Korešpondenčný seminár z programovania
XXI. ročník, 2003/04
Katedra vyučovania informatiky FMFI UK,
Mlynská Dolina, 842 48 Bratislava

KSP finančne podporujú: aSc – Applied Software Consultants spol. s r.o.
MICROSTEP-MIS spol. s r.o.

Vzorové riešenia 2. kola letnej časti

Milí riešitelia!
Ďalší ročník KSP je úspešne za nami (a vami). Všetkým vám blahoželáme k dosiahnutým výsledkom a dúfame, že ste sa čo-to naučili a bavilo vás to. S najlepšími z vás sa na jeseň stretneme na týždňovom sústredení, pozvánky očakávajte tak koncom septembra. Ale ako to už chodí, ešte pred tým vás čakajú zaslúžené prázdniny plné slnka, vody a skvelých kamarátov. Poriadne si ich užite! Keď sa oddýchnutí vrátite späť do škôl, vráti sa aj váš obľúbený seminár.

War doesn't show who's right, just who's left.

KSPáci

1. O binárnom kamzíkovi

opravovala Monika
(max. 15 bodov)

Väčšina vašich riešení sa dala rozdeliť do dvoch skupín: riešenie sústavy lineárnych rovníc pomocou Gaussovej eliminácie alebo použitie backtracku. Za riešenie lineárnych rovníc som dávala plný počet bodov, za backtrack som dávala 9-7 bodov. Za chýbajúci odhad pamäťovej a časovej zložitosti som strhávala po bode za každý.

Vzorové riešenie V prvom rade je treba si uvedomiť, že výmenou poradia, v akom majú skupiny robiť výmeny, sa nič nezmení. Ďalej je nutné uvedomiť si, že pokiaľ by mala urobiť nejaká skupina dve výmeny, je to to isté, ako keby neurobila žiadnu. Teda stačí uvažovať o tom, že skupina urobí najviac jednu výmenu.

Uvažujme, že máme m skupín a n ľudí. Nech $a_{i,j}$ je 1, ak i-ty človek je v skupine j a 0 inak. Nech c_i je stav, v ktorom je človek i na začiatku. Stav človeka môže byť 0, vtedy je v podrepe alebo 1 a vtedy stojí. Zaujíma nás, ktoré skupiny zmenia svoj stav. Preto pre každú skupinu budeme mať jednu neznámu – x_i, ktorá môže tiež nadobúdať hodnotu 0 (skupina svoj stav nezmení) alebo 1 (skupina svoj stav zmení). Pre každého človeka i si všimnime nasledujúci výraz: $a_{i,1} * x_1 + a_{i,2} * x_2 + \ldots a_{i,m} * x_m$. Tento výraz predstavuje počet zmien, ktoré človek i urobí – ak pre nejake $a_{i,j} = x_j = 1$, tak zmení vďaka skupine j svoj stav (pokiaľ $a_{i,j} = 1$, znamená to, že človek sa v skupine j nachádza a pokiaľ $x_j = 1$, znamená to, že skupina urobí výmenu). Každý človek má skončiť v stoji, preto pre každého človeka má byť číslo $c_i + a_{i,1} * x_1 + a_{i,2} * x_2 + \ldots a_{i,m} * x_m$ nepárne. Toto môžeme zapísať ako rovnicu $c_i + a_{i,1} * x_1 + a_{i,2} * x_2 + \ldots a_{i,m} * x_m = 1$, pričom sčítanie uvažujeme modulo 2 (teda 1+1=0).

Po vytvorení takejto rovnice pre každého človeka dostaneme sústavu rovníc. Ako sa taká sústava lineárnych rovníc dá vyriešiť? Odpoveď dáva algoritmus zvaný Gaussová eliminácia. Vysvetlím túto metódu vo všeobecnosti (naša sústava rovníc je len jej špeciálnym prípadom). Majme teda sústavu lineárnych rovníc:

$$a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,m}x_m = b_1$$
$$a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,m}x_m = b_2$$
$$\ldots$$

$$a_{n,1}x_1 + a_{n,2}x_2 + \ldots + a_{n,m}x_m = b_m$$

Nech $a_{i,j} \in R,\ 1 \leq i \leq n,\ 1 \leq j \leq m$ a $b_i, 1 \leq i \leq m$ sú známe hodnoty a $x_i,\ 1 \leq i \leq m$ sú hľadané neznáme. S touto sústavou bez toho, aby sme niečo pokazili (teda zmenili množinu riešení) môžeme robiť napríklad tieto operácie:

1. vymeniť ľubovoľné dva riadky (tým sa sústava nezmení)
2. vynásobiť ľubovolnú rovnicu nenulovou konštantou c (riešenie sa nezmení, pretože pri výpočte môžeme krátiť touto konštantou celú rovnicu)
3. pripočítať nejakú rovnicu k inej rovnici (ak riešenie vyhovovalo rovniciam A a B, vyhovuje zjavne aj rovnici $A + B$, podobne ak vyhovuje rovniciam A a $A + B$, vyhovuje aj B)
4. vymeniť dva stĺpce v sústave (tým akoby preznačíme neznáme, teda prehodíme x_i a x_j)

Tieto operácie zvykneme volať ekvivalentné úpravy. Budeme sa snažiť pôvodnú sústavu rovníc upraviť do nasledujúceho tvaru (budeme ho ďalej nazývať trojuholníkový tvar):

$$\begin{aligned} 1x_1 + c_{1,2}x_2 + c_{1,3}x_3 + \ldots + c_{1,m}x_m &= d_1 \\ 1x_2 + c_{2,3}x_3 + \ldots + c_{2,m}x_m &= d_2 \\ 1x_3 + \ldots + c_{3,m}x_m &= d_3 \\ \ldots & \\ 1x_m &= d_m \end{aligned}$$

Pritom $c_{i,j}$ a d_i sú čísla, ktoré vznikli počas upravovania. Keby sme si hodnoty $c_{i,j}$ zapísali do štvorcovej matice, vidíme, že na jej hlavnej diagonále (zľava zhora doprava dole) budú jednotky a pod ňou bude trojuholník núl. Keď máme sústavu v takomto tvare, jej riešenie $x_1, x_2, \ldots, x_m$ vypočítame takto: Posledná rovnica je tvaru: $1x_m = d_m$ a teda vieme, že $x_m = d_m$. Predposledná rovnica je tvaru: $1x_{m-1} + c_{n-1,m}x_m = d_{m-1}$ a z predchádzajúcej rovnice vieme hodnotu x_m. Z toho ľahko vypočítame aj x_{m-1}: $x_{m-1} = d_{m-1} - c_{n-1,m}x_m$. Netreba zabúdať, že poznáme všetky hodnoty d_{m-1} aj $c_{n-1,m}$. Takýmto spôsobom postupujeme ďalej a dopočítame ostatné neznáme.

