Studentský matematicko-fyzikální časopis

Ročník XX / Číslo 3

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.

Zadání úloh

Termín odeslání třetí série: 13. 1. 2014

Lední medvěd Barulkos si zamyšleně prohrábl srst na hlavě a překulil se na druhý bok. Období zimního spánku už dávno začalo, ale on ne a ne usnout. Možná by mohl navštívit jednoho ze svých kamarádů a zjistit, jestli taky nemají potíže se spaním. Kdyby si ovšem pamatoval, kde bydlí... Ale moment! Vždyť má přece vzadu v koutě vyrytou do ledu mapu!

Úloha 3.1 – Lední mapa (4b)

Na mapě (nekonečné rovině) jsou vyznačeny příbytky medvědů. Pro každý z nich platí, že v jeho okolí se nachází právě tři další doupata ve vzdálenosti menší než 5 km. Kolik může být na mapě doupat?[1]

Barulkos se vyvalil z pelechu, vyklepal ze svého ledního kožichu lední blechy a vydal se k nejbližšímu sousedovi – svému kamarádu Valibukovi. Po cestě ho však zastihnul řádný hlad, a tak bez velkého přemýšlení prorazil díru v ledu a začal chytat ryby.

Úloha 3.2 – Rybaření (3b)

Zkoušeli jste někdy chytat ryby holýma rukama? Není to vůbec jednoduché ... mj. proto, že ryba se nachází blíže a hlouběji, než ji vidíme. Jak velká je odchylka zdánlivé polohy ryby od té skutečné? Jako bonus se můžete zamyslet nad tím, jakou nepřesnost vnesou do určení polohy ryby nerovnosti hladiny.

Poté co se najedl, dorazil konečně k Valibukovi, který také nemohl najít polohu pro pořádný zimní spánek. Zasedli spolu k šálku ledového čaje a začali probírat poslední velký medvědí sraz, kterého se oba zúčastnili. Takový medvědí sraz je velmi formální záležitost. Medvědi si potrpí na protokol a tak mají organizátoři vždy velké starosti s jejich rozdělením do skupin.

Úloha 3.3 – Organizace skupinek (4b)

Ve skupině 30 medvědů se každý zná s alespoň 10 dalšími (tento vztah je vzájemný). Je možné je rozdělit do 2 až 3 členných skupinek tak, aby každý ve své skupince někoho znal?

Když dopili čaj, dostali chuť ještě na něco malého k snědku a Valibuk vyhrabal v rohu rohu své ledové jeskyně úplně zmrzlou pizzu – přesně tak, jak to mají lední

[1] Ptáme se na všechny možné počty, ne jen na příklad jednoho takového počtu.

medvědi nejraději. A pokud by vás snad zajímalo, kde medvědi berou pizzu a ledový čaj… Nu, řekněme, že tomu, kdo jim je přinesl, už tyhle věci chybět nebudou.

Úloha 3.4 – Hra na hamouna (2b)

Pizza, kterou Valibuk s Barulkosem rozbalili, je nařezaná na 8 dílků. Dílky však jsou nařezané nerovnoměrně[2], a tak není snadné pizzu rozdělit. Dohodli se proto na následujícím postupu. Začne Valibuk a vybere si libovolný dílek pizzy. Potom se ve vybírání střídají, ale vždy musí zvolit jeden z dílků sousedních s už vyjedenou oblastí. A samozřejmě se oba snaží získat pizzy co nejvíc. Dokažte, že existuje strategie, při které Valibuk získá alespoň polovinu pizzy.

Po dobrém jídle se oba rozvalili na záda a oddychovali, až se jim pomalu začaly klížit oči. Ledový strop ledové jeskyně se medvědům rozplynul před očima a nahradily jej vzpomínky na poslední polární den.

Úloha 3.5 – Riki (2b)

Zkuste na papír zachytit Rikiho[3] všední den, či zavzpomínat na jeho prázdninová dobrodružství. Nejhezčí obrázky otiskneme v některém z dalších čísel.

[2] Pizza tedy vypadá jako kruh s osmi výsečemi, řezy jdou jen od středu na okraj.
[3] Riki je náš maskot, lišák, kterého potkáváte na ilustracích v každém čísle.

Řešení úloh

Úloha 1.1 – Ping-pongový turnaj (3b)

Zadání:
Turnaje ve stolním tenise se zúčastnilo dvakrát více dívek než chlapců. Každý s každým hrál právě jeden zápas, který nikdy neskončil remízou. Poměr mezi výhrami dívek a výhrami chlapců byl 7:5. Kolik hráčů se turnaje mohlo zúčastnit?

Řešení:
Označme počet chlapců n. Pak celkový počet hráčů je $3n$. Celkový počet zápasů je $\frac{3n(3n-1)}{2}$, protože každý z $3n$ hráčů hrál se všemi ostatními a takto jsme započítali každý zápas dvakrát. Protože poměr mezi vyhranými zápasy děvčat a chlapců má být 7:5, musí $12 \mid \frac{3n(3n-1)}{2}$, po úpravě dostaneme $8 \mid n(3n-1)$. Protože n a $(3n-1)$ mají rozdílnou paritu, tak buď $8 \mid n$ nebo $8 \mid (3n-1)$. Z toho dostáváme, že turnaje se mohlo zúčastnit jedině $24k$ nebo $24k+9$ hráčů, kde $k \in \mathbb{N}$.

Tímto bohužel většina řešení končila. Zatím jsme ale ze zadání vyvodili pouze nějakou podmínku na počet hráčů, která nám nezaručuje, že turnaj s takovým počtem hráčů mohl proběhnout. Speciálně jsme podmínku na poměr vyhraných zápasů dívek a chlapců využili jen k dělitelnosti dvanácti. Zkuste si rozmyslet, že třeba poměru 1:11 nedokážeme nikdy dosáhnout. Pokud hrají dva chlapci, je jasné, že vyhraje chlapec. Obdobně i pro dívky. Musíme tedy ukázat, že počet výher chlapců (resp. dívek) je menší nebo roven $\frac{5}{12}$ (resp. $\frac{7}{12}$) všech odehraných zápasů.

Požadované nerovnosti stačí roznásobit a platnost jednoduše vyplyne:

Chlapci:

$$\frac{n(n-1)}{2} \leq \frac{5}{12} \cdot \frac{3n(3n-1)}{2} \leq (33n-3)n$$

Pokud $n = 0$, nastane rovnost. Jinak je $(33n-3)$ i n kladné, a tedy nerovnost platí.

Dívky:

$$\frac{2n(2n-1)}{2} \leq \frac{7}{12}\frac{3n(3n-1)}{2} \leq 15n^2 + 3n$$

Nerovnost platí, protože na pravé straně sčítáme dva nezáporné členy.

