Studentský matematicko-fyzikální časopis

M&M

Ročník XXIII / Číslo 4

Časopis M&M a stejnojmenný korespondenční seminář je určen pro studenty středních škol, kteří se zajímají o matematiku, fyziku či informatiku. Během školního roku dostávají řešitelé zdarma čísla se zadáním úloh a témat k přemýšlení. Svá řešení odesílají k nám do redakce. My jejich příspěvky opravíme, obodujeme a pošleme zpět. Nejzajímavější řešení otiskujeme.

Milí čtenáři,

první číslo M&M v roce 2017 je tady. Můžete si v něm prostudovat vzorová řešení druhé série a přečíst první díl seriálu o programování v assembleru a principech fungování mikroprocesorů.

Určitě si nezapomeňte vyhradit čas pro nadcházející akce Matfyzu: 1. února se koná Jeden den s informatikou a matematikou (JDIM) a 16. února Jeden den s fyzikou (JDF). Přesné místo konání a program akcí naleznete na stránkách příslušných akcí[1]. Na JDIM můžete navíc navštívit stánek semináře M&M a pozdravit organizátory.

Začaly probíhat horlivé přípravy jarního soustředění, které proběhne 11.–19. března, tak doufáme, že budete dál pilně řešit, abychom se na něm mohli potkat.

Spoustu zábavy u zimních sportů a poznávání vědy přejí

Vaši organizátoři

Zadání úloh

Termín odeslání čtvrté série: 28. 2. 2017
(14. 2. 2017 pro účast na jarním soustředění)

Temně modrá obloha na obzoru získávala fialový nádech, který pomalu přecházel do červeno-růžové a oranžové. A jak se obzor rozjasňoval, objevilo se i slunce, pomalu a líně šplhalo nad vrcholky stromů a zalévalo celou krajinu jasným světlem. Seděl jsem na kameni a nemohl odtrhnout oči od té podívané. Ten pohled byl uklidňující jako nic na světě. Pomalu jsem upil kávu z hrnku a dál sledoval, jak slunce vychází nad obzor. Hrdinové v knihách často říkají něco jako: „V té chvíli jsem si připadal jako jediný člověk na světě.“ Ale já si tak nepřipadal. Naopak, já se v té chvíli cítil jako součást nějakého většího celku, protože… Protože ve všech ostatních chvílích jsem byl *poslední člověk na světě.*

No dobrá, možná jsem nebyl úplně poslední, možná ještě někde nějací jiní lidé přežili, ale já jsem o žádných nevěděl. Už téměř dva roky jsem žádného živého člověka nepotkal a, po pravdě řečeno, dokud jsem ještě nějaké lidi potkával, často jsem si přál, aby se to nestalo.

I když už slunce vystoupalo nad vrcholky stromů, začala mi být zima – čas se zvednout a pokračovat v cestě. Ještě jednou jsem si vychutnal pohled na zalesněné svahy kopců ozářené vycházejícím sluncem a pak jsem se jal uhasit oheň, na kterém jsem vařil kávu. V ešusu mi ještě trocha zbyla, tak jsem ji přelil do otlučené termosky, však ona ještě cestou přijde vhod.

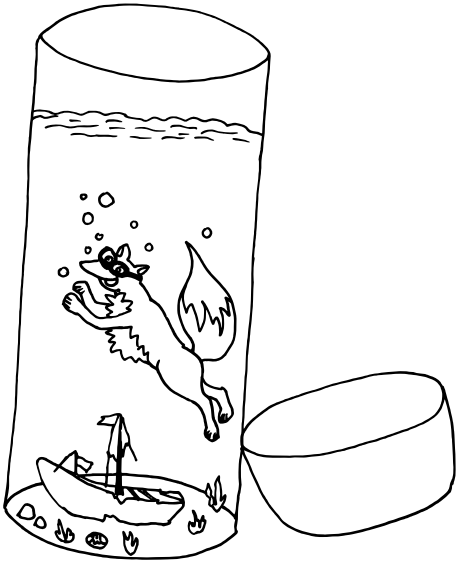
[1] JDIM – `http://www.mff.cuni.cz/verejnost/jdim/`
JDF – `http://www.mff.cuni.cz/verejnost/jdf/`

Úloha 4.1 – Termoska (3b)

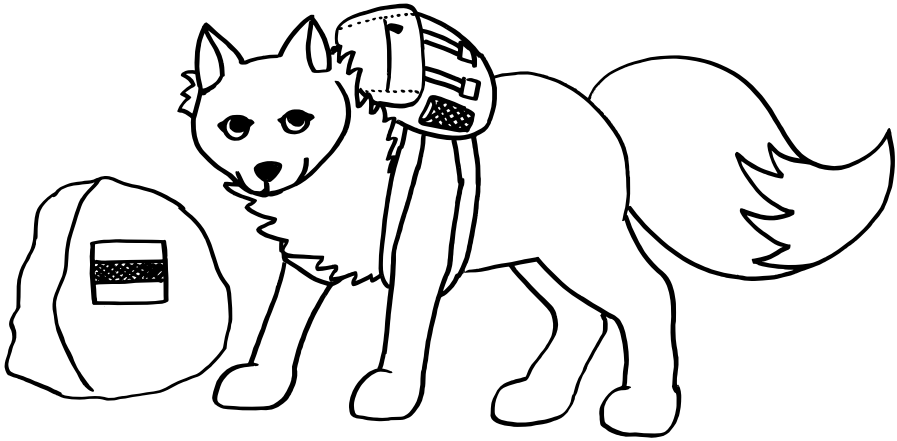
Mějme termosku tvořenou uzavřenou vnitřní a vnější plechovou stěnou, mezi kterými je vakuum. Jediný významný mechanismus přenosu tepla zevnitř ven je tepelným zářením přes evakuovanou mezeru.

Termoska je naplněna horkou kapalinou, která postupně chladne. Chladla by pomaleji, pokud bychom za jinak stejných podmínek vložili doprostřed vakuové mezery další uzavřenou „slupku“ z tenkého plechu ze stejného materiálu jako stěny nádoby, která nebude mít tepelný kontakt ani s vnější, ani s vnitřní stěnou? Pokud ano, jak moc pomaleji? Co když takových navzájem se nedotýkajících slupek přidáme více?

Můžete předpokládat, že teplota vnější stěny termosky je prakticky stejná jako teplota okolí, a nebude tedy záviset na kvalitě izolace. Přidávané vrstvy jsou natolik tenké a jejich vlastní tepelná kapacita tak nízká, že během zanedbatelně krátké doby dosáhnou rovnovážné teploty, a stačí tedy uvažovat jen tento ustálený stav.



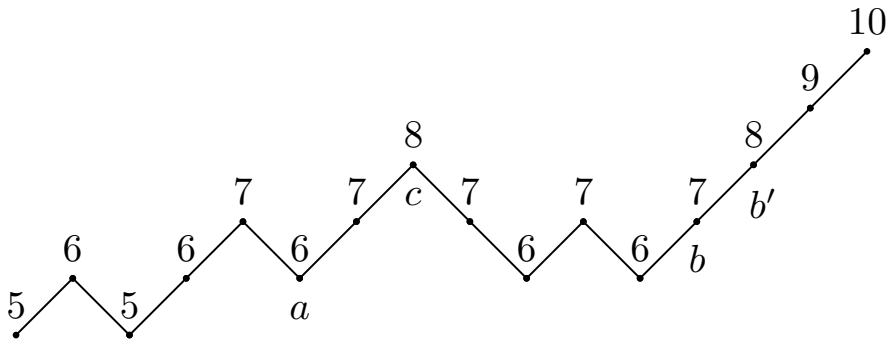
Hodil jsem si na záda krosnu a začal sestupovat po kamenných schůdcích zpět na cestu. Zpátky domů to ještě nějaký kus mám, obzvláště s tím, jak cesta pořád stoupá a klesá – když jsem tudy procházel zhruba před rokem poprvé, tak jsem ji překřtil na „Horskou dráhu“. Na mapě sice byly vyznačené nějaké zkratky, kterými bych si mohl pár výšlapů ušetřit, ale nechtěl jsem riskovat, že se ztratím – kdo ví, jak ty zkratky po takové době bez lidí vypadají. Zato z Horské dráhy jsem sejít nemohl, před Tím to byla oblíbená hřebenovka plná krásných výhledů, takže byla plná chodníčků, zábradlí a jiných stop lidské přítomnosti.