Teraz nám ostáva ešte zopár otázok: Ako upravíme počiatočnú sústavu rovníc do tohto nového tvaru? A ide to vôbec vždy urobiť? Pokúsme sa upraviť pôvodnu sústavu do trojuholníkového tvaru. Na začiatku zistíme, či $a_{1,1}$ je nenulové. Ak nie je, tak si nájdeme ľubovolný iný riadok, ktorý má číslo $ai, 1$ nenulové a vymeníme ich. Ak takýto riadok neexistuje, vymeníme stĺpec obsahujúci neznáme x_i za stĺpec na konci, ktorý nie je nulový. Teraz už máme $a_{1,1}$ nenulové. Budeme postupovať pre všetky $i > 1$ takto: Riadok i prenásobíme $a_{1,1}$ a odčítame od neho riadok 1 prenásobený číslom $a_{i,1}$. Výsledkom bude rovnica:

$$0x_1 + (a_{1,1}a_{i,2} - a_{i,1}a_{1,2})x_2 + \ldots + (a_{1,1}a_{i,m} - a_{i,1}a_{1,m})x_m = a_{1,1}b_i - a_{i,1}b_1$$

Takýmto spôsobom sa nám podarilo získať koeficient pri x_1 nulový. Rovnaký postup uplatníme aj na ostatných riadkoch a získame sústavu rovníc, kde v prvom stĺpci budeme mať $a_{1,1}x_1$ a pod ním uz iba $0x_i$, $i > 1$. Už by sme len potrebovali aby sme mali v prvom riadku namiesto $a_{1,1}x_i$ iba $1x_i$. To urobíme jednoducho - prvý riadok prenásobíme číslom $1/a_{1,1}$. Práve sa nám podaril prerobiť prvý stĺpec do tvaru ako by sme potrebovali mať v trojuholníkovom tvare. Takýto postup budeme uplatňovať aj ďalej. Skúsim tento postup popísať všeobecnejšie: Nech máme pôvodnú sústavu v takomto tvare, pričom $e_{i,i}$ je nenulové:

$$\begin{aligned} x_1 + e_{1,2}x_2 + \ldots + e_{1,m}x_m &= f_1 \\ 0x_1 + x_2 + \ldots + e_{2,m}x_m &= f_2 \\ 0x_1 + 0x_2 + x_3 + \ldots + e_{3,m}x_m &= f_3 \end{aligned}$$

$$\ldots$$

$$0x_1 + 0x_2 + 0x_3 + \ldots + 0x_{i-1} + e_{i,i}x_i + e_{i,i+1}x_{i+1} \ldots e_{i,m}x_m = f_i$$

$$0x_1 + 0x_2 + 0x_3 + \ldots + 0x_{i-1} + e_{i+1,i}x_i + e_{i+1,i+1}x_{i+1} \ldots e_{i+1,m}x_m = f_{i+1}$$

$$\ldots$$

$$0x_1 + 0x_2 + 0x_3 + \ldots + 0x_{i-1} + e_{n,i}x_i + e_{n,i+1}x_{i+1} \ldots e_{n,m}x_m = f_n$$

Ďalšie úpravy budú takéto: pre každé $j > i$ ak $e_{j,i}$ je nenulové, prenásobíme riadok j s hodnotou $e_{i,i}$ a odčítame od neho súčinu riadku i a $e_{j,i}$. Takto získame v sústave namiesto nenulového $e_{j,i}$ hodnotu 0. Na záver ešte prenásobíme i-ty riadok číslom $1/e_{i,i}$ a získame ďalší stĺpec do trojuholníkového tvaru.

Aké môžu nastať problémy?

- počet rovníc je vyšší ako počet neznámych. Vtedy vytvoríme diagonálnu maticu, vypočítame hodnoty neznámych a do zvyšných rovníc, ktoré sme nepoužili dosadíme aby sme overili, či vypočítané hodnoty neznámych sedia aj tu.
- počet rovníc je menší ako počet neznámych. V tomto prípade dosiahneme taký tvar, že v poslednej rovnici bude za diagonálou viac neznámych. Vtedy si stačí zvoliť ľubovolné hodnoty neznámych za týmto miestom a zvyšné dopočítať podľa upravených rovníc. Tento spôsob môžeme vždy použiť, pretože takáto sústava má nekonečno veľa riešení a my si vyberieme jedno riešenie.
- pokiaľ pri výpočtoch narazíme na riadok, ktorý obsahuje pri neznámych nulové koeficienty. Tu môže nastať prípad, že celá rovnica (aj ľavá aj pravá strana) je nulová. V takomto prípade túto rovnicu môžeme presunúť na koniec a ďalej neuvažovať (pre ľubovolné hodnoty neznámych bude vždy platiť). Iný prípad je, pokiaľ na ľavej strane dostaneme koeficienty nulové, a teda celá ľavá strana je nulová, avšak na pravej strane získame nenulové číslo. V takomto prípade sústava nemá žiadne riešenie.[1]
- pokiaľ počas úprav dostaneme každý koeficient pri neznámej x_i rovný 0. Vtedy stačí zvoliť ľubovoľné číslo za x_i.[2]

Listing programu:

```
const MAXN = 10;
      MAXM = 66;

var N,M:integer;
    matica : array[1..MAXN,1..MAXM+1] of boolean;
    vysledok : array[1..MAXM] of boolean;
    kto : array[1..MAXM] of integer;{sluzi na pamatanie si skupin ak robim vymenu
stlpcov}

Procedure vstup;
var i,j,K,r : integer;
begin
  read(N);
  read(M);

  for i:=1 to N do
    for j:=1 to M do
      matica[i,j] := false;
```

[1] Keďže naše úpravy boli ekvivalentné, každé riešenie musí spĺňať aj túto rovnicu. Nech ale do nej dosadíme čokoľvek, stále dostaneme $0 = c \neq 0$, čo platiť nebude.