Tím jsme ukázali, že turnaje se opravdu mohlo zúčastnit $24k$ nebo $24k+9$ hráčů ($k \in \mathbb{N}$).

Martin

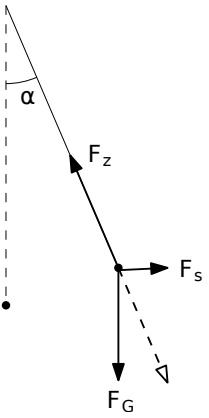
Úloha 1.2 – Kývající se přezka (3b)

Zadání:

Agent č. 7368 nasedl do vlaku a našel si volné kupé. Na horní poličku si odložil zavazadlo a pohodlně se usadil. Když se vlak začal rozjíždět, všiml si, že popruh, jenž má na konci přezku a který mu visí z batohu volně do prostoru, se začal kývat. Jak se bude přezka pohybovat během rozjezdu s konstantním zrychlením a a během jízdy, kdy se vlak pohybuje konstantní rychlostí v? Místo přezky a popruhu můžete uvažovat matematické kyvadlo s hmotností m a délkou závěsu l.

Řešení:

Jelikož agent č. 7368 sedí ve vlaku, rozebereme si chování přezky v neinerciální soustavě, tj. soustavě spojené s vlakem, ne se zemí. Přezku na popruhu si aproximujeme matematickým kyvadlem o délce závěsu l a hmotnosti závaží (přezky) m. Stojí-li vlak ve stanici, přezka je vůči pozorovateli v klidu (nepohybuje se) a visí svisle dolů, tj. ve směru působení tíhové síly $\vec{F_G}$. Při rozjíždění vlaku ($a = \text{konst.}$) na přezku kromě tíhové síly $\vec{F_G}$ a tahové síly závěsu $\vec{F_z}$ působí ještě setrvačná síla $\vec{F_s}$, působící proti směru zrychlení vlaku (obr.1). Výslednice těchto tří sil musí být nulová, jelikož kyvadlo je vůči pozorovateli v klidu. Přezka zůstane vychýlena o úhel α od svislého směru v rovině určené $\vec{F_G}$ a $\vec{F_s}$. Úhel α určíme z podmínky rovnováhy sil:



Obrázek 1

$$\tan\alpha = \frac{|\vec{F_s}|}{|\vec{F_G}|} = \frac{ma}{mg} = \frac{a}{g} \qquad (1)$$

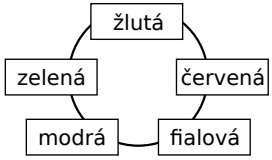
Při přechodu z $a = \text{konst.}$ na $v = \text{konst.}$ přestane působit setrvačná síla. Kyvadlo má tendenci se dostat do rovnovážné polohy, kdy se vyrovná tíhová síla a tahová síla závěsu, tj. $\alpha = 0$, čímž dojde k rozkývání přezky. V aproximaci matematického kyvadla, kdy se zanedbává odpor vzduchu při pohybu kyvadla i tření v závěsu, by se přezka do nekonečna kývala kolem vertikální osy procházející bodem uchycení kyvadla se stejnou maximální výchylkou α na obě strany a periodou kmitání $T = 2\pi\sqrt{\frac{l}{g}}$. Uvážíme-li vliv odporu vzduchu, amplituda kmitů se bude postupně zmenšovat, až kmity zcela zaniknou a přezka bude opět nehybně viset ve směru tíhové síly. Při zpomalování vlaku se přezka opět vychýlí o úhel α daný vztahem (1), avšak na opačnou stranu než při zrychlování (ve směru působení setrvačné síly, tj. proti směru zrychlení vlaku).

Peťa

Úloha 1.3 – Šátky (3b)

Zadání:
Agent nosí každý den jeden z pěti barevných šátků. Ústředí tajné služby pořídí při přeletu špionážní družice jeden satelitní snímek, ze kterého pozná barvu šátku, který má agent zrovna na hlavě. Kvalita snímku ale není moc velká, a barvy šátků jsou tak mezi sebou částečně zaměnitelné – každé dvě barvy, které spolu v obrázku níže sousedí, si spolu můžou splést (zelená tedy může být na snímku vidět jako žlutozelená nebo modrozelená). Agent se za 10 dní snaží přenést co nejvíce informace nějakým předem domluveným kódem, což si můžeme představit jako přenesení jednoho konkrétního čísla z co největšího rozsahu. Například za jeden den může volbou červená nebo modrá přenést číslo z rozsahu 1–2. Ale jaký postup je optimální pro více dní, a jaký rozsah je tedy možné za 10 dní přenést?

Řešení:
Nejprve si úlohu trochu přeformulujme v řeči grafů, jako třeba graf zaměnitelných šátků v zadání. Barvy budeme označovat prvními písmenky jejich názvů (Č, Ž, Z, M, F).

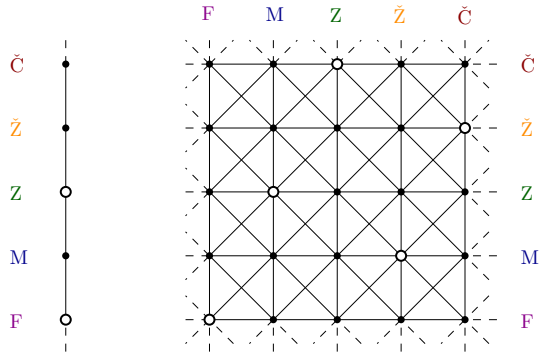


Pro optimální strategii na jeden den se agent může jen dopředu domluvit, jakou podmnožinu barev šátků může použít. Tato podmnožina má být co největší, ale přitom nesmí být šátky zaměnitelné. Právě mezi zaměnitelnými zprávami povede hrana.

I bez této úvahy je celkem jasné, že za jediný den nemůže odeslat více než jednu ze dvou různých zpráv. Ale i tak můžeme během 10 dnů odvysílat jednu z 1024 zpráv. Zvolme, že F znamená 0 a Z znamená 1. Desetidenní zpráva pak odpovídá deseticifernému číslu ve dvojkové soustavě, tedy jednomu číslu z rozsahu $0 \ldots 1023$, též 1 bit za den.

Jde to ale ještě lépe, když se zamyslíme nad strategií pro dva dny. Uvažujme všechny dvojice (barva první den, barva druhý den) a jejich zaměnitelnost. Například dvojice (Č, Z) a (F, M) zaměnit lze – pokud by první den pozorovali červeno-fialovou a druhý den modro-zelenou. Ale dvojice *(Č, Z)* a (F, Č) už zaměnit nelze – zelenou a červenou druhý den si nikdy nespletou. Grafy zaměnitelnosti barev za jeden a dva dny jsou na obrázku, v obou případech přerušované hrany jdoucí z obrázku pokračují z druhé strany.

