Vyloženě triviální – řešení

Řešení této vskutku netriviální úlohy vyžaduje několik kroků, vyžadujících trochu programátorské zručnosti, trochu teorie a trochu trpělivosti. Jen na okraj poznamenejme, že i takto triviální jazyk je turingovsky úplný (tedy že v něm lze, dle Churchovy teze, vyjádřit libovolný algoritmus).
Prvním krokem je bezpochyby volba vhodného nástroje, protože čísla, se kterými musíme pracovat, jsou dost velká. Téměř každý rozumný programovací jazyk nabízí nějakou formu práce s velkými celými čísly: Python počítá bez omezení rozsahu přímo nativně (stačí použít typ long), Java má BigDecimal, v PHP lze využít knihovnu bcmath a jistě byste něco našli i pro svůj oblíbený jazyk. Jen jde o to, s jakým úsilím...
Mohlo by se zdát, že programovat vlastně vůbec nebude třeba – že bude stačit jen dost přesná kalkulačka. Samozřejmě, můžeme program vyhodnotit buď ručně, nebo si napsat interpret či překladač do jiného jazyka. Tak či tak, řešit musíme stejné problémy.
Jedním problémem je rozhodnutí pro dané číslo n, o jaký příkaz se vlastně jedná. Pohlédnutím na kódování jednotlivých příkazů snadno rozlišíme přiřazovací příkazy od složených: pokud je n dělitelné 2, je to cyklus nebo sekvence; naopak pokud 2 nedělí n, jde o nějaké přiřazení.
Druhý případ je jednoduchý: vyzkoušíme postupně dělitelnost 5 a 7 (příkaz $x_i := x_j - 1$), 3 a 7 (příkaz $x_i := x_i * x_j$) a nakonec pouze 5 (příkaz $x_i := x_i + C$; pozor, že n nemusí být dělitelné 3, protože C může být 0). Faktorizací n (rozložením n na prvočinitele) snadno zjistíme i parametry jednotlivých příkazů.
Zbývá sekvence a cyklus. Ty jsou zapeklité tím, že máme dán předpis funkce t, ale ne její inverze, kterou k vyhodnocení programu potřebujeme. Pro dané k bychom rádi znali p, q takové, že $t(p, q) = k$. Řekněme zatím, že to umíme (k tomu je samozřejmě nutné, aby t byla bijekce, tj. aby inverze byla jednoznačná – dokážeme to úplně na závěr, ale rozhodně to není nutnou součástí úlohy). Pak snadno rozlišíme sekvenci od cyklu: sekvence má $p=0$ a q pak tvoří tělo sekvence. Naopak cyklus má $p>0$ a q tvořící kód těla cyklu.
Pro cyklus určíme faktorizací p parametry (které dvě proměnné se porovnávají), tělo cyklu vyhodnotíme jako samostatný program rekurzivně.
Sekvenci rozbalíme takto: voláme stále inverzi funkce t, dávající vhodné p, q, až do okamžiku, než dostaneme $p = 0$. Pak q z tohoto posledního volání je kódem prvního příkazu sekvence. Hodnota q z předchozího volání je kódem druhého příkazu, atd. až q z prvního volání je kódem posledního příkazu. Vše postupně vyhodnotíme a máme hotovo.
K dokončení už schází jenom efektivní výpočet inverze funkce t, tedy nalezení p, q tak, že $t(p, q) = n$. Řekněme pro začátek, že umíme uhodnout hodnotu součtu $p+q$, označme jej s. Pak je to snadné – přímo z předpisu funkce t dostaneme $p = n - s\cdot(s+1)/2$. Dopočteme $q = s - p$. Zhruba vzato je $t(p, q)$ druhou mocninou s. Mohli bychom tedy zkusit aproximovat od odmocniny z n, ale to bude pomalé. Spočtěme rovnou vzoreček pro výpočet s. Pro dané n bude s největší celé číslo takové, že $s\cdot(s+1)/2 \leq n$ (to je zřejmé z předpisu funkce t – protože $0 \leq p, q \leq s$). Upravme nerovnost: $\frac{s^2}{2}+\frac{s}{2}-n\leq 0$
Spočteme kořeny tohoto polynomu: jsou to $\frac{-1}{2}\pm\sqrt{\frac{1}{4}+2\mathrm{n}}=\frac{-1\pm\sqrt{1+8\mathrm{n}}}{2}$. Z nich uvažujeme jen ten nezáporný, protože víme, že i $s \geq 0$. Ve výsledku tedy hledáme největší celé $s\leq\frac{-1+\sqrt{1+8\mathrm{n}}}{2}$ a to už je snadné: $s=\left[\frac{-1+\sqrt{1+8\mathrm{n}}}{2}\right]$ (dolní celá část uvedeného zlomku).
Máme tedy s, dopočteme p, q. Pro dané n tedy umíme v logaritmickém čase (v čase výpočtu odmocniny) zjistit inverzi $t(n)$. A to bylo to poslední, co nám pro rozkódování programu scházelo.

Abychom učinili korektnosti zadost, ještě ověřme, že t je skutečně bijekce, tj. že žádné dvě dvojice (p_1, q_1), (p_2, q_2) nezobrazí na stejnou hodnotu. Označme $n_1 = t(p_1, q_1)$, $n_2 = t(p_2, q_2)$, $s_1 = p_1 + q_1$, $s_2 = p_2 + q_2$ a rozlišme dva případy:

1) $s_1 = s_2$, pak dle definice t platí, že $n_1 - n_2 = p_1 - p_2$. Pokud $n_1 = n_2$, musí i $p_1 = p_2$, a protože s_1

$= s_2$, pak i $q_1 = q_2$, tedy dvojice jsou nutně stejné.

2) $s_1 \neq s_2$, bez újmy na obecnosti předpokládejme $s_1 < s_2$. Pro spor předpokládejme $n_1 = n_2$ a dosaďme: $\frac{1}{2}s_1 \cdot (s_1+1) + p_1 = \frac{1}{2}s_2 \cdot (s_2+1) + p_2$. Upravme:

$$\frac{s_2}{2}(s_2+1) - \frac{s_1}{2}(s_1+1) = p_1 - p_2$$
$$s_2^2 - s_1^2 + s_2 - s_1 = 2(p_1 - p_2)$$
$$(s_2 - s_1)(s_2 + s_1 + 1) = 2(p_1 - p_2)$$

Avšak ono platí:

$$(s_2 - s_1)(s_2 + s_1 + 1) > 2(p_1 - p_2)$$

protože $p_1 \leq s_1$ a $p_2 \geq 0$, a proto pravá strana má hodnotu nejvýše $2s_1$. Levá strana je větší:

$s_2 + s_1 + 1 > 2s_1$ a zároveň $s_2 - s_1 > 0$ (protože $s_2 > s_1$).