[2] Premyslite si to.

```
  for i:=1 to N do
    matica[i,M+1] := true;

  for i:=1 to M do
    kto[i] := i;

  for i:=1 to M do
    begin
      read(K);
      for j:=1 to K do
        begin
          read(r);
          matica[r,i] := true;
        end;
    end;
end;

Procedure gauss;
var i,j,k : integer;
    temp : boolean;
    poc,temp2 : integer;
begin
  i := 1;
  poc := 0;

  while (i<= N) and (i <= M) do
    begin
      if not matica[i,i] then
        begin{hladam riadok s nenulovym matica[j,i+d]}
          j:=i+1;
          while (j <= N) and not (matica[j,i] ) do
            j:=j+1;
          if (j <= N) then  {ak som nasla riadok s nenulovym matica[j,i+d]}
            begin
              for k:=i to M+1 do{vymenim riadky}
                begin
                  temp := matica[i,k];
                  matica[i,k] := matica[j,k];
                  matica[j,k] := temp;
                end;
            end
          else{ak som nenasla riadok, tak cely stlpec necham tak a posuniem sa do
dalsieho}
            begin
              if M-poc > i then{ak este je "nenulovy" stlpec}
                begin
                  for j:=1 to N do
                    begin
                      temp := matica[j,i];
                      matica[j,i] := matica[j,M-poc];
                      matica[j,M-poc] := temp;
                    end;
                  temp2 := kto[i];
                  kto[i]:= kto[M-poc];
                  kto[M-poc] := temp2;
                  poc := poc+1;
                end
              else
                break;
            end;
```

```
      end
    else
      begin
        for j:=i+1 to N do
          begin
            if matica[j,i] then
              for k:=i to M+1 do
                matica[j,k] := matica[j,k] xor matica[i,k];
          end;
        i:=i+1;
      end;
  end
end;


Function dosadenie : boolean;
var i,j,min : integer;
   nulovy,spolu  : boolean;
begin

  for i:=1 to N do   {zistovanie nulovosti riadkov}
    begin
      nulovy:= true;
      for j:=1 to M do
        if matica[i,j] then nulovy:=false;
      if nulovy then{ak je na lavej strane 0}
        if matica[i,M+1] then{na pravej strane musi byt tiez nula}
          begin
            writeln('NIE JE RIESENIE');{inak nie je riesenie}
            dosadenie:=false;
            exit;
          end;
    end;

  min:=N;
  if min>M then min:=M;

  for i:= 1 to N do{osetrenie riadkov, ktore su nulove aj v diagonale}
    begin
      spolu := false;
      if not matica[i,i] then
        for j:=i+1 to M+1 do
          begin
            spolu := spolu xor matica[i,j];
          end;
      if spolu then vysledok[N]:=not vysledok[N];
    end;

  for i:=min downto 1 do
    begin
      if matica[i,i] then
        begin
          spolu := false;
          for j:=i+1 to M+1 do
            if matica[i,j] then
              spolu := spolu xor vysledok[j];
          if spolu then vysledok[i]:=false
          else vysledok[i]:=true;
        end;
```

```
    end;


  for i:=min+1 to N do {overenie, pokial je viac ludi, ci sedia vysledky}
    begin
      spolu := false;
      for j:=1 to M do
        if matica[i,j] then
          spolu := spolu xor vysledok[j];
      if spolu <> matica[i,M+1] then{nie je riesenie}
          begin
            writeln('NIE JE RIESENIE');
            dosadenie:=false;
            exit;
          end;
    end;

  dosadenie:=true;
end;

Procedure vysled;
var i : integer;
begin
  writeln('Zoznam skupin, ktore maju urobit zmenu:');
  for i:=1 to M do
    if vysledok[i] then
      write(kto[i],' ');
end;

begin
  vstup;
  gauss;
  if dosadenie then
    vysled;
end.
```

opravoval Paľo
(max. 15 bodov)

2. O zlatokopovi Kleofášovi

Potešili ste ma. Väčšina vašich riešení bola podobná vzoráku a mohla dostať maximálne 15 bodov. Pár exponenciálnych riešení mohlo dostať 8 bodov. Iné riešenia sa vlastne ani nevyskytovali. Poďme však rýchlo k veci:

Bane a autobusové spoje medzi nimi nám tvoria ohodnotený graf. Ukážeme si riešenie využívajúce techniku tzv. dynamického programovania. Označme $z_{i,j}$ maximálny počet kilogramov zlata, ktorý Kleofáš môže vyťažiť počas prvých j hodín tak, že sa po j hodinách bude nachádzať v bani i. Všetky $z_{i,j}$ nám tvoria tabuľku veľkosti $N \times T$, ktorú budeme postupne vypĺňovať. Všimnime si, že my máme vlastne zrátať maximálnu hodnotu v T-tom stĺpci našej tabulky – inými slovami maximálny počet kilogramov, ktoré Kleofáš môže vyťažiť počas prvých T hodín a skončí v ľubovoľnej bani.

Otázka znie, ako spočítať nejaké $z_{i,j}$. Treba rozobrať dve možnosti:

- Kleofášovi sa oplatilo, aby bol v bani i už hodinu pred tým (teda v čase $j-1$) a počas tejto hodiny ťažil v bani i. Dokopy teda mohol v tomto prípade vyťažiť $z_{i,j-1} + c_{i,j}$ kilogramov zlata.
- Kleofáš do bane i prišiel až v čase j z nejakej bane k a cesta mu z nej trvala d hodín. V tomto prípade ešte nestihol z bane i nič vyťažiť a dokopy teda mohol vyťažiť toľko

isto, ako pred začiatkom cesty z bane k do bane i, teda maximálne $z_{k,j-d}$ kilogramov zlata.

Takže nám stačí položiť $z_{i,j}$ rovné väčšiemu číslu z týchto dvoch prípadov. V druhom prípade musíme samozrejme vyskúšať všetky bane, z ktorých premáva autobus do bane i. Treba ešte ošetriť špeciálny prípad, keď sa do nejakej bane nedá vôbec v nejakom čase dostať. Toto si tiež môžeme zapamätať v našej tabuľke.

Ešte treba vyriešiť jeden menší problém. My sme mali nielen spočítať maximálny počet kilogramov zlata, ktoré vie Kleofáš vyťažiť, ale aj nájsť spôsob, akým to má urobiť. Nie je nič ľahšie. Na každom políčku našej tabulky si budeme pamätať okrem $z_{i,j}$ aj spôsob, akým sme ho našli. Konkrétne, ak sa Kleofášovi oplatí prísť do bane i až po j hodinách(nastal druhý prípad), tak si v tabuľke na i-tom riadku a j-tom stĺpci zapamätáme číslo bane, z ktorej sa Kleofáš do bane i dostal a ešte počet hodín, ktoré mu cesta trvala. V opačnom prípade (prvý prípad) si len zapamätáme, že Kleofáš v bani i už bol (a počas j-tej hodiny tam ťažil zlato).

Teraz môžeme postupne skonštruovať Kleofášovu cestu po baniach odzadu. Vieme, že Kleofáš bol v čase T v bani i. Pozrieme sa, odkiaľ sa tam dostal. Nech je to baňa j a bol v nej v čase k. Znovu sa pozrime do tabuľky, odkiaľ sa Kleofáš dostal do bane j v čase k, atď. Cestu si môžeme zapamätať do poľa a na konci vypísať odpredu.

