Korepondenčný seminár z programovania
XIX. ročník, 2001/2002
Katedra vyučovania informatiky FMFI UK,
Mlynská Dolina, 842 48 Bratislava

KSP finančne podporuje nadácia Open Society Fund Bratislava a
IUVENTA – zariadenie pre vožný čas detí, mláde« e a dospelých
MICROSTEP spol s.r.o.

Vzorové riešenia 1. kola letnej časti

Milí kovboji,

Prišla jar. Preto vytiahnite svoje oceľové tátoše zo stajne a hor sa bicyklovať. „Kto sa nebicykluje, nech ani neje“, tak znie heslo moderných kovbojov. A komu sa lení alebo nechová doma žiadneho tátoša, nech si sadne doma na zadok a prečíta si vzoráky KSP. Nech sa potom ale nečuduje, že dostane z toľkého štúdia vlka či očibôľ.

Maturita je ako plot – keď neprelezieš sám, prehodia ťa.

KSPáci

1. Zanzibarské jednosmerky

opravovalo YoYo δ
(max. 15 bodov)

Aj keď bol tento príklad najťažší, predsa len bolo riešení neúrekom. A boli aj správne. Niektoré. A za tie najrýchlejšie z nich bežiace v čase ($O(N+M)$), či už podobné vzorovému, alebo nie, sa dal získať plný počet bodov. Potom boli riešenia s jednoduchým prístupom: „ak sa dá dostať z každého vrcholu do každého, tak sa dá, ak sa nedá, tak sa nedá“[1]. Tie jednoducho vyskúšali pre každú dvojicu križovatiek, či sa z jednej dá dostať do druhej. Mohli dostať 12 a menej bodov, podľa časovej zložitosti (prinajlepšom $O(NM)$). No a potom tu boli ostatné riešenia, správne mohli dostať až 8 bodov, nesprávne nemohli.

Ako teda zistíme, či existuje medzi každými dvoma križovatkami (v každom smere) cesta? Ak sa dá naozaj dostať odvšadiaľ všade, môžeme si pri ceste z ľub. miesta A na ľub. miesto B vždy spraviť okľuku cez hlavné námestie. To preto, lebo určite existovala cesta z A na námestie a z námestia na B. Ale naopak, ak sa vieme z ľub. miesta A dostať na námestie a známestia na ľub. miesto B, vieme sa vlastne dostať z ľub. miesta A no ľub. miesto B (stačí tie dve cesty pripojiť za seba). Máme teda novú podmienku: či sa dá dostať z každej križovatky na námestie[2] a zároveň z námestia na každú križovatku. No a to sa nám bude zisťovať oveľa ľahšie.

Stačí nám teda zistiť, či sa dá dostať zovšadiaľ do nejakej konkrétnej križovatky (napríklad č. 1) a čí sa z nej dá dostať všade. Ako na to? Pekne postupne[3]. Aby sme zistili, kam všade sa vieme z križovatky 1 dostať, pustíme z nej jednoduché prehľadávanie. To nám „ofarbí“ všetky križovatky, do ktorých sa vieme dostať. Ako teraz zistíme, či sa dá dostať zovšadiaľ na do 1? Predsatvme si, že v 1 sadneme do auta, zaradime spiatočku[4] a začneme cúvať po jednosmerkách. Zrejme z každej križovatky, do ktorej by sme takýmto spôsobom prišli sa vieme dostať do do 1 (jednoducho dopredu prejdeme po ceste , ktorou sme docúvali). Takže jednoducho otočíme smery jednosmeriek a znova pustíme prehľadávanie z 1. Ak opäť

[1] dovolil som si citovať jedného z riešiteľov

[2] každému je dúfam jasné, že námestie môže byť ľubovoľná križovatka

[3] ale odzadu :-)

[4] prirovnanie prebraté od jedného z riešiteľov

ofarbíme všetky križovatky, dá sa z každej dostať do každej križovatky. Použijeme dve prehľadávania, ktorých časová zložitosť je $O(M + N)$, načítanie rovnako, takže celý algoritmus má časovú zložitosť $O(M + N)$.

Pár poznámok k vzorovému riešeniu: Po prehľadávaní netreba kontrolovať každú križovatku, či bola prejdená. Stačí si pri prehľadávaní, vždy keď nájdeme neoznačenú križovatku zvýšiť nejaké počítadlo a potom skontrolovať iba či je rovné N (premenná `pocet`). Použité je prehľadávanie do hĺbky, použitím rekurzie sa nemusíme starať o frontu alebo zásobník, pascal to spraví za nás. Čo sa týka prehľadávania v opačnom smere, oveľa jednoduchšie (aj rýchlejšie) je hneď pri načítaní si vytvoriť dva grafy (`G1` a `G2`).

