Korešpondenčný seminár z programovania
XXVI. ročník, 2008/09
Katedra základov a vyučovania informatiky FMFI UK,
Mlynská Dolina, 842 48 Bratislava

KSP finančne podporujú: MICROSTEP-MIS spol. s r.o.
Whitestein Technologies spol. s r.o.

Vzorové riešenia 2. kola letnej časti

Milé naše riešiteľky, milí naši riešitelia,

práve sa vám dostali do rúk najnovšie a posledné vzoráky tohto ročníka KSP. Keďže ide leto, tak sme sa rozhodli spraviť nasledovnú súťaž o čokoládu. Pošlite nám fotku vás a našich vzorákov z čo najzaujímavejšieho miesta, ktoré v lete navštívite. Víťaza vyberie odborná porota skladajúca sa z KSPákov. Fotky zasielajte na adresu `mic@sutaz.ksp.sk`.

Vedeli ste, že 74% štatistík je vymyslených?

KSPáci

1. Záhadné štvorce

Opravovala Dominika
(max. 10 bodov)

Všetci ste prišli na to, že ak do štvorca napíšeme čísla zaradom tak tento štvorec spĺňa všetky podmienky záhadného štvorca. Prečo to tak ale je, to ste všetci nenapísali a za to som strhávala dosť bodov. Tiež som stiahla bod, ak ste nenapísali odhad zložitosti.

Tak a poďme teda na vysvetlenie: Ak máme čísla zapísané zaradom, tak pri štvorci veľkosti n si každé číslo môžeme napísať v tvare $s + n \cdot (r - 1)$, kde s je stĺpec a r riadok. Ak si teraz vyberieme n čísel, každé z iného riadku a iného stĺpca, dostaneme vždy rovnaký súčet:

$$[1 + 2 + \cdots + n] + [n \times 0 + n \times 1 + n \times 2 + \cdots + n \times (n - 1)] = n \cdot (n^2 - 1)/2$$

(v prvej zátvorke sú čísla stĺpcov – každý práve raz; v druhej zátvorke je $n \times$ číslo riadku $- 1$ – každý práve raz).

Časová zložitosť je kvadratická $O(n^2)$, pamäťová je konštantná $O(1)$.

Listing programu:

```
program stvorec;
var n, i : longint;
begin
  readln(n);
  for i := 1 to n*n do
    if i mod n = 0 then writeln (i:4) else write (i:4);
end.
```

2. Zeus a rodokmene

Opravovali Halucinka a Ivan
(max. 10 bodov)

Buď poznáme rodičov danej bytosti alebo poznáme jeho božskosť (0 alebo 1). Naším cieľom je vypočítať božskosť bytosti 1. Rodokmeň si môžeme predstaviť ako binárny strom. Koreňom

je bytosť 1. Rozvetvuje sa vždy na dvoje, pretože každý vrchol, ak nie je listom, má dvoch rodičov. Listy stromu sú tie bytosti, ktorých božskosť poznáme (pretože vo vstupe máme miesto ich rodičov napísanú ich božskosť). Takže, keď už máme takúto binárnu predstavu `:-)`, ako zrátame tú božskosť?

Ak poznáme božskosť jeho oboch rodičov, tak sme vyhrali – spriemerujeme a máme to. Ak božskosť aspoň jedného z rodičov nepoznáme, tak sa pustíme do pátrania božskosti rodiča. A to urobíme presne rovnakým spôsobom. Teda pozrieme sa na rodičov tohto rodiča a ak poznáme ich božskosti, tak ich spriemerujeme, ak nie, tak ideme pátrať po ich božskosti. Snáď vám je už jasné, že pôjde o rekurzívne volanie funkcie. Vytvoríme si funkciu `božstvo`, ktorá nám vypočíta rovno božskosť našej bytosti. V tele tejto funkcie bude volanie samej seba (pre zistenie božskosti rodiča našej bytosti, ak ju nepoznáme).