Taková tzv. *nezávislá množina* (tedy podmnožina vrcholů, mezi kterými nevede hrana) je třeba (F, F), (Z, M), $(\check{\mathrm{C}}, \mathrm{Z})$, $(\mathrm{M}, \check{\mathrm{Z}})$ a $(\check{\mathrm{Z}}, \check{\mathrm{C}})$, a ta je v tomto grafu největší (více dvojic už nutně musí obsahovat zaměnitelnou dvojici). To nám dává o dost lepší výsledek – pokud si dvojice očíslujeme $0 \ldots 4$ a zapíšeme desetidenní pozorování jako pěticiferné číslo v pětkové soustavě, přeneseme libovolné číslo v rozsahu $0 \ldots 3124$, což též můžeme formulovat jako $\log_2(3125)/10 = 1.16$ bitů za den.



Mohli bychom podobně zkoušet i grafy odpovídající více dnům najednou (tedy takové tří- a více-rozměrné zahuštěné mřížky), ale ukazuje se, že více než průměrně 1.16 bitů za den není možné odvysílat a 3124 za 10 dnů je opravdu maximum. Dokázat to je už o něco složitější, ale velmi pěkný a čitelný popis tzv. Shannonovy kapacity grafu (kterou jsme tu vlastně počítali) a důkaz, že $\Theta(C_5) = \sqrt{5}$ a tedy $\Theta(C_5^2) = 5$, najdete ve 12. kapitolce sborníčku *Šestnáct miniatur* od J. Matouška[4].

Tomáš

Úloha 1.4 – Betonová (3b)

Zadání:

Když platíme účty za elektřinu, zajímá nás, kolik stojí jedna kilowatthodina korun. Jak je to ale s betonovou náročností výroby elektřiny? Kolik metrů krychlových použitého betonu připadá na výrobu jedné kilowatthodiny různými způsoby? Vychází z tohoto pohledu nejlépe elektrárna větrná, vodní, jaderná nebo jiná? A co kdyby nás zajímal celkový objem použitých materiálů, nebo třeba zastavěná plocha? Své závěry nezapomeňte zdůvodnit a uvést, odkud jste pro ně čerpali data. Iniciativě se meze nekladou, můžete si zasloužit i bonusový bod!

Řešení:

Jako ilustrační případ uvedeme výpočet betonové náročnosti Jaderné elektrárny Temelín. Na její výstavbu bylo použito cca 2 060 000 m^3 betonu [1]. Její celkový instalovaný výkon činí 2 000 MW [2]. Počítáme-li s tím, že hustota použitého betonu je zhruba 2 500 $\mathrm{kg.m}^{-3}$ [3], vychází nám betonová náročnost Temelína na

$$\frac{2\,500\,\mathrm{kg.m}^{-3} \cdot 2\,060\,000\,\mathrm{m}^3}{2\,000\,000\,000\,\mathrm{W}} = 2{,}58\,\mathrm{kg} \cdot \mathrm{W}^{-1}$$

Ve skutečnosti ale není tato hodnota přesná, protože elektrárna nepracuje neustále na plný výkon. Nakolik je využíván výkon energetického zdroje vyjadřuje tzv. koeficient ročního využití [4]. V případě JE Temelín činí tento koeficient 79,4 % [2]. Betonová náročnost Temelína je tedy

[4] `http://kam.mff.cuni.cz/~matousek/la-appls.ps`

$$\frac{2\,500\,\mathrm{kg}\cdot\mathrm{m}^{-3}\ \cdot\ 2\,060\,000\,\mathrm{m}^3}{2\,000\,000\,000\,\mathrm{W}\ \cdot\ 0{,}794} = 3{,}24\,\mathrm{kg}\cdot\mathrm{W}^{-1}$$

Podívejme se nyní v porovnání na větrné elektrárny. Počítáme-li s větrnou elektrárnou, jejíž konstrukce je postavená především na ocelovém sloupu (dnes už se začínají rozmáhat i sloupy betonové), spotřebujeme na základy jednoho sloupu 1 104 000 kg betonu. Větrná elektrárna z našeho příkladu má výkon 2 MW [5] a její koeficient ročního využití je 12,71 %. Její betonová náročnost pak vychází

$$\frac{1\,104\,000\,\mathrm{kg}}{2\,000\,000\,\mathrm{W}\ \cdot\ 0{,}1271} = 4{,}34\,\mathrm{kg}\cdot\mathrm{W}^{-1}$$

Snadno nahlédneme, že kdybychom nepočítali s koeficientem ročního využití, vycházela by větrná elektrárna na spotřebu betonu lépe než jaderná. Ve chvíli, kdy tento faktor zahrneme, je tomu přesně naopak. Pokud bychom chtěli být ještě důkladnější, mohli bychom zvážit např. projektovanou životnost jednotlivých typů elektráren a také ji zahrnout do výpočtu.

[1] `http://www.transportbeton.cz/jaderna-elektrarna-temelin.html`
[2] `http://cs.wikipedia.org/wiki/Jaderná_elektrárna_Temelín`
[3] `http://www.ebeton.cz/pojmy/hmotnost-betonu`
[4] `http://cs.wikipedia.org/wiki/Koeficient_ročního_využití`
[5] `http://www.csve.cz/clanky/betonovy-zaklad/305`
[6] `http://cs.wikipedia.org/wiki/Větrná_elektrárna#.C3.9A.C4.8Dinnost`

Slovo závěrem

Pokud provádíme výzkum na nějaké téma, je důležité uvést zdroje, ze kterých jsme čerpali. Pokud uvedeme k jednomu údaji více nezávislých zdrojů, je to jedině dobře. Pokud žádné relevantní zdroje nemáme k dispozici, snažíme se alespoň svoje tvrzení řádně podpořit argumenty a necucáme si čísla z prstu. Je dobré položit si otázku, co všechno může ovlivňovat výsledek výpočtu a jak moc (v našem případě např. koeficient ročního využití). Pokud jsme přesvědčeni o tom, že můžeme nějaké vlivy zanedbat, můžeme to udělat, ale musíme o tom čtenáře zpravit a vysvětlit, proč považujeme dané vlivy za zanedbatelné.

Honza

Řešení témat

Téma 1 – Zobrazování počasí

V nesoustředkovém termínu prvního čísla dorazila řešení Anety Lesné, Jany Noskové a Bc.MM Vladimíra Bartoviče.

Aneta Lesná a Bc.MM Vladimír Bartovič se zabývali použitím šišky pro zobrazení počasí. Bohužel ani jeden z jejich příspěvků není psán jako souvislý text, proto otiskujeme jejich shrnutí.