Úloha 4.2 – Hřebenovka (3b)

Na celé cestě je N bodů tak, že mezi každými dvěma sousedními body cesta buď o jeden metr klesne, nebo o jeden metr stoupne. Cestovatele vždy zajímá, jaký je nejvyšší bod na nějakém úseku cesty, neboť z něj bude nejhezčí výhled. Na dotaz „Jaký je nejvyšší bod na cestě mezi body a a b?“ tedy jako odpověď očekává bod c, který leží mezi body a a b (včetně) a zároveň má největší možnou nadmořskou výšku (viz Obrázek 1).

Vaším cílem je navrhnout co nejefektivnější algoritmus, který pro danou cestu (zadanou jako posloupnost nadmořských výšek bodů na ní tak, jak jdou za sebou) bude odpovídat na dotazy. Dotazy samozřejmě neznáte předem, ale můžete předpokládat, že počet dotazů K je řádově větší než N – počet bodů na cestě.



Obrázek 1: Příklad hřebenovky s 16 body, nad každým bodem je jeho nadmořská výška. Pro dotaz na kus cesty mezi body a a b by správnou odpovědí byl bod c s výškou 8. Pro dotaz na cestu mezi a a b' můžeme odpovědět buď opět bodem c, nebo bodem b', jelikož oba mají stejnou výšku.

Jak jsem šel známým úsekem cesty, začal jsem se nořit hlouběji a hlouběji do vzpomínek. Už jsem zmínil, že jsem možná poslední člověk na světě. Sám už vlastně moc nevím, jak se to stalo. Ale bylo to hrozně náhlé, jako bych šel večer spát a ráno se probudil do jiného světa. Lidé umírali po stovkách, po tisících. Nikdo pořádně nevěděl proč. Někdo tvrdil, že je to nějaký zmutovaný virus, jiný zas, že se někde zvrtnul výzkum nanorobotů, ale faktem bylo, že lidé z ničeho nic omdleli a pak prostě umřeli. Jen tak. Někteří šli spát a už se nevzbudili. Nikdo nevěděl, jak se to šíří, ale pomalu bylo jasné, že nemá smysl se ptát, jestli mě to dostane taky, ale kdy. Ti, kteří přežili prvních pár dní, začali rabovat a plenit, protože veškeré zákony a pravidla se zhroutily.

Proto jsem taky utekl sem, do hor. Ne, že bych tehdy věřil, že se mi to povede, před tou epidemií se nedalo utéct. Často jsem si před spaním říkal: „Tak zítra, zítra už se neprobudím," ale vždycky jsem se probudil. Nakonec jsem se dostal do hezké vesničky, kde jsem se usadil v jednom z prázdných domů, a smířil se s tím, že jsem přežil.

Zhruba kolem poledne jsem zastavil na jedné z vyhlídek, abych se naobědval. Odtud už byla vidět vesnice, ve které bydlím. Vzal jsem to nějak rychleji, takže se domů dostanu ještě za světla. Vytáhl jsem z krosny pár pruhů sušeného masa, jablko a, mmmm, zbytek kávy od snídaně! A kupodivu jsem tentokrát nebyl úplně sám, společnost mi dělaly jakési ploštice.

Úloha 4.3 – Lezoucí brouci (3b)

Mějme rovnostranný trojúhelník s délkou strany 1. Na každém z jeho vrcholů stojí brouk, na vrcholu A stojí brouk B_A, na vrcholu B brouk B_B a na vrcholu C brouk B_C. Brouk B_A leze tak, že neustále míří za broukem B_B, stejným způsobem leze brouk B_B za broukem B_C a brouk B_C za broukem B_A.

Všichni brouci lezou jednotkovou rychlostí. Jak dlouho bude trvat, než se všichni brouci potkají?

Po obědě jsem pokračoval v cestě, která už spíš jen klesala. Kolena si sice stěžovala, ale já je vesele ignoroval, neboť jsem se už těšil domů. Je zvláštní, jak rychle mi to místo přirostlo k srdci. Ale i tak jsem čas od času dostal chuť odtamtud zmizet, a proto jsem chodil na tyhle túry. Pár dní v lese, kde jsem si mohl alespoň trochu připadat, jako by se nikdy nic nestalo, mi vždycky zlepšilo náladu.

Konečně jsem měl před sebou roubenku, která označovala začátek vesnice, kde bydlím. Tedy, nejbližší další stavení bylo asi tři kilometry dál po cestě, ale podle mapy už tady vesnice začínala. Pravděpodobně tu sídlili místní myslivci. Když jsem to tu poprvé prohledával, tak jsem ve sklepě našel trezor s nějakými jejich dokumenty – a hned vedle něj krásnou sbírku likérů a pálenek, kterými nejspíš zapíjeli obzvláště vydařené úlovky.

Úloha 4.4 – Trezor (3b)

Jedenáctičlenné sdružení místních myslivců má v trezoru uložené tajné materiály. Chtějí trezor opatřit zámky tak, aby jej libovolná šestice dokázala otevřít a zároveň žádná pětice otevřít nedokázala. Kolika nejméně zámky je potřeba trezor opatřit a kolika klíči vybavit každého člena sdružení tak, aby se jim to podařilo? (Každý myslivec může mít více klíčů a zároveň od jednoho zámku může mít klíč více myslivců. Zároveň každý klíč otvírá právě jeden zámek.)

Když jsem se trochu přiblížil, začal ve mě hlodat pocit, že tu něco nesedí. Něco… Něco prostě nebylo tak, jak by mělo. Zpomalil jsem a začal jsem pořádně prohlížet okolí. Křoví! Křoví a tráva, které vyrostly na cestě ke vchodu, vypadaly, jako by skrz ně někdo prošel. A jedno z oken vypadalo mnohem méně špinavě než zbylá, jako by ho někdo vyčistil, aby mohl vidět ven. Někdo tu musel být! Zamrazilo mě a sáhnul jsem pro pistoli. Možná to zní zvláštně, že s sebou nosím pistoli a hned ji vytáhnu, když objevím stopy dalšího člověka, ale věřte mi, kdybyste zažili to, co já…

Opatrně jsem se vydal ke dveřím. Vůbec mě v tu chvíli nenapadlo, že by možná bylo lepší prostě utéct. Jak jsem se snažil neslyšně prodrat křovím, došlo mi, že to nebyl ten nejlepší nápad, ale už nebylo cesty zpět. Když jsem byl u dveří, opatrně jsem zmáčknul kliku – šla podezřele lehce – a vsunul hlaveň do vzniklé mezery. Pomalu jsem mezeru zvětšoval, ale o panty se nikdo nestaral, a tak hlasitě zavrzaly. V tu chvíli zpoza převráceného stolu vyskočila postava a pokusila se zaběhnou za roh. „Ani se nehni!“ zakřičel jsem a cizinec strnul. Opatrně se na mě podíval. Vypadal dost vystrašeně, ale to já asi taky. Chvíli jsme na sebe mlčky zírali, až on pronesl: „Prosím, nech mě žít. Já… Já si chtěl jen najít hezký dům, kde bych mohl v klidu dožít. Tady to vypadalo… Nevěděl jsem, že ještě někdo…“ Sakra, přece nemůžu jen tak zabít jediného dalšího člověka. Ale co když… Co když on zabije mě? Ne… Pomalu jsem sklonil pistoli. „Já jsem Tobiáš. Bydlím kus dál po cestě ve vesnici.“ „Já jsem Marek, těší mě.“ Pomalu jsme k sobě přišli a podali si ruce.

Řešení úloh 2. série

Úloha 2.1 – Prásk (4b)

Zadání:
Spočítejte, za jak dlouho se zavřou obdélníkové dveře výšky h o hmotnosti m, pokud jsou otevřeny o úhel α, tlak na jedné straně dveří je p_1 a na druhé p_2. Neuvažujte velké proudění vzduchu.

Řešení:
Na dveře z obou stran působí tlaková síla. Nechť tlak v místnosti, do které jsou dveře otevřeny, je p_1 a tlak v druhé místnosti je p_2. Potom se dveře zavřou, pokud $p_1 > p_2$. Označme si rozdíl tlaků $p_1 - p_2 = \Delta p$ a šířku dveří s.

Z mechaniky hmotného bodu známe vztahy pro dráhu rovnoměrně zrychleného tělesa

$$x = \frac{1}{2}at^2$$

a sílu

$$F = ma.$$

Při rotaci platí obdobné zákony, jen musíme počítat s jinými veličinami. Dráhu nahradí úhel α, zrychlení bude úhlové zrychlení ϵ, místo síly použijeme moment síly M a hmotností bude v tomto případě moment setrvačnosti J.

