Korešpondenčný seminár z programovania
XXVIII. ročník, 2010/11
Katedra základov a vyučovania informatiky FMFI UK,
Mlynská Dolina, 842 48 Bratislava

KSP finančne podporujú: MICROSTEP-MIS spol. s r.o.
Agentúra na podporu výskumu a vývoja

Vzorové riešenia 1. kola zimnej časti

opravoval Feki

1. Zbláznil sa fyzikár... (max. 10 bodov)

Väčšina riešiteľov sa zachovala k tomuto príkladu vzorovou formou, aspoň z hľadiska kódu. Horšie to bolo s popisom a odhadom časovej a pamäťovej zložitosti.

Jedno z možných riešení využíva fakt, že sa v tabuľke striedajú dva typy riadkov: `+-+-+...+-+` a `|␣|␣|...|␣|` (znak ␣ predstavuje medzeru). Obidva typy môžeme vypísať pomocou `for`-cyklu. Stačí len S-krát vypísať dvojicu znakov `+-` (prípadne `|␣`) a nakoniec dodať znak `+` (prípadne `|`). Celé to obalíme ďalším `for`-cyklom – R-krát vypíšeme dvojicu riadkov prvého a druhého typu a tabuľku ukončíme riadkom prvého typu.

Iné riešenie: očíslujeme si riadky výstupu od 0 do $2 \cdot R$ a stĺpce od 0 do $2 \cdot S$. Keď si teraz vezmeme nejakú pozíciu vo výstupe $[r, s]$, vieme určiť, ktorý znak sa má na nej nachádzať, len pomocou zvyškov r a s po delení 2. Napríklad, ak je číslo riadka aj stĺpca párne, na danej pozícii bude znak `+`. Ak je ale číslo riadka párne a číslo stĺpca nepárne, bude tam `-`.

Obe vzorové riešenia bežia v čase $O(R \cdot S)$ a využívajú konštantne veľa ($O(1)$) pamäte. Ak ste si zapamätali celý riadok v reťazci a potom ho vypisovali, zhoršila sa vám pamäťová zložitosť na $O(S)$.

Bodovanie:

- Riešenie blízke vzorovému dostalo plný počet.
- Strhávalo sa pomerne často za chýbajúci alebo nedostatočný popis (-5 bodov za chýbajúci, inak podľa kvality popisu).
- Za odhad každej zložitosti bol 1 bod. Chýbala vám, alebo mali ste ju nesprávne? Body nedostanete.
- Za vypisovanie zbytočnej medzery na konci riadka som nekompromisne strhol pomerne veľa bodov, keďže takéto riešenie nebolo správne.

Listing programu:

```
var r, s, i, j: integer;

begin
  ReadLn(r, s);

  for i := 1 to r do begin
    for j := 1 to s do
      Write('+-');
    WriteLn('+');

    for j := 1 to s do
      Write('| ');
    WriteLn('|');
  end;
```


Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe zmluvy č. LPP-0103-09

```
  for j := 1 to s do
    Write('+-');
  WriteLn('+');
end.
```

Listing programu:

```
const znak: array[0..1] of array[0..1] of char = (('+', '-'), ('|', ' '));
var r, s, i, j: integer;

begin
  ReadLn(r, s);

  for i := 0 to 2 * r do begin
    for j := 0 to 2 * s do
      Write(znak[i mod 2][j mod 2]);
    WriteLn;
  end;
end.
```

2. Zrkadielko, zrkadielko, povedz že mi...

Táňa & Mio
(max. 10 bodov)

Keďže nemalá časť z vás mala problémy s dodržaním obmedzení zadania, tak si ich zopakujeme (a upresníme):

- *Váš program si má daný náhrdelník pamätať v jednej premennej typu reťazec, ktorej obsah môže meniť.* – To znamená, že nesmiete ani meniť jeho veľkosť (t.j. pridávať znaky na koniec reťazca...).
- *Nesmiete však používať ďalšie pomocné reťazce ani polia.* – To znamená, že nesmiete používať ani funkcie, ktoré interne používajú ďalší reťazec alebo pole (napr. funkcie `copy`, `strcat`...).
- *Na výstup vypíšte obsah premennej typu reťazec, do ktorej ste na začiatku načítali náhrdelník zo vstupu (vypíšte ho jediným príkazom, nie po znakoch).*

Najprv si označme dĺžku náhrdelníka n a počet znakov, o ktoré treba náhrdelník otočiť, k.

Bolo treba myslieť aj na to, že k mohlo byť aj väčšie ako dĺžka reťazca a programy, ktoré nefungovali pre $k > n$ sme považovali za nefunkčné. V zadaní síce nebolo napísané ani to, že $k \geq 0$, ale vzhľadom na to, že väčšina z vás to neošetrila, rozhodli sme sa za to body nestrhávať.

Bodovanie:

Hodnotili sme prísne, ale je to pre vaše dobro, pretože chceme, aby ste boli čo najlepší. Body sme rozdávali takto:

- Vzorové riešenie v čase $O(n)$ mohlo získať 10 bodov.
- Riešenie v čase $O(n^2)$ (t.j. také, ktoré si k zmenšilo tak, aby nebolo väčšie ako n) mohlo získať 8 bodov.
- Riešenie v čase $O(n \cdot k)$ (t.j. také, ktoré si k nezmenšilo) mohlo získať 7 bodov.
- Nefunkčné riešenie nezískalo viac ako 4 body.

Strhávali sme body za chýbajúci popis, zdroják, chybné alebo chýbajúce odhady časovej a pamäťovej zložitosti a samozrejme za porušenie obmedzení zadania.

Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe zmluvy č. LPP-0103-09

Viacerí z vás mali problém s určovaním zložitostí. Ak v programe používame reťazec, ktorý nie je konštantnej dĺžky, tak pamäťová zložitosť musí určite závisieť aj od dĺžky reťazca, čiže pamäťová zložitosť bola $O(n)$, nie $O(1)$. Čo sa týka časovej zložitosti, najväčším problémom bolo určovanie časových zložitostí funkcií `delete`, `insert` a pod. Ak mažeme niečo zo začiatku reťazca, tak funkcia `delete` posunie každý znak reťazca o vymazaný počet znakov dopredu a teda má časovú zložitosť $O(n)$. Takisto ak pridávame niečo na začiatok reťazca, tak funkcia `insert` posunie celý zvyšok reťazca o pridaný počet znakov ďalej a má časovú zložitosť $O(n)$.

Riešenie:

Na úvod spomenieme, že ak náhrdelník otáčame o $k = pn + q$, tak je to to isté, ako keby sme náhrdelník otočili o $k = q$ (pretože ak náhrdelník otočíme o n, dostaneme ten istý náhrdelník). Takže predtým, ako ideme načítať náhrdelník, zmenšíme si k na $k \bmod n$.