Listing programu:

```
Program Zanzibarske_jednosmerky;
const MaxN=100;
type TG=array[1..MaxN,0..MaxN] of integer;

var
    G1,G2:TG;
    i,a,b,pocet:integer;
    N,M:integer;
    bol:array[1..MaxN] of boolean;


procedure Prejdi(var G:TG; i:integer);
var j:integer;
begin
  if not bol[i] then begin          {ak sme tuto krizovatku nenavstivili}
    bol[i]:=true;                                    {tak ju navstivime}
    inc(pocet);
    for j:=1 to G[i,0] do Prejdi(G,G[i,j]); {prejdeme vsetky cesty z nej}
  end;
end;

begin
  readln(N);
  readln(M);
  for i:=1 to N do begin
    G1[i,0]:=0;
    G2[i,0]:=0;
  end;
  for i:=1 to M do begin
    readln(a,b);
    inc(G1[a,0]); G1[a,G1[a,0]]:=b;             {Ulozime si jednosmerku}
    inc(G2[b,0]); G2[b,G2[b,0]]:=a;           {do G2 ulozime opacny smer}
  end;
  pocet:=0;
  for i:=1 to N do bol[i]:=false;
  Prejdi(G1,1);
  if pocet<>N then writeln('Neda sa.') {neda sa dostat vsade z krizovatky 1}
  else begin                                    {ak sa da dostat z 1 vsade}
    pocet:=0;                            {skontrolujeme este, ci sa da dostat}
    for i:=1 to N do bol[i]:=false;
```

```
    Prejdi(G2,1);                                        {odvsadial do 1}
    if pocet=N then writeln('Da sa.')                           {ak ano }
    else writeln('Neda sa.');                                {ak nie :))}
  end;
end.
```

opravoval Goober
(max. 15 bodov)

2. Zábudliví trpaslíci

Tento príklad ma veľmi potešil, lebo prišlo veľa riešenía a všetky boli správne (aj keď nie všetky vzorové). Bodové hodnotenie príkladu teda záviselo hlavne od časovej zložitosti riešenia – $O(n^2)$ = **9 bodov**, $O(n)$ alebo $O(1)$ = **15 bodov**. Približne tri body mohli ísť dolu za chýbajúci dôkaz správnosti a/alebo popis a odhad časovej a pamäťovej náročnosť.

Ako sa to malo robiť ? Nuž, základom úspechu bolo zistenie, že správny stromček sa nachádza čo najbližšie k stredu ulice (čo do počtu domčekov a stromčekov, nie vzdialeností). Na to väčšina z vás prišla. Lenže, nestačí na to len prísť, treba to aj dokázať. A tu bol veľmi častý zdroj problémov – mnohí z vás prehlásili, že stromček v strede je najlepší, lebo ak by sme vzali stromček tesne napravo alebo tesne naľavo od neho, tak by bol horší. To samozrejme nestačí ... lebo ďalej sa to môže zlepšiť. Správny dôkaz by mohol vyzerať tak, že ak vezmeme *ľubovoľný* stromček naľavo alebo napravo, tak bude cesta aspoň taká dlhá, ako pre nami vybratý stromček (druhá možnosť je, že pre každý iný stromček ukážeme, že ak sa posunieme smerom k stredu, celková vzdialenosť klesne – teda každý okrem stredného možno zlepšiť). Náš dôkaz využíva druhý postup.

Dôkaz: Predpokladajme, že najvýhodnejší je i-ty stromček a rozoberme tri prípady – ak je $i < \lfloor \frac{N}{2} \rfloor$, tak si všimnime stromček $i+1$ (ten je bližšie k stredu). Všetci trpaslíci v domčekoch $1 \ldots i$ budú musieť prejsť oproti pôvodnej ceste ešte úsek medzi i-tym a $(i+1)$-vým stromčekom (označme ho D). Naopak, trpaslíkom z domčekov $i+2 \ldots n$ sa cesta skráti o rovnaký kus. Trpaslík v dome $(i+1)$ isto nebude musieť oproti pôvodnému plánu prejsť viac ako D (namiesto doľava pôjde doprava). Z podmienky pre i vyplýva, že celková prejdená cesta sa takto skráti aspoň o D. Teda i-ty stromček nemôže byť za takejto podmienky najlepší. Druhý prípad je, že $i > \lceil \frac{N}{2} \rceil$. Tam si všimneme $(i-1)$-vý stromček, a zvyšok posupu je rovnaký. Takže, ukazuje sa, že najlepší stromček môže byť len $(\lfloor \frac{N}{2} \rfloor)$-vý, alebo $(\lceil \frac{N}{2} \rceil)$-vý. Ak je N párne, sú si tieto čísla rovné, a teda najlepší strom je jednoznačne určený. Ak je N nepárne, sú dve možnosti – dva stromčeky tesne vedľa stredného domu. Tu už nemožno „zanedbať“ stredného trpaslíka, lebo ostatní majú k obom stromčekom presne rovnakú celkovú dĺžku cesty. Preto treba vybrať stromček, ktorý má ku strednému domčeku bližšie.