V každém bodě dveří působí na plošku δS síla

$$\delta F = \Delta p \cdot \delta S.$$

Síla F působí kolmo na dveře. Moment síly, který působí na plošce δS, je

$$\delta M = \delta F \cdot r,$$

kde r je vzdálenost od zárubně. Tento moment síly není na celé ploše dveří konstantní, roste lineárně s r. Abychom získali celkový moment síly M, musíme si uvědomit, že průměrný moment síly je $\delta F \cdot s/2$ a působí uprostřed dveří. O kolik je tento moment větší ve větší vzdálenosti r, o tolik je menší v menší vzdálenosti r. Celkový moment síly tedy je

$$M = F \cdot \frac{s}{2} = \Delta phs \cdot \frac{s}{2} = \frac{1}{2}\Delta phs^2,$$

kde $F = \Delta p \cdot S = \Delta p \cdot hs$ je celková síla působící na dveře. Stejný výsledek získáme integrací momentu síly podle r od nuly do s.

Moment setrvačnosti dveří je stejný jako moment setrvačnosti tyče délky s a hmotnosti m otáčející se kolem osy procházející jejím koncem. Můžeme najít, že tento moment setrvačnosti je

$$J = \frac{1}{3}ms^2.$$

Nyní již stačí dosadit do rovnice pro moment síly

$$M = J\epsilon$$
$$\frac{1}{2}\Delta phs^2 = \frac{1}{3}ms^2\epsilon$$

a vyjádřit úhlové zrychlení

$$\epsilon = \frac{3}{2}\frac{\Delta ph}{m}.$$

Čas, za který se dveře zavřou, spočítáme ze vztahu

$$t = \sqrt{\frac{2\alpha}{\epsilon}} = \sqrt{\frac{4\alpha m}{3\Delta ph}}.$$

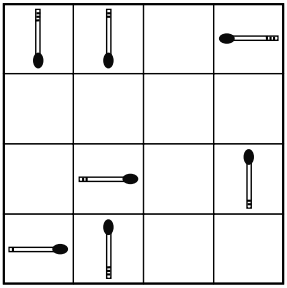
Viktor

Úloha 2.2 – Cesta po sirkách (3b)

Zadání:

Mějme čtvercovou síť o velikosti 4×4. Vezmeme si 16 sirek a uděláme na nich čárky následovně: na jednu sirku nakreslíme jednu čárku, na osm sirek dvě čárky, na sedm sirek tři čárky.

Položíme-li na políčko sirku s jednou čárkou, ukazuje ve směru hlavičky na sousední pole, sirka se dvěma čárkami o jedno pole dále a sirka se třemi čárkami na třetí políčko v daném směru. Podíváme se na pole, na které ukazuje sirka, a pokud na něm leží nějaká další, pokračujeme podle ní. Putování může skončit různě, nás však bude zajímat takové rozložení sirek, že se po navštívení poslední sirky vrátíme opět na začátek (viz Obr. 2).



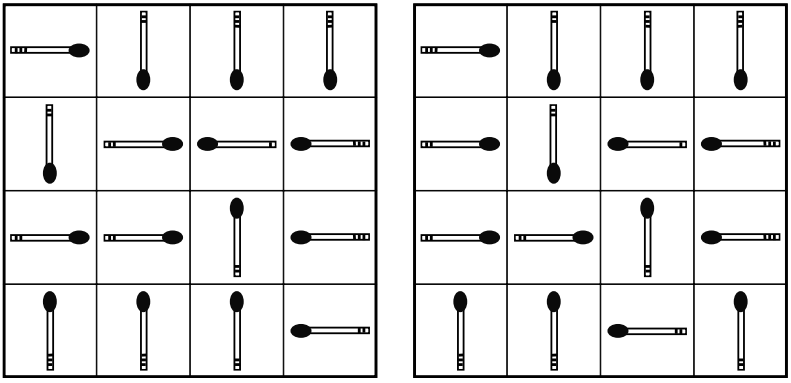
Obrázek 2: Ukázka cyklu na síti s použitím sedmi sirek (jedna s jednou čárkou, tři se dvěma čárkami a tři se třemi čárkami).

Dokážete umístit všech 16 sirek na síť tak, aby tvořily uzavřenou cestu (viz Obr. 2)? Existuje jiné označení sirek, které umožní vytvořit delší uzavřenou cestu (za délku cesty se považuje počet políček)?

Řešení:
Při řešení se nejdříve zamyslíme nad tím, jak můžeme umístit sirky se třemi čárkami. Proč začít zrovna jimi? Tento druh sirek je možné umístit pouze na okrajová pole čtvercové sítě. Kdybychom totiž uložili sirku se třemi čárkami na některé z vnitřních polí, ukazovala by mimo síť. Navíc pouze v rozích můžeme tříčárkované sirky položit dvěma způsoby, na ostatních okrajových polích musí sirka směřovat dovnitř, jinak by opět ukazovala mimo síť.

Celkem tedy máme 12 polí, kam můžeme umístit tříčárkové sirky na 7 tříčárkových sirek. Nemůžeme však použít všech osm polí zároveň. Kdybychom položili dvě sirky naproti sobě, vytvoříme cyklus. Z osmi polí, která jsou na okraji sítě, ale nejsou v rozích, můžeme použít pro tříčárkové sirky pouze 4, které nejsou protilehlé. Jak to vypadá s rohovými poli? Sirky musíme pokládat jedním směrem, protože kdybychom položili jednu sirku opačným směrem, opět vznikne cyklus. Při obsazení všech rohů vznikne cyklus o čtyřech sirkách, takže do rohů můžeme položit nejvýše 3 tříčárkové sirky. Celkem tedy máme poměrně přesně stanoveno, kde se musí nacházet tříčárkové sirky.

Nyní určíme polohu zbývajících sirek. Začneme uložením dvoučárkové sirky do posledního zbývajícího rohu. Můžeme ji položit dvěma způsoby. Dále pokládáme sirky tak, abychom umístili zbylé 4 tříčárkové sirky na okraj. Výsledná dvě možná řešení viz Obrázek 3. Délka cyklu je stejná jako součet čárek na všech sirkách, tedy 38.



Obrázek 3: Dvě možnosti, jak naskládat všech 16 sirek tak, aby tvořily uzavřený cyklus.

Jak je to s možností přeznačit sirky tak, aby vznikla delší uzavřená cesta? Problém si přeformulujeme: Hledáme přeznačení sirek tak, aby součet všech čárek byl větší než 38 a zároveň jsme z nich dokázali sestavit uzavřený cyklus.

Čtyř- a více čárkové sirky zřejmě použít nemůžeme, jelikož bychom se dostali mimo síť. Už při hledání uzavřeného cyklu ze všech sirek jsme si ukázali, že polí, kam je možné umístit tříčárkové sirky, je právě 7. Přeznačit dvou- nebo jednočárkovou sirku za tříčárkovou tedy nemůžeme. Zároveň máme jen jednu jednočárko-

vou sirku, takže nemáme možnost ponížit trojčárkovou sirku na dvojčárkovou a poté přeznačit ostatní sirky, aby se aktuální počet čárek zvýšil alespoň o dvě.

Zbývá tedy pouze možnost přeznačit jednočárkovou sirku na dvoučárkovou. Pro toto označení sirek však také nedokážeme nalézt uzavřený cyklus. Proč? Pokud chceme prodloužit původní cyklus, můžeme to udělat pouze tak, že si „zajdeme“ nějakým směrem. Když si ale „zajdeme“ dále jedním směrem, musíme se pak i vrátit. Délka cesty tak musí zůstat sudá, jak nahlédlo mnoho z vás. Zajímavý argument použil Mgr.MM Pavel Turinský. Obarvíme si čtvercovou síť jako šachovnici a všimneme si, že počet sirek, které ukazují na bílá pole, je stejný, jako počet sirek, která ukazují na černá pole. Tím, že přeznačíme jednočárkovou sirku na dvoučárkovou, změníme barvu pole, na které ukazuje. Jelikož žádnou jinou sirku nepřeznačujeme, způsobili jsme nerovnováhu mezi bílou a černou.

Z toho plyne, že nalezený cyklus je nejdelší možný. Existuje ještě jedno řešení o stejné délce, které se skládá z šesti tříčárkových sirek a devíti dvoučárkových sirek. Dokážeš ho najít?