Pre každú baňu si budeme pamätať zoznam baní, z ktorých do nej premáva autobus. Pamäťová zložitosť je potom zjavne $O(M + TN)$. Na spočítanie obsahu políčka tabulky na i-tom riadku a j-tom stĺpci potrebujeme prezrieť každého suseda i-tej bane. Teda na zrátanie jedného stĺpca potrebujeme urobiť $O(M + N)$ operácií. Na zistenie Kleofášovej cesty potrebujeme urobiť $O(T)$ operácií. Teda dokopy je to $O(T(M + N))$ operácií.

Listing programu:

```
#include <stdio.h>

#define MAXN 1000
#define MAXT 1000
#define INF 2000000000

struct SUSED {
   int v,d;
};

SUSED bane[MAXN][MAXN];
int pocsus[MAXN];
int n,m,t;

int maxtaz[MAXT][MAXN];
int pred[MAXT][MAXN];
int predt[MAXT][MAXN];
int akttaz[MAXN];

int cesta[MAXT];
int cas[MAXT];
int pces;

int main()
{
   int i,j,k;
   int a,b,c;
   int max;

   scanf("%d %d", &n,&m);

   for (i = 0; i < n; i++) {
```

```
    pocsus[i]=0;
    maxtaz[0][i]=-INF;
    pred[0][i]=-1;
    predt[0][i]=-1;
}

for (i = 0; i < m; i++) {
    scanf("%d %d %d", &a,&b,&c);
    b--;a--;
    bane[b][pocsus[b]].v=a;
    bane[b][pocsus[b]].d=c;
    bane[a][pocsus[a]].v=b;
    bane[a][pocsus[a]].d=c;
    pocsus[b]++;
    pocsus[a]++;
}


scanf("%d", &t);


pred[0][0]=0;
predt[0][0]=0;
maxtaz[0][0]=0;
for (i = 1; i <= t; i++) {
    for (j = 0; j < n; j++) {
        scanf("%d", &akttaz[j]);
    }
    for (j = 0; j < n; j++) {
        pred[i][j]=-1;
        predt[i][j]=-1;
        maxtaz[i][j]=-INF;
        for (k = 0; k < pocsus[j]; k++) {
            if (i-bane[j][k].d >= 0) {
                if (maxtaz[i][j] < maxtaz[i-bane[j][k].d][bane[j][k].v]) {
                    maxtaz[i][j] = maxtaz[i-bane[j][k].d][bane[j][k].v];
                    pred[i][j]=k;
                    predt[i][j]=i-bane[j][k].d;
                }
            }
        }
        if (maxtaz[i-1][j] != -INF) {
            if (maxtaz[i][j] < maxtaz[i-1][j]+akttaz[j]) {
                maxtaz[i][j]=maxtaz[i-1][j]+akttaz[j];
                pred[i][j]=-2;
                predt[i][j]=i-1;
            }
        }
    }
}

max=0;
for (i = 1; i < n; i++) {
    if (maxtaz[t][i] >= maxtaz[t][max]) {
        max=i;
    }
}

printf("Dokopy ziskas %d kilogramov zlata.\n",maxtaz[t][max]);
```

```
    cesta[0]=max;
    pces=1;
    i=t;
    do {
        cas[pces-1]=0;
        while (i>0 && pred[i][cesta[pces-1]]==-2) {
            cas[pces-1]++;
            i--;
        }
        if (i==0) break;
        cesta[pces]=pred[i][cesta[pces-1]];
        i=predt[i][cesta[pces-1]];
        cas[pces]=0;
        pces++;
    } while (i>0);


    printf("Doluj v %d. bani %d hodin.\n", cesta[pces-1]+1, cas[pces-1]);
    for (i = pces-2; i >= 0; i--) {
        printf("Presun sa do %d. bane.\n", cesta[i]+1);
        printf("Doluj v %d. bani %d hodin.\n", cesta[i]+1, cas[i]);
    }

    return 0;
}
```

3. O čokoláde

opravoval MišoF.
(max. 15 bodov)

Vďaka nie práve najjasnejšej formulácii zadania viacerí z vás riešili o niečo ľahšiu úlohu, keď predpokladali, že čokoládu lámeme tak, že si zvolíme čiaru a pozdĺž nej zlomíme postupne všetky aktuálne kúsky. Pôvodné zadanie chcelo povedať, že v jednom kroku si vyberieme jeden kúsok a prelomíme ho podľa niektorej z čiar. Keďže šlo sčasti o chybu z našej strany, rozhodol som sa hodnotiť najviac 15 bodmi oba druhy riešení.

V tomto prípade sa vzorové riešenie skladá z troch krokov. Prvý – uhádnuť myšlienku – je ľahký a zvládli ho takmer všetci. Druhý – dokázať túto myšlienku – je o dosť ťažší. Zostáva už len pomerne ľahká implementácia, s ktorou problémy nemal nik. Kde ale problémy boli? Pri dôkaze správnosti. Ten v mnohých riešeniach úplne chýbal a mnohí ďalší ho odbili dvoma vetami v duchu „je jasné, že keď to spravíme neskôr, bude to len horšie“.

To nie je dôkaz. Dôkaz je dostatočne podrobné zdôvodnenie, ktorým o správnosti algoritmu presvedčíte človeka, ktorý ho dovtedy nevidel. Spraviť poriadny dôkaz je tiež dobré na to, aby ste **vy** vedeli, že je vaše riešenie správne. Ak na praktickej súťaži začnete písať riešenie s nesprávnou myšlienkou, stratíte na tom minimálne hodinu cenného času. Zvyknite si dokazovať správnosť svojich algoritmov ešte pred tým, ako ich vôbec začnete písať, ešte sa vám to oplatí.

Keďže v tomto prípade bol podľa mňa práve spomínaný dôkaz správnosti potrebnou a najťažšou časťou riešenia, vyžadoval som ho a dával zaň viac bodov, ako býva zvykom. Ale dosť slov do duše, poďme k riešeniu.

Ukážeme jeden postup pre našu, ťažšiu verziu zadania. Výsledná postupnosť lámaní bude zároveň riešením ľahšej verzie.

Veta. *Jedno možné* najlepšie riešenie vyzerá tak, že si vždy vyberieme niektorý kúsok čokolády a zlomíme ho pozdĺž *ľubovoľnej* najdrahšej hrany.

(Všimnite si zvýraznené časti a zamyslite sa, načo tam sú.)