Důkaz, že t je bijekce je tedy hotov.

A nyní, když už umíme rozkódovat zadaný program, ať už ručně nebo programem, dostaneme ekvivalent zapsaný v Pascalovské syntaxi:

```
begin
    x1 := x1 + 1;
    x2 := x2 + 3;
    x4 := x4 + 42;
    x5 := x5 + 5;
    while x4 <> x3 do
        begin
            x1 := x1 * x2;
            x3 := x3 + 1
        end;
    x1 := x1 * x5
end
```

Prohlédneme-li program, zjistíme, že na výstup dá hodnotu $3^{42} \cdot 5 = 547094945657561796045$. Což je i kódem této úlohy. Na čtenáři již ponecháme dovtípit se, kód jakého programu náš zadaný program počítá :-)

Na závěr přikládáme program překládající programy z číselného jazyka do Pascalovské syntaxe. Samozřejmě v Pythonu (jen je třeba implementovat odmocninu nad typem *long* – ta vestavěná počítá pouze s typem *float*).

```
import sys

# funkce pro vytisteni ciselneho programu v Pascalovske syntaxi
def print_program(program, indent):
	# zkusit delitelnost 2 pro rozliseni jednoduchych a slozenych prikazu
	if program % 2 == 0:
		# extrahovat slozeny prikaz
		composite = program / 2
		(head, code) = invt(composite)
		if head == 0:
			# begin ... end
			if code == 0:
				print_indented('begin end', indent)
			else:
				print_indented('begin\n', indent)
				print_seq(code, indent+1)
				print ''
				print_indented('end', indent)
		else:
			# while ... do ...
			f = factorise(head)
```

```
                print_indented('while x%d <> x%d do\n' % (f[2], f[3]), indent)
                print_program(code, indent+1)
        else:
                # nektery z jednoduchych prikazu
                f = factorise(program)
                if 5 in f and 7 in f:
                        print_indented('x%d := x%d - 1' % (f[5], f[7]), indent)
                elif 3 in f and 7 in f:
                        print_indented('x%d := x%d * x%d' % (f[3], f[3], f[7]), indent)
                elif 5 in f:
                        c = f[3] if program % 3 == 0 else 0
                        print_indented('x%d := x%d + %d' % (f[5], f[5], c), indent)
                else:
                        print_indented('** UNKNOWN PROGRAM: %d' % program, indent)

# funkce pro vytisteni sekvence prikazu
def print_seq(seq, indent):
        (i,j) = invt(seq)
        if i != 0:
                print_seq(i, indent)
                print ';'
        print_program(j, indent)

def print_indented(code, indent):
        sys.stdout.write(' ' * 4*indent + code)

# faktorizace cisla n; na vystup da mapovani prvocisel na odpovidajici exponenty
def factorise(n):
        f = {}
        if n <= 0:
                return f
        d = 2
        while n != 1:
                dd = 0
                while n % d == 0:
                        dd += 1
                        n /= d
                if dd > 0:
                        f[d] = dd
                d += 1
        return f

# inverze funkce t
def invt(n):
        s = (-1 + isqrt(1 + 8*n)) // 2
        i = n - (s**2 + s) // 2
        j = s - i
        return (i, j)

# taken from http://www.codecodex.com/wiki/Calculate_an_integer_square_root#Python
def isqrt(n):
    xn = 1
    xn1 = (xn + n/xn)/2
    while abs(xn1 - xn) > 1:
        xn = xn1
        xn1 = (xn + n/xn)/2
    while xn1*xn1 > n:
        xn1 -= 1
    return xn1


program =
48778341323092779256993355044802122645250614765371435359474353067077660136431191144256307
13389897052407370041926715313723307321555823818091581272341278046283359456679574910294553
78142435139220926765482517170455844942889658039010978541537951293525987470285449084417352
38211784891834106932535972145868176909753276413378470018688343400395091264554512085795831
10366995643571807732227507777458374529780060866117453501848710037184069324936296288095345
24837125953359859882375369981664099637163599690501019299633638639458805322345590609255548
44436244984941195769318079868042463157286489391047237888607987554274589295571616883463552
81972238677098362820529517505845236431445658409897214617687293218338476556347363096528843
2869897392L
print_program(program, 0)
```

P2 MI(o)sání *(řešení)*

InterLoS 2012

Zbežný pohľad na zadanie nám navráva, že ak by krabičiek bolo dosť veľa, tak bude na konci (keď už nebude žiaden možný ťah) rozmiestnenie cukríkov v miskách zodpovedať binárnemu zápisu n. Ak je krabičiek menej, zvyšné cukríky ostanú v poslednej krabičke, odkiaľ sa už nemôžu pohnúť.

Keďže každý cukrík, ktorý nie je v krabičkách na konci Norbert zjedol, stačí nám od počiatočného počtu odpočítať počet cukríkov v krabičkách na konci a máme hľadaný výsledok.

správné heslo: 6348394782344010705755343481200723473709 4

```
int main() {
    long long n,k;
    for (int q = 0; q < 5; q++){
        cin >> n >> k;
        long long vysledok = 0;
        long long n2 = n;
        for (int i = 0; i < k-1; i++) {
            vysledok += (n%2);
            n /= 2;
        }
        vysledok += n;
        cout << (n - vysledok) << " ";
    }
    cout << endl;
}
```

P3 Sudoku netradičně *(řešení)*

InterLoS 2012

Úlohu možno naprogramovat technikou *brute-force*, tj. program bude postupne zkoušet všachna možná rozmístnění čísel a pro každé ověří, jestli je platným řešením. Tuto metodu je možné vylepšit mírným prořezáváním a nezkoušet řešení, kde se oapkují čísla a pod.