Anet

Úloha 2.3 – Zapeklitý trojúhelník (3b)

Zadání:

Mějme rovnoramenný trojúhelník ABC. Nechť M je střed jeho základny AB a N je bod osově souměrný s M podle přímky BC. Rovnoběžka s AB procházející bodem N protíná přímku AC v bodě K. Určete velikost úhlu AKB.

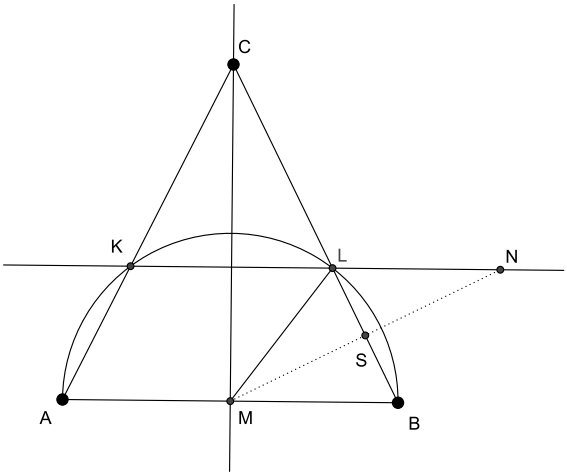
Řešení:

Označme L průsečík přímek BC a NK (viz Obrázek 4). Protože je trojúhelník ABC rovnoramenný se základnou AB a LK je rovnoběžná s AB, tak bod L je osově souměrný s bodem K dle osy MC, a tím pádem $|\sphericalangle AKB| = |\sphericalangle ALB|$. Stačí tedy určit velikost úhlu $\sphericalangle ALB$.

Označme S průsečík MN a BC. Potom z rovnoběžnosti NL a BM a osové souměrnosti bodů M a N dle BL platí, že trojúhelníky BMS a LNS jsou shodné dle věty *usu*. Z toho vyplývá, že $|BS| = |SL|$. Tedy úhlopříčky BL a MN se navzájem půlí a jsou na sebe kolmé, takže $MBNL$ je kosočtverec. Tedy $|AM| = |MB| = |ML|$. Bod L proto leží na Thaletově kružnici nad AB, proto je velikost $\sphericalangle ALB$ (a tím i AKB) roven $90°$.

Je dobré posílat k řešení geometrických úloh i obrázek. Mnoho z vás to tentokrát neudělalo a občas na to i doplatilo. Obrázek pomáhá opravovateli pochopit vaše řešení a občas mu i napoví, že jste se jen přepsali. A zvláště důležitý je ve chvíli, kdy si pojmenujete nějaký bod neobvykle, či v rozporu se všeobecnými zvyklostmi, případně dokonce v rozporu se zadáním. Dobrým nástrojem je třeba program GeoGebra[2] (dostupný i v online verzi bez instalace), který umož-

[2] `www.geogebra.org`



Obrázek 4: Trojúhelník z úlohy 2.3

ňuje rýsování na PC a mnoho dalšího. Z něj se dá poté jednoduše vyexportovat PNG soubor, který můžete vložit do řešení.

Petr

Úloha 2.4 – Vyrovnání dluhů (3b)

Zadání:
Pět kamarádů spolu vyrazilo na akční hru Hranostaj. Verča zaplatila startovné, Tadeáš koupil lístek pro čtyři z nich do Ferdinandova, Olda si koupil svoji jízdenku a zaplatil také zákusky v cukrárně, Luboš se postaral o účet v restauraci. No a na Evelínu zbylo vymyslet, jak se co nejjednodušeji vyrovnat.

Obecněji máme n kamarádů a m dluhů, což jsou trojice (kdo, komu, kolik dluží). Nalezněte co nejlepší horní odhad na počet transakcí, který bude pro daná n a m na vyrovnání vždy stačit. Pak vymyslete co nejrychlejší algoritmus, který takové vyrovnání nalezne. Pokud dokážete, že úloha nalezení vyrovnání s minimálním *počtem transakcí je NP-úplná*[3]*, bonusové body vás neminou!*

Řešení:
Jako první si ujasníme, že se úloha skládá ze tří různých otázek. První otázka se týká horního odhadu na počet transakcí, jenom ze znalosti n a m, bez znalosti konkrétní instance problému. Druhá otázka spočívá v nalezení algoritmu, který nalezne vyrovnání (posloupnost transakcí) pro konkrétní instanci problému. Tento algoritmus by měl najít vyrovnání, které není horší než námi nalezený horní odhad na počet transakcí. Dále by tento algoritmus měl být co nejrychlejší. Pokud jde o rychlost algoritmu, tak nás již nezajímá počet transakcí, ale počet kroků

[3]O NP-úplných problémech byla přednáška na soustředění. Jinak si můžete přečíst např. kuchařku KSP: `http://ksp.mff.cuni.cz/kucharky/tezke-problemy/`

výpočtu, které k nalezení řešení vedou. Zatímco při stanovování horního odhadu jsme museli být velmi precizní, tak při určování časové složitosti algoritmu je vhodné použít asymptotickou notaci, která nám umožní zanedbat inkrementální a multiplikativní konstanty. Třetí otázka se týká NP-úplnosti hledání vyrovnání dluhů na co nejnižší počet transakcí. Jedná se o problém, kde je opravdu potřeba najít vyrovnání na co nejnižší počet transakcí pro konkrétní instanci problému – nestačí tu tedy již dodržení limitu pro obecný problém jen ze znalosti n a m.

Tvrdíme, že nejlepší horní odhad na vyrovnání m dluhů mezi n lidmi je menší z čísel $n-1$ a m. Pokud je dluhů opravdu málo, tak se druhy můžou vyrovnat mezi stejnými dvojicemi lidí, jako vznikly (transakcí bude m). Jinak předpokládejme, že dluhů je více než lidí. Pak je výhodné pro každého člověka spočítat, kolik peněz celkem dluží, kolik peněz je mu celkem dluženo a jaká je výsledná bilance. Následně můžeme sestavit řetěz předávání peněz, kde všichni lidé, kteří v souhrnu dluží peníze, budou před lidmi s kladnou pohledávkou. První člověk pošle druhému peníze, které dluží zbytku skupiny v součtu. Druhý člověk k tomu přidá svoji výslednou dluženou částku a součet pošle dál. Když se peníze dostanou k věřitelům, začnou si z těchto peněz postupně brát tolik, kolik mají v součtu dostat, a zbytek peněz, které jim dorazily, posílat zbytku věřitelů. Až nakonec k poslednímu věřiteli dorazí přesně tolik peněz, kolik měl v součtu dostat. Počet transakcí v tomto řetězu je $n-1$.

Tento horní odhad je nejlepší možný. Pokud jednomu člověku všichni kamarádi dluží 1 Kč, pak aby proběhlo vyrovnání, musí mít každý dlužník odchozí transakci. Těchto transakcí bude $n-1=m$.

Algoritmus pro nalezení vyrovnání plyne z našeho zdůvodnění horního odhadu. Nejprve pro každého člověka sečteme všechny jeho dluhy a pohledávky, což nám zabere $O(n+m)$ kroků. Pak roztřídíme lidi podle znaménka součtu do dvou seznamů, což nám zabere $O(n)$ kroků. Nyní začneme vypisovat lidi ze seznamu dlužníků a jejich dluhy postupně přičítáme k částce, kterou si musí mezi sebou předávat. Nakonec budeme vypisovat lidi z druhého seznamu, přičemž od částky, kterou si musí předávat, budeme odčítat jejich pohledávky. To se opět vejde do $O(n)$ kroků. Časová složitost našeho algoritmu tudíž je $O(n+m)$.

Nyní k bonusové otázce. Na podrobné představení NP-úplnosti zde nemáme prostor, lepší vhled do problematiky můžete získat např. v již zmiňované kuchařce KSP. Připomeňme jen, že NP-úplné problémy jsou právě ty, které jsou ve třídě NP (to znamená, že se dají vyřešit na **N**edeterministickém Turingově stroji v **P**olynomiálním čase) a zároveň jsou NP-těžké (to znamená, že se na ně každý problém v NP dá *převést* – vyřešit pomocí nich s tím, že se jeho vstup při převodu může zvětšit jen polynomiálně).

Hodí se umět o problému dokázat, že je NP-úplný, protože když se nám to podaří, můžeme se přestat snažit nalézt efektivní (tj. polynomiální) algoritmus, který jej řeší. Ne že by někdo s jistotou věděl, že to nejde. Slavná otázka jestli P = NP (P je třída problémů řešitelných v polynomiálním čase na obyčejném Turingově stroji) stále není vyřešená. Ale snažilo se o to už hodně lidí a všeobecně se má za to, že P $\neq$ NP.