Naše tvrdenie dokážeme matematickou indukciou podľa veľkosti čokolády.

Pre čokoládu s plochou 1 aj 2 triviálne platí. Majme teraz nejakú čokoládu rozmerov $M \times N$, pričom pre všetky čokolády s plochou menšou ako MN dokazované tvrdenie platí. Zoberme ľubovoľnú najdrahšiu hranu h. Potrebujeme ukázať, že existuje najlepšie riešenie, v ktorom začneme prelomením hrany h.

Uvažujme ľubovoľné najlepšie riešenie R, nech začína hranou k. Ak $k = h$, vyhrali sme. Inak nám ostávajú dva prípady. Ľahší je ten, ak k a h sú rovnobežné. Po prvom kroku riešenia R dostaneme dva kusy čokolády, pričom hrana h je celá v jednom z nich a je najdrahšia v ňom. Označme tieto kusy A a B, pričom h je v A.

Zjavne riešenie R môžeme preusporiadať tak, aby najskôr rozlámalo celú časť A, až potom B. Ale z IP vieme, že časť A sa dá optimálne rozlámať tak, že začneme zlomením hrany h.[3] Zostrojme teda riešenie R', ktoré vyzerá rovnako ako R, len časť A lámeme našim optimálnym spôsobom. Toto riešenie je zjavne rovnako dobré ako R.

Všimnime si teraz prvé dva kroky riešenia R': prelomenie podľa k a podľa h. Keďže sú tieto hrany rovnobežné, cena sa nezmení, ak tieto dve prelomenia vykonáme v opačnom poradí. Tým sme zostrojili optimálne riešenie, ktoré začína rozlomením hrany h a vyhrali sme.

Ťažší prípad bude fungovať podobne. Nech teda k a h sú na seba kolmé. Podobne ako v predchádzajúcom prípade prvým rozlomením dostaneme dva kusy A a B. Tentokrát h zasahuje do oboch z nich a v oboch je najdrahšia. Z indukčného predpokladu vieme, že aj A aj B sa dajú optimálne rozlámať tak, aby sme v prvom kroku prelomili príslušný kus hrany h. Takto dostaneme (možno iné, možno rovnaké) optimálne rozlámanie. Preusporiadajme ho teraz tak, aby sme v druhom a treťom kroku zlomili A a B (opäť, keďže lámania A a B sa navzájom neovplyvňujú, môžeme si to dovoliť).

Všimnime si teraz prvé tri kroky nášho optimálneho riešenia: prelomenie podľa k a dvakrát podľa dvoch kusov h. Čo sa stane, ak by sme prelomili čokoládu najskôr podľa h a potom (dvakrát) podľa k? Dostaneme presne rovnakú situáciu, takže ďalej môžeme použiť presne ten istý postup. Zaujíma nás teda len cena týchto troch prelomení. V prvom prípade sme dvakrát platili cenu h a raz cenu k, v druhom prípade je to naopak. Keďže ale h bola najdrahšia hrana, druhá možnosť je aspoň taká dobrá ako prvá, a teda nutne vedie k optimálnemu riešeniu.[4]

Riešenie úlohy. Keďže nám stačí spočítať cenu optimálneho rozlámania, môžeme si spomedzi (potenciálne viacerých) možných optimálnych rozlámaní vybrať také, ktorého cenu ľahko spočítame. Bude to nasledujúci postup: Utriedime všetky hrany (vodorovné aj zvislé) podľa ceny. Potom ideme postupne od najdrahšej hrany k najlacnejšej a vždy pozdĺž aktuálnej hrany prelomíme všetky kúsky, cez ktoré prechádza.[5]

Utriediť hrany vieme v čase $\Theta((M+N)\log(M+N))$. Ako potom efektívne spočítať výslednú cenu? Jediné, čo potrebujeme, je vedieť pre hranu povedať, koľko kúskov pozdĺž nej potrebujeme prelomiť. Tých je ale o jeden viac ako počet na ňu kolmých hrán, ktoré sme dovtedy spracovali. Preto nám stačí, keď si budeme pamätať, koľko vodorovných a koľko zvislých hrán sme už prelomili. Samotné spracovanie vieme potom spraviť v lineárnom čase.

Listing programu:

```
#include <stdio.h>
#include <stdlib.h>

typedef struct { int cena, smer; } tHrana;
```

[3] Tu sa nám oplatilo to slovo „ľubovoľnej“ v znení vety.

[4] Navyše to znamená, že nutne majú h a k rovnakú cenu.

[5] Všimnite si, že toto je riešenie našej všeobecnejšej verzie, ale takisto je to aj riešenie zjednodušenej verzie a zjavne je aj pre ňu optimálne (prečo?).

```
int tHranaCmp(const void *a, const void *b) {
  tHrana *aa=(tHrana *)a, *bb=(tHrana *)b;
  return (bb->cena) - (aa->cena);
}

tHrana H[1000]; // ceny hran
int M,N; // rozmery
int WAS[2]; // kolko hran ktorym smerom sme uz videli
int cost=0; // cena optimalneho riesenia

int main(void){
  int i;
  // nacitame
  scanf("%d %d ",&M,&N);
  for (i=0;i<M;i++) { scanf("%d ",&H[i].cena); H[i].smer=0; }
  for (i=M;i<M+N;i++) { scanf("%d ",&H[i].cena); H[i].smer=1; }
  qsort(H,N,sizeof(tHrana),tHranaCmp);
  WAS[0]=WAS[1]=0;
  for (i=0;i<M+N;i++) {
    cost+=H[i].cena * (1 + WAS[1 - H[i].smer]);
    WAS[ H[i].smer ]++;
  }
  printf("%d\n",cost);
  return 0;
}
```

opravoval Dávidko
(max. 15 bodov)

4. O plnej izbe

Väčšina riešiteľov prišla na to, že úloha je nejakým spôsobom spojená s paritou počtu inverzií krabíc. (Ak ste predchádzajúcej vete nerozumeli, nezúfajte, o chvíľu si všetko vysvetlíme.) Korektný dôkaz, kedy a prečo ide izba upratať a kedy nie, mal málokto. Zväčša ste ho buď nemali, alebo ste ho mali zle alebo ste len uviedli nejaký obskúrny zdroj, z ktorého ste čerpali. Algoritmy, ktoré rátali inverzie krabíc boli troch typov, podľa čoho dostali aj body:

- triviálne riešenie s časovou zložitosťou $O(N^4)$ – 10 bodov,
- riešenie s časovou zložitosťou $O(N^2 \log N)$ – 12 bodov,
- vzorové riešenie s časovou zložitosťou $O(N^2)$ – 15 bodov.