správné heslo: 72

L1 Interlosí domino *(řešení)*

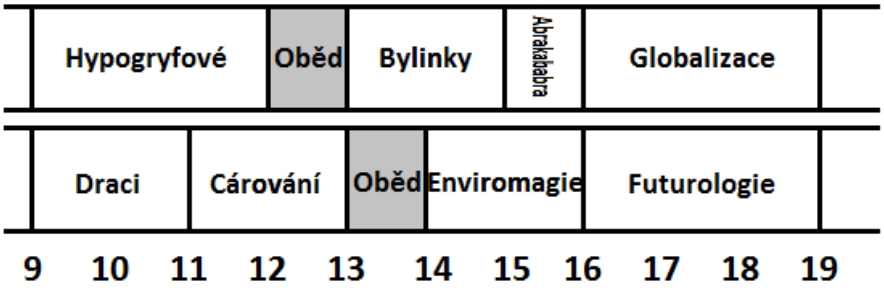
InterLoS 2012

O	O	R	S	S	I	S	O	L
L	L	N	R	T	L	T	T	S
R	E	E	R	I	R	S	T	L
I	L	I	E	I	O	O	S	S
E	N	N	E	O	E	N	N	I
O	S	T	T	O	R	T	N	L
N	N	N	E	I	R	E	R	L
I	T	L	R	T	O	I	E	S

správné heslo: SSVSSSSSVS

L2 Hermionin rozvrh *(řešení)*

InterLoS 2012

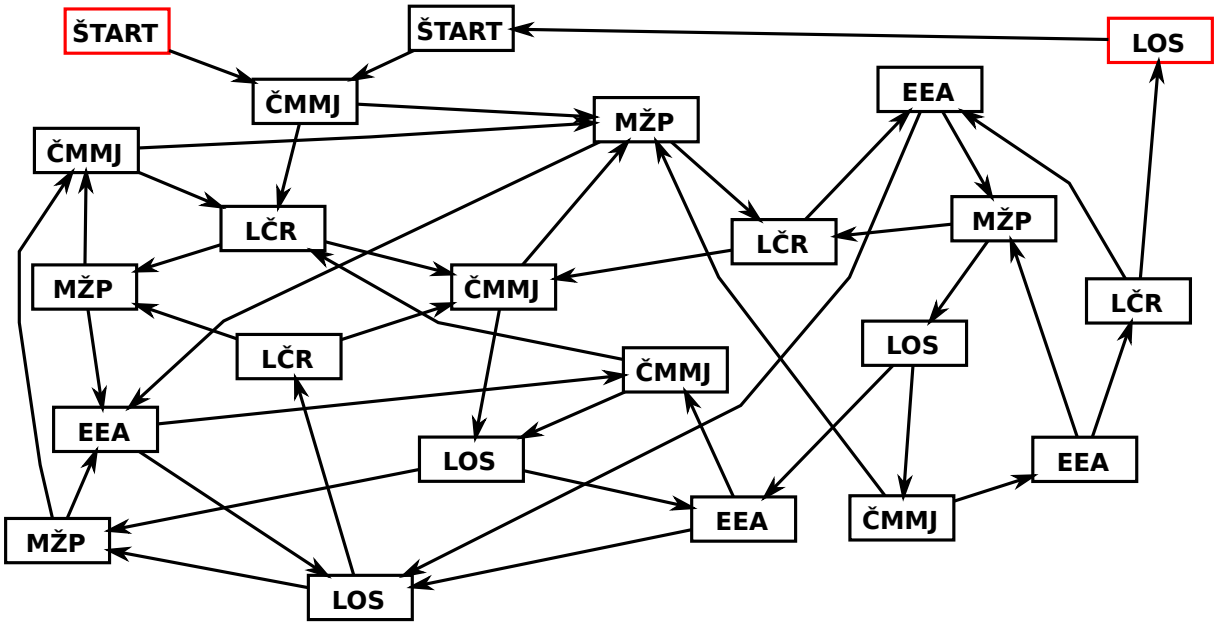


správné heslo: A15B13C11D9E14F16G16H9

L3 Povolení na lov losů *(řešení)*

InterLoS 2012

Stav v tomto „byrokratickém bludišti“ je určen kombinací organizace, kde se právě nacházíte a organizace, kde jste se nacházeli v předešlém kroce. Plán tohoto „bludiště“ se nachází níže. Počáteční a koncový stav jsou v vyznačeny červeně. Na posledním stanovišti se dozvíte, že heslo úlohy je POVOLENIZABIJET. Pro zajímavost poznamenáváme, že v původní verzi úlohy byla řešením nejkratší cesta tímto bludištěm :-).



správné heslo: POVOLENIZABIJET

S1 Ví bůch *(řešení)*

InterLoS 2012

V textu šifry se vyskytuje nápadně mnoho přídavných jmen v 1. nebo 4. pádě mužského rodu. Jejich koncovky odpovídají koncovkám názvů chemických sloučenin (ný, natý, itý, ičitý, ičný/ečný, ový, istý, ičelý). Přiřadíme jim hodnotu (oxidačního čísla) a v rámci vět sečteme. Převedeme na písmeno dle pořadí v abecedě.

správné heslo: ZVIRATKO

S2 Čítanka *(řešení)*

InterLoS 2012

Šest nahrávek obsahuje úryvky z šesti různých příběhů, postavy jsou však cyklicky posunuté. Tedy např. v příběhu o Adamovi a Evě vystupuje Romeo a Julie, v příběhu o Romeovi a Julii vystupuje Harry Potter a Ginny, v Harry Potterovy vystupuje Evžen Oněgin a Taťána atd. Dohoromady tedy tvoří nahrávky cyklus, kde následnictví je určeno tím, ze kterého příběhu jsou postavy, které v současném příběhu vystupují.

Kdo však poslouchá pozorně, určitě zvládne určit i počátek a konec posloupnosti nahrávek. Začíná se příběhem o Adamovi a Evě a končí Zločinem a trestem (Adam a Eva - jako počátek lidstva, narážka na „začínání“ se vyskytuje i v jejich dialogu, také se dovídáme, že "to končí vraždou", a dialog v Zločinu a trestu končí slovy "To bude konec").