Jak dokážeme o problému nalezení minimálního počtu transakcí, že je NP-úplný? Uvažme jinou verzi problému: pro daných n lidí a m dluhů nalézt vyrovnání na právě t transakcí, pokud existuje. Označme si tento problém *Tr*. Původní problém pomocí *Tr* dokážeme vyřešit jednoduše tak, že budeme zkoušet počet transakcí $t = 1, 2, \ldots$ a skončíme, jakmile nalezneme první vyrovnání. A naopak – jestli lze najít vyrovnání na právě t transakcí zjistíme z jednoho zavolání minimalizace. Pokud je minimální počet transakcí větší než t, vyrovnání na t transakcí nalézt nelze. Naopak pokud je minimum menší než t, vždy můžeme přidat další transakce předávající 0 peněz.

Jelikož umíme problém převést tam i zpět, je minimalizace počtu transakcí NP-úplná právě tehdy, když je NP-úplný *Tr*.

Nyní dokážeme, že *Tr* je NP-úplný. Zaprvé je v NP – pokud nám někdo ukáže, jak by prováděl t transakcí (dá nám tzv. „certifikát“), dokážeme snadno v polynomiálním čase ověřit, jestli tyto transakce zajistí vyrovnání (pro každého člověka spočítáme, jestli změna stavu jeho účtu odpovídá vyžadované). Zadruhé potřebujeme dokázat, že je NP-těžký. To uděláme tak, že na něj převedeme známý NP-úplný problém zvaný *problém dvou loupežníků*: Dva loupežníci si „vypůjčili“ k předmětů, které mají obecně různé hodnoty. Existuje rozdělení předmětů na dvě hromádky tak, aby součty hodnot předmětů v nich byly stejné?

Převod dvou loupežníků na *Tr*: Za každý předmět a každou hromádku si vytvoříme jednoho člověka. Dlužit budou lidé odpovídající předmětům polovinu své hodnoty každé ze dvou hromádek. Takto každý předmět celkem zaplatí svou hodnotu a každá hromádka celkem dostane polovinu hodnoty všech předmětů. Abychom našli rozdělení předmětů mezi hromádky, spustíme *Tr* s $t = k$. Máme k dispozici jen jednu transakci na předmět a všechny peníze z předmětů musí dotéct do hromádek (ale nemusí přitéct přímo, mohou to vzít přes jiné předměty a utvořit tak jakési stromy transakcí). Pokud tedy *Tr* nalezne vyrovnání, získáme správné rozdělení předmětů na hromádky a vyřešíme problém dvou loupežníků. Pokud *Tr* vyrovnání nenalezne, neexistuje ani správné rozdělení předmětů na hromádky.

Převedli jsme problém dvou loupežníků na *Tr*, tím jsme dokončili důkaz jeho NP-úplnosti, a díky tomu víme, že i minimalizace počtu transakcí je NP-úplná.

Martin a Matěj

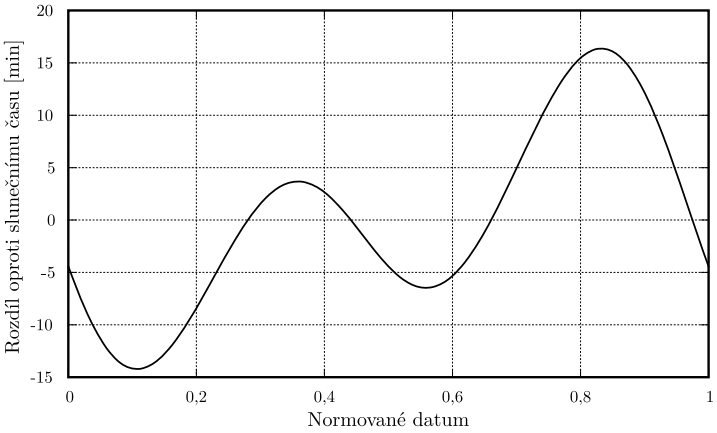
Řešení témat

Téma 1 – Sluneční hodiny

S řešeními ke druhému číslu nové příspěvky nepřibyly. Tento článek shrnuje základní poznatky o fungování slunečních hodin, ze kterých můžete při řešení vyjít.

Princip slunečních hodin

Sluneční hodiny na pevném stanovišti fungují všechny podobně – na principu hodinového úhlu. Slunce zdánlivě oběhne Zemi za 24 h, tj. jeden sluneční den. Zemi, aproximovanou sférou, si pak můžeme pomyslně rozdělit poledníky na 24 stejných dílů tak, že 15° připadá na jednu hodinu. Pokud zvolíme ukazatel slunečních hodin rovnoběžný s osou rotace Země, otočí se kolem něj Slunce zdánlivě každou hodinu o 15°. Tento princip je dobře patrný na hodinách, kde je rovina číselníku rovnoběžná s rovinou rovníku (tzv. rovníkové sluneční hodiny). To znamená, že stínový ukazatel bude k rovině číselníku kolmý. Rysky značící čas budou rozmístěné na číselníku po patnácti stupních.



Obrázek 5: Průběh časové rovnice v průběhu roku. Nula na x-ové ose je 1. leden.

Země ale obíhá kolem Slunce po elipse (1. Keplerův zákon) a rychlost oběhu je různá v závislosti na vzdálenosti od Slunce (2. Keplerův zákon). To má za následek, že zdánlivý pohyb Slunce po obloze je nerovnoměrný, tj. úsek pro 1 h na

slunečních hodinách mění v průběhu roku svou délku. Hodnota rozdílu mezi časem zobrazeným na slunečních hodinách a časem rovnoměrně plynoucím (střední sluneční čas – tedy čas na našich hodinkách) se nazývá časová rovnice. Její průběh během roku je vykreslený na Obrázku 5.

Dominigga

Téma 2 – Časosběr

V adventním období k nám dorazila dvě videa od jednoho autora. První záznam zachycuje pohled z okénka dopravního letadla během letu. Na tomto videu bych doporučil pomaleji přehrávat vzlet a přistání, které považuji za informačně nejhustější. Samotný let ve výšce bych klidně pustil ještě rychleji. Na videích tohoto typu si můžeme všímat třeba vývoje oblačnosti, výšky a techniky letu nebo změn posunující se krajiny. Druhé video ukazuje pohyb lidí na ulici pohledem z okna. Tento snímek mohl být trochu propracovanější, přece jen bych doporučil alespoň položit telefon nebo fotoaparát na pevnou podložku. Pokud nedrží samostatně, tak jej můžeme opřít třeba o knížku nebo o hrnek. Zde by mohlo být zajímavé pozorovat třeba rozdíl mezi zalidněností ulice ve špičce a mimo špičku.

Ve vašich příspěvcích zohledňuji i informační hodnotu vašeho videa. V rámci tohoto tématka je video demonstrující nějakou vlastnost nebo událost mnohem hodnotnější než umělecký snímek. V dalších příspěvcích uvítám hlubší zamyšlení nad obsahem, který vaše časosběrné video prezentuje. Všem řešitelům přeji vše nejlepší do nového roku a těším se na další zrychlené záznamy dějů kolem nás.

Béda

Téma 3 – Filmoví poradci

K tématku zatial prišiel jediný príspevok od Doc.^{MM}Petra Šimůnka, ktorý si vybral film *Marťan* (2015). Tento film sa dá veľmi ľahko rozobrať, stačilo by si vypočítať alebo zistiť pár údajov o Marse. Tieto údaje by potom rýchlo dokázali chybu vo filme. Išlo by hlavne o: tlak a hustotu atmosféry, teploty pri povrchu a tiažovú silu. Je veľmi dobré citovať zdroje, ktoré potvrdia vaše úvahy alebo ich podložiť výpočtami.

Hneď na začiatku filmu Petr spochybňuje, že by vietor utrhol anténu. Bohužiaľ svoje tvrdenie o tom, že vietor by nemal anténu utrhnúť nepodkladá žiadnym ani aproximačným výpočtom ani zdrojom. Hneď potom si ale nevšimol ich rakety. Ak by platila fyzika, ktorú načrtol film, raketa by sa pravdepodobne zrútila vo vetre. Na Zemi sa rušia lety do vesmíru ak je nepriaznivé počasie, a tu kvôli tomu štartujú! Ak však fyzika vo filme nepracuje v skútočnosti tak, ako ju vo filme načtli, neutrhlo by počasie anténu a ani nezranilo Marka.