Odhad časovej a pamäťovej náročnosti. Ak ste pochopili uvedený postup riešenia, malo by byť jasné, že nám vlastne stačia nanajvýš štyri údaje – počet stromčekov (N), poloha stredného stromčeku, alebo dvoch stredných stromčekov a stredného domčeka. Pomocou týchto údajov vieme úlohu vyriešiť na konštatný počet operácii s konštantným počtom pomocných premenných, teda časová aj pamäťová zložiosť by bola $O(1)$. Problém je s formátom vstupu – ak by sme chceli striktne dodržiavať ukázaný formát zo zadania, museli by sme prečítať aj zbytočné údaje (až do polovice), a teda by časová zložitosť vzrástla na $O(N)$. Tento problém sa zväčša rieši tak, že sa formát vstupu upraví na vhodnejší tvar (napríklad, že by sme sa mohli pýtať na relevantné údaje). Iná možnosť je predpokladať, že vstup už máme načítaný a zaujíma nás len samotný algoritmus.

Listing programu:

```
var
 n,k,l,p,d:integer;
```

```
 begin
  write('Pocet stromcekov: '); readln(n);
  k:=n div 2;
  if n mod 2=0 then
{ Párny počet domčekov }
   begin
    write('Stromcek #',k,': '); readln(k);
   end
  else
{ Nepárny počet domčekov }
   begin
    write('Stromcek #',k,': '); readln(l);
    write('Stromcek #',k+1,': '); readln(p);
    write('Domcek #',k+1,': '); readln(d);
    if (d-l)>(p-d) then k:=p else k:=l;
   end;
  writeln('Stretnu sa pri stromceku na pozicii ',k);
 end.
```

3. Záhady Kiribatského rybolovu

opravovala Janka
(max. 15 bodov)

Vaše riešenia boli viac-menej troch druhov. Za riešenia, ktoré spočívali v postupnom vypísaní radu druhých mocnín, či už do súboru alebo do stringu a následnom pozretí sa na k-tu pozíciu ste mohli získať maximálne **8 bodov**. Riešenia, v ktorých ste postupne počítali druhé mocniny čísel, kým súčet počtov cifier týchto mocnín neprekročil k boli ohodnotené najviac **10 bodmi**. Za riešenia, ktoré postupne počítali počet jednociferných, dvojciferných, ... mocnín, až kým by počet cifier v rade nepresiahol k sa dalo získať **15 bodov**.

Mnohí z vás určovali počet cifier nejakého čísla tak, že ho postupne celočíselne delili desiatimi, až kým nedosiahli nulu. Takýto cyklus nie je potrebný. Stačí si uvedomiť, že počet cifier čísla je približne jeho desiatkový logaritmus. Napríklad pre x od 100 až 999 leží $\log x$ v intervale $\langle 2, 3)$, odtiaľ už ľahko nahliadneme, že počet cifier (prirodzeného!) čísla je $\lfloor \log x \rfloor + 1$.

Keďže v Pascale máme len funkciu pre prirodzený logaritmus, môžeme použiť známy vzorec: $\log_a b = \frac{\log_c b}{\log_c a}$. Určenie počtu cifier nášho čísla x by teda v Pascale vyzeralo takto:
`cifier:=1+trunc(ln(x)/ln(10));`