U všech úryvků začíná poslední promluva slovy "Někdo něco ŘEKL/ŘÍKÁ". To, co říká je obzvláště důležité a pokud máme posloupnost příběhů seřazenou správně, tak jsou to kroky, které máme provést, abychom dospěli k cíli:

1. **setřídit si to**
   (Tedy seřadit si ukázky podle návaznosti určené migrací postav.)

2. **vědět naprosto jistě, že začátek a konec jsou na svém místě**
   (Upozornění na to, že je důležité, kde je začátek a kde konec posloupnosti příběhů.)

3. **tvoje jméno je pojem - a proto tě lidi budou pořád vyhledávat**
   (Jméno souboru je důležité - k řešení vede i to, že ho zkopírujete a necháte si ho vyhledat třeba Googlem. (I když existuje ještě o trošku přímější cesta - pokud posloucháte nahrávky pozorně, určitě vás napadne, že se názvy souborů odkazují na videa na YouTubu – pak stačí vložit název videa jako parametr do standardní YouTube adrese).

4. **čas si přesně nastavte**
   (Nastavit čas ve videu podle času ukázky. K tomu navádí několik narážek v rozhovorech, z nichž patrně nejvýraznější je přímé zmínění délky druhé nahrávky - „na konci 64. vteřiny“).

5. **to, co vidíš, si piš**
   (Sepsat to, co vidí ve videu v nastavený čas. Pokaždé je to jedna číslice, a to (v tomto pořadí): 1, 1, 2, 3, 5, 8)

6. **...že mi to pořádně spočítájí. Víš, co bude následovat.**
   (Následovat bude číslo 13 - pro uvedenou posloupnost platí, že každé číslo získáme jako součet předchozích dvou (Fibonacciho posloupnost). Odpověď tedy spočítáme jako $5 + 8 = 13$.)

S2 Čítanka *(řešení)*

InterLoS 2012

Pro všechny uvedené kroky existuje v dialozích hromada různých náznaků. Pokud si díky nim řešitelé uvědomí, že když nestačí zvuk, musí vidět video a spojí si to s tím, že mají použít internet a že názvy jsou důležité, mohlo by je přímo napadnout, že mají jméno souboru dosadit do částečné adresy společné všem videím na YouTubu (`http://www.youtube.com/watch?v=`) a vyhledat ona videa. Nicméně i pokud název dají do Googlu, tak jim to jako první odkaz příslušné video vyhledá.

Náznaky v nahrávkách:

1.
   - použití internetu je nutné pro řešení
   - byla by chyba pouze poslouchat
   - "Mě by jen zajímalo, JAK DLOUHO TO POTRVÁ"- důležitost délky nahrávek
   - "přemýšlet o tom, co přijde dál"- následující prvek posloupnosti
   - setřídit si to
2.
   - Na konci 64. vteřiny slunce vysvitne. - důležitost délky nahrávek (1:04 je délka této nahrávky)
   - Své jméno nezapírej, neb je nesmírné ceny – důležitost názvu nahrávek
   - Vědět naprosto jistě, že začátek a konec jsou na svém místě – je důležité, které video je první v řadě (a které naopak poslední)
3.
   - Minulý týden napadli 2 osoby a tenhle týden už dokonce 4. Kolik lidí to asi schytá příští týden - (hledání dalšího prvku posloupnosti)
   - Počet obětí roste geometrickou řadou – Fibonacciho posloupnost je ve skutečnosti geometrická (i když uznávám, že to na první pohled není vidět)
   - Měli by otevřít oči (zase: nestačí jenom zvuk)
   - králík (Fibonacci)
   - tvoje jméno je pojem - a proto tě lidi budou pořád vyhledávat
4.
   - Na to, co chci vám sdělit, nestačí slova (zase: nestačí jenom zvuk)
   - kde a kdy přesně (je třeba nastavit přesný čas ve videích)
   - v udaný čas, přesně na vteřinu
   - mé oči Vás vidět musí
   - čas si přesně nastavte

S2 Čítanka *(řešení)*

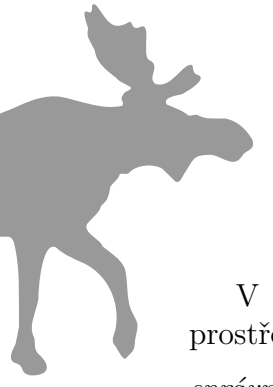
InterLoS 2012

5.
- Uši ti nestačí, musíš použít i zrak
- Uvidíš tajemných číslic stín
- měřil čas
- to co vidíš si piš

6.
- umí si dát dvě a dvě dohromady (sčítání dvou čísel)
- že mi to pořádně spočítají
- víš, co bude následovat

Na Fibonacciho posloupnost navádí i to, že samy nahrávky tvoří posloupnost (první krok řešení).

správné heslo: TRINACT



S3 Sousedi *(řešení)*

InterLoS 2012

V každém slově je trojice písmen, která jsou v abecedě za sebou. Heslo je tvořeno prostředními písmeny z trojic.

správné heslo: OSTEN

P6 Prolomení hesla *(řešení)*

InterLoS 2012

Ryze programovací úloha. Samozřejmě nebylo možné zkoušet všechny možné klíče, kterými byl text zašifrován, to by trvalo dost dlouho. Důležité je, že klíč se zahašuje a šifruje se již pouze hašem. Takže nebudeme tipovat klíče, ale přímo haše. I tak ale zkusit všech 256^8 možných hašů nelze stihnout. Jistěže to ale není nutné, protože reálná entropie není rozhodně tak velká. Podívejme se, jak se haš tvoří.

Začněme od poslední fáze hašování. Reálně se zde vezmou 4 byty ($r0$ až $r3$) a naplní se z nich 8 bytů. Že se některé části ještě rotují nehraje roli, důležité je, že veškerou informaci už nesou původní byty $r0$ až $r3$. Zkoušet by tak stačilo pouze tyto byty. 256^4 je ale pořád dost, takže jedeme dál.

Druhá fáze vezme byty $k0$ až $k3$ a zrotuje je o nějaká čísla. Smysl má ale uvažovat rotaci nejvýše o 7 míst, protože rotace o 8 je identita, rotace o 9 je stejná jako rotace o 1 atd. Budeme-li tedy znát hodnoty $k0$ až $k3$, máme jen 8^4 možností, jak z nich vyrobit byty $r0$ až $r3$.