Jeden riešiteľ napísal riešenie založené na backtrackingu, ten dostal 8 bodov. Za chyby (občas až fatálne) som stŕhal bod.

Vzorové riešenie

Na pochopenie vzorového riešenia treba čo, to vedieť o permutáciách. *Permutácia* M prvkov je usporiadaná[6] M-tica čísel $\{1, 2, \ldots, M\}$ taká, že každé číslo sa v nej vyskytuje práve raz. Napríklad $[7, 3, 6, 2, 5, 4, 1]$ je permutácia siedmich prvkov. Situáciu, keď v permutácii stojí väčšie číslo pred menším, budeme volať *inverzia*. Napríklad $7, 3$ je inverzia v permutácii $[7, 3, 6, 2, 5, 4, 1]$. Dokopy má táto permutácia 16 inverzií. Ak máme nejakú permutáciu, tak výmenou ľubovoľných dvoch jej prvkov dôjde iste k akejsi zmene počtu inverzií. Dá sa však ľahko ukázať, že každou takou výmenou sa zmení *parita* počtu inverzií. Napríklad permutácia $[7, 3, 6, 2, 5, 4, 1]$ má párny počet inverzií, výmenou prvkov $3, 4$ dostaneme permutáciu $[7, 4, 6, 2, 5, 3, 1]$, ktorá má nepárny počet inverzií. Skúste vymeniť v permutácii $[7, 3, 6, 2, 5, 4, 1]$ iné dva prvky a zistíte, že vždy ich dostanete nepárny počet. Všeobecný

[6] t.j. záleží na usporiadaní čísel v nej

dôkaz je ľahký a prenechávame ho na vás. Ľahko si potom uvedomíme, že keď budeme vymienať dvojice prvkov, tak aby sme dostali usporiadanú permutáciu $[1, 2, \ldots, M]$, tak parita počtu výmen bude rovná parite počtu inverzií. Treba si uvedomiť, že toto platí nech budeme triediť akýmkoľvek spôsobom.

Vráťme sa k našej úlohe, kedy vie Janko usporiadať krabice v izbe. Prázdne políčko volajme pre jednoduchosť *diera*. Všimnime si permutáciu na $M = N^2 - 1$ číslach krabíc v izbe čítanú po riadkoch, pričom dieru ignorujeme. Všímajme si, čo sa stane s paritou počtu inverzií, ak budeme dieru posúvať. Posunom diery v riadku sa permutácia nemení, a teda sa nemení ani parita počtu inverzií. Skúmajme posun diery hore alebo dole. Označme k krabicu, ktorá sa posunula. Ak sa pozrieme na permutáciu, tak v nej prvok k preskočil práve $N - 1$ iných prvkov. Ak k bolo v inverzii s nejakým preskočeným prvkom, tak po preskočení nebude, a naopak ak nebolo, tak bude. Takže parita počtu inverzií sa zmenila podľa parity čísla $N - 1$.

Všímajme si na chvíľu situácie, keď je diera v pravom dolnom rohu. Ak nejako poposúvame krabice a nakoniec dieru zase vrátime do toho istého rohu, tak parita inverzií sa nemohla zmeniť, pretože vertikálnych posunov sme spravili párny počet – niekoľko hore a toľko isto dole. Keďže upratraná pozícia má nula inverzií, tak situácia s dierou vpravo dole a nepárnym počtom inverzií upratať nejde.

Naopak, ukážeme, že ak máme dieru vpravo dole a párny počet inverzií, tak situácia upratať ide. Dôkaz urobíme indukciou na N. Báza induckie je $N = 2$, tam dokazované tvrdenie prevedieme ručným preskúšaním všetkých možností, je ich $3! = 6$. Indukčný krok sa urobí nasledovne: Pre $N > 2$ vieme upratať horný riadok a ľavý stĺpec, t.j. dostať krabice $1, 2, \ldots, N$ a $N+1, 2N+1, \ldots, (N-1)N+1$ na svoje miesta a dieru naspäť dostať do pravého dolného rohu. (Chvíľu sa pohráte a uvidíte, že to ide.) Keďže sme vrátili dieru do pravého dolného rohu, tak parita inverzií sa nezmenila. Ľahko sa tiež ukáže, že parita počtu inverzií len na číslach bez horného riadku a ľavého stĺpca je rovnaká ako s ním, t.j. párna. (To si zase dokážte sami.) Ak teraz zabudneme na horný riadok a ľavý stĺpec, dostávame situáciu, kde už máme rozmer izby len $N - 1$, a táto podľa indukčného prepokladu ide upratať. Q.E.D.

Na záver ešte rozeberme, čo ak diera nie je v pravom dolnom rohu. Vtedy ju ale vieme do neho poposúvať. Každým posunom o riadok nižšie spôsobíme zmenu parity o rovnakú paritu ako $N - 1$. Inak, povedané ak je N nepárne, tak k parite inverzií nepripočítame nič. Ak je N párne, tak ju zmeníme o paritu riadku. Izbu upratať ide, práve vtedy, ak je výsledná parita párna.

Zistili sme, že to, či sa dá alebo nedá izba upratať závisí len od parity N, parity riadku s dierou a hlavne od parity počtu inverzií. Počet inverzií sa dá triviálne zrátať v čase $O(M^2) = O(N^4)$, netriviálne v čase $O(M \log M) = O(N^2 \log N)$. Nám ale ide len o paritu a nie o počet, a to ide urobiť v lineárnom čase!

Pripomeňmne si najprv pár ďalších pojmov. *Cyklus* v permutácii $p = [p_1, p_2, \ldots, p_M]$ je postupnosť rôznych prvkov permutácie $(a_1, a_2, \ldots, a_k)$ taká, že $a_{i+1} = p_{a_i}$ pre $i = 1, 2, \ldots, k-1$ a $a_1 = p_{a_k}$. Slovami, na a_i-tom mieste permutácie leží číslo a_{i+1}. Napríklad permutácia na siedmich prvkoch $[7, 3, 6, 2, 5, 4, 1]$ má tri cykly: $(4, 2, 3, 6)$, $(1, 7)$ a (5). Nájsť cykly permutácie ide hravo v lineárnom čase: Budeme si každý prvok permutácie odškrtávať. Ideme zaradom po prvkoch permutácie, a ak je prvok nezaškrtnutý, tak obídeme cyklus, v ktorom tento prvok leží, pričom všetky prvky na cykle zaškrtáme. Teraz si už len stačí uvedomiť, že na utriedenie prvkov na cykle dĺžky k stačí $k - 1$ výmen.[7] Napríklad v našej permutácii $[7, 3, 6, 2, 5, 4, 1]$ stačia na utriedenie cyklu $(4, 2, 3, 6)$ tri výmeny $(4, 6)$, $(4, 3)$ a $(4, 2)$, na cyklus $(1, 7)$ jediná výmena a na cyklus (5) žiadna. Spolu teda treba $(4 - 1) + (2 - 1) + (1 - 1) = 4$ výmeny, a teda aj počet inverzií je párny. Vo všeobecnosti

[7] Spomeňte si, že parita počtu inverzií je zároveň paritou počtu výmen potrebných na utriedenie.

treba $M - C$ výmen, kde C je počet cyklov. Teda paritu počtu inverzií vieme určiť len z počtu cyklov.