Teraz uvedieme myšlienku $O(n)$ riešenia, ktoré posúva znaky vždy o k. Zapamätáme si 1. znak a v cykle si na jeho miesto dáme $(k+1)$-vý, na jeho miesto $(2k+1)$-vý atď. (pozície počítame modulo n), až kým sa nedostaneme k prvému znaku. Toto ale zakape na tom, že ak sú n a k súdelitelné, tak sa k niektorým znakom nedostaneme. Celý trik spočíva v tom, že týchto cyklov spravíme viac a vždy začneme o znak ďalej. Ľahko vidieť, že keď týchto cyklov spravíme toľko, ako je najväčší spoločný deliteľ n a k, presunieme všetky znaky, každý práve raz, a teda to beží v čase $O(n)$. Toto riešenie bolo celkom vymysliteľné a ďalej sa mu nebudeme venovať. Namiesto toho si ukážeme pekné, ľahko naprogramovateľné trikové riešenie.

Ak spravíme reverz reťazca (otočíme ho zrkadlovo) tak prvých k znakov sa nám zrazu ocitne na konci ako posledných k znakov – akurát v opačnom poradí. Posledných $n-k$ znakov bude teraz prvými $n-k$ znakmi a tiež odzadu. Takže ak tieto 2 časti znovu zrkadlovo otočíme, máme pootočený náhrdelník. Algoritmus teda vyzerá tak, že spravíme reverz celého reťazca, reverz prvej časti s veľkosťou $n-k$ a reverz druhej časti s veľkosťou k. Reverz spravíme tak, že vymeníme vždy i-ty znak s $(m-1-i)$-tym, kde m je dĺžka podreťazca, na ktorom vykonávame reverz. Reverz má časovú zložitosť $O(n)$, takže taká je aj časová zložitosť riešenia. Pamäťová zložitosť je lineárna od dĺžky reťazca, takže tiež $O(n)$.

Jednoduché, nie? `:-)` Teraz si môžete užiť vzorák:

Listing programu:

```
var s: string;
    k, n: integer;

procedure reverse(od, po:integer);
var t: char;
    i: integer;
begin
  for i := 0 to (po - od + 1) div 2 - 1 do begin
    t := s[od + i];
    s[od + i] := s[po - i];
    s[po - i] := t;
  end;
end;

begin
  ReadLn(s);
  ReadLn(k);

  n := Length(s);  { dlzka nahrdelniku }
  k := k mod n;  { zmodulujeme k }
  if k < 0 then
    k := k + n;  { posunutie o -k dolava je to iste ako posunutie o n-k dolava }
```


Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe zmluvy č. LPP-0103-09

```
  reverse(1, n);
  reverse(1, n - k);
  reverse(n - k + 1, n);

  WriteLn(s);
end.
```

opravoval Mišo
(max. 10 bodov)

3. Zase ďalšia Alica?

Najprv trochu matematických úvah: chceme porovnať x, ktoré má za sebou d_x výkričníkov s y, ktoré má za sebou d_y výkričníkov. Porovnanie dopadne rovnako, ako keď od každého odoberieme jeden výkričník. To vyplýva z toho, že faktoriál je rastúca funkcia. Takže môžeme odobrať od každého rovnaký počet výkričníkov tak, aby pri jednom čísle neostal žiaden výkričník. Pri druhom ostane rozdiel z počtu výkričníkov. Takže napr. z porovnania 4!!! a 22!! ostane 4! a 22. Potom ideme rátať faktoriály z toho čísla, pri ktorom ostali výkričníky. To robíme dovtedy, kým už nebude väčšie ako to druhé číslo. Pozor, túto podmienku musíme dať do vnútra funkcie faktoriál, aby sme zabránili pretečeniu.

V programe si na začiatku vymeníme x a y tak, aby x malo viac výkričníkov. Potom ošetríme špeciálne prípady, aby sme zvytočne nerátali veľa faktoriálov dvojky alebo jednotky.

Strhával som body za neošetrenie pretečení. Predsa len na uloženie čísla 4!!! treba viac bitov ako je atómov vo vesmíre, preto ste za toto mohli dostať 3 až 4 body. 2 body som strhával za zlé alebo chýbajúce odhady zložitostí. Za chýbajúci zdroják/popis išla polovica bodov dole.

Listing programu:

```
var x, xx, xxx, dx, y, dy, i, j: longint;
    s, t, u: string;

begin
  ReadLn(x, s);
  dx := Length(s) - 1;  { -1 kvoli medzere navyse }
  ReadLn(y, t);
  dy := Length(t) - 1;

  if dy > dx then begin
    i := dy; dy := dx; dx := i;
    u := s; s := t; t := u;
    i := y; y := x; x := i;
  end;

  if x < 3 then begin
    if y < x then
      WriteLn(x, s)
    else if y = x then
      WriteLn('rovnake')
    else
      WriteLn(y, t);
    Exit();
  end;

  xxx := x;
  for i := 1 to dx - dy do begin
    xx := x;
```


Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe zmluvy č. LPP-0103-09

```
    x := 1;
    for j := 1 to xx do begin
      x := x * j;
      if x > y then begin
        WriteLn(xxx, s);
        Exit();
      end;
    end;
  end;

  if x = y then
    WriteLn('rovnake')
  else
    WriteLn(y, t)
end.
```

opravovalo sa samo, vzorák Usamec
(max. 15 bodov)

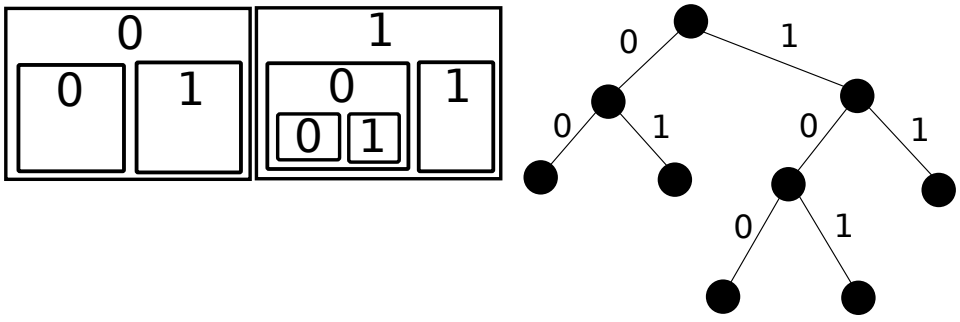
4. Zabité dekódovanie

Najprv si ujasníme, čo ideme robiť a potom si ukážeme, ako to robiť dostatočne rýchlo. Už v zadaní je napísaný hint, že postupnosť sa dá dekódovať jednoznačne. Takže keď niekde na začiatku zakódovanej postupnosti nájdeme nejaký kód, môžeme ho hneď veselo dekódovať. Je to tým, že tento kód nie je prefix žiadneho iného kódu, teda žiadny iný kód tu už začínať nebude.