Po tom, čo Mark našiel sondu, Petr teoretizuje nad kompatibilitou elektroniky sondy a nového vybavenia. Či bude sonda používať rovnaký napájací kábel ako lab. Tu by chcelo naozaj vyhľadať informácie na internete či NASA používa niečo ako spätnú kompatibilitu v nových sondách, bez toho nejde určiť či je to chyba filmu alebo nie.

Po výbuchu v labe v ňom nastane dekompresia. Mark má popraskané sklo na skafandri a zalepí ho lepiacou páskou z vonku. Petr dobre upozornil na to, že páska by sa mohla odlepiť kvôli tlaku v skafandri.

Potom Mark opravuje lab páskou a fóliou. Petr taktiež ako pri páske poukázal na to, že by fólia nevydržala, avšak neuviedol, akým silám čelí fólia a páska. Neskôr vo filme je ukázané, ako sa fólia vlní vo vetre, napriek tomu, že na jednej strane je teraz tlak v labe. Má na Marse vietor dostatočnú silu, aby spolu s tlakom v atmosfére prekonali tlak v labe?

Pri dekompresii sa poškodila jeho farma na zemiaky. Petr uvažuje či by to zemiaky mohli prežiť. Bielkoviny a sacharidy sú v prírode v tuhej forme. Problém by predstavovala voda. Mohla by nastať obdoba Kesonovéj choroby, v zemiakoch by sa vytvorili bublinky vzduchu. Človek vie dočasne tieto podmienky prežiť. Rastlinné pletivá nie sú tak náchylné na poškodenie ako živočíšne tkanivá. Môže zemiak zostať iba zamrznutý?

Pri príprave na odlet Mark odľahčuje raketu a to aj o aerodynamické prvky. Cez dieru v nose rakety pretiahne plachtu. Herci tvrdia, že im aerodynamiku netreba kvôli riedkej atmosfére. Petr na to upozorňuje, ale neoveruje ich tvrdenia.

Kubo

Níže nabízíme pár podnětů k tomu, na co se můžete dále zaměřit:

- Jak vypadá ve filmu dopad vystřelené kulky? Jaká může být teplota kulky? Je reálné jiskření při dopadu na kovové předměty? A co obláček, který se objeví u hlavně těsně po výstřelu?

- Zkuste se zamyslet nad výbuchy. Nechybí k nim někdy vůbec jakýkoliv důvod? Za jak dlouhou dobu po autonehodě vůbec může nastat výbuch?

- Ve filmech se často stává, že vznikne díra v lodi nebo jiném plavidle, konkrétně například ve filmu *Asterix a Obelix: Mise Kleopatra* (2002). Jak vysoko může v takovém případě vytrysknout voda (pokud vůbec)?

- Podívejte se pozorně na tabuli při hodině matematiky ve filmu *Billy Madison* (1995).

Při hledání a popisování chyb se pokuste své domněnky řádně podložit výpočtem, dohledáním potřebných informací nebo alespoň podloženým odhadem. Pokuste se také navrhnout úpravu příslušné scény, aby byla věcně správná. Věnovat se můžete i opravování chemických a biologických omylů a nepřesností.

Těšíme se na vaše filmová pozorování.

Anet

Seriál: Malý úvod do assembleru

Assembler – jazyk procesoru. Pojďte se se mnou na chvíli podívat do hlubin počítače.

Už po přečtení nadpisu slyším: „Proč bychom se měli zabývat assemblerem?“ Mezi běžné argumenty lidí, kteří se s ním nesetkali, nebo setkali jen zběžně, patří: Assembler je zastaralý. Už nepotřebujeme assembler, máme vysokoúrovňové programovací jazyky. A na nízkoúrovňové programování stačí C nebo jazyk podobného typu. Psát v assembleru je časově neefektivní, náchylné k chybám, těžko čitelné a zbytečně složité. Zkusím tyto námitky jednu po druhé rozebrat.

- *Assembler je zastaralý.* Toto tvrzení je nepřesné. Prvně, assemblerů je mnoho – dalo by se říci, že každá skupina procesorů s podobnou architekturou má svůj. Takže – některé assemblery nepochybně zastaralé jsou. Ale naproti tomu existují takové, které se používají stále. Příkladem budiž assembler pro mikrokontroléry AVR.
- *Máme vysokoúrovňové programovací jazyky, případně C jakožto nízkoúrovňový jazyk.* To je nepochybně pravda, jenže assembler se využívá na jiné typy úloh. Oproti běžným programovacím jazykům má minimálně dvě výhody:

  1) I v dnešní době, kdy mnozí z vás nosí v kapse počítač s výkonem o několik řádům větším, než měly počítače ještě před desítkou let, můžeme narazit na potřebu optimalizovat spotřebu paměti nebo rychlost provádění kritické části kódu za hranice toho, co dokáže překladač vyššího programovacího jazyku. Třeba pro mikrokontroléry s malou pamětí, které mají své místo v ovladači kávovaru, čtečce platebních karet, krokoměru a podobně.
  2) Některé operace specifické pro konkrétní hardware, může být problém vyjádřit v univerzálních programovacích jazycích včetně C, které se snaží od hardwarové vrstvy abstrahovat. V takových případech se může kontrola nad každou jednotlivou instrukcí programu nebo jeho části hodit.

- *Neefektivní pro psaní kódu.* Nepochybně. Po assembleru většinou sáhneme až ve chvíli, kdy nemáme jinou možnost. Nebo když se někdo nudí. (Potkali jste už někdo nějaký jazyk ze soutěže zvané codegolf? Ještě stále vám assembler připadá složitý?)
- *Náchylný k chybám.* Ano, assembler nenabízí komplexnější řídící struktury, často neumožňuje mít vyhrazené oddělené proměnné, jak bychom podle jejich významu v kódu chtěli, a vůbec má svou sadu drobných zákeřností. Asi tak jako každý jiný jazyk.
- *Těžko čitelný.* Ano.
- *Zbytečně složitý.* Jak se to vezme. Assembler je ve své podstatě velmi jednoduchý. O to složitější je z něj něco smysluplného vytvořit.

Myslím, že i v dnešní době má smysl se na assembler podívat. A to ne jen jako na historickou kuriozitu. Je pravda, že kdo nechce, nejspíše se takto blízko železu nemusí ani přiblížit. Ale na druhou stranu může být i pro vysokoúrovňového programátora užitečné o strojovém kódu něco málo vědět, kdyby už jenom proto, aby se přesvědčil, že v hlubinách počítače nesedí velmi chytrý skřítek (zploštělý, aby se vám vlezl do kapsy).

Ukázali jsme si, proč má smysl se assemblerem zabývat, ale zaznívají jiné hlasy, konkrétně: „Ale já neumím programovat.“ Nevadí. V assembleru totiž běžné programovací techniky mají jen omezené využití. Sice budete v nevýhodě, ale aspoň se můžete cítit jako průkopníci, kteří nic než assembler neměli. Oni na začátku neměli ani ten assembler.

Assembler je poměrně široké téma. Kde tedy začít?

Kdo se podívá na internet, narazí na mnoho manuálů. Proč si tedy lámat hlavu zrovna s tím mým? Většina jich totiž řeší, jak psát v assembleru. Já zkusím trochu jiný úhel: jak pochopit assembler. Pro tyto účely budu často odbíhat, trochu zjednodušovat a co nejvíce vysvětlovat. Možná trochu polopaticky, ale budu počítat s tím, že někteří z vás neumí programovat, a z vlastní zkušenosti vím, že začátky s assemblerem nebývají jednoduché.

Inu začněme. Pojďme se společně zamyslet nad zcela fiktivním assemblerem, který nazvu *PAPIRAS* – papírový assembler. Vezměte si prosím papír a tužku, budeme si kreslit a počítat. *PAPIRAS* se bude podobat AVR-assembleru, který si však upravím pro své potřeby. Rád bych teď rozptýlil vaše obavy ze skřítka v počítači, ale to by se do tohoto miniseriálu nevešlo. Budu se tedy k procesoru chovat jako k černé skříňce – nebudu řešit, jak je zadrátovaný uvnitř. Nebudu řešit logické obvody. Mějme tedy abstraktní procesor.