Vzorové riešenie: V prvom rade chceme nájsť číslo, v ktorého mocnine je k-ta cifra radu, označme ho x. Ako bolo už vyššie spomenuté, budeme postupne počítať, koľko druhých mocnín má práve 1 cifru (tento počet označíme a_1), koľko ich má dve cifry (a_2), ... Zároveň budeme počítať, koľko cifier má rad tvorený všetkými jednocifernými mocninami, dvojcifernými mocninami, ..., až by dĺžka (počet cifier) takéhoto radu presiahla k. Ako určíme počet i-ciferných mocnín? Nuž najväčšia i-ciferná mocnina bude zrejme mocnina čísla $\lfloor \sqrt{10^i - 1} \rfloor$. Podobne najväčšia $(i-1)$-ciferná mocnina bude mocnina čísla $\lfloor \sqrt{10^{i-1} - 1} \rfloor$. Počet i-ciferných mocnín bude rozdiel týchto dvoch čísel. Rad tvorený všetkými i-cifernými mocninami má počet cifier rovný ia_i. Nech teda $a_1 + 2a_2 + \ldots + ia_i < k$, ale $a_1 + 2a_2 + \ldots + ia_i + (i+1)a_{i+1} >= k$. Nech $l = a_1 + 2a_2 + \ldots + ia_i$. Teda vieme, že x má $i+1$ cifier. A tiež vieme, že pred číslom x je v rade $\lfloor \frac{k-l-1}{i+1} \rfloor$ $(i+1)$-ciferných mocnín. Nech y je posledná i ciferná mocnina (t.j. $y = \lfloor \sqrt{10^i - 1} \rfloor$), potom $x = y + \lfloor \frac{k-l-1}{i+1} \rfloor + 1$. Teraz nám už len stačí nájsť $(k-l) \bmod (i+1)$-tú cifru čísla x. Všeobecne, j-tu cifru i-ciferného čísla nájdeme tak, že číslo vydelíme číslom 10^{i-j} a z toho vypočítame zvyšok po delení desiatimi.

Ako funguje náš program? V cykle postupne od `k` odčítavame dĺžku (`dlzka`) radu tvoreného mocninami, ktoré majú `cifier` cifier. Cyklus končí, ak by po ďalšom odčítaní bolo `k` ≤ 0. Po skončení cyklu vieme, že hľadaná cifra je `k`-tou cifrou v rade `cifier`-ciferných mocnín. Z toho podľa vyššie opísaného postupu vypočítame číslo, v ktorého mocnine sa `k`-ta cifra nachádza (`cislo`). Toto číslo už len umocníme a určíme jeho (`k mod cifier`)-tú cifru.

Odhad časovej zložitosti: Označme si číslo, ktorého druhá mocnina obsahuje k-tu cifru x. Hlavný cyklus v našom programe sa vykoná toľkokrát, koľko má x cifier. Ostatné časti programu sa vykonávajú s konštantou časovou zložitosťou. Skúsme určiť, koľko bude mať x rádovo cifier v závislosti od čísla k. Keďže každá mocnina v našom rade má určite aspoň jednu cifru, tak platí, že $x \leq k$. Vieme, že číslo k má rádovo $\log_{10} k$ cifier, teda aj číslo x má najviac rádovo $\log_{10} k$ cifier. Časová zložitosť nášho programu bude teda rádovo $\log_{10} k$, značíme to $O(\log k)$.

Listing programu:

```
program z;
uses crt;
var
 k,cifier,dlzka,posledne,nove,desat,vysledok:longint
 cislo:longint       {cislo, ktoreho mocnina obsahuje k-tu cifru radu}
 dlzka:longint       {je pocet cifier radu tvoreneho len i-cifernymi mocninami}

begin
  writeln('Zadaj poziciu zbrane: ');
  readln(k);
  cifier:=0;
  dlzka:=0;
  posledne:=0;
  nove:=0;
  desat:=10;
  while k>dlzka do
  begin
    k:=k-dlzka;
    posledne:=nove;
    inc(cifier);
    nove:=trunc(sqrt(desat-1));
    dlzka:=(nove-posledne)*cifier;
    desat:=desat*10;
  end;
  cislo:=(k+cifier-1) div cifier;
  cislo:=cislo+posledne;
  k:=k mod cifier;
  if k=0 then vysledok:=sqr(cislo) mod 10
  else
  begin
    k:=cifier-k;
    vysledok:=sqr(cislo) div round(exp((k)*ln(10)));  {delime cislom 10^k }
    vysledok:=vysledok mod 10;
  end;
  writeln('Cifra na k-tej pozicii je: ',vysledok);
end.
```

4. Z

opravoval Martin
(max. 15 bodov)

Túto úlohu ste riešili dvoma spôsobmi. Niektorí z vás načítali mapu a pre každú žiadosť prehľadali celú oblasť uvedenú na žiadosti, či sa v nej nachádza iba more. Časová zložitosť takého riešenia je $O(ZMN)$, kde M a N sú rozmery mapy a Z je počet žiadostí. Za takéto riešenie ste získali 7 bodov, prípadne ak ste urobili aj nejaké výrazné zlepšenia tohto algoritmu, mohli ste získať ešte 2 body naviac. Ostatných riešiteľov napadlo, že po načítaní mapy si môžu niečo predpočítať tak, aby vyhodnotenie každej žiadosti sa dalo zvládnuť v konštantnom čase, alebo inými slovami, aby časová zložitosť programu bola $O(MN + Z)$. Tieto riešenia mohli získať najviac 13 bodov. Ďalej ste mohli získať 2 body za kvalitný popis vášho algoritmu, alebo stratiť niekoľko bodov za chybný alebo chýbajúci odhad časovej alebo pamäťovej zložitosti, za škaredý a neprehľadný program a za množstvo chýb v ňom.