Oba zmiňují, že parametry šišky se mění s vlhkostí. Aneta Lesná uvádí (krom schlíplosti, což je velmi špatně specifikovatelný parametr) i barvu a hmotnost, a uvádí, že tento postup byl v místě jejího bydliště aplikován s přesvědčivými výsledky, bohužel ale tyto výsledky neuvádí, ani se na ně nikam neodkazuje, takže tomuto tvrzení nemusíme věřit. Vladimír Bartovič navrhuje využití změn objemu šišky, dále navrhuje měření objemu Archimédovou metodou, ale sám uznává, že ponořením šišky do kapaliny změní vlhkost v jejím okolí (a navíc nás zajímá spíše objem obalového útvaru šišky, než objem samotného dřeva – šiška roztahuje šupinky, hmota samotná se nijak výrazně neroztahuje), nebo laserový 3D skener, což je pro změnu nedostupné v zálesáckých (domácích) podmínkách. Dokáže někdo pomocí některé z uvedených veličin nakalibrovat šišku k měření vlhkosti?

Oba se rovněž zabývají popisem větru. Uvádí, že nárazy větru se projeví trhavými pohyby šišky. Aneta Lesná doporučuje dobře zvážit umístění šišky, aby rychlost a směr větru nebyl ovlivněn okolními budovami či krajinnými útvary. Bc.MM Vladimír Bartovič uvádí, že čím větší bude rychlost větru, tím více se vychýlí provázek se šiškou ze svislé polohy. Tvrdí, že jde o přímou úměru (tedy lineární závislost rychlosti větru na výchylce), ničím ale toto tvrzení nepodpořil. Teoretické či experimentální ověření či vyvrácení tohoto tvrzení je tedy ponecháno někomu dalšímu. Dále Bc.MM Vladimír Bartovič uvádí, že maximální změřitelná rychlost větru je závislá na hmotnosti šišky a provázku. Dalším úkolem je tedy odvodit tuto závislost. A je maximální měřitelná rychlost ovlivněná ještě něčím jiným?

Všichni zmínění se zabývali jinými způsoby zobrazování počasí, nejpodrobnější popis podala Jana Nosková, proto vám přinášíme přepis jejího příspěvku

Další návrhy (3b)

Jana Nosková

Domeček s panáčkem a panenkou je založen na vláknu (např. koňské žíně), které se kroutí v závislosti na vlhkosti vzduchu. Vlákno se tak natahuje nebo smršťuje a pohybuje figurkami. Panenka určuje suché a jasné počasí, panáček vlhko a deštivo.

Větrníky vyrobené z lehkého nylonu reagují na velmi slabý vánek, proto jsou mnohem citlivější, než větrníky z plastu nebo plechu. Větrník neukazuje sílu větru.

Korouhve na střechách, např. ve tvaru kohouta s ukazatelem světových stran, jsou dokonale vyvážené, takže se mohou snadno pohybovat kolem své osy. Při větru se otáčí a šipka ukazuje směr větru na směrovkách.

Kromě již uvedených způsobů lze předpovídat počasí ještě pomocí mraků: ranní červánky předpovídají déšť, večerní naopak jasné počasí. Známé je rovněž předpovídání počasí podle chování zvířat a rostlin. Pokud má být pěkně, tak vlaštovky létají vysoko, pavouci tkají sítě a hmyz se rojí; pokud má být deštivo, tak žížaly vylézají na povrch, vlaštovky létají nízko, některé rostliny zavírají květy, a starší lidi bolí klouby.

Tento příspěvek přináší některé otázky, jimiž se můžete dále zabývat: Jaká vlákna reagují na vlhkost, které byste si vybrali do svého vlastního domečku? Vyrobí někdo domeček s panáčkem a panenkou? Skutečně větrník nedokáže alespoň orientačně zobrazit sílu větru? Nedalo by se to přece jen nějak udělat? Navrhne někdo dokonale vyváženou korouhev lišáka Rikiho? Proč se zvířata a rostliny chovají tak, jak se chovají? Klouby zatím pomineme... Na jaký měřitelný parametr počasí reagují?

Jak vidíte, došlé příspěvky přináší víc otázek než odpovědí. Čeká nás ještě spousta bádání, aby nezůstaly nezodpovězeny.

Zuzka

Téma 2 – Kozy

V termínu po soustředění přišly další příspěvky, za které děkujeme. Vymysleli jste další zajímavé útvary, které můžete nalézt na webu.

Doc.^{MM} Aneta Šťastná poslala formalizaci problému a naučila nás, jak situaci na louce popsat bez kreslení obrázků. Pokud budete posílat další příspěvky zapsané pomocí této formalizace nebo její modifikace, ulehčíte tím sobě i nám pochopení vašich myšlenek.

Formalizace (2b)

Doc.^{MM} Aneta Šťastná

Povolené prostředky jsou očka na zatlučených kůlech, kůl se dvěma oky, lano přivázané k oku, dalšímu lanu nebo koze a koza samotná. Pro každou věc si můžeme zavést označení (N je přirozené číslo, A znak abecedy):

- zatlučený kůl – kN, např. k3,
- koza – Z,
- kůl se dvěma oky – KN(první bod, druhý bod, délka), např. K1(k3, Z, 5 m),
- lano – LN(první bod, druhý bod, délka), např. L1(k1, k2), L2(k1, Z, 3 m), L3(L1, Z, 2 m),

- pomocný bod – pN,

Dále bychom měli zavést vzájemnou polohu prvků, definovanou vzdáleností, případně souřadnicemi nebo geometrickým vztahem. Zde můžeme převzít matematické značení:

- vzdálenost – |první prvek, druhý prvek|
- kolmost – $L1 \perp L2$, pro jednodušší textové zpracování L1 T L2
- rovnoběžnost – L1 || L2
- souřadnice – k1[2, 3]
- náleží – $p1 \in L2$ (lano vede nad bodem p1)

xlfd

Téma 3 – FlatFox

K tématu FlatFox od vás přišlo celkem osm příspěvků, a to jak řešení online hlavolamů, tak různé teoretické úvahy, užitečné podprogramy a návrhy na vylepšení. Vzhledem k omezenému prostoru tištěného čísla bylo nutné některé příspěvky zkrátit a vynechat větší programy. Kompletní příspěvky s barevnými programy najdete na webu tématka[5], stejně jako autorská řešení hlavolamů.

Z teoretických příspěvků byly nejčastější ty popisující užitečné komponenty pro operace jako nulování registru, sčítání, přesun a kopírování hodnot, násobení a celočíselné dělení se zbytkem, ale i složitější, jako práce se zásobníkem či mocniny. Níže zveřejňujeme články od Dr.MM Mateje Lieskovského a Mgr.MM Václava Rozhoně. Podobné příspěvky nám zaslali též Doc.MM Aneta Šťastná, Doc.MM Markéta Calábková, Jonáš Havelka a Bc.MM Otto Hollmann. Článek Mgr.MM Dominika Krasuly je zatím v přípavě ke zveřejnění.