Začnime s triviálnym riešením. Zoberieme náš zakódovaný text a vyskúšame každý kód. Ak je nejaký kód zhodný so začiatkom, tak vypíšeme patričné slovo a zakódovaný text skrátime. Akú to má časovú zložitosť? Na prvý pohľad $O(MNL)$, kde L je dĺžka najdlhšieho slova, lebo prinajhoršom v každom písmene zakódovaného textu vyskúšame všetky slová a vyskúšanie jedného slova trvá L krokov. Reálne je čas ale oveľa lepší, lebo väčšinou odoberieme zo zakódovaného textu viac ako jedno písmeno.

Teraz si popíšeme lepšie riešenie. Predstavme si, že sme prečítali zo zakódovaného textu nulu. Tak teraz môžeme všetky kódy začínajúce na jednotku zahodiť. Teraz ak ešte existuje viac kódov začínajúcich na 0, prečítame ďalšie písmeno. Nech je to teraz jednotka. Takže môžeme zahodiť ďalšie kódy a to tie, ktoré začínajú na 00. A tento postup môžeme opakovať. Otázka znie, ako tieto kódy usporiadať, aby sa to ľahko kódilo.

Zoberme si dve krabice. Napíšme na nich značky 0 a 1. Do krabice označenej 0 nahádžeme všetky kódy začínajúce na 0, do krabice označenej 1 nahádžeme tie začínajúce na 1. Zatiaľ sme si moc nepomohli. Ale keď sú kódy uložené v nejakej krabici, tak tú zase môžeme rozdeliť na dve a tam tie kódy rozhodiť. A toto môžeme opakovať dovtedy, kým v krabici nemáme len 1 kód, ale ešte lepšie, keď už sme kód rozdelili podľa všetkých písmen. Z informatického hľadiska si toto môžeme predstaviť ako strom (všimnite si, že označenia 0, 1 sme dali na hrany stromu, aby vyjadrovali, cez čo ideme). Na obrázku sú krabice a strom pre kódy 00, 01, 100, 101, 11.



Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe zmluvy č. LPP-0103-09

Programuje sa to už celkom jednoducho. V každom vrchole si pamätáme odkazy na synov (alebo že tam nič nie je). A pokiaľ tam končí nejaký kód, pamätáme si tam jeho preklad. Dôležité ale je nemať v každom vrchole pole dĺžky okolo 20 na preklady, lebo sa nezmestíme do pamäte. Takže buď si preklady uložíme externe do poľa a do stromu si zapamätáme len index prekladu, alebo sa trochu pohráme s dynamickými poliami.

A keď už máme vybudovaný strom, tak dekódovanie sa spraví ľahko. Posúvame sa po strome a keď narazíme na preklad, vypíšeme ho a vrátime sa v strome do koreňa (začíname odznovu).

Časová zložitosť: strom vybudujeme v čase lineárnom od dĺžky kódov $O(LM)$. A prebehneme v ňom v čase $O(N)$ (na každom znaku máme jeden posun). Pamäťové nároky na strom sú $O(ML)$ (musíme si uložiť všetky kódy).

Listing programu:

```
type Vrchol = record
       syn: array[0..1] of longint;
       preklad: longint;
     end;

{ strom obsahuje vrcholy stromu, 0 je koren stromu }
var strom: array[0..5000000] of Vrchol;
    slova: array[0..1000000] of ansistring;
    M, N, i, j, nextv, cur, medz, kam: longint;
    riadok, kod: ansistring;
    first: boolean;

procedure pripravVrchol(x: longint);
begin
  strom[x].syn[0] := -1;
  strom[x].syn[1] := -1;
  strom[x].preklad := -1;
end;

begin
  ReadLn(M);
  { nextv je prvy nepouzity vrchol z pola }
  nextv := 1;
  pripravVrchol(0);
  for i := 1 to M do begin
    { pascal je skaredy a nevie nacitanie delit medzerou }
    ReadLn(riadok);
    medz := Pos(' ', riadok);
    slova[i] := Copy(riadok, 1, medz - 1);
    kod := Copy(riadok, medz + 1, Length(riadok) - medz);
    cur := 0;
    { najdeme v strome spravne miesto }
    for j := 1 to Length(kod) do begin
      kam := Ord(kod[j]) - Ord('0');
      { ked tam kde chceme ist este nic neni, vyrobime novy vrchol }
      if strom[cur].syn[kam] = -1 then begin
        pripravVrchol(nextv);
        strom[cur].syn[kam] := nextv;
        Inc(nextv);
      end;
      cur := strom[cur].syn[kam];
    end;
    { a na spravne miesto zapiseme preklad }
    strom[cur].preklad := i;
```


Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe zmluvy č. LPP-0103-09

```
  end;

  ReadLn(N);
  ReadLn(riadok);
  first := true;
  cur := 0;
  { behame po strome }
  for i := 1 to Length(riadok) do begin
    kam := Ord(riadok[i]) - Ord('0');
    cur := strom[cur].syn[kam];
    { ak sme dosli k nejakemu prekladu vypiseme ho }
    if strom[cur].preklad <> -1 then begin
      if not first then
        Write(' ');
      Write(slova[strom[cur].preklad]);
      first := false;
      cur := 0;
    end;
  end;
  WriteLn;
end.
```

5. Otrávený duel

opravoval Piťo
(max. 15 bodov)

V tejto úlohe si bolo treba najskôr uvedomiť, že kruhy boli naozaj len chyták a na ich počte a veľkosti nezáleží. Môžeme si to takto zjednodušiť, pretože kruhy lordov vôbec neobmedzujú vo výbere pohárov. V jednom ťahu môžu odobrať $1 \dots L$ pohárov z ľubovoľných kruhov a na konci duelu je v každom možnom vývoji hry len jeden kruh, preto rozhoduje len počet vypitých pohárov v danom ťahu.

Duel si teda vieme predstaviť ako rad pohárov očíslovaných od 1 po N, kde lordi môžu vypiť 1 až L pohárov z konca a ten, kto vypije pohár s číslom 1, zomrie. Vieme, že obaja lordi hrajú optimálne, to znamená, že ak existuje postupnosť ťahov, pri ktorej vyhrajú, tak túto postupnosť urobia (inak povedané nerobia hlúposti).

Nazvime si situáciu, kedy je lord na ťahu a má pred sebou M pohárov, pozíciou M. Navyše, ak neexistuje postupnosť optimálnych ťahov, pri ktorej vyhrá, budeme ju volať prehrávajúcou pozíciou. Jednou takouto pozíciou je pozícia 1. Lordovi vtedy neostáva nič iné, len vypiť posledný pohár a zomrieť. Vo všeobecnosti je prehrávajúcou pozícou taká, z ktorej všetky ťahy vedú do vyhrávajúcej, teda akýmkoľvek ťahom umožníme súperovi vyhrať.