Byty $k0$ až $k3$ sice neznáme, ale prohlédneme-li, jak jsou tvořeny, zjistíme, že rozhodně nemohou nabývat lecjakých hodnot...

Byte $k0$ je součtem znaků klíče, s tím mnoho neuděláme. Byte $k1$ je ale `xor` několika čísel, z nichž každé je osmou mocninou nějakého znaku klíče. Důležité je, že aritmetické operace probíhají pouze na hodnotách 0 až 255. Podíváme-li se na binární zápis osmých mocnin čísel, zjistíme, že 4. – 7. bit je vždy 0, a tak nám zbývají pouze 4 bity informace, které je třeba vyzkoušet.

Byte $k2$ je zcela závislý na klíči a hodnoty můžou být ledajaké. Byte $k3$ je ale definován podobně. Když definici prohlédneme, zjistíme, že $k3$ má stejnou hodnotu jako $k3$, jen první a druhá čtveřice bitů je naopak.

Celkově tak stačí zkoušet hodnoty pro byty $k0$, $k1$ a $k2$, přitom pro $k1$ stačí hádat pouze 1., 2., 3. a 8. bit. Zbytek už je daný. Máme tak $2^8 \cdot 2^4 \cdot 2^8 = 1048576$ možností.

Napíšeme tedy program, který zkouší různé skutečně možné hodnoty pro byty $k0$ až $k3$ (to je 1048576 možností), každou možnost ještě různě orotuje, a pokusí se text s výsledným klíčem rozšifrovat a zkontrolovat na výskyt podřetězce *hashovaci*. Spoustu práce si ale ještě můžeme ošetřit cachováním již vyzkoušených hodnot – z rotací dostaneme spoustu možností, které se zkusily již pro dřívější iterace.

Jediný detail je, jak s daným hašem text rozšifrovat – jednoduše provedeme inverzně to, co se děje při šifrování.

Rozšifrovaný text je: *jmenoviteznehohashovacihoalgoritmuvyhlasenehovnedavnoukoncenepetiletesoutezijakoshatri*

Stačí vygooglit, že v říjnu tohoto roku byl vyhlášen vítěz soutěže o hašovací algoritmus SHA-3.

správné heslo: KECCAK

P4 ReMorse *(řešení)*

InterLoS 2012

Jednalo se o poměrně jednoduchou programátorskou úlohu. Myšlenku možného řešení ilustruje následující program:

```
file = fopen('data_morse.txt');
fid = fopen('new_morse.txt','w');

tic;
while ~feof(file)
    curr = fscanf(file,'%c',1);
    if strcmp(curr,'.')
        fprintf(fid, '.');
    else curr = fscanf(file,'%c',1);
        curr = fscanf(file,'%c',1);
        curr = fscanf(file,'%c',1);
        if strcmp(curr,'.')
            fprintf(fid, '-');
        else
            fprintf(fid, '/');
        end
    end
    while curr ~= '/'
        curr = fscanf(file,'%c',1);
    end
end

fclose(file);
fclose(fid);
toc;
```

V morzeovce je uložen následící věta z filmu Pelíšky: ..."KDE UDELALI SOUDRUZI Z NDR CHYBU?"

správné heslo: PELISKY

P5 Losí komprese *(řešení)*

InterLoS 2012

Nejdřívě uděláme následující pozorování:

1. Pokud jsou všechny znaky v řetězci stejné, nelze provést žádnou operaci.
2. Pokud jsou v řetězci dva stejné znaky vedle sebe (a alespoň z jedné strany je jiný znak), můžeme je „odstranit“. Např. $\ldots LLO \ldots \rightarrow \ldots LS \ldots \rightarrow \ldots O \ldots$

Pomocí pozorování 2 můžeme odstanit libovolnou skupinu stejných znaků u sebe (odstraňujeme dvojice až zbyde jen 1 nebo žádný). Pokud v žádném místě nejsou dva znaky u sebe, provedeme operaci na prvních dvou znacích a odstraníme dvojici stejných znaků, pokud touto operací vznikla. Tímto způsobem můžeme řetězec zkracovat, než nemá délku 1 nebo 2(2 stejné znaky). Pokud tedy na začáku nejsou všechny znaky stejné, je řešením 1 nebo 2. Zbývá rozhodnout, která z těchto možností nastane.

Pozorování 3: Libovolná operace měni paritu počtu každého znaku. Na příklad, pokud je na začátku počet znaků L, O, S (sudý, sudý, lichý), bude jejich počet v dalším kroku (lichý, lichý, sudý) atd. Parita všech nikdy stejná a nemůžeme proto skončit s řetězcem délky 2 (tam jsou totiž počty 0, 0, 2, tedy všechny sudé). Naopak, pokud je na začátku parita všech stejná, bude pořád stejná a nemůžeme proto skončit s řetězcem délky 1. Stačí tedy spočítat počet znaků L, O, S na začátku - pokud mají stejnou paritu, je odpověď 2, jinak 1.

správné heslo: 14722111212

```
def LOS(s):
        a = [s.count(x) for x in "LOS"]
        if a.count(0)==2:
                return sum(a)
        return 3 - len({x%2 for x in a})
```