Vzhledem k tomu, že většina z vás ve svých programech opakovaně používala komponenty pro sčítání, kopírování, násobení a tak podobně, inspirováni vašimi podprogramy jsme rozšířili FlatFox o další operace a vznikl tak **FlatFox++**, o kterém se rovněž dočtete níže. Na stránce tématka najdete též nové bodované hlavolamy pro FlatFox++.

S větší silou jazyka se nabízí i další a zajímavější aplikace, problémy, rozšíření a otázky (několik jich najdete pod popisem FlatFoxu++). Budeme se na ně těšit ve vašich příspěvcích!

FlatFox++

FlatFox++ je dvourozměrný jazyk zpětně kompatibilní s FlatFoxem. Přibývají hlavně nové instrukce pro efektivnější operace s čísly a jedna instrukce pro ladění

[5] `http://mam.mff.cuni.cz/flatfox`

programů. Ve většině nových instrukcí má červený registr `R` speciální roli takzvaného *akumulátoru*, všechny nové instrukce ho používají jako jeden z operandů. Díky tomu stačí méně nových instrukcí, ale nelze tak například přímo sečíst registry `G` a `B`.

- `o` – Tato instrukce pozastaví provádění programu, třeba za účelem ladění a inspekce. Rychlou simulaci programu je pak možné obnovit nebo třeba pokračovat krokováním. Při vyhodnocování hlavolamů se pole chová jako prázdné. Tuto instrukci inspirovaly návrhy Dr.^{MM} Mateje Lieskovského a Mgr.^{MM} Václava Rozhoně.

- `R0, G0, B0, C0, M0, Y0` – Tyto instrukce přímo vynulují daný registr.

- `GA, BA, CA, MA, YA` – Přičtení *(Add)* hodnoty k registru `R`, hodnota zdrojového registru se nezmění.

- `GS, BS, CS, MS, YS` – Odečtení *(Substract)* hodnoty od registru `R`, hodnota zdrojového registru se nezmění, pokud by mělo vyjít záporné číslo, výsledkem bude `R=0`.

- `GM, BM, CM, MM, YM` – Vynásobení *(Multiply)* registru `R` jiným registrem.

- `GD, BD, CD, MD, YD` – Vydělení *(Divide)* registru `R` jiným registrem se zbytkem. V registru `R` bude zbytek po dělení (jako bychom od něj opakovaně odečítali, dokud je to možné), ve druhém registru pak podíl.

- `GL, BL, CL, ML, YL` – Zkopírování *(Load)* hodnoty registru `R` do jiného registru. Hodnota registru `R` se nezmění.

Tyto operace určitě drasticky sníží počet kroků (nebo též *časovou složitost*) různých operací. Nabízí se ale otázka, kterých a jak moc. Jak byste upravili programy své či ostatních a jakého zrychlení by tak šlo dosáhnout? Co soudíte o volbě právě tohoto typu rozšíření, jaké to má výhody a nevýhody a proč? A jaká další zlepšení FlatFoxu++ byste navrhli?

Na webu už vás čeká vylepšený interpret spolu s trochu těžší dávkou online úloh pro FlatFox++. Jedním z témat úloh je vztah složitosti přičítání 1, sčítání a násobení.

Zatímco si mnozí všimli, že násobit $a \times b$ nebylo možné rychleji než řádově $a \times b$ kroky, násobit $a \times b$ s pomocí nových instrukcí sčítání apod. ale *bez* použití instrukcí pro násobení a dělení lze o dost rychleji než v řádově $\min\{a, b\}$ krocích. Jak na to?

Díky za všechny zajímavé příspěvky a těším se na vaše další nápady, reakce, konstrukce, řešení a komentáře!

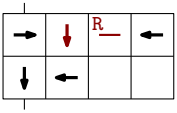
Tomáš

FlatFox (16b)

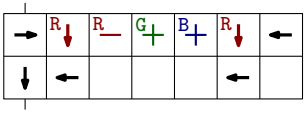
Dr.[MM] *Matej Lieskovský*

Když jsem dostal tento programovací jazyk, chvilku jsem si s ním hrál a pak jsem si položil jednoduchou otázku: „Co všechno to umí?“ Takový hezký základ by bylo dokázat, že je FlatFox Turingovsky úplný. A to se nejlépe dokazuje odsimulováním Turingova stroje. Jelikož to není úplně triviální, nejdříve si vyrobím několik základních součástek, které pak budu moct použít (nejen) v Turingově stroji.

Pokud možno, tak hlava bude vstupovat do komponenty levým horním rohem a opouštět ji levým dolním rohem. Tím docílíme snadnějšího řetězení operací.



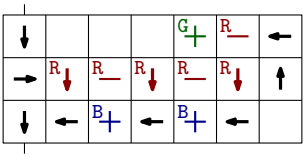
Vynulování registru.



Kopírování hodnoty (též do více registrů).

Pro násobení konstantou přidáme v přesunu další pluska.

Celočíselné dělení konstantou, ať už jako modulo konstanta (vynecháme `G+`), dělení beze zbytku (vynecháme `B+`) nebo divmod:

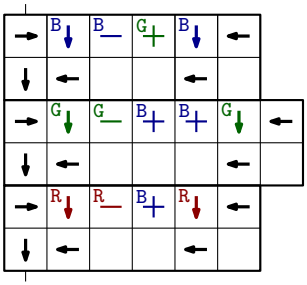


Jelikož umíme přesouvat mezi registry, můžeme nadefinovat in-place operace, které ale potřebují pomocné registry. Tím se hezky dostáváme k pokročilejší správě registrů. FlatFox má jenom 6 registrů. Pokud chceme z jednoduchých funkcí poskládat větší program, musíme se ujistit, že nám bude 6 registrů stačit. Jednoduché instrukce si vystačí s 2–3 registry, ale co když chceme mít například 10 proměnných? Elegantním řešením by bylo poskládat několik registrů do jednoho.

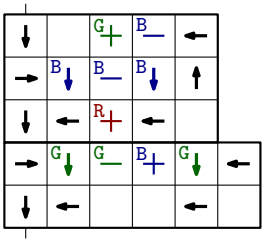
Zásobník

Nejdříve si vyrobme z jednoho registru zásobník. Do zásobníku bude sice možné ukládat pouze čísla od 0 do $n-1$, ale výměnou se nám do něj vejde těchto čísel libovolně mnoho.

V ukázce je `R` prvek, `G` manipulační prostor a `B` samotný zásobník. Číslo n je pevně nakódované do toho, jakou konstantou násobím nebo dělím `B`. Ukázka používá $n = 2$.



Push(R)



Pop(R)

Celý princip je hezky vidět, pokud si představíme, jak bude číslo v B vypadat v soustavě o základu n. Nové prvky jsou pak jen cifry připsané na konec tohoto čísla.