Vyhrávajúcou pozíciou nazveme takú, z ktorej existuje ťah do prehrávajúcej (urobíme práve tento ťah a súper bude musieť prehrať). Takéto pozície sú napríklad 2 až $L + 1$, pretože vezmeme 1 až L pohárov a dostaneme tým súpera do pozície 1. Pozícia $L + 2$ je opäť prehrávajúca, pretože odobratím $1 \dots L$ pohárov sa vieme dostať len do vyhrávajúcich pozícií. Nasledujúcich L je opäť vyhrávajúcich.

Teraz si môžeme všimnúť, že sa tu opakuje postupnosť prehrávajúcej a L vyhrávajúcich pozícií. S týmto zistením už vieme vymyslieť riešenie. Stačí, keď si uvedomíme, kedy lord Norton začína v prehrávajúcej pozícií. To sa stane práve vtedy, keď má pred sebou $m \cdot (L + 1) + 1$ pohárov ($m \in \mathbb{N}_0$) čiže $N \bmod (L + 1) = 1$.

Toto riešenie beží v čase $O(1)$, vyžaduje pamäť $O(1)$ (ignorujeme druhý riadok vstupu s veľkosťami kruhov) a dalo sa zaň získať 15 bodov. Niektorí ste ich však čítali celý vstup, čo vám v skutočnosti zhoršilo časovú zložitosť na $O(K)$, a ak ste si ich nebodaj aj pamätali, tak aj pamäťovú. Nebral som za to body, no v prípade, že už zo vstupu nič nepotrebujete, tak to zbytočne nečítajte.

Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe zmluvy č. LPP-0103-09

Medzi ostatnými riešeniami boli aj také, ktoré nerobili operáciu modulo ale $N/(L+1)$ odčítaní, a teda potrebovali $O(N/L)$ času a $O(1)$ pamäte. Tie mohli dostať 11 bodov. Našli sa aj také, ktoré si v poli vyfarbovali pozície, podľa toho, či sú vyhrávajúce alebo prehrávajúce, a preto potrebovali $O(N \cdot L)$ času a $O(N)$ pamäte a mohli dostať 9 bodov. Body som strhával za slabý popis, ako ste sa dostali k riešeniu, za nedostatočné zdôvodnenie, prečo na kruhoch nezáleží a za chýbajúci alebo nesprávny odhad zložitosti.

Tento problém je špeciálnym prípadom problému Nim, nazývaný Subtraction game, viac si môžete prečítať napríklad na Wikipédii: `http://en.wikipedia.org/wiki/Nim`.

Listing programu:

```
var N, K, L: integer;
begin
  ReadLn(N, K, L);
  if N mod (L + 1) = 1 then
    WriteLn('Vyhra lord Berkley')
  else
    WriteLn('Vyhra lord Norton');
end.
```

6. Oliaty koberec

opravoval Ivan
(max. 20 bodov)

Z definície susednosti a povoleného obklopovania si môžeme všimnúť, že nám na riešenie úlohy stačí vedieť, ktoré oblasti sú v ktorých, teda dvojfarebný graf, konkrétne strom oblastí alternujúcich farieb. Použitie čistiaceho resp. špiniaceho prostriedku na oblasti potom zodpovedá prefarbeniu vrcholu a zjednoteniu so svojimi susedmi, čo nám zanechá alternujúcu povahu farieb.

Úlohou je zredukovať strom na jeden biely vrchol. Najprv si môžeme všimnúť, že biele listy sú zanedbateľné – prefarbovať ich nemá zmysel, čistenie koberca nám vôbec neovplyvnia a môžeme ich zmazať na zjednodušenie. Druhú vec, ktorú si treba všimnúť a veľa riešiteľov si nevšimlo, je, že koreň stromu nemá žiaden špeciálny význam.

Ak má najdlhšia cesta dĺžku l (l je počet hrán v ceste a je párne), tak určite potrebujeme aspoň $l/2+1$ farbení, lebo pri jednom farbení cestu môžeme skrátiť maximálne o 2 a keďže listy sú čierne, na konci nám zostane čierny vrchol, ktorý potrebujeme ešte prefarbiť na biely. Tvrdenie platí pre všetky grafy okrem prázdneho (na začiatku bol iba biely list), ktorý farbiť nepotrebujeme.

Aby sme ukázali, že nám $l/2+1$ aj stačí a je teda riešenie, stačí si uvedomiť, že ak začneme prefarbovať v strede najdlhšej cesty, a opakujeme to $l/2+1$ ráz, koberec vyčistíme. Inými slovami. Strom zavesíme tak, aby hĺbka stromu bola čo najmenšia.

Na spočítanie najdlhšej cesty budeme opakovane z grafu odstraňovať všetky listy. Počet opakovaní bude polovica z maximálnej dĺžky cesty, ktorá je párna, takže nám na konci určite zostane jeden vrchol, nie dva spojené hranou.

Náš program najprv pomocou DFS ofarbí každú oblasť jednou farbou, potom z nich urobí strom a horeuvedeným spôsobom spočíta riešenie. Časová a pamäťová zložitosť: Vyrábanie stromu z mapy trvá lineárny čas od veľkosti mapy a tiež si potrebujeme celú mapu pamätať. Teda časová a pamäťová zložitosť tejto časti sú $O(RS)$. A odmazávanie listov zo stromu trvá lineárne od veľkosti stromu a tá nebude väčšia ako veľkosť mapy. Takže celková časová a pamäťová zložitosť je $O(RS)$.

Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe zmluvy č. LPP-0103-09

Listing programu:

```
#include <iostream>
#include <cstdlib>
#include <vector>
#include <string>
#include <stack>

using namespace std;

int N, M;
vector<string> koberec; // vstup
vector<vector<int> > znak; // znacenie oblasti
int oblasti;

int dx[8] = {-1, 0, 0, 1,-1,-1, 1, 1};
int dy[8] = { 0,-1, 1, 0,-1, 1,-1, 1};

// graf
vector<vector<int> > susedia;
vector<int> hran; // kolko zostalo hran
vector<bool> spinavy;

void nacitaj()
{
    cin >> N >> M;
    koberec.resize(N);
    for (int i = 0; i < N; ++i)
        cin >> koberec[i];
}

void dfs_oznac_oblast(int y, int x, int zn)
{
    if (znak[y][x])
        return;
    znak[y][x] = zn;
    char t = koberec[y][x];
    int ds = (t == '.' ? 8 : 4);
    for (int i = 0; i < ds; ++i) {
        int y1 = y + dy[i];
        int x1 = x + dx[i];
        if ((y1 >= 0) && (y1 < N) &&
                (x1 >= 0) && (x1 < M) &&
                (koberec[y1][x1] == t))
            dfs_oznac_oblast(y1, x1, zn);
    }
}

void oznac_oblasti()
{
    znak.resize(N);
    for (int i = 0; i < N; ++i)
        znak[i].resize(M);

    oblasti = 1;
    for (int i = 0; i < N; ++i)
        for (int j = 0; j < M; ++j)
            if (!znak[i][j])
                dfs_oznac_oblast(i, j, oblasti++);
}
```


Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe zmluvy č. LPP-0103-09

```
void vytvor_graf()
{
    susedia.resize(oblasti);
    spinavy.resize(oblasti, false);
    vector<bool> t(oblasti, false);
    t[1] = true;
    for (int i = 0; i < N; ++i)
        for (int j = 0; j < M; ++j)
            if (!t[znak[i][j]]) {
                int z = znak[i][j];
                int z1 = (j>0 ? znak[i][j-1] : 0);
                t[z] = true;
                spinavy[z] = (koberec[i][j] == '*');
                susedia[z].push_back(z1);
                susedia[z1].push_back(z);
            }

    hran.resize(oblasti);
    for (int i = 0; i < oblasti; ++i)
        hran[i] = susedia[i].size();
}

int zostavajuca_hrana(int z)
{
    for (int i = 0; i < susedia[z].size(); ++i)
        if (hran[susedia[z][i]] != 0)
            return susedia[z][i];
}

int main() {
    nacitaj();
    oznac_oblasti();

    // sledujeme hrany. Ak mame iba 1 flak,
    // ziadne hrany nam nezostanu
    if (oblasti == 3) {
      cout << "1\n";
      return 0;
    }

    vytvor_graf();

    // odstranime ciste listy
    for (int i = 0; i < oblasti; ++i)
        if (!spinavy[i] && hran[i] == 1) {
            hran[i]--;
            hran[susedia[i][0]]--;
        }

    // hlavny algoritmus
    stack<int> Q[2];
    for (int i = 0; i < oblasti; ++i)
        if (hran[i] == 1)
            Q[1].push(i);

    int rounds = 0;
    int r = 1;
    while (!Q[r].empty()) {
        while (!Q[r].empty()) {
```


Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe zmluvy č. LPP-0103-09

```
            int z1 = Q[r].top();
            Q[r].pop();
            if (!hran[z1])
                continue;
            int z2 = zostavajuca_hrana(z1);
            hran[z1]--;
            hran[z2]--;
            Q[!r].push(z2);
        }
        r = !r;
        rounds++;
    }

    cout << rounds << endl;
    return 0;
}
```

7. Odhaľ komunikačnú sieť

opravovala Halucinka
(max. 20 bodov)

Ideme odhaľovať strom. Keď počujeme slovo strom, tak väčšinou prvá užitočná vec je si ho niekde zakoreniť. Takže vrchol 1 prehlásime za koreň stromu. Teraz sa naša úloha mení na to, aby sme každému ďalšiemu vrcholu našli otca (potom vieme všetky hrany určit veľmi ľahko, bude to pár $X, otec(X)$).

To, čo vieme ľahko nájsť, sú všetci predkovia vrcholu X. Títo určite ležia medzi koreňom a vrcholom X. Takže spýtame sa N otázok typu $(1, X, Y)$ pre všetky možné vrcholy Y. Tam kde dostaneme kladnú odpoveď vieme, že Y je predok X. Takto si vieme zistiť predkov pre všetky vrcholy v strome na $O(N^2)$ otázok a zároveň si ich niekde uložiť (minieme na to $O(N^2)$ pamäte).

Jediné, čo ostáva, je vyrobiť z týchto informácií otca pre každý vrchol. Keď máme vrchol X, tak vieme, že má rovnakých predkov ako jeho otec, až na to, že tam má o jedného predka viac – samotného otca. A zároveň otec X má počet predkov o 1 menší ako X. Takže jediné čo treba spraviť na to, aby sme našli otca X je pozrieť sa na jeho predkov a zistiť u ktorého z nich je počet predkov menší o 1. Týmto prístupom sme na $O(N^2)$ otázok v $O(N^2)$ čase a pamäti získali celý strom.

To ale stále nie je optimálne riešenie. To, čo by sme chceli, je znížiť pamäťovú náročnosť (keďže časom ukážeme, že na menej ako $O(N^2)$ otázok sa to vlastne nedá). Spravíme to nasledovne: Vyrobíme si pole $best[i]$, kde si pre každý vrchol pamätáme jeho zatiaľ najbližšieho predka (na začiatku to inicializujeme na hodnotu koreňa, čiže 1). Teraz zoberieme vrchol X a zistíme všetkých jeho potomkov (otázkami typu $(1, Y, X)$). Nech Y je nejaký potomok X. Potom chceme vedieť, či X je bližší predok Y ako doteraz nájdený $best[Y]$. To spravíme jednoducho. Ak je $best[Y]$ tiež potomok X, tak X je vzdialenejší predok, takže netreba nič vylepšovať. Ak naopak $best[Y]$ nie je potomok X, tak sme našli bližšieho predka pre Y.

Keď tento postup spravíme pre všetky vrcholy X, tak sme potom každému vrcholu našli najbližšieho predka – otca. Takže máme počet otázok a čas $O(N^2)$ a pamäť $O(N)$.

Ešte krátke pojednanie o tom, prečo to na menej ako $O(N^2)$ otázok nejde. Predstavme si fakt veľký strom taký, že všetky vrcholy visia za jeden vrchol (jeden vrchol má $N-1$ synov a ostatné visia pod ním). A nech program položí rádovo menej ako N^2 otázok a správne odhalí tento strom. Z počtu jeho otázok vyplýva, že na nejakú dvojicu synov sa vôbec nespýtal (táto dvojica sa spolu nevyskytla v žiadnej otázke). Ten zmeňme náš strom tak, že týchto 2 synov zavesíme jedného pod druhého (strom bude taký, že koreň bude mať $N-2$ synov z toho jeden z nich má ešte syna a ostatné sú listy). Keďže sa program na túto pozmenenú dvojicu

Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe zmluvy č. LPP-0103-09

nikdy naraz nepýtal, tak dostane úplne rovnaké odpovede ako pre pôvodný strom a teda musí odpovedať zle.

Listing programu:

```
#include <cstdio>

bool ask(int a, int b, int c) {
    printf("Lezi %d medzi %d a %d? (0 pre nie, hocico ine pre ano)\n", c, a, b);
    int x;
    scanf("%d", &x);
    return x != 0;
}

int main(){
    int N;
    scanf("%d", &N);
    int *best = new int[N + 1];
    bool *pot = new bool[N + 1];
    for(int i = 1; i <= N; i++)
        best[i] = 1;

    for(int i = 2; i <= N; i++){
        // Zistime vsetkych potomkov i
        for(int j = 1; j <= N; j++){
            if(j == i)
                pot[j] = false;
            else
                pot[j] = ask(1, j, i);
        }

        for(int j = 1; j <= N; j++){
            if(!pot[j])  // Ak j nie je potomok nic nerobime
                continue;
            if(pot[best[j]])  // Ak sucasny najlepsi predok je potomok i, tiez
              continue;       // nic nerobime

            best[j] = i;  // Inac vylepsime
        }
    }

    // Vypiseme dvojice (vrchol, otec)
    for(int i = 2; i <= N; i++)
        printf("%d %d\n", i, best[i]);
}
```

8. O dračej invázii

vzorák písal Bob
(max. 25 bodov)

Základom úspešného vyriešenia tejto úlohy bolo nezľaknúť sa jej vysokého čísla. Vám, odvážlivcom, ktorí získali 25 bodov, gratulujem... a pre ostatných je tu aspoň vzorák.