Mapa Kiribati je matica $M \times N$, ktorej štvorček ležiaci na súradniciach i, j označíme $Mapa_{i,j}$. Ak na súradniciach i, j je more, tak $Mapa_{i,j} = 0$, ináč $Mapa_{i,j} = 1$. Ďalej, nech $P(x_1, y_1, x_2, y_2)$, kde $x_1 \leq x_2$ a $y_1 \leq y_2$, značí počet pevnín v obdĺžniku so súradnicami x_1, y_1 a x_2, y_2 a pre $x_1 > x_2$ alebo $y_1 > y_2$ nech platí $P(x_1, y_1, x_2, y_2) = 0$. Nech A je matica taká, že $A_{i,j} = P(1, 1, i, j)$, pre $0 \leq i \leq M$ a $0 \leq j \leq N$. T.j. $A_{i,j}$ je počet pevnín, ktoré sú na súradniciach nanajvýš i,j.

Najprv si vyrátame maticu A. Pre všetky $0 \leq i \leq M; A_{i,0} = P(1, 1, i, 0) = 0$ a všetky $0 \leq j \leq N; A_{0,j} = P(1, 1, 0, j) = 0$. Teraz majme $i > 0$ a $j > 0$ a predpokladajme, že poznáme $A_{i-1,j-1}, A_{i,j-1}$ a $A_{i-1,j}$. Na základe týchto predpokladov vypočítame aj $A_{i,j}$. V súčte $Mapa(i, j) + A_{i,j-1} + A_{i-1,j}$ je raz alebo dvakrát zarátaná každá pevnina v obdĺžniku s rohmi $(1, 1)$ a (i, j). Pritom dvakrát sú zarátané práve tie pevniny, ktoré ležia v obdĺžniku s rohmi $(1, 1)$ a $(i - 1, j - 1)$. Tých je ale práve $A_{i-1,j-1}$. Teda platí:

$$A_{i,j} = Mapa_{i,j} + A_{i,j-1} + A_{i-1,j} - A_{i-1,j-1}$$

Teraz ideme vyrátať $P(x_1, y_1, x_2, y_2)$ pomocou A. Opäť funguje podobná úvaha, ako v odstavci vyššie. A_{x_2,y_2} sú všetky pevniny so súradnicami nanajvýš x_2, y_2. Keď od toho odčítame A_{x_1-1,y_2} a A_{x_2,y_1-1}, tak sme odrátali všetky pevniny mimo nášho obdĺžniku, až na to, že pevniny so súradnicami nanajvýš $x_1 - 1$,$y_1 - 1$ sme odrátali dvakrát. Preto ešte pričítame A_{x_1-1,y_1-1}, čím dostávame vzťah:

$$P(x_1, y_1, x_2, y_2) = A_{x_2,y_2} + A_{x_1-1,y_1-1} - A_{x_2,y_1-1} - A_{x_1-1,y_2}$$

$$P(x_1, y_1, x_2, y_2) = A_{x_2,y_2} + A_{x_1-1,y_1-1} - A_{x_2,y_1-1} - A_{x_1-1,y_2} = 38 + 13 - 30 - 16 = 5.$$

Číslo $P(x_1, y_1, x_2, y_2)$ nám o každom obdĺžniku so súradnicami x_1, y_1 a x_2, y_2 hovorí, koľko je v ňom pevnín, preto ak $P(x_1, y_1, x_2, y_2) = 0$, tak v obdĺžniku je len more. Maticu A si predrátame, preto na každý obdĺžnik vieme odpovedať v konštantnom čase.

Samotná realizácia: Pri výpočte matice A ju vypĺňame postupne po riadkoch. Všetky jej prvky nachádzajúce sa v nultom riadku alebo v nultom stĺpci majú hodnotu nula a všetky ostatné prvky matice vieme z predošlých prvkov matice vypočítať. Keď máme maticu A predpočítanú, môžeme začať spracovávať jednotlivé žiadosti. Obdĺžnik je v žiadosti zadaný pomocou dvoch protiľahlých vrcholov x_1, y_1 a x_2, y_2. Vzťah pre $P(x_1, y_1, x_2, y_2)$ však dokáže vypočítať počet pevnín v obdĺžniku len vtedy, ak $x_1 \leq x_2$ a $y_1 \leq y_2$, čo však v zadanom štvorci nemusí platiť. Preto musíme pracovať s ľavým horným, resp. pravým dolným vrcholom zadaného obdĺžnika, ktorých súradnice sú $min(x_1, x_2), min(y_1, y_2)$, resp. $max(x_1, x_2), max(y_1, y_2)$. Pre zadaný obdĺžnik vypočítame počet pevnín v ňom obsiahnutých. Ak je tento počet rovný nule, tak žiadosť pridelíme, v opačnom prípade ju zamietneme.