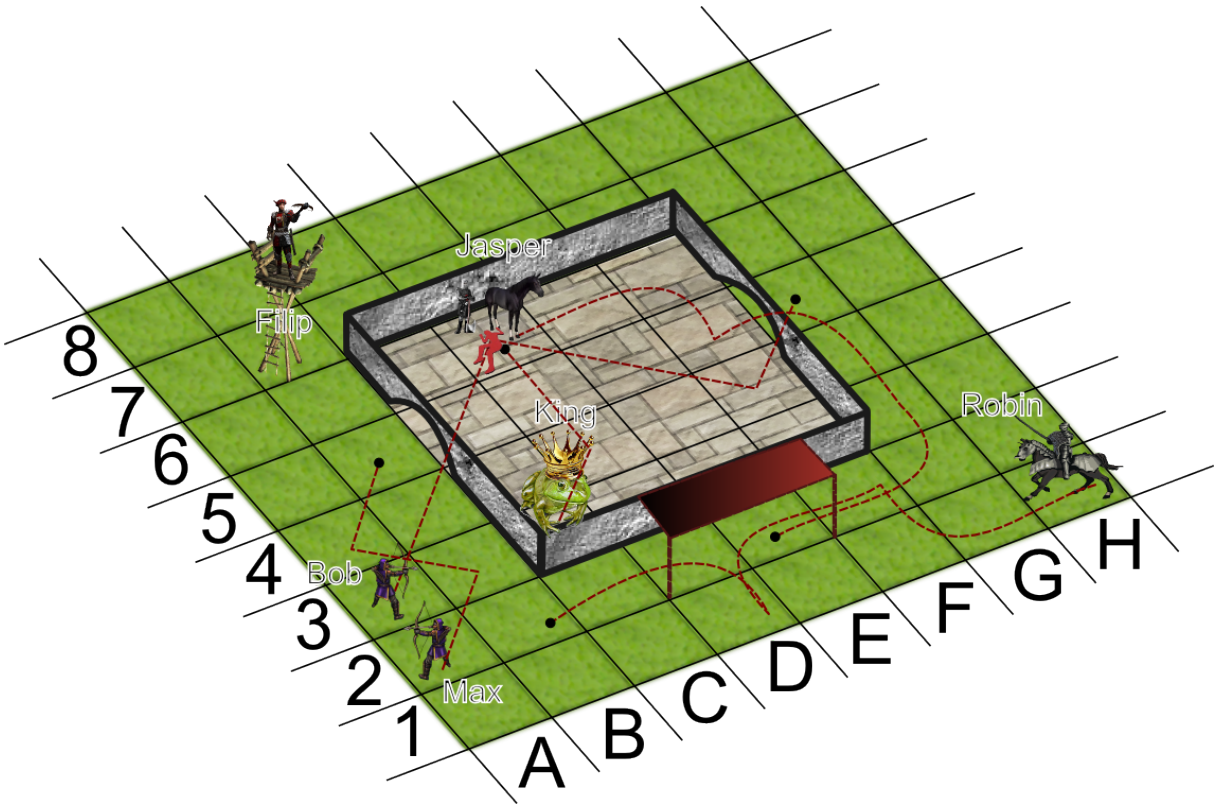
L4 Záhada v královském paláci
(řešení)

InterLoS 2012

Každá postava znázorňuje šachovou figurky a pohybuje se stejně jako ona. Filip je věž, Max A Bob jsou střelci, Jasper a Robin jezdci a King je král.

Max královnu zabít nemohl, protože se na její pole jako černopolný střelec nikdy nedostane. Dává navíc alibi Filipovi, který jen stál a koukal (tj. pozice Filipa je B7). Jelikož je Filip nevinný, říká pravdu, tedy že Max šel stále k bráně. Jako střelec tedy musela být jeho trasa A2-B3-A4-B5. Stejně tak je nevinný Robin, protože jako jezdec by to během tří tahů nestihl. Podle Filipa se vynořil zpoza rohu, proto musel stát před tím na D1. Nejbližší roh, z kterého se dá vyběhnout, je pak H1. Podle Filipovi výpovědi je nevinný i King, protože vrah uprchl bránou a King byl po třech tazích v paláci. Vrahem je tedy Jasper, který se jako jezdec mohl tahy D6-F5-G3-E2 dostat z místa činu až do stájí. Rohy, kde krále královna neohrožuje, jsou 33 a 63. Minimální prostor pro pohyb je ale jen v rohu C3. A Bob stál vedle Maxe (A2) a jediným tahem prošel branou. Proto musel stát na poli A3.

správné heslo: A3B7D6C3A2H1



L5 Logická řada *(řešení)*

InterLoS 2012

Je potřeba doplnit pátou trojici písmen. Ve skutečnosti se jedná o tři podřady, tvořené levými / prostředními / pravými písmeny v rámci trojice.

Levá podřada:

Vzdálenosti písmen v abecedě tvoří posloupnost +2, -4, +6. Dalčí člen je od písmena T vzdáleno -8, jedná se tedy o písmeno L.

Prostřední podřada:

Jedná se o abecedně seřazené samohlásky A, E, I, O. Je tedy potřeba doplnit U.

Pravá podřada:

Vzdálenosti písmen v abecedě jsou konstantně -6. Další znak je tedy H.

správné heslo: LUH

L6 Prvočíselné sudoku *(řešení)*

InterLoS 2012

15 7	5	19 2	17	19	26 3	23	43 11	13
3	43 11	13	43 23	2	5	27 17	7	19
25 23	19	32 17	7	13	39 11	2	3	16 5
2	62 13	3	5	46 23	19	7	37 17	11
17	23	19	2	11	7	5	13	3
16 5	7	66 11	13	3	52 17	19	2	40 23
11	26 2	23	19	38 7	13	3	35 5	17
35 13	17	7	3	5	23	11	19	32 2
19	3	16 5	11	17	15 2	13	23	7

správné heslo: 232513131723195137273523

S4 Použij nůžky! *(řešení)*

InterLoS 2012

Stříháme podle čárkovaných čar. Podle plných čar pak proužky ohneme do pravých úhlů. Tři části losa by pak při správném pohledu měly splynout. Podobně také rozházenáná písmena se spojí v řádky. Čtením z obou kolmých směrů pak postupně dostáváme: 1. řádek: KAM PRO KRESLO 2. řádek: EKTORP TULLSTA

Otázka tedy je, kam pro křeslo *Ektorp Tullsta*. Toto křeslo zakoupíte jedině v IKEI, heslo je tedy IKEA.

správné heslo: IKEA

S5 Nie som, čo som *(řešení)*

InterLoS 2012

Správnou odpovědí byla fráze „GIFAR“ související s principem šifry. Pro zvídavé doporučujeme http://en.wikipedia.org/wiki/Gifar. A jak k této odpovědi dojít?

Nejprve použijeme nápovědu z nadpisu a obrázku zdrhovadla (lidově „zip“) a zkusíme přejmenovat na archiv `01.zip`. Takto vzniklý archiv rozbalujeme dokud to jde. Dostaneme archivy `02.zip`, `03.zip`, `04.zip` a `05.zip`.

