 – Vrcholové pokrytí

V páté a poslední sérii pro vás máme výzvu.

Na webu se objevil desátý graf, který je o něco větší, než ty předchozí. Vaším úkolem je pro tento graf najít co nejmenší vrcholové pokrytí. Netrapte se, pokud se vám nebude dařit nalézt minimální vrcholové pokrytí, nějaké body dostanete i za neoptimální řešení.

Můžete zkusit využít pozorování Mgr.$^{\text{MM}}$ Lenky Kopfové, která nahlédla, že velikost minimálního pokrytí daného grafu musí být alespoň

- dolní celá část poloviny délky nejdelší cesty
- horní celá část poloviny délky nejdelší kružnice.

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

Výsledková listina 3. čísla

Poř.	Jméno	R.	$\sum_{-1}$	Úlohy t1	t2	t4	t5	t6	$\sum_0$	$\sum_1$
1.	Mgr.MM T. Sourada	4	37,1			2,3			2,3	37,1
2.	Mgr.MM O. Chlubna	2	36,8			3,5	9,0		12,5	36,8
3.	Mgr.MM V. Materna	3	27,0	10,0					10,0	27,0
4.	Dr.MM M. Kalousková	3	50,3			12,5			12,5	24,8
5.	Mgr.MM M. Souza de Joode	2	40,3						0	20,5
6.	Mgr.MM K. Hloušková	3	23,5				1,0		1,0	19,0
7.	Doc.MM K. Rosická	4	132,5		9,7				9,7	15,7
8.–10.	Bc.MM T. Flídr	1	13,0						0	13,0
	Bc.MM K. Kamenářová	4	15,7				5,0		5,0	13,0
	Bc.MM B. Kopčák	3	16,0		1,0	12,0			13,0	13,0
11.	Bc.MM P. Kolář	3	12,0		3,0			9,0	12,0	12,0
12.–13.	Mgr.MM M. Boček	Z9	21,4						0	11,0
	Bc.MM M. Vícha	Z9	11,0						0	11,0
14.	J. Štěpo	Z9	9,5						0	9,5
15.	Dr.MM L. Kundratová	4	69,7						0	9,0
16.	F. Bujnovský	1	8,8						0	8,8
17.	Mgr.MM O. Gonzor	2	29,9		3,7				3,7	8,2
18.	Mgr.MM J. Pallová	4	36,8						0	8,0
19.–21.	Bc.MM A. Foglarová	4	12,6						0	7,0
	Dr.MM J. Havelka	3	98,4						0	7,0
	Mgr.MM L. Kopfová	4	38,7						0	7,0
22.	J. Piroutek	3	6,7		6,7				6,7	6,7
23.	Dr.MM B. Hroncová	4	80,8						0	5,6
24.	V. Jůzková	2	5,5						0	5,5
25.	A. Hollmannová	2	9,0	5,0					5,0	5,0
26.	J. Kvapil	1	3,7						0	3,7
27.	L. Kunčarová	3	3,6						0	3,6
28.	J. Kováč	4	3,5						0	3,5
29.	Mgr.MM J. Růžička	4	35,9		3,2				3,2	3,2
30.	Dr.MM K. Balej	4	83,4						0	3,0
31.	Mgr.MM E. Vítková	3	28,3		1,3				1,3	2,8
32.	R. Zavřel	3	2,6		0,3				0,3	2,6
33.	E. Neumannová	2	2,1		2,1				2,1	2,1
34.–36.	N. Koscelanská	3	2,0						0	2,0
	L. Vomelová	3	2,0		2,0				2,0	2,0
	V. Žák	3	2,0		2,0				2,0	2,0

Poř.	Jméno	R.	$\sum_{-1}$	Úlohy t1	t2	t4	t5	t6	$\sum_{0}$	$\sum_{1}$
37.–38.	J. Přerovská	3	1,0						0	1,0
	M. Turinská	2	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