Tento algoritmus najprv predpočíta maticu A v čase $O(MN)$ a potom každú žiadosť rozhodne v konštantnom čase. Ak máme Z žiadosti, tak celková časová zložitosť je $O(MN + Z)$. Pamäťová zložitosť algoritmu je $O(MN)$, pretože si pamätáme maticu A.

Listing programu:

```
const
  MaxVyska = 100;
  MaxSirka = 100;

var
  A: array[0..MaxVyska, 0..MaxSirka] of Integer;    { Matica A       }
  Vyska, Sirka: Integer;                            { Rozmery mapy   }
  Ziadosti: Integer;                                { Pocet ziadosti }
  x1, x2, y1, y2: Integer;                          { Ziadost        }
  i, j, t: Integer;
  ch: Char;

begin
  Readln(Vyska, Sirka);

  for i := 1 to Vyska do A[i,0] := 0;
  for j := 1 to Sirka do A[0,j] := 0;

  for i:=1 to Vyska do begin
    for j:=1 to Sirka do begin
      A[i,j] := A[i-1,j] + A[i,j-1] - A[i-1,j-1];   { Vztah (2)      }
      Read(ch);
      if ch = '1' then A[i,j] := A[i,j] + 1;
    end;
    Readln;
  end;

  Readln(Ziadosti);
  for i := 1 to Ziadosti do begin
    Readln(y1,x1,y2,x2);

    { Pracujeme s lavym hornym a pravym dolnym vrcholom obdlznika. }
    if y1 > y2 then begin t := y1; y1 := y2; y2 := t end;
    if x1 > x2 then begin t := x1; x1 := x2; x2 := t end;

    if A[y2,x2] + A[y1-1,x1-1] - A[y1-1,x2] - A[y2,x1-1] = 0
```

```
      then Writeln('Pridelit')                    { Vztah (1)      }
      else Writeln('Zamietnut');
  end;
end.
```

5. Zablúdená správa

Opravovala Ňaňka
(max. 10 bodov)

Ako sa to už v piatom príklade stáva, väčšina riešení, ktoré sme dostali, bola správna, čiže obsahovala dekódovanú správu nasledujúceho obsahu:

PATRANIE PO NAJCASTEJSIE DREVENOM PREDMETE, KTORY MOZE SLUZIT NA UDRZIAVANIE STABILITY JEDINCOVI DRUHU HOMO SAPIENS, KTORYCH VEK VYSOKO PREKRACUJE MEDIAN TEJTO VELICINY NA SUBORE VSETKYCH ZIJUCICH JEDINCOV TOHTO DRUHU, SA U REFLEKTANTA ATAKU SELMY CELADE CANIDAE STRETA S USPECHOM ZA AKYCHKOLVEK PODMIENOK.

Za takúto správu s popisom ako vznikla (alebo s programom, ktorý ju vygeneroval) som udeľovala **10 bodov**. Keď niektorá časť chýbala, strhla som **2 body**. Iná správa samozrejme tiež mohla vzniknúť (veď spôsobov šifrovania je požehnane...), ale v slovenčine asi nedávala zmysel (preto za iný spôsob bolo menej bodov).

Ale myslím, že väčšina z tých, ktorým nevyšlo nič zmysluplné, ani riešenie tohto príkladu neposlali. A tak teraz niečo hlavne pre nich.

Akým spôsobom sa tento príklad dal riešiť:

- **Zložitým:** skúšať analyzovať, hľadať kľúč, spôsob šifrovania...
- **Jednoduchším:** pozrieť sa na obrázok, ktorý tam nebol náhodou a nájsť v ňom dve slová, ktoré už o spôsobe ako pokračovať čosi napovedia...

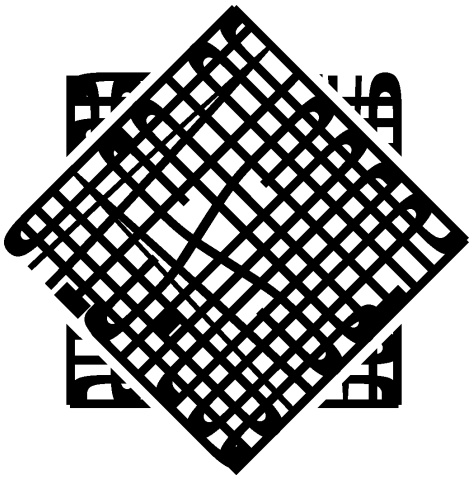
My sa budeme samozrejme zaoberať tým jednoduchším spôsobom, lebo ten zložitejší je na dlhšie rozprávanie a hlavne na dlhšie riešenie a to sme od vás nechceli. Preto bol pri zadaní obrázok. Keď ste si ho poriadne zo všetkých strán a uhlov pozreli (najlepšie je asi držať papier takmer vodorovne na úrovni očí), mohli ste v ňom objaviť dve slová: *PODUSTVA* a *DUTOSVAP*. Tieto samozrejme majú za úlohu napomôcť k vyriešeniu šifry.