Pozn. red.: Z článku jsme vynechali program, který složí binární zápisy dvou čísel do jednoho registru a pak jej rozloží na původní hodnoty. Najdete ho na webu témátka, stejně jako barevný a přehlednější obrázek Turingova stroje prezentovaného níže.

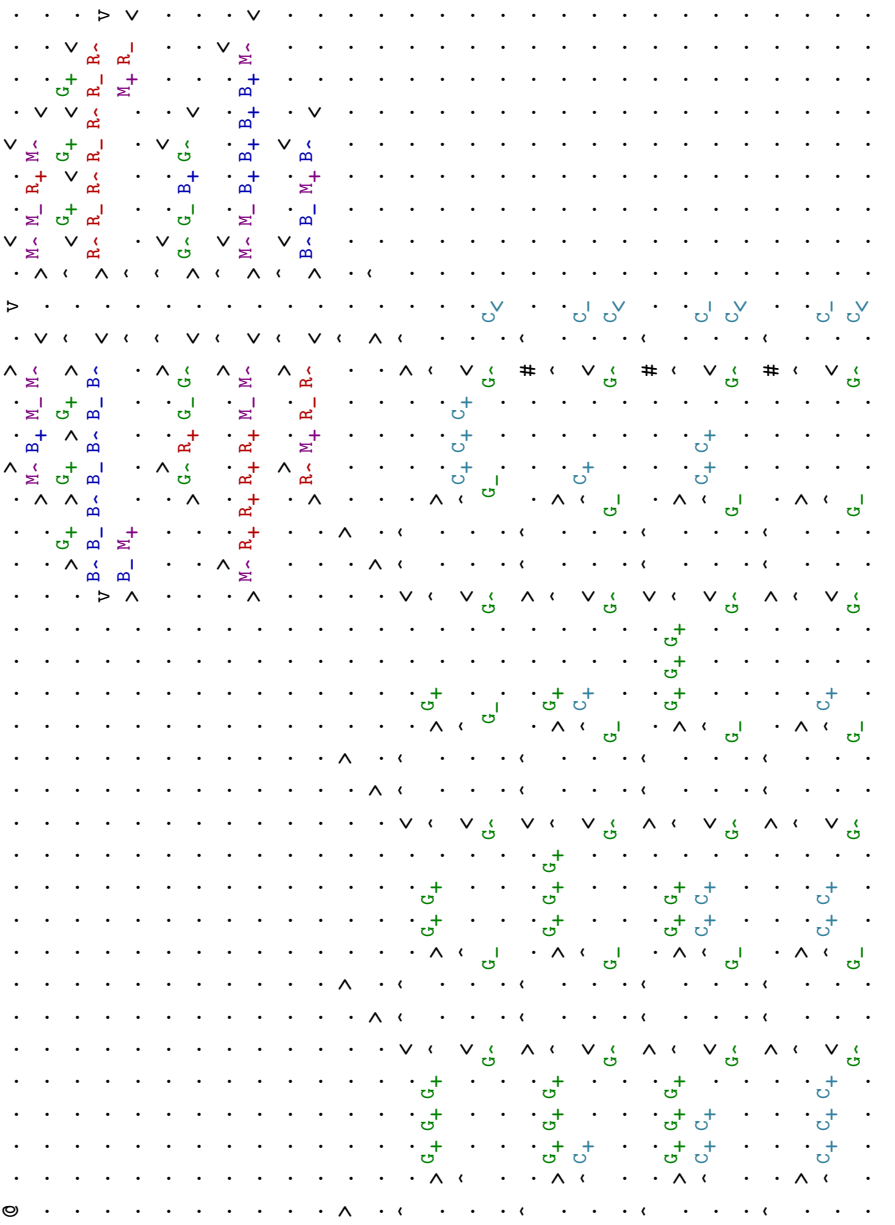
Tímto končí můj první soupis podprogramů ve FlatFoxu. Pro stavbu Turingova stroje máme již všechny potřebné součástky, dejme se tedy do jeho stavby. Pozn.: Uvedené programy nejsou optimalizovány, snažil jsem se spíše o čitelnost.

Turingův stroj

Turingův stroj je teoretický model počítače popsaný Alanem Turingem v roce 1936. Je vybaven nekonečnou páskou rozdělenou na políčka, kde v každém políčku je zapsán jeden znak z nějaké předem zvolené abecedy, která je konečná a obsahuje mezeru. Nad páskou se pohybuje hlava, která umí přečíst znak zapsaný v políčku pod ní a případně jej i přepsat na jiný. Činnost stroje ovládá řídící jednotka, která se vždy nachází v jednom z konečného počtu stavů. Program je kromě abecedy a seznamu stavů definován rozhodovací tabulkou, která pro každou kombinaci stavu a znaku pod hlavou určí, zda a na jaký znak má být políčko pod hlavou přepsáno, zda a kterým směrem se má hlava posunout o jedno políčko a do kterého stavu se má řídící jednotka přepnout.

Navzdory své jednoduchosti je Turingův stroj schopen modelovat jakýkoliv algoritmus. Pokud by se někdo chtěl dovědět o Turingově stroji více, doporučuju přečíst si první část letošního seriálu Korespondenčního Semináře z Programování, která je dostupná na `http://ksp.mff.cuni.cz/tasks/26/tasks1.html#task8`.

Představme si Turingův stroj s abecedou o velikosti a a počtem stavů s. Nejdříve postavíme tu nekonečnou pásku se čtecí hlavou. Pole aktuálně pod čtecí hlavou bude v registru G, páska před hlavou R, páska za hlavou B, kde R a B budou zásobníky s $n = s$. Posun pásky budou zařizovat dva bloky, jeden pro posun dopředu a druhý pro posun dozadu. Posun dopředu provedeme $\texttt{Push}_B\texttt{(G)}$ a pak $\texttt{G=Pop}_R\texttt{()}$, pro posun dozadu zaměnit R a B.



Turingův stroj rozhodující, zda je na vstupu (kódovaném ve 4-kové soustavě) stejně znaků 1 a 2.

A teď rozhodovací tabulka. Nejdříve potřebujeme vědět, jak bude vypadat jedna buňka. Buňka musí být schopná nastavit libovolný stav a zapsat libovolný znak – to určitě umíme v prostoru o straně $O(\sqrt{sa})$ i bez nějakých pokročilejších triků. Zda se má páska posunout doleva, nebo doprava bude určovat, zda buňku opustíme podél její levé, nebo pravé hrany. V tabulce si tedy nejdříve nalezneme

řádek odpovídající aktuálnímu stavu a pak sloupec odpovídající aktuální hodnotě v `G` – tím současně obojí vynulujeme.