Z archivu `05.zip` získáme obrázky, které nám napoví skutečný formát příslušných formátů:

- Obrázek `01.jpg` zachycoval logo formátu HTML5. Po přejmenování `01.zip` na `01.html` a otevření nám velký text prozradí, že první písmeno je **G**.
- Obrázek `02.jpg` zobrazoval sluchátka, což mělo naznačit nejčastější hudební formát. Po přejmenování `02.zip` na `02.mp3` a poslechnutí zjistíme, že další písmenko je **I**.
- Obrázek `03.jpg` ukazoval paňďuláčka docenta, z čehož je zřejmé, že po přejmenování `03.zip` na `03.doc` a otevření uvidíte losa a větu s dalším písmenem – **F**.
- Obrázek `04.jpg` vystavoval světu ikonu, přejmenováním `04.zip` na `04.ico` a otevřením v příslušném editoru jste jednoduše zjistili další písmenko – tedy **A**.
- Obrázek poslední, `05.jpg` ukazoval ruskou (rekurzivní) hračku – matrošku. Po krátkém zapátrání mezi formáty bylo jasné, že správný název archivu `05.zip` je `05.mkv`, po shlédnutí videa v editoru zjistíte, že poslední písmenko je **R**.

S6 Co chybí? *(řešení)*

InterLoS 2012

V zadání je 9 skupin písmen. Všechny připomínají názvy slovních druhů (nejvíce asi tatne jmen, catice...). Je potřeba doplnit písmena, která chybí. Jsou to písmena a skupiny písmen, která představují předložky. Žádný řádek však neodpovídá předložkám. Pro vytvoření hesla je nutné použít slovo „předložka“, které se transformuje stejným způsobem, jakým jsou transformovány ostatní slovní druhy v zadání. PRED**L**OZK**A** → LA.

správné heslo: LA

P7 Závody pixelů *(řešení)*

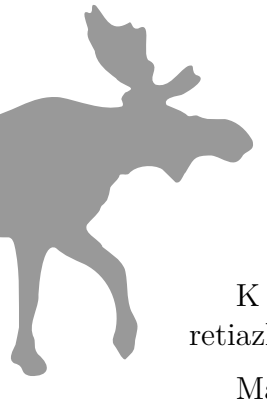
InterLoS 2012

Nejdříve je potřeba zkonstruovat si danou permutaci pixelů. Simulujeme tedy celý závod od začátku, čímž zjistíme pořadí závodníků v cíli. To lze udělat například následovně (ukázka v jazyce C#):

```
const int MaratonLength = 42195;
const int DiceFirst = 42;
const int DiceModulus = 9876553;
var vcili = new bool[Size];
var ubehl = new int[Size];
var cil = new List<int>();
var a = (Int64)DiceFirst;
while (cil.Count < Size)
{
    for (var i = 0; i < Size; i++)
    {
        if (vcili[i]) continue;
        ubehl[i] += (int)((a % 100) + 1);
        a = (a * a) % DiceModulus;
        if (ubehl[i] >= MaratonLength)
        {
            vcili[i] = true;
            cil.Add(i);
        }
    }
}
return cil.ToArray();
```

Poté stačí popřeházet posledních 25600 bytů v daném BMP souboru.

správné heslo: KRTEK



P8 Retiazka *(řešení)*

InterLoS 2012

K riešeniu tohto príkladu je najlepšie pristupovať z opačného smeru - akú najdlhšiu retiazku vieme rozdeliť vybratím k článkov tak, aby šli vyskladať retiazky dĺžky 1 až n?

Máme teda nazačiatku k jednočlánkových retiazok. Najkratšia časť rozdelenej retiazky potom musí mať $k+1$ článkov – všetky dĺžky do k vieme vyskladať z jednočlánkových retiazok a $k+1$ už nevieme. Druhý najkratší kúsok potom bude mať $(k+k+1)+1$ článkov, a takto postupne až kým nebudeme mať $k+1$ kúskov (nepočítame jednočlánkové). Po chvíľke rátania sa dostaneme k tomu, že reťaz môže mať najviac $(k+1)\cdot 2^{(k+1)}-1$ článkov.

Keďže vidíme, že samotné k je oproti vstupu veľmi malé, stačí nám skúšať k od 1, až kým netrafíme také, že najväčšia dĺžka retiazky pre toto k je väčsia alebo rovná nášmu zadanému n.