Co si pod tím pojmem představit? Je to stroj s pamětí, který vykonává instrukce, které mu dodáme. K instrukcím se dostaneme vzápětí, první je potřeba se seznámit s vnitřní pamětí procesoru. Ta reprezentuje okamžitý stav procesoru, a spolu se zrovna prováděnou instrukcí plně určuje, co konkrétně se bude v následujícím okamžiku dít. Buňky této paměti nazýváme registry, v našem případě jich bude devět (Obr. 6).[4]

Program Counter (`PC`)
`A`
`B`
`C`
`D`
`E`
`F`
`G`
`H`

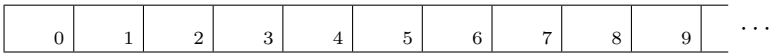
Obrázek 6: Vnitřní paměť procesoru tvořená devíti registry, každý o velikosti 8 bitů.

Každý z našich registrů je buňka velikosti 8 bitů, tedy čísla v rozsahu 0–255. Desítková soustava je sice čitelnější pro lidi, ale nás budou zajímat konkrétní bity, takže se přesuneme do soustav pro procesor přirozenějších. Pokud tedy dále v textu uvidíte číslo ve tvaru `0xAB`,

[4] Existuje alternativní architektura, založená na zásobníku – akumulátoru. Ta je však mimo rozsah našeho článku.

tak se nelekněte – předpona `0x` značí šestnáctkovou soustavu, předpona `0b` soustavu dvojkovou. Registry `A` až `H` jsou registry pro běžné použití, registr `PC` je speciální – většina instrukcí k němu nemá přístup.

Kromě registrů, které reprezentují vnitřní stav procesoru, potřebujeme ještě vnější paměť, ze které budeme číst instrukce programu. Tu si můžeme představit jako očíslovanou nekonečnou pásku, viz Obrázek 7. Jedno každé políčko pásky má taktéž velikost 8 bitů.



Obrázek 7: Vnější paměť procesoru ve které je uložený program, každé políčko má velikost 8 bitů a své pořadové číslo (adresu).

Takže teď už máme registry sloužící k počítání i programovou paměť, ve které se procesor dozví, co má počítat. Je tedy na čase jej zapnout. Říkali jsme, že procesor bude vykonávat zadané instrukce. Jako první ale musí vědět, kde ty instrukce vezme. Přečte si tedy obsah svého registru `PC` (Program Counter neboli česky programový čítač). Protože jsme procesor právě zapnuli, jsou všechny registry zatím prázdné – je v nich hodnota `0x00`. Hodnota `PC` je nula, procesor se tedy podívá na nulté místo paměti,[5] ve kterém si přečte instrukci.

Je tam nějaká? Vždyť my jsme zatím žádnou instrukci nezapsali?! To ale ničemu nevadí. Protože jsme do paměti nic nezapsali, je prázdná – jsou v ní samé nuly. Nulová instrukce, taky instrukce. V našem případě se jedná o instrukci `NOP` (*no operation*), nedělej nic.[6] Procesor tedy poctivě nic neudělá a bude chtít další instrukci. Zvýší tedy hodnotu v registru `PC` o jedna a čte dál. Situace se opakuje, a procesor čte instrukce, dokud se nedočte až na konec programové pásky. O té jsme si před chvílí řekli, že je nekonečná.

Tak počkat, tady něco nehraje. Kam až se dostane? Zkuste se chvíli zamyslet, než budete pokračovat ve čtení.

Každý registr, tedy i `PC`, je 8bitový. Načteme instrukci číslo 255, přičteme jedničku a $255+1 = 0$. Možná se to zdá podivné, ale je to přirozený důsledek toho, že máme jen 8 bitů na uložení adresy. Těchto osm bitů má maximální hodnotu 255 neboli `0b11111111`. Při přičtení jedničky získáme číslo `0b100000000`, ale ona jednička, která značí hodnotu 256, je 9. bit v pořadí, který se nám už do registru nevleze a je smazán – registr takzvaně přeteče. Počítání v 8bitových registrech je tedy počítání modulo 256. Takže poté, co procesor 256krát nevykoná nic, mu přeteče programový čítač a on se vrátí zpět na začátek. Právě jsme si v assembleru napsali nekonečný cyklus.

Teď jsme se dostali k zajímavému omezení – dokud budeme mít registr `PC` jen 8bitový, tak naše assemblerovské programy nemohou mít více než 256 instrukcí.

[5] Hle, počátek číslování (indexování) od nuly.

[6] Proč je zrovna prázdná instrukce instrukcí `NOP`? Protože nechceme, aby procesor, který nedostal instrukce, začal dělat něco nepředvídatelného.

Prozatím nám to stačí – programy, které budeme psát, budou mít zatím jen pár řádek.

Abychom nějaké smysluplné programy mohli začít psát, je potřeba rozšířit náš repertoár příkazů. Ještě předtím ale bude vhodné se podívat, jak procesor tyto příkazy rozlišuje.

Jak jsme si už řekli, jedna buňka programové paměti má 8 bitů – to znamená, že programová paměť může obsahovat 256 různých hodnot. Každé hodnotě přiřadíme nějakou operaci. Už jsme si řekli, že hodnota 0 znamená „nedělej nic, pokračuj“.

Dále můžeme například určit, že pokud procesor najde v paměti hodnotu `0b00000001`, přičte 1 k registru `A`. Když budeme chtít totéž vykonat pro registr `B`, bude tomu v paměti odpovídat hodnota `0b00000010`. A tak dále – zvýšení hodnoty registru `E` bude `0b00000101`, registru `F` pak `0b00000110`, ...

Čímž jsme právě zavedli instrukci `INC Rd`, po jejímž zavolání procesor zvýší hodnotu registru uvedeného na místo `Rd` o jedna. Zároveň zjišťujeme nepříjemnou věc – takto jednoduchá instrukce nás připravila o 8 z našich drahocenných 256 instrukcí. A teď si představte, že budeme chtít sčítat dva registry v plném rozsahu `A` až `H`. K tomu budeme potřebovat 8 hodnot pro jeden registr a 8 hodnot pro druhý registr. Abychom mohli sčítat každý s každým, dostaneme se na 64 kombinací, tedy 6 bitů.[7] Zbudou dva bity, což znamená, že operace, které podporují libovolné dva registry, můžeme mít nejvýše čtyři – pokud nebudeme mít žádné jiné. Ale my už máme `INC` a `NOP`, takže jsme na třech. Au.

Jak tuhle zapeklitou situaci budeme řešit? To bude úkol pro vás. Zkusíme se teď vcítit se do někoho, kdo navrhuje instrukční sadu.

Úloha 4.5 – Návrh instrukční sady (2b)

Navrhněte 8bitové instrukce[8], s jejichž pomocí budete schopni realizovat dále uvedené operace, a napište, jak bude tato realizace vypadat.[9] Co budeme po naší sadě chtít?

- Zachovat operace `NOP` a `INC Rd`.
- Umět nastavit registry na libovolnou zadanou hodnotu z celého jejich rozsahu. Není nutné, aby to byla operace na jednu instrukci.

[7] V instrukčních sadách se v takovýchto případech používají souvislé bloky bitů – je to přehlednější a lépe se to „drátuje“.

[8] Instrukce je číselná hodnota v rozsahu 0–255, která spolu s okamžitým obsahem registrů jednoznačně určuje, co procesor následně udělá (například jak upraví hodnoty svých registrů). Pak tuto instrukci „zapomene“ a načte novou podle adresy v registru `PC`.

[9] K provedení jedné operace může být potřeba jedna i více 8bitových instrukcí. Tato „realizace“ operace se může lišit například v závislosti na tom, na kterých konkrétních registrech má celá operace proběhnout.

Jednou z variant by mohlo být zavést instrukci `SHL Rr`, která posune bity registru `Rr` doleva o jeden bit.[10] Potom by uložení hodnoty 255 do registru `C` vypadalo takto:

```
INC C   (v C je 0b1)
SHL C   (v C je 0b10)
INC C   (v C je 0b11)
SHL C   (v C je 0b110)
...
```

Po několika dalších opakování získáme v registru `C` hodnotu `0b11111111`, jak jsme chtěli. Ale bude nás to stát 15 instrukcí, což není právě lichotivé, nehledě na to, že každé číslo budeme do paměti cpát jinak dlouho. Navíc má uvedený příklad jeden zásadní nedostatek – bude fungovat jen krátce po zapnutí procesoru dokud je v registru nezměněná počáteční nulová hodnota. Zkuste vymyslet lepší řešení.

- Dále chceme umět operace:

`ADD Rd, Rr`	součet `Rd` a `Rr` zapiš do `Rd`, operace přetéká;
`SUB Rd, Rr`	`Rd` − `Rr` → `Rd`, operace podtéká[11];
`NOT Rd`	bitovou negaci[12] `Rd` zapiš zpět do `Rd`;[13]
`AND Rd, Rr`	`Rd AND Rr` → `Rd`, bitový logický součin registrů;[14]
`OR Rd, Rr`	`Rd OR Rr` → `Rd`, bitový logický součet registrů;
`SHL Rd`	bitový posun doleva;
`SHR Rd`	bitový posun doprava;
`STOP`	ukončí chod procesoru.