Pro ilustraci jsem sestavil Turingův stroj, který kontroluje, zda je na pásce stejný počet 1 a 2. Jo, je to ten nejjednodušší Turingův stroj, který mě napadl, ale myslím, že princip konstrukce libovolného jednopáskového Turingova stroje je zřejmý. Dalším krokem by bylo sestrojení Univerzálního Turingova stroje (Turingova stroje, který na vstupu dostane popis jiného Turingova stroje a vstup, na který má být tento stroj spuštěn), ale vzhledem k nízké rychlosti FlatFoxu je takový stroj neprakticky pomalý i pro triviální vstupy.

Abeceda je $\{0,1,2,3\}$, kde 0 je mezera, 1 a 2 jsou znaky na vstupu a 3 je speciální znak, kterým budeme přepisovat už použité znaky. Stavy jsou $\{0,1,2,3\}$, kde 0 je počáteční stav, ve kterém se čtecí hlava vždy přesune na začátek vstupu. Pak stroj přejde do stavu 3, kdy budeme hledat 1 nebo 2, následně přejdeme do odpovídajícího stavu, nalezneme znak opačný a ve stavu 0 se vrátíme na začátek. Pokud algoritmus nějaký znak nenajde, skončí a `C` bude indikovat, kterého znaku je víc. 0 značí, že znaků bylo stejně.

FlatFox (14b)

Mgr.^{MM} Václav Rozhoň

Pozn. red.: Mgr.^{MM} Václav Rozhoň zaslal příspěvek se součástkami dost podobnými článku Dr.^{MM} Mateje Lieskovského, některé podobné součástky jsme vynechali. Celý článek si můžete přečíst na webu, tam najdete i většinu programů, které se nám svou velikostí už do čísla nevešly.

Základní součástky

Pokusil jsem se o implementaci některých základních flatfoxových operací, v zásadě jsem se držel zadání. Problém u výpočtu funkcí ve flatfoxu je samozřejmě ten, že přičítání jedniček činí všechny postupy hrozně pomalými – pokud chci vypočítat $f(x)$, musím udělat aspoň $|f(x)-x|$ přičtení nebo odečtení jedniček, což je třeba u mocnin dost zdlouhavé.

Všude počítám jen s funkcemi zabírajícími konečně veliký prostor a s konečně velkými registry (doufám, že se nerozcházím se zadáním).

Fibonnaciho čísla. Celkem pěkný příklad na sčítání jsou imho Fibonacciho čísla. Potřebuju tři proměnné, ve kterých budu mít F_n, F_{n+1} a nulu a budu mezi nimi přelévat stylem $(F_n, F_{n+1}, 0) \to (F_{n+2}, 0, F_{n+1}) \to (0, F_{n+2}, F_{n+3}) \to (F_{n+3}, F_{n+4}, 0)$ atd. a mezitím budu snižovat counter. Šlo by to sice jednodušeji jako $(F_n, F_{n+1}, 0) \to (F_{n+2}, 0, F_{n+1}) \to (F_{n+1}, F_{n+2}, 0)$ atd., ale to by bylo o trošku pomalejší.

Záporná čísla. Odčítání (`R-G`) můžeme dělat podobně jako sčítání s tím, že v případě, kdy se R vynuluje dřív než G, do R po skončení cyklu přesuneme zbytek G a pomocnou proměnnou Y nastavíme na 1.

Samozřejmě bychom tu jedničku mohli zkomprimovat jako poslední číslici R a zbytek čísla posunout o 1 doleva, nicméně by to bylo výrazně pomalejší a pracnější – znaménko bychom testovali jako zbytek po dělení dvěma a při nějakém počítání bychom se pak podle znaménka rozhodli, jakou operaci vlastně chceme udělat (sčítání/odčítání) a po jejím provedení zase přilepili znaménko. Nakonec by asi bylo celkem dlouhé rozmyslet všechny možné příklady sčítání/odčítání kladných/záporných čísel.

Největší společný dělitel. Vzhledem k tomu, že umíme pouze odčítat jedničky, budeme prostě odčítat menší číslo od většího, dokud obě nebudou stejná. Na to kromě dvou vstupních proměnných potřebujeme i jednu pomocnou, do které si budeme ukládat odčítané číslo. Navíc ještě musíme vyřešit případy, kdy jedno z čísel bude nula. Asymptoticky to bude trvat $O(a+b) = O(max(a,b))$, což lépe nejde, protože obě čísla musím vynulovat, abych aspoň zjistil jak jsou velká (můj program by šel vylepšit, kdybych vedle sebe postavil několik stejných cyklů jako ten, který používám, jen v jiné barevné permutaci, a pak mezi nimi různě přecházel, než abych po každém cyklu přeléval registr `B` do `R` nebo `G`). Výsledek je v `B`, `R` a `G` jsou vynulovány.

Násobení. Na násobení $a \cdot b$ už potřebujeme čtyři proměnné. Operace vypadá tak, že přičítám a do výsledné proměnné a snižuju counter b při každém cyklu o 1. Abych si ale a zapomatoval, musím si jej současně uložit do nějaké jiné proměnné, mám tedy dvě proměnné mezi kterými přelévám a a dva odpovídající cykly – každý pro jednu proměnnou (na místo by bylo úspornější mít jenom jeden přičítací cyklus, po kterém bych si do první pomocné proměnné nalil a z té druhé a pak cyklus začal znovu, nicméně by to bylo trochu časově náročnější).

Testování prvočíselnosti

Nejlepší mně známý 100% spolehlivý algoritmus na otestování prvočíselnosti čísla N je prostě vzít všechna prvočísla od 2 do $\sqrt{N}$ a vymodulit jimi N. Těchto prvočísel bude asymptoticky něco jako $O(\sqrt{N}/\log N)$, což znamená, že teoreticky nemusíme testovat všechna čísla od 1 do $\sqrt{N}$, takže by složitost byla $O(N\sqrt{N})$, ale třeba bychom mohli udělat nějaké Eratosthenovo síto v čase $O(\sqrt{N} \cdot \log\log N)$ a pak testovat pouze prvočísla; nicméně nevím, jak by se tohle dalo implementovat, takže ve svém programu pouze tupě testuji všechna čísla od 1 do $\sqrt{N}$. Modulení trvá $O(N)$ kroků, protože z N musíme odčítat až na nulu. Asymptotická složitost je pak $O(N \cdot \sqrt{N})$.