Časová aj pamäťová zložitosť bude lineárna od veľkosti vstupu, t. j. $O(N^2)$.

Listing programu:

```
Program O_plnej_izbe;

const MAX = 100;

var a:array[1..MAX*MAX-1] of integer;
   N,i,j,k,riadok,cyklov:integer;

begin
readln(N);

      { precitame vstup }
k:=1;
for i:=1 to N do
for j:=1 to N do
begin
read(a[k]);

if a[k]=0 then  { nula je diera }
                riadok:=i   { zapamatame si riadok s dierou }
             else
                inc(k);
end;

      cyklov:=0;             { pocet cyklov }

{ ratanie cyklov }
      for i:=1 to N*N-1 do
      if a[i]>0 then
      begin
          inc(cyklov);{ novy cyklus permutacie }
          j:=i;
          while(a[j]>0) do   { prechadzame cyklus permutacie }
          begin
              k:=a[j];
              a[j]:=0;      { vyskrtneme prvok }
              j:=k;         { skocime na dalsi prvok cyklu }
          end;
      end;

      if ((N*N-1-cyklov)+riadok*(N-1)) mod 2 = 0
        then writeln('da sa')
        else writeln('neda sa');
end.
```

5. O mravcoch IV

opravoval Tom
(max. 15+ bodov)

Predpokladajme, že ceny hrán sú navzájom rôzne. Úlohu budeme riešiť paralelizovaným Sollinovym algoritmom. Ten pracuje v krokoch. Každý krok pozostáva z dvoch fáz. V prvej sa ku každému vrcholu nájde najlacnejšia z neho odchádzajúca hrana. Tieto hrany sa pridajú

do vznikajúcej kostry. (Dôkaz toho, že tam naozaj patria ponechávam ako cvičenie pre čitateľa.) Takto vybrané hrany vytvárajú v grafe komponenty. V druhej fáze každý z týchto komponentov skontrahujeme do jedného vrchola. Dostaneme tak nový (menší) graf, ktorý je vstupom do ďalšieho kroku.

Keďže každý komponent vyznačený v prvej fáze obsahuje aspoň 2 vrcholy, v druhej fáze zmenšíme graf na najviac polovicu. Bude nám teda treba vykonať $O(N \log N)$ (kde N je počet vrcholov grafu) krokov.

Zostáva ešte doriešiť pár otázok.

Ako skontrahujeme tie komponenty? Predpokladajme, že výsledky prvej fázy máme uložené v poli D tak, že $(i, D[i])$ je najlacnejšia hrana odchádzajúca z vrchola i. Zrejme ak $i \neq D[D[i]]$, tak hrana $(D[i], D[D[i]])$ je lacnejšia ako hrana $(i, D[i])$. Teda v slede $i, D[i], D[D[i]], D[D[D[i]]], \ldots$ ideme po hranách s klesajúcou cenou (čiže sa necyklíme) až kým sa nedostaneme k vrcholu k takému, že $k = D[D[k]]$. Takto je ku každému vrcholu i určená nejaká neusporiadaná dvojica vrcholov $k, D[k]$, ktoré sú zrejme v tom istom komponente vyznačeného podgrafu ako i. Zrejme k vrcholom z rovnakého komponentu je takto určená rovnaká dvojica. (Jednoduchý dôkaz opäť ponechávam ako cvičenie pre čitateľa.) Jeden z vrcholov z tej dvojice (napríklad ten menší) budeme považovať za reprezentanta tohto komponentu.

Pre každý vrchol nájdeme reprezentanta komponentu, do ktorého patrí. To spravíme jednoducho tak, že najprv priradíme $D[i] := i$ pre každé i také, že $D[D[i]] = i$. Potom nikoľko krát pre každý vrchol priradíme $D[i] := D[D[i]]$. Po každej takejto iterácii sa nám cesta k reprezentantovi skráti na polovicu. Bude teda treba $O(\log N)$ iterácii.

Teraz si to zjednodušíme a komponent neskontrahujeme, ale iba zrušíme hrany vnútri komponentu. Jediné, čo sa tým zhorší bude práca nášho algoritmu. (Kto by chcel vidieť nezjednodušenú verziu algoritmu, nech si prečíta – Joseph JáJá: An Introduction to Parallel Algorithms, algoritmus 5.3.) To nám ale mierne skomplikuje prvú fázu. Už nebudeme hľadať najlacnejšiu hranu odchádzajúcu z vrchola, ale z komponentu. To ale nie je príliš ťažké. Najprv si pre každý vrchol nájdeme najlacnejšiu hranu z neho odchádzajúcu a podom pre každý komponent (teraz komponent znamená množinu vrcholov s rovnakým reprezentantom) najlacnejšiu hranu spomedzi tých, ktoré sme našli pre vrcholy v tom komponente. Obe minimá vieme nájsť v čase $O(\log N)$ s (N^2) počítačmi.

Celý algoritmus bude mať teda časovú zložitosť $O(\log^2 N)$, pamäťovú zložitosť $O(N^2)$ a počet počítačov $O(N^2)$.

Ešte zopár poznámok Formát vstupu je: v `Mem[0]` je počet vrcholov grafu N, v `Mem[1..N*N+1]` je symetrická matica susedností, čiže v `Mem[(i-1)*N+j]` je cena hrany (i, j), alebo nekonečno, ak taká hrana v grafe nie je.

Pri inicializácii pridám do grafu ďalší vrchol, ktorý bude izolovaný.