Na mapu kráľovstva sa budeme pozerať ako na graf (hrad a mestá budú vrcholmi a cesty zase hranami). Zoberme si nejaké mesto u, v ktorom žije nešťastný zemepán. Keďže sa z u nedá dostať do hradu, nemôže to ísť ani zo žiadneho susedného mesta. Preto si ofarbíme všetky vrcholy, ktoré susedia s mestami nešťastných zemepánov.

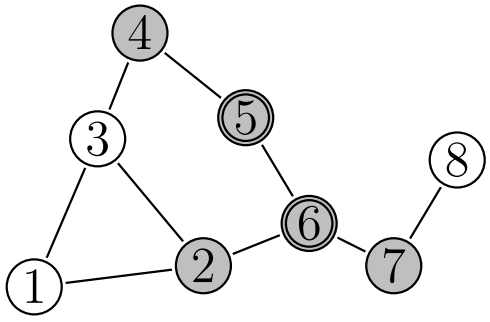
Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe zmluvy č. LPP-0103-09

Môžeme si byť istí, že kto vkročí na ofarbený vrchol, ten už do hradu nedôjde (dokonca ho v tom vrchole môže zožrať drak). Preto v našom grafe rovno zakážme prechod cez ofarbené vrcholy.

Takže teraz o niektorých vrcholoch už s určitosťou vieme, že sú od hradu odrezané (tie ofarbené a navyše tie, z ktorých sa nedá dôjsť do hradu bez navštívenia ofarbeného vrcholu). Otázkou zostáva, či sa dá dosiahnuť, aby to boli len tieto vrcholy (a žiadne navyše).

Odpoveď: dá. Napríklad ak by sa draci usadili práve v tých ofarbených vrcholoch, z ktorých sa neozvali zemepáni, vysvetlilo by to všetky sťažnosti na vstupe a počet miest odrezaných od hradu by sa nezmenil.

Príklad: na grafe vpravo sa z vrcholov 5 a 6 sťažovali zemepáni. Vrcholy 2, 4, 5, 6 a 7 sme ofarbili, tým zostal vrchol 8 odrezaný od hradu. Draci sa mohli usadiť napríklad vo vrcholoch 2, 4 a 7.



Na výstup potrebujeme vypísať počet vrcholov, z ktorých sa nedá dostať do hradu. Jednoduchšie je ale prehľadávaním z vrcholu 1 zistiť K' – počet vrcholov, z ktorých sa do hradu dostať dá, a vypísať $N - K'$.

Časová aj pamäťová zložitosť riešenia je lineárna od veľkosti grafu, teda $O(M + N)$.

Ak sa budete cez prázdniny nudiť, skúste si vyriešiť inú verziu tejto úlohy, v ktorej nás zaujíma minimálny počet drakov. Pozor, je trochu ťažšia.

Listing programu:

```
#include <iostream>
#include <vector>
using namespace std;

vector<vector<int> > G;
vector<bool> V; // V[x] <=> x je ofarbený alebo navštívený DFS-kom
int k = 0;

void DFS(int x){
    V[x] = true;
    ++k;

    for(vector<int>::iterator i = G[x].begin(); i != G[x].end(); ++i)
        if(!V[*i])
            DFS(*i);
}

int main(){
    int n, m, z;
    cin >> n >> m >> z;

    G.resize(n);
    for(int i = 0; i < m; ++i){
        int a, b;
        cin >> a >> b;
        --a, --b;
        G[a].push_back(b);
        G[b].push_back(a);
    }
```


Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe zmluvy č. LPP-0103-09

```
    V.resize(n, false);
    for(int i = 0; i < z; ++i){
        int x;
        cin >> x;
        --x;

        for(vector<int>::iterator j = G[x].begin(); j != G[x].end(); ++j)
            V[*j] = true;  // ofarbíme susedov
    }

    DFS(0);
    cout << n - k << endl;
}
```


Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe zmluvy č. LPP-0103-09

Výsledková listina po 1. kole kategórie KSP-Z

	Meno a priezvisko	Škola	Trieda	1	2	3	4	5	Σ
1	Drevenák Marek	Gym. Veľká Okružná Žilina	3	10	8	10	15	15	58
1	Greššák Jerguš	Gym. Mudroňova Prešov	2	9	10	9	15	15	58
3	Krajčovič Matej	Gym. Jura Hronca BA	2	10	7	10	15	15	57
4	Hudec Roman	Gym. Nové Zámky	3	10	8	10	15	13	56
4	Šafin Jakub	Gym. P. Horova Michalovce	2	10	6	10	15	15	56
6	Balog Matej	Gym. Grösslingová BA	4	10	5	10	15	15	55
6	Mikuš Milan	Gym. Ul. 1. mája Trenčín	3	10	10	10	15	10	55
8	Marko Jozef	Gym. Lettricha Martin	2	10	7	7	15	15	54
9	Farkaš Viktor	Gym. Jura Hronca BA	4	8	5	10	15	14	52
9	Klembarová Barbora	Gymnázium iné	3	10	5	7	15	15	52
11	Rydzi Filip	GLN Tomášikova BA	3	8	5	9	15	14	51
11	Vozárová Viktória	Gym. Jura Hronca BA	1	8	4	10	15	14	51
13	Ivan Lukáš	Gym. Jura Hronca BA	1	8	4	10	15	13	50
13	Rabatin Rastislav	Gym. Jura Hronca BA	2	10	9	1	15	15	50
13	Trungel Tomáš	Gymnázium Levice	4	10	6	4	15	15	50
16	Gallo Matej	SPŠE Nové Zámky	2	9	7	3	15	13	47
17	Součková Kamila	Ev. lýceum BA	2	10	6	10	15	5	46
18	Mandák Lukáš	Gym. Rajec	3	10	6	10	15	2	43
19	Petrucha Jaroslav	Gym. Metodova BA	2	10	7	10	0	14	41
20	Ille Ondrej	Gym. Jura Hronca BA	4	9	7	10	1	13	40
21	Sklenčárová Gabriela	Gym. Sečovce	2	8	6	10		15	39
22	Pecha Petr	SPŠS Vsetín	4	6	7	10	0	15	38
23	Hoos Andrej	Gym. Jura Hronca BA	3	7	4	8	4	14	37
23	Jasenčáková Katarína	Gym. Veľká Okružná Žilina	3	7	5	10		15	37
23	Kováčová Barbora	Škola pre mim. nadané deti BA	1	10	6	6		15	37
26	Kucera Roman	Gym. Nedožerského Prievidza	3	7	6	10	11		34
26	Turlík Tomáš	Gym. Mudroňova Prešov	2	8	5	3	15	3	34
28	Batmendijn Eduard	Gymnázium iné	0	9	6	4		13	32
28	Paulovič Jakub	Gym. Metodova BA	2	10	6	3		13	32
30	Smolík Michal	Gym. Grösslingová BA	2	9	3	4	15		31
30	Vajdová Mária	Gym. P. de Coubertina Piešťany	3	8	3	3	4	13	31
32	Lukačka Miloš	Gym. Ľudovíta Štúra Trenčín	3	10	4	1	15		30
32	Vlček Andrej	Evanjelické gym. J. Tranovského Lipt. Mikuláš	3	10		8		12	30
34	Filek Martin	Gym. Trstená	4	6	7	10	4		27
34	Halajová Barbora	Gym. Veľká Okružná Žilina	3	10	5			12	27
34	Macko Vladimír	Gym. Ľ. Štúra Zvolen	2	3	10			14	27
34	Šuppa Marek	SKŠ Nitra, Gym. sv. Cyrila a Metoda	2	8	6	4	9	0	27
38	Kubincová Petra	Škola pre mim. nadané deti BA	4	7	5		0	14	26
38	Porázik Matúš	Gym. Alejová Košice	1	8	4	3	0	11	26
40	Tokarova Natalia	Gym. Mudroňova Prešov	2	7	3	4		11	25
41	Gafurov Askar	Gym. Grösslingová BA	2	9	4	10			23
41	Kurdelová Alžbeta	Škola pre mim. nadané deti BA	1	8				15	23
43	Ficková Klára	Gym. Poštová Košice	3	10	8	4			22
43	Minárik Jakub	Gym. Jura Hronca BA	4	6	6	8	1	1	22
45	Bednár Stanislav	Gym. Jura Hronca BA	1	7	3	0		11	21
45	Kališ Vladimír	SOŠ Handlová	3	8	6	1	4	2	21
45	Kurtulík Matej	Gym. Námestovo	3	8	5	8	0		21
45	Slivka Norbert	Gym. Tajovského B. Bystrica	1	6	7	3		5	21
45	Šuník Martin	Gym. Veľká Okružná Žilina	3	10	6	5	0		21
50	Bednár Stanislav	Gym. Jura Hronca BA	1	7	2			11	20

Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe zmluvy č. LPP-0103-09

	Meno a priezvisko	Škola	Trieda	1	2	3	4	5	Σ
51	Pančík Juraj	Gym. Tajovského B. Bystrica	1	7	6	2	4		19
52	Berta Peter	Gym. Jura Hronca BA	1	5	2			11	18
52	Bodinger Marek	Gym. Jura Hronca BA	1	5	2			11	18
52	Nagy Imrich	Gym. Hronská BA	1	10	4	4			18
55	Rabatin Branislav	Gym. Jura Hronca BA	3	7	5	5			17
55	Tunová Anna	Gym. Párovská Nitra	2	7	6	4			17
57	Bock Michal	Gym. Grösslingová BA	2	10	6		0		16
58	Bočan Peter	Gym. P. Horova Michalovce	2	9	6				15
58	Glončák Vladan	Gym. Ľudovíta Štúra Trenčín	2	9	6		0		15
60	Branderský Gabriel	SPŠE Piešťany	4	4	5	0	5		14
60	Pistrakova Alexandra	Gym. Poštová Košice	3	6	4	4			14
60	Smolík Martin	Gym. Grösslingová BA	3	5	3	6			14
63	Bajtoš Martin	Gym. Školská Spiš. Nová Ves	2	8	4	1			13
63	Dlužanský Michal	Gym. P. Horova Michalovce	2	7	5	1	0		13
63	Fabián Štefan	Gym. Krompachy	3	5	7	1	0		13
63	Hraška Peter	Gym. Grösslingová BA	2	9	4		0		13
63	Lúčna Nina	Gym. P. de Coubertina Piešťany	2	4	5	1	3		13
63	Marko Jozef	SPŠ Myjava	4	9	2	1	0	1	13
69	Jalovecký Jerguš	Gym. Jura Hronca BA	1	7	2	3			12
69	Klugová Ema	Gym. Jura Hronca BA	1	8	3	1	0		12
71	Badin Matej	Gym. Jura Hronca BA	1	4	5	2	0		11
71	Cvacho Rudolf	Gym. Trstená	3	4	4	3	0		11
73	Dancs Tibor	Gymnázium iné	2	6	3			1	10
73	Juhos Ján	Gym. P. Horova Michalovce	3	4	5	1			10
73	Korbela Michal	Gym. Bánovce nad Bebravou	1	8	2	0	0		10
73	Lipovský Mário	Gym. Jura Hronca BA	1	7	3		0		10
77	Brisuda Rudolf	Gym. Trstená	4	4	5				9
77	Krumpal Rudo	Gym. Jura Hronca BA	4	5	3	1		0	9
77	Macák Martin	Gym. Šurany	4	6	3				9
80	Hvostaľ Vladimír	Gym. P. Horova Michalovce	2	6	2				8
80	Ivanov Marián	Gym. Jura Hronca BA	2	5		3			8
80	Karlík Radoslav	Súkromná SOŠ Humanus Via	2	5	1	1	0	1	8
80	Šalata Pavol	Gym. P. Horova Michalovce	2	8					8
84	Bobuľa Daniel	SPŠ Martin	0	4	3				7
85	Boža Vladimír	Gym. Tatarku Poprad	7			–	6		6
85	Juračka Marek	Súkromná SOŠ Humanus Via	3	4	1	1	0		6
87	Staňo Jozef	Gym. Veľká Okružná Žilina	3	0					0

Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe zmluvy č. LPP-0103-09

Výsledková listina po 1. kole kategórie KSP-O

	Meno a priezvisko	Škola	Trieda	4	5	6	7	8	Σ
1	Horňák Marián	Gym. Párovská Nitra	3	15	15	20	20	25	95
2	Hozza Ján	Gym. Jura Hronca BA	4	15	13	18	20	25	91
3	Anderle Michal	Gym. Haličská Lučenec	4	15	15	4	20	25	79
3	Mariš Andrej	Gym. Piaristická Nitra	2	15	15	4	20	25	79
5	Greššák Jerguš	Gym. Mudroňova Prešov	2	15	15	2	17	25	74
6	Kučera Martin	Gym. Golianova Nitra	4	15	14	6	13	25	73
7	Korbaš Rafael	Gym. Hronská BA	4	15	15	3	13	25	71
8	Brandys Jozef	Gym. Námestovo	3	15	15	4		25	59
8	Součková Kamila	Ev. lýceum BA	2	15	5	5	9	25	59
10	Balog Matej	Gym. Grösslingová BA	4	15	15		17		47
11	Mandák Lukáš	Gym. Rajec	3	15	2			25	42
12	Fulla Peter	SPŠ strojnícka Spišská Nová Ves	6	15				25	40
12	Špano Marek	Gym. Jura Hronca BA	4	15				25	40
14	Drevenák Marek	Gym. Veľká Okružná Žilina	3	15	15	6			36
15	Rabatin Rastislav	Gym. Jura Hronca BA	2	15	15	2			32
15	Večerík Matej	Škola pre mim. nadané deti BA	4	15	14	3			32
17	Farkaš Viktor	Gym. Jura Hronca BA	4	15	14	2			31
17	Marko Jozef	Gym. Lettricha Martin	2	15	15	1			31
17	Trebichavský Richard	Gym. Jura Hronca BA	3	8	9	1	13		31
20	Dresslerova Anna	Gym. Jura Hronca BA	4	15	15				30
20	Klembarová Barbora	Gymnázium iné	3	15	15				30
20	Krajčovič Matej	Gym. Jura Hronca BA	2	15	15				30
20	Livora Tomáš	Gym Javorová Spiš. Nová Ves	4	15	15				30
20	Mrocková Mária	Gym. Jura Hronca BA	4	15	15				30
20	Pinter Martin	Gym. Jura Hronca BA	4	0	15	2	13		30
20	Šafin Jakub	Gym. P. Horova Michalovce	2	15	15			0	30
20	Trungel Tomáš	Gymnázium Levice	4	15	15				30
28	Mrocek Jakub	Gym. Jura Hronca BA	3	11	15	3			29
28	Rydzi Filip	GLN Tomášikova BA	3	15	14				29
28	Vozárová Viktória	Gym. Jura Hronca BA	1	15	14				29
31	Gallo Matej	SPŠE Nové Zámky	2	15	13				28
31	Hudec Roman	Gym. Nové Zámky	3	15	13				28
31	Ivan Lukáš	Gym. Jura Hronca BA	1	15	13				28
34	Toman Viktor	Gym. Golianova Nitra	4	15	11				26
35	Mikuš Milan	Gym. Ul. 1. mája Trenčín	3	15	10				25
36	Hoos Andrej	Gym. Jura Hronca BA	3	4	14				18
36	Pecha Petr	SPŠS Vsetín	4	0	15	3			18
36	Turlík Tomáš	Gym. Mudroňova Prešov	2	15	3				18
39	Vajdová Mária	Gym. P. de Coubertina Piešťany	3	4	13				17
40	Marko Jozef	SPŠ Myjava	4	0	1	4	11		16
41	Birkus Róbert	SPŠE Nové Zámky	4	15					15
41	Hübsch Ondřej	Gymnázium iné	1	15					15
41	Ille Ondrej	Gym. Jura Hronca BA	4	1	13	1			15
41	Jasenčáková Katarína	Gym. Veľká Okružná Žilina	3		15				15
41	Kováčová Barbora	Škola pre mim. nadané deti BA	1		15				15
41	Kurdelová Alžbeta	Škola pre mim. nadané deti BA	1		15				15
41	Lukačka Miloš	Gym. Ľudovíta Štúra Trenčín	3	15					15
41	Sklenčárová Gabriela	Gym. Sečovce	2		15				15
41	Smolík Michal	Gym. Grösslingová BA	2	15					15
50	Kubincová Petra	Škola pre mim. nadané deti BA	4	0	14				14

Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe zmluvy č. LPP-0103-09

	Meno a priezvisko	Škola	Trieda	4	5	6	7	8	Σ
50	Macko Vladimír	Gym. Ľ. Štúra Zvolen	2		14				14
50	Petrucha Jaroslav	Gym. Metodova BA	2	0	14				14
50	Stahr Daniel	Gym. J. Jungmanna Litoměřice	4	14					14
54	Batmendijn Eduard	Gymnázium iné	0		13				13
54	Paulovič Jakub	Gym. Metodova BA	2		13				13
56	Halajová Barbora	Gym. Veľká Okružná Žilina	3		12				12
56	Vlček Andrej	Evanjelické gym. J. Tranovského Lipt. Mikuláš	3		12				12
58	Bednár Stanislav	Gym. Jura Hronca BA	1		11				11
58	Bednár Stanislav	Gym. Jura Hronca BA	1		11				11
58	Berta Peter	Gym. Jura Hronca BA	1		11				11
58	Bodinger Marek	Gym. Jura Hronca BA	1		11				11
58	Kucera Roman	Gym. Nedožerského Prievidza	3	11					11
58	Porázik Matúš	Gym. Alejová Košice	1	0	11				11
58	Tokarova Natalia	Gym. Mudroňova Prešov	2		11				11
65	Šuppa Marek	SKŠ Nitra, Gym. sv. Cyrila a Metoda	2	9	0				9
66	Boža Vladimír	Gym. Tatarku Poprad	7	6					6
66	Kališ Vladimír	SOŠ Handlová	3	4	2				6
66	Plavák Dušan	Gym. Trstená	4	6					6
69	Branderský Gabriel	SPŠE Piešťany	4	5					5
69	Lipovský Mário	Gym. Jura Hronca BA	1	0				5	5
69	Slivka Norbert	Gym. Tajovského B. Bystrica	1		5				5
72	Filek Martin	Gym. Trstená	4	4					4
72	Pančík Juraj	Gym. Tajovského B. Bystrica	1	4					4
74	Lúčna Nina	Gym. P. de Coubertina Piešťany	2	3					3
75	Minárik Jakub	Gym. Jura Hronca BA	4	1	1				2
76	Dancs Tibor	Gymnázium iné	2		1				1
76	Karlík Radoslav	Súkromná SOŠ Humanus Via	2	0	1				1
78	Badin Matej	Gym. Jura Hronca BA	1	0					0
78	Bock Michal	Gym. Grösslingová BA	2	0					0
78	Cvacho Rudolf	Gym. Trstená	3	0					0
78	Dlužanský Michal	Gym. P. Horova Michalovce	2	0					0
78	Fabián Štefan	Gym. Krompachy	3	0					0
78	Glončák Vladan	Gym. Ľudovíta Štúra Trenčín	2	0					0
78	Hraška Peter	Gym. Grösslingová BA	2	0					0
78	Juračka Marek	Súkromná SOŠ Humanus Via	3	0					0
78	Klugová Ema	Gym. Jura Hronca BA	1	0					0
78	Korbela Michal	Gym. Bánovce nad Bebravou	1	0					0
78	Krumpal Rudo	Gym. Jura Hronca BA	4		0				0
78	Kurtulík Matej	Gym. Námestovo	3	0					0
78	Šuník Martin	Gym. Veľká Okružná Žilina	3	0					0

Táto práca bola podporovaná Agentúrou na podporu výskumu a vývoja na základe zmluvy č. LPP-0103-09