Ještě předtím ale musíme vypočíst odmocninu z N. Naštěstí není potřeba nějaký extra přesný nebo extra rychlý odhad, můžu například vzít jedničku, přičítat k ní 1 a dívat se, jestli druhá mocnina není větší než N. Takhle dostanu nejlepší možný celočíselný horní odhad $\sqrt{N}$; v i-tém kroku mě mocnění stojí $O(i^2)$ a porovnávání s N to samé (i^2 je menší nebo cca stejné), celkově mě

tedy nalezení odmocniny stojí

$$O\left(\sum_{i=1}^{\sqrt{N}} i^2\right) = O\left(\frac{1}{6}\left(2\sqrt{N}^3 + 3\sqrt{N}^2 + \sqrt{N}\right)\right) = O(N\sqrt{N}),$$

což nám celkovou složitost nezhorší. Můžeme ale postupovat rychleji, když zase začneme na jedničce, ale pak ji budeme násobit dvěma. Najdeme tak číslo z intervalu $[\sqrt{N}, 2\sqrt{N})$, které je určitě v $O(\sqrt{N})$. Časová složitost je pak

$$O\left(\sum_{i=0}^{\frac{1}{2}\log N} (2^i)^2\right) = O\left(\sum_{i=0}^{\frac{1}{2}\log N} 4^i\right) = O\left(4^{\frac{1}{2}\log N}\right) = O(N).$$

Takhle to v programu dělám já. Pokud bychom pak odhad chtěli ještě upřesnit, mohli bychom dále použít půlení intervalu, což by trvalo řádově $\log N$ kroků, z nich každý by zabral $O(N)$, takže celkem asi tak $O(N \log N)$ času.

V druhé části pak jen modulíme čím dál vyšším číslem (když začneme s menšími čísly, tj. od 2, tak program obecně skončí dřív) a snižujeme counter. Pokud je číslo složené, najdeme ještě navíc jeho nejmenší dělitel (krom 1). Ačkoliv má program složitost $O(N\sqrt{N})$, pokud předpokládáme náhodné číslo, je v průměru o něco rychlejší.

V mojí implementaci odhadnu číslo z intervalu $[\sqrt{N}, 2\sqrt{N})$ (vlevo nahoře) a pak modulím (vpravo dole); na obě části potřebuju pět proměnných.

Návrhy

Přidal bych barvičky – u programu můžeme měřit na kolik barviček jej dáme, ať už máme k dispozici šest nebo deset barviček.

Hodilo by se udělat barvy trochu víc kontrastními – v samotném programu vypadá červená jako růžová, zelená jako tyrkysová a modrá jako šedá (ale nevím, do jaké míry je to jen mým monitorem).

Celkem by se hodilo umět volat funkci, umožní to dělat větší a hezčí programy.

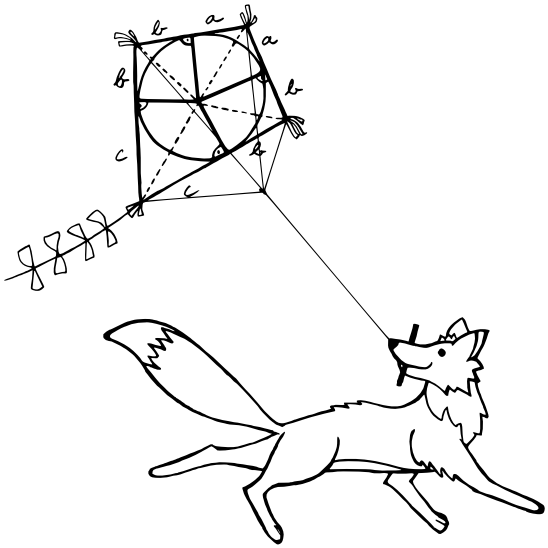
Myslím, že by mohla být užitečná funkce, která robota zastaví do stisku klávesy/kliknutí myši někam. Třeba program na testování prvočísla by se pak snadno předělala na program postupně vypisující jeho dělitele.

Výsledková listina 1. čísla

Poř.	Jméno	R.	$\sum_{-1}$	Úlohy							$\sum_0$	$\sum_1$
				r1	r2	r3	r4	t1	t2	t3		
1.	Mgr.MM D. Krasula	1.	27	3	3	2	3	4	2	10	27	27
2.	Mgr.MM V. Rozhoň	3.	21	3		4				14	21	21
3.	Doc.MM M. Calábková	3.	103	3	3	4			5	5	20	20
4.	Mgr.MM P. Souček	2.	29	3	3				5	7	18	18
5.	Bc.MM O. Hollmann	4.	17	2	3	3			3	6	17	17
6.–7.	Dr.MM M. Lieskovský	4.	83							16	16	16
	Bc.MM L. Studená	4.	16	3	2	2	3		6		16	16
8.	Doc.MM A. Šťastná	4.	103	3			3		6	2	14	14
9.	Dr.MM P. Nácovský	3.	50	3	3	2				5	13	13
10.	Bc.MM V. Bartovic	2.	11	2	1	3	1	4			11	11
11.	Bc.MM A. Teichmann	4.	18	3			3			3	9	9
12.–13.	Mgr.MM J. Kušnír	3.	49	0	2	3	3				8	8
	A. K. Lesna	1.	8		0	2	3	2	1		8	8
14.	T. Paliesek	2.	7	2	3	2					7	7
15.–18.	J. Havelka	1.	6	3					1	2	6	6
	Mgr.MM L. Langerová	3.	42	2					2	2	6	6
	D. Tanglová	1.	6	0	1			5			6	6
	J. Václavek	2.	6	2	2	2					6	6
19.–21.	J. Liška	2.	5	1	3	1					5	5
	J. Nosková	4.	5	0	0	1	1	3			5	5
	A. Šedová	2.	5	2			1		2		5	5
22.–24.	R. Hlavinka	2.	4	1	1	1	1				4	4
	Mgr.MM M. Poljak	2.	38						4		4	4
	J. Stanovský	2.	4	2	1	1					4	4
25.–30.	D. Dimitrov	3.	3	0	3						3	3
	Dr.MM F. Homza	4.	90	3							3	3
	Dr.MM A. Hrušková	4.	59	3							3	3
	K. Kolář	2.	6		3						3	3
	J. Škvára	3.	6		3						3	3
	Dr.MM P. Vincena	3.	61	3							3	3
31.–34.	Bc.MM J. Dittrich	2.	14	2							2	2
	Doc.MM J. Kadlec	3.	100	2							2	2
	Mgr.MM V. Skoupý	4.	44		2						2	2
	Bc.MM M. Šafek	3.	14		2						2	2
35.–37.	Bc.MM K. Ilievová	3.	12	1							1	1
	V. Václavík	4.	9	1							1	1

Poř.	Jméno	R.	$\sum_{-1}$	Úlohy r1	r2	r3	r4	t1	t2	t3	$\sum_0$	$\sum_1$
	F. Zajíc	1.	1	1							1	1
38.–40.	Mgr.[MM] J. Cerman	2.	33	0							0	0
	Bc.[MM] D. Macháčová	4.	16	0							0	0
	M. Müller	4.	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, UK MFF
Ke Karlovu 3
121 16 Praha 2

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

Časopis M&M je zastřešen Oddělením pro vnější vztahy a propagaci Univerzity Karlovy, Matematicko-fyzikální fakulty a vydáván za podpory středočeské pobočky Jednoty českých matematiků a fyziků.