Čo sa s takými slovami dá robiť. Pravdepodobne sú to nejaké kľúče. Iste hneď mnohých napadla klasická šifra, kde sa pod text podpíše kľúč, sčítajú sa hodnoty... A akí ste boli sklamaní, keď to nič zmysluplné nedávalo...

Tu sa treba zastaviť a pozrieť si poriadne tie slová. Sú dve a v obidvoch sú rovnaké písmená, iba nejako vhodne poprehadzované. Čo keby sme to isté spravili aj s textom, ktorý sme dostali? Konkrétne, rozdeľme si text na osmice a každú z nich poprehazdujme.

Aj tak nám však ostali dve možnosti: Buď budeme prehadzovať písmenká zo slova DUTOSVAP na *PODUSTVA*, alebo naopak zo slova *PODUSTVA* na slovo *DUTOSVAP*. V jednom z prípadov nám však na začiatku vzniká slovo *IAERAPTN*, čo nevyzerá vôbec slovensky. V druhom nájdeme slovo *PATRANIE* a ďalej text vyššie napísanej správy, ktorý aj keď sa to možno nezdá po slovensky a je tiež správne dešifrovaným textom.

Bonusová úloha: Viete, čo vlastne tá správa znamená?



Výsledková listina po 3. sérii kategórie KSP-Z

	Meno a priezvisko	kola	Trieda		31	32	33	34	35	Σ
1.	Jakub Závodný	Gym. Grösslingová BA	2	15	15	15	15	10	70	
2.	Matúš Petruľák	Gym. Grösslingová BA	1	15	14	15	15	10	69	
	Martin Rejda	Gym. Grösslingová BA	2	14	15	15	15	10	69	
4.	Michal Kevický	Gym. Grösslingová BA	2	15	15	14	14	10	68	
5.	Michal Ďuriš	Gym. Grösslingová BA	1	15	15	13	14	10	67	
6.	Milan Šatka	Gym. Liptovský Hrádok	3	15	14	15	11	10	65	
7.	Michal Hrobár	Gym. Jura Hronca BA	2	12	14	15	12	10	63	
8.	Michal Nánási	Gym. Jura Hronca BA	1	11	15	10	15	10	61	
9.	Miroslav Štolc	Gym. Párovská Nitra	2	12	14	15	7	10	58	
	Imrich Živčák	Gym. Komenského Trebišov	4	11	14	10	13	10	58	
11.	Michal Repovský	Gym. Štefánika Trebišov	2	15	14	10	8	10	57	
12.	Juraj Kollar	Gym. Jura Hronca BA	1	10	14	15	6	10	55	
13.	František Šmitala	Gym. Farská Nitra	2	11	9	15	7	10	52	
14.	Peter Perešíni	ZS Radvanská B. Bystrica	0	11	15	15	9		50	
15.	Maroš Bajtoš	Gym. Jura Hronca BA	1	9	14	15	11		49	
	Lukáš Poláček	Gym. K. Štúra Modra	2	12	15	15	7		49	
	Juraj Porubský	Gym. Farská Nitra	2	7	9	15	8	10	49	
18.	Rastislav Lenhardt	GMMH Liptovský Mikuláš	2	8	13	10	7	10	48	
	Andrej Mikulík	Gym. Grösslingová BA	2	9	9	14	6	10	48	
20.	Pavol Szórád	Gym. Farská Nitra	2	6	9	13	9	10	47	
21.	Jozef Legeny	Gymnázium	1	9	9	10	8	10	46	
22.	Tibor Blénessy	Gym. Poštová Košice	2	12	14	10	9		45	
	Stano Bustor	Gym. Jura Hronca BA	1		13	11	11	10	45	
	Ivan Trejbal	Gym. Jura Hronca BA	2		13	15	7	10	45	
	Ján Valaska	Gym. Jura Hronca BA	2	10	9	6	10	10	45	
26.	Martin Bies	Gym. Jura Hronca BA	2		14	11	9	10	44	
27.	Magdaléna Jurkovičová	Gym. Golianova Nitra	3	3	15	10	5	10	43	
28.	Michal Dobiš	Gym. Hollého Trnava	3	5	9	10	8	10	42	
29.	Ľuboš Duda	Gym. Nová Baňa	3	7	9	8	7	10	41	
30.	Marek Jančuška	Gym. Párovská Nitra	2	10	14	5	2	9	40	
31.	Katarína Holešová	Gym. Haanova BA	4	2	9	10	8	10	39	
32.	Michal Kottman	Gym. Jura Hronca BA	1		13	8	6	10	37	
	Radoslav Zapotocky	Gym. Alejová Košice	2	12	8	10	7		37	
34.	Peter Ambroz	Gym. Jura Hronca BA	2		14		12	10	36	
	Miroslav Jagoš	Gym. Varšavská Žilina - Vlčince	1		9	10	7	10	36	
	Igor Rjabinin	Gym. Farská Nitra	2	3	9	7	7	10	36	
37.	Miloš Čiernik	Gym. Lettricha Martin	1		9	10	6	10	35	
38.	Michal Lukáč	Gym. Alejová Košice	4	3	9	8	6	8	34	
	Veronika Petráková	Gym. Jura Hronca BA	1		9	9	6	10	34	
	Peter Ružička	Gym. Jura Hronca BA	1	0	9	8	7	10	34	
41.	Miroslav Priesol	Gym. Dolný Kubín	4	2	14		7	10	33	
	Marek Tomacha	Gym. sv. Uršule BA	4		9	15	9		33	
43.	Michal Dzetkulič	Gym. P. Horova Michalovce	1	1	14	10	7		32	
	Róbert Šimon	Gym. Hlohovec	2	7	9	10	6		32	
45.	Mariana Kuchynárová	Gym. Jura Hronca BA	2	9			12	10	31	
46.	Katarína Gergelyová	Gym. Jura Hronca BA	1		9	7	6	8	30	
	Pavel Struhár	Gym. Jura Hronca BA	1		5	8	7	10	30	
	Matúš Svrček	Gym. Stará Ľubovňa	2	12	13	5			30	
	Marek Zeman	Gym. Jura Hronca BA	1		9	11		10	30	
50.	Ľubomír Kundrák	Ev. lýceum BA	1	3	9	10	7		29	