správné heslo: 261517209

P9 Slovní mutace *(řešení)*

InterLoS 2012

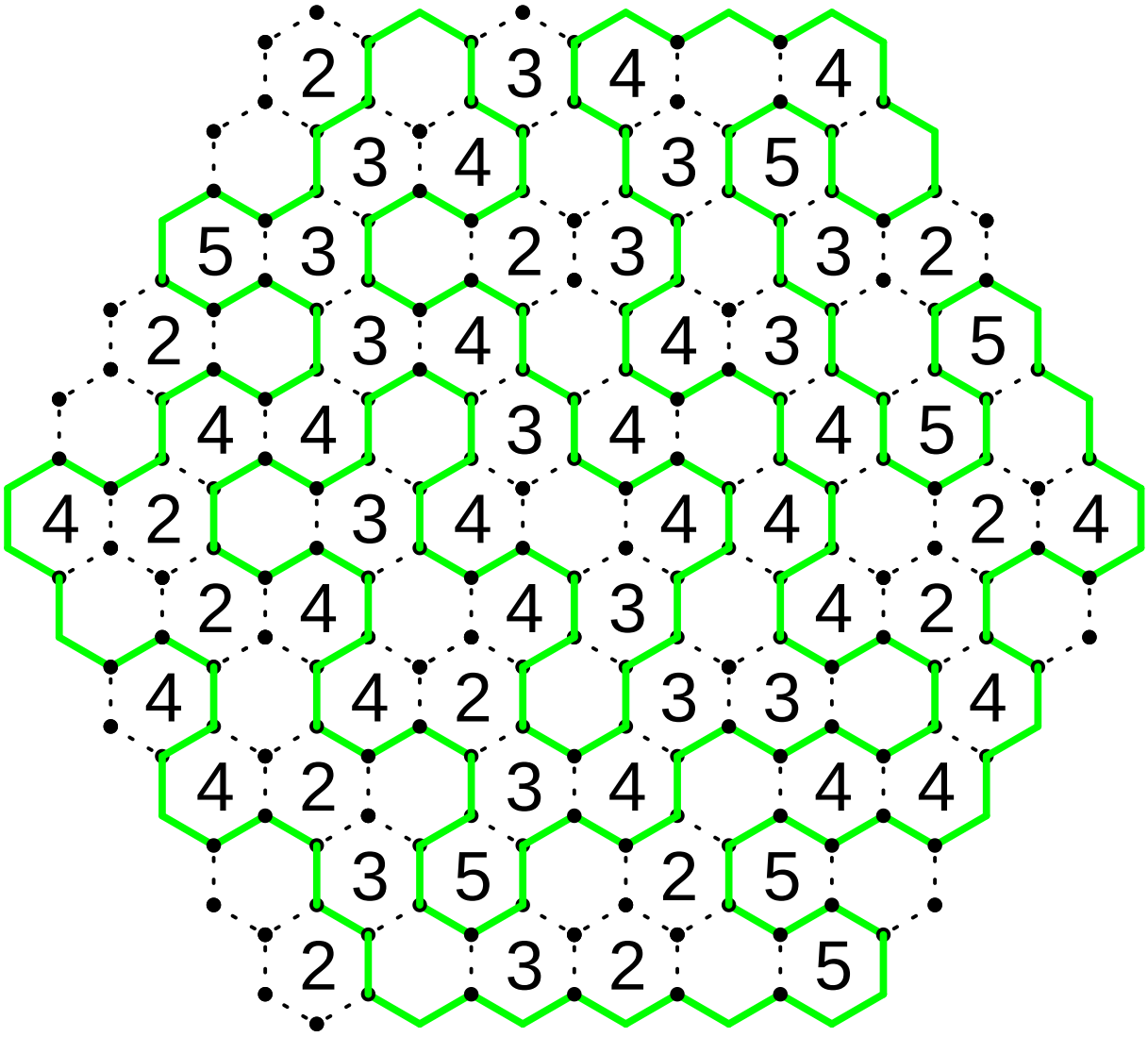
Správné řešení mohlo například prohledávat strom slov do šířky. V kořeni tohoto stromu bylo počáteční slovo, hrany vedli ke slovům vzniklých korektními úpravami.

Ukázkový zdrojový kód pro *MatLab* najdete v esterních souborech *mutace1.m* a *mutace2.m*.

správné heslo: PPPPPPPTT

L7 Ohrádka pro losátka *(řešení)*

InterLoS 2012



správné heslo: 4333553533

L8 Počítáme *(řešení)*

InterLoS 2012

Dámy a pánové, začneme něčím docela jednoduchým. Tahle jedna samotná věta má ve svých slovech celkem **jedenact** éček. Čtvrtá věta tohoto odstavce má celkový počet téček o **šest** menší než tahle. Postupujme dál, tahle skvělá věta má sama o sobě **deset** áček a právě **sedm** esek. Součet výskytů písmene es v této větě a ve druhé větě tohoto odstavce může být maximálně číslo **deset**. V následující, s Mississippi nesouvisející, větě je **deset** péček. V předchozí, s populární Mississippi popravdě právě stejně pramálo provázané, větě je **devět** esek. V celém odstavci se pak vyskytuje **deset** slov, která začínají déčkem.

správné heslo: JEDENACTSESTDESETSEDMDESETDESETDEVETDESET

POZNÁMKA: *Jak si několik týmů všimlo, úloha má bohužel i další řešení.*

L9 Krychle krychlí *(řešení)*

InterLoS 2012

Text na krychli zní:

P O S L E T E N A M Z P R A V U K R O K O D Y L

správné heslo: KROKODYL

S7 Vlnění *(řešení)*

InterLoS 2012

V řetězci se vyskytuje právě jednou číslo 2 → B, dvakrát 25 → Y, třikrát 12 → L, čtyřikrát 15 → O, pětkrát 26 → Z, šestkrát 18 → R, sedmkrát 1 → A, osmkrát 22 → V, devětkrát 5 → R, desetkrát 3 → C. Jde tedy o to čísla seřadit podle četnosti výskytu a v tomto pořadí je převést na písmena podle pořadí v abecedě.

správné heslo: BYLOZRAVEC.

S8 Rozměrná *(řešení)*

InterLoS 2012

Řada má celkem 64 znaků. Z toho je 40 nul a jedniček - dohromady 8 pětic (dle první nápovědy $0 + 1 = 5 \cdot 8$). Každou pětici 0 a 1 přepepočítáme na číslo a pak odpovídající písmeno, dostáváme „První je T“. Máme tedy první písmeno. Z druhé nápovědy $0 + 1 + 2 + 3 = 8 \cdot 8$ je zřejmé, že se dostáváme do druhého rozměru. Naskládáme si všechny číslicie do mřížky 8×8, dvojky pak vytvoří písmena R a I. Ještě nám zbývá druhá část druhé nápovědy $0 + 1 + 2 + 3 = 4 \cdot 4 \cdot 4$ a dostáveme se tedy do třetího rozměru - vyskládáme si všechny číslice do krychle o velikosti 4. V předchozích fázích jsme ještě nečetli číslici 3, tak se na ně zaměřme. Zjistíme, že v krychli je napříč z trojek vyskládáno písmeno O. Postupně tedy dostáváme T, R, I a O.

správné heslo: TRIO.

S9 Budiž světlo! *(řešení)*

InterLoS 2012

Poslední šifrovací úloha byla spíš pro odlehčení. Většina z vás snad vytušila, že v rozsvěcování světýlek nejste sami. Každý tým měl přiděleny nějaké body, které po stisku svého tlačítka rozsvítil. Dokud jste nerozsvítili, nic jste neviděli, ale po stisku tlačítka se vám ukázaly všechny body, které už někdo rozsvítil (včetně těch vašich). Z rozsvícených bodů se postupně vykreslovalo písmeno, a pokud vás bylo dost, bylo zřetelně čitelné. Ale jen na minutu a pak vše znovu zhaslo! Protože se začalo vykreslovat další písmeno hesla. A opět, dokud jste sami nerozsvítili, nic jste neviděli. Postupně se takto opakovaně kreslilo heslo: `OREZAVATKO`. Nějaká světýlka s postupem času vykresloval samotný systém, takže i když jste se k šifře dostali později a neřešilo ji tou dobou už tolik týmů, pořád byla řešitelná.

správné heslo: OREZAVATKO