V programe použijem pre lepšiu čitateľnosť nasledujúce označenia:

- `N` bude to isté, ako `Mem[0]`
- `X` bude to isté, ako `(cpuid mod (N+1))`
- `Y` bude to isté, ako `(cpuid div (N+1))`
- `Wio[i,j]` bude to isté, ako `Mem[(i-1)*N+j]`
- `W[i,j]` bude to isté, ako `Mem[N*N+(i-1)*(N+1)+j]`
- `Min[i,j]` bude to isté, ako `Mem[2*N*N+2*N+1+(i-1)*(N+1)+j]`
- `From[i,j]` bude to isté, ako `Mem[3*N*N+4*N+2+(i-1)*(N+1)+j]`
- `D[i]` bude to isté, ako `Mem[4*N*N+6*N+3+i]`

Listing programu:

```
var Nk,t,a,b,ia,ib:integer;
```

```
   if cpuid>(N+1)*(N+1) then halt;
   if (X<=N)and(Y<=N) then W[X,Y]:=Wio[X,Y] else W[X,Y]:=infinity;
   if (X<=N)and(Y<=N) then Wio[X,Y]:=infinity else nop;
   if cpuid<=N+1 then D[i]:=D[i];
   Nk:=N+1;

7: Min[X,Y]:=Y;   prva faza; najdeme najlacnejsiu hranu odchadzajucu z kazdeho vrchola
   t:=N+1;
1: if Y*2-1<=t then a:=Min[X,Y*2-1] else a:=N+1;
   if Y*2<=t then b:=Min[X,Y*2] else b:=N+1;
   if W[X,a]<W[X,b] then Min[X,Y]:=a else Min[X,Y]:=b;
   t:=t div 2 + t mod 2;
   if t<=1 then goto 2 else goto 1

2: Min[X,Y]:=Min[Y,1];      a teraz najlacnejsiu hranu odchadzajuju z kazdeho komponentu
   From[X,Y]:=Y;
   if D[X]<>D[Y] then Min[X,Y]:=N+1 else nop;
   t:=N+1;
3: if 2*Y-1<=t then ia:=2*N-1 else ia:=t;
   if 2*Y<=t then ib:=2*N else ib:=t;
   if W[From[X,ia],Min[X,ia]]<W[From[X,ib],Min[X,ib]] then m:=ia else m:=ib;
   if 2*Y-1<=t then Min[X,Y]:=Min[X,ia] else nop;
   if 2*Y-1<=t then From[X,Y]:=From[X,ia] else nop;
   t:=t div 2 + t mod 2;
   if t<=1 then goto 4 else goto 3

4: if (cpuid<=N)and(D[cpuid]=cpuid)and(W[From[cpuid,1],Min[cpuid,1]]<infinity) then
     Wio[From[cpuid,1],Min[cpuid,1]]:=W[From[cpuid,1],Min[cpuid,1]] else nop;
   if (cpuid<=N)and(D[cpuid]=cpuid)and(W[From[cpuid,1],Min[cpuid,1]]<infinity) then
     Wio[Min[cpuid,1],From[cpuid,1]]:=W[From[cpuid,1],Min[cpuid,1]] else nop;
           ;pridame hrany do kostry
   if (cpuid<=N+1)and(D[cpuid]=cpuid)and(W[From[cpuid,1],Min[cpuid,1]]<infinity) then
     D[cpuid]:=D[Min[cpuid,1]] else nop;
           ;vyznacime najdene hrany (dame ich medzi reprezentantov)


   t:=1;  druha faza
   if (cpuid<=N+1)and(cpuid=D[D[cpuid]])and(cpuid<D[cpuid]) then
     D[cpuid]:=cpuid else nop; najdeme reprezentantov
5: if cpuid<=N+1 then D[cpuid]:=D[D[cpuid]] else nop;
   t:=2*t;
   if t>2*N+2 then goto 6 else goto 5;
6: if D[X]=D[Y] then W[X,Y]:=infinity else nop;  zrusime hrany vnutri komponentu
   Nk:=Nk div 2 + Mk mod 2;
   if Nk>1 then goto 7 else halt;
```

Výsledková listina po 2. kole kategórie KSP

	Meno a priezvisko	Škola	Ročník		21	22	23	24	25	Σ
1	Cicko Miroslav	Gym. Tajovského B. Bystrica	3	70	15	15	14	15	10	139
2	Simančík František	Gym. Grösslingová BA	3	62	15	15	15	12		119
3	Buštor Stano	Gym. Jura Hronca BA	3	41	12	12	13	12	15	105
3	Nánási Michal	Gym. Jura Hronca BA	3	41	15	15	12	11	11	105
5	Imriška Jakub	Gym. Jura Hronca BA	2	46	15		14	8	11	94
6	Bundala Daniel	Gym. Jura Hronca BA	2	35	10	15	7	11	11	89
7	Perešíni Peter	Gym. Tajovského B. Bystrica	2	43	15	15		15		88
8	Kuštárová Tamara	Bilingválne gym. Sučany	3	43	5	4	11	12	10	85
9	Plžík Milan	Gym. Ľ. Štúra Zvolen	3	45	9	1	7	9	10	81
10	Beleš Lukáš	Gym. Čadca	3	43	8	6	10		12	79
11	Takáč Slavomír	Gymnázium	2	25	15	15	15			70
12	Schlosáriková Eva	Gym. P. de Coubertina Piešťany	3	42		9	5		11	67
13	Krchniak Matej	Gym. Jura Hronca BA	2	39	7	4	10			60
14	Beran Jakub	Gym. Alejová Košice	2	24	7		9	9	10	59
15	Špalek Lukáš	Gym. Čadca	3	18	9	8	10		10	55
16	Ludha Marek	Gym. Tajovského B. Bystrica	4	0	15	13	10	12		50
17	Trnovcová Zuzana	Gym. Jura Hronca BA	3	32			15			47
18	Petruľák Matúš	Gym. Grösslingová BA	3	45						45
19	Zeman Marek	Gym. Jura Hronca BA	3	20	8		12			40
20	Lenhardt Rastislav	Gym. Jura Hronca BA	4	38						38
21	Poláček Lukáš	Gym. K. Štúra Modra	4	37						37
22	Urminský Tomáš	SPŠ Nové Mesto nad Váhom	2	17			9			26
23	Kuchynárová Mariana	Gym. Jura Hronca BA	4	24						24
24	Ružička Peter	Gym. Jura Hronca BA	3	0			9	9		18
25	Kováč Michal	Gym. Grösslingová BA	2	9			8			17
25	Mindek Peter	Škola pre mim. nadané deti BA	2	17						17
27	Goga Peter	Gym. 17. novembra Topoľčany	4	15						15
27	Smažáková Dana	Gym. Jura Hronca BA	4	15						15
29	Ďurfina Lukáš	Gym. Golianova Nitra	3	13						13
30	Bodnár Jozef	Gym. Fiľakovo	3	12						12
31	Glaus Peter	Gym. Jura Hronca BA	4	11						11
32	Vadkerti Miroslav	SPŠE Nové Zámky	4	9						9
33	Morihladko Peter	Gym. Školská Spiš. Nová Ves	3	8						8
34	Vanderka Peter	Gym. Trstená	3	7						7