	Meno a priezvisko	kola	Trieda		31	32	33	34	35	Σ
	Martin Valacsai	Gym. Jura Hronca BA	1	10	13		6		29	
52.	Denis Donauer	Gym. Jura Hronca BA	1		7	7	6	8	28	
	Michal Kramarič	Gym. Jura Hronca BA	2		13		5	10	28	
	Roman Petrinec	Gym. Jura Hronca BA	2		12		6	10	28	
55.	Jana Adamková	Gym. Grösslingová BA	2	3	4	8	2	10	27	
	Martin Paulech	Gym. Hlohovec	2		3	10	6	8	27	
	Martin Slavik	Gym. Jura Hronca BA	2		12		5	10	27	
58.	Tomáš Balogh	Gym. Jura Hronca BA	2		9		6	10	25	
	Peter Čunderlík	Gym. Jura Hronca BA	2		9	8	8		25	
	Ján Lalinský	Gym. Varšavská Žilina - Vlčince	1		9	10	6		25	
	Robert Sasak	SPŠE Piešťany	4		9		8	8	25	
	Ondrej Székely	GLN Tomášikova BA	3		9		6	10	25	
63.	Štefan Krištofík	SPŠ Levice	3		9	8	7		24	
	Jakub Šimko	Gym. Jura Hronca BA	1		8		6	10	24	
65.	Martin Salaj	Gym. Čadca	3		8	8	6		22	
	Andrea Štefánková	Gym. Jura Hronca BA	1		14			8	22	
67.	Michal Beňo	Gym. Kremnica	2		9	5	5	2	21	
68.	Anna Kovacicova	Gym. Jura Hronca BA	1		9			10	19	
	Vojtech Villaris	Gym. Jura Hronca BA	1		8	6	5		19	
70.	Peter Ivanov	Gym. Jura Hronca BA	1		7			10	17	
	Lucia Kuklicová	Gym. Jura Hronca BA	2				7	10	17	
72.	Igor Andruška	SPŠ Levice	1			10	6		16	
	Michal Lipták	Gym. Jura Hronca BA	4				6	10	16	
	Daniela Mikušová	Gym. Jura Hronca BA	1				6	10	16	
75.	Peter Goga	Gym. 17. novembra Topoľčany	2	1		8	6		15	
76.	Martin Hudec	Gym. Jura Hronca BA	1		3			3	6	
	Michal Mraz	Gym. Jura Hronca BA			3			3	6	
	Marian Ridilla	Gym. Jura Hronca BA	1		3			3	6	

0