Fantazii se meze nekladou, odměnou za zajímavá a elegantní řešení mohou být další bonusové body. Pokud se vám zdá, že vám nějaká operace chybí, klidně si ji dodefinujte, budete-li mít dost bitů nazbyt. Těm z vás, kteří již programovat umí, budou nepochybně chybět řídící struktury – větvení a skoky. K těm se dostaneme příště.

Pro teď bych rád, abyste se zamysleli nad tím, jak co nejefektivněji využít 8 bitů ke kódování instrukcí. Za míru efektivity můžete vzít například počet kódů instrukcí, které si musíme zarezervovat pro danou operaci vůči počtu kroků, které bude provedení operace trvat.

[10] Bitový posun doleva: když si napíšete bity vedle sebe, zahodíte ten nejvíce vlevo (tedy ten, který odpovídá nejvyšší mocnině dvojky v binárním zápisu čísla) a doprava dopíšete nulu, získáte bitový posun doleva. Bitový posun doprava je analogicky obráceným směrem. (Výsledkem je hodnota vynásobená, respektive podělená dvěma.)

[11] Analogicky k přetečení: $0 - 1 = 255$.

[12] Bitová negace: zapište si jednotlivé bity vedle sebe a znegujete každý jednotlivý bit. Z čísla `0b00000011` tak vznikne číslo `0b11111100`.

[13] Bitové operace si dobře zapamatujte, budeme je v assembleru používat v budoucnu velmi často – schválně, k čemu by se nám mohly hodit?

[14] Bitový and/or: tentokrát si sepište pod sebe bity dvou registrů a na každou odpovídající dvojici aplikujte logický součin respektive součet. Tedy `0b11001100 AND 0b10101010 = 0b10001000`.

Až sadu budete mít, předveďte její funkčnost spočítáním následujícího výrazu a uložením výsledku do registru `D`:

$$255 - 10 + ((5 - 3)\ \texttt{OR}\ 64) \to \texttt{D}\,.$$

K využití operací vám poskytnu následující příklad. Chceme spočítat výraz $1 + 2 - (16 \gg 3)$ a výsledek uložit do registru `B`. Symbol $\gg 3$ značí bitový posun doprava o tři bity.

```
            ; ';' bude v PAPIRASU značit komentář pro zbytek
            ; řádku.
LDI B, 1    ; Načte hodnotu 1 do B.
LDI C, 2    ; Načteme do registru C hodnotu 2.
ADD B, C    ; V PAPIRASU si příště zadefinujeme ADD, který
            ; sečte dva registry a výsledek uloží do
            ; registru B. Vy nejspíše nebudete mít dost bitů
            ; a budete muset najít nějakou alternativu.
LDI C, 16   ; Přepíšeme registr C hodnotou 16, předchozí
            ; hodnotu už nepotřebujeme.
SHR C, 3    ; Za předpokladu, že funkce SHR posunuje o zadaný
            ; počet bitů. Opět nejspíše narazíte na bitová
            ; omezení. Jedna z variant by byla SHR C, SHR C,
            ; SHR C, pokud posunujeme vždy jen o jedna.
SUB B, C    ; SUB B, C je instrukce B = B - C a jsme hotovi.
STOP        ; konec
```

Tímto končím první díl úvodu do assembleru. Pokud vám dnešní část přišla složitá, nezoufejte, příště to bude jednodušší – budeme se méně zabývat jednotlivými bity a více samotnými instrukcemi. Myslím si však, že dnešní část je důležitá k pochopení některých vlastností, které assemblery mají, a to i v dnešní době 64bitových mnohojádrových procesorů.

Na co se těšit příště: Větší registry, více paměti. Nové instrukce – zejména skoky a větvení. Počítání instrukcí, generování Fibonacciho čísel. A nejen to.

Tomáš Bartoněk

Výsledková listina 4. čísla

Poř.	Jméno	R.	$\sum_{-1}$	Úlohy r1	r2	r3	r4	t2	t3	$\sum_0$	$\sum_1$
1.	Mgr.[MM]K. Rosická	2	22,0	4,0	3,0	2,5				9,5	22,0
2.	Mgr.[MM]P. Turinský	4	45,6		3,0	3,1				6,1	20,1
3.	Dr.[MM]A. Mlezivová	3	75,3		2,8	3,0				5,8	19,8
4.	Bc.[MM]V. Pavlíček	1	18,3				2,8			2,8	18,3
5.	Mgr.[MM]K. Čížková	3	23,5		3,0	1,5	2,2			6,7	17,0
6.	Bc.[MM]K. Balej	2	16,8	4,0		1,5	2,0			7,5	16,8
7.	Bc.[MM]K. Vnuková	2	16,7							0	16,7
8.	Bc.[MM]M. Holubička	1	16,3		1,8					1,8	16,3
9.	Bc.[MM]B. Hroncová	2	15,0			2,0				2,0	15,0
10.	Bc.[MM]Z. Lüköová	2	13,1	1,1	2,5					3,6	13,1
11.	Bc.[MM]M. Bukvaj	1	13,0							0	13,0
12.	Doc.[MM]P. Šimůnek	4	155,5						1,3	1,3	12,8
13.	Bc.[MM]R. Olšák	2	12,0			3,0				3,0	12,0
14.	Mgr.[MM]J. Suchánek	3	26,9							0	11,7
15.	Bc.[MM]O. Gonzor	Z9	11,2				1,6	3,9		5,5	11,2
16.	Mgr.[MM]B. Požár	3	30,9							0	10,8
17.	Mgr.[MM]J. Paidar	3	29,4		3,0	1,0				4,0	10,3
18.	Bc.[MM]T. Kulichová	2	10,2		2,0					2,0	10,2
19.	Mgr.[MM]T. Hladíková	3	45,4	0,3						0,3	9,8
20.	Mgr.[MM]J. Pallová	2	28,8			3,0				3,0	9,5
21.	Mgr.[MM]O. Buček	3	27,6							0	9,3
22.	Mgr.[MM]F. Čermák	3	44,8			2,9				2,9	9,2
23.	V. Procházka	3	9,0							0	9,0
24.	T. Drobil	2	8,7							0	8,7
25.	Mgr.[MM]L. Vincenová	3	45,5			3,0				3,0	8,4
26.–27.	Mgr.[MM]D. Chytilová	3	28,8							0	8,0
	Bc.[MM]A. Neubauerová	3	16,0			2,0				2,0	8,0
28.	Mgr.[MM]O. Knopp	3	34,3							0	7,7
29.	J. Heller	4	7,5							0	7,5
30.	L. Šajnarová	3	7,3							0	7,3
31.–32.	Bc.[MM]L. Bujnovská	3	12,0							0	7,0
	J. Pelc	3	7,0							0	7,0
33.–34.	Mgr.[MM]L. Kopfová	2	25,7							0	6,7
	J. Šrejbr	1	6,7							0	6,7
35.	T. Dolák	4	6,3							0	6,3
36.	A. Mírková	2	6,0							0	6,0

Poř.	Jméno	R.	$\sum_{-1}$	Úlohy r1	r2	r3	r4	t2	t3	$\sum_0$	$\sum_1$
37.	Bc.[MM]J. Růžička	2	11,5							0	5,5
38.	Mgr.[MM]L. Kundratová	2	32,7							0	5,4
39.	Bc.[MM]F. Zajíc	4	15,7							0	3,7
40.	Mgr.[MM]S. Lukeš	4	42,4							0	3,2
41.	P. Martínek	2	3,1	0,1		0,0				0,1	3,1
42.–44.	D. Daubner	2	3,0							0	3,0
	R. Luc	4	7,0			3,0				3,0	3,0
	M. Picek	2	3,0							0	3,0
45.	T. Poláková		2,8		2,8					2,8	2,8
46.–47.	M. Machalová	4	4,7							0	2,7
	M. Sejkorová	2	2,7							0	2,7
48.	A. Šebestíková	2	2,0		2,0	0,0				2,0	2,0
49.	F. Kmječ	1	1,5			1,5				1,5	1,5
50.	L. Kubacki	4	0,0		0,0	0,0				0,0	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

Adresa redakce:

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

E-mail: mam@matfyz.cz
WWW: http://mam.matfyz.cz