Už teda máme aj predstavu o našej funkcii, bude sa správať v skratke takto: Ak poznáš božskosť rodičov našej bytosti, spriemeruj a ulož božskosť našej bytosti, ak nepoznáš božskosť nejakého z jeho rodičov zisti ju. Tu zavolá sama seba s tým, že už zisťuje božskosť inej bytosti. Takto pokračuje, až kým sa nedostane k predchodcom, o ktorých vieme ich božskosť. Potom sa bude späť vynárať až zráta božskosť našej bytosti, pretože potrebné informácie prídu vždy s jedným vynorením sa.

Na to, aby sme mohli vypísať božskosť Hermiosa potrebujeme vyriešiť, ako si budeme tú božskosť pamätať. Jedna možnosť je pamätať si číslo ako zlomok – t.j. mať vždy dve premenné – čitateľ a menovateľ. Potom si musíme napísať funkcie alebo procedúry, ktoré s takýmito zlomkami vedia počítať. Napríklad $a/b + c/d = (ad + bc)/(bd)$.

Jednoduchšie riešenie je pamätať si božskosti ako reálne čísla a potom z výsledku zrekonštruovať zlomok. Ale vieme si toto reálne číslo zapamätať presne? A vieme ho prerobiť na zlomok presne? Uvedomme si, že v menovateli zlomku môžu byť iba mocnina dvojky (teda, že zlomok sa dá upraviť na tvar, kde v menovateli bude len mocnina dvojky). Dokážeme si to indukciou.

Báza indukcie: Vezmime si bytosť, kde poznáme božskosť oboch rodičov. Ak sú obaja bohovia, tak aj ich potomok je boh, teda menovateľ je $1 = 2^0$. Ak je práve jeden rodič boh, tak božskosť potomka je $1/2$, teda menovateľ je 2^1. A nakoniec ak ani jeden z rodičov nie je boh, tak nie je boh ani potomok.

Indukčný krok: Predpokladajme, že náš predpoklad platí pre rodičov bytosti x. Chceme dokázať, že aj bytosť x bude mať v menovateli mocninu dvojky. Nech božskosť rodičov tohto prvku je $a/2^i$ a $b/2^j$ a nech $i \leq j$. Potom božskosť potomka je: $(a \cdot 2^{j-i} + b)/2^j$. Teda aj potomok má v menovateli len mocninu dvojky.

Teda ak máme božskosť bytosti v reálnom čisle r, potom $r = 2r/2 = 4r/4 = 8r/8 \ldots$, a takto postupujeme, až kým čitateľ nebude celé číslo.

Počítač si reálne čísla uchováva v dvojkovej sústave, kde každá číslica je 2^i , pričom i je pozícia číslice. V reálnych číslach i je aj záporné, to znamená, že číslo $0,011$ v dvojkovej sústave bude $0 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3}$. A my chceme uchovávať čísla tvaru p/q, kde q je 2^n. To znamená, že to reálne číslo bude presné a nemusíme sa báť, že nám nejaká desatinka utečie `:-)`.

Takže všetky problémy sú vyriešené. Už iba spomenúť, že si pamätáme všetky bytosti v strome, teda pamäťová zložitosť závisí lineárne od počtu bytostí: $O(n)$. A časová zložitosť je taktiež $O(n)$, pretože prejdeme každú bytosť najviac raz (božskosť každej bytosti potrebujeme na vypočítanie božskosti bytosti 1).

Listing programu:

```
type TBytost = record
       rodic1, rodic2 : longint;
       bozstvo : real;
     end;
var i, N, cislo, q : longint;
```

```
  p : array of TBytost;
  hermios: real;

function bozstvo (bytost : longint) : real;
begin
  if (p[bytost].bozstvo >= 0) then
    bozstvo := p[bytost].bozstvo
  else
    bozstvo := (bozstvo(p[bytost].rodic1) + bozstvo(p[bytost].rodic2))/2;
end;

begin
  readln(N);
  SetLength(p, N + 1);
  for i:=1 to N do p[i].bozstvo := -1;
  for i:=1 to N do begin
    read(cislo);
    if cislo=1 then readln(p[i].rodic1, p[i].rodic2)
               else readln(p[i].bozstvo);
  end;
  hermios := bozstvo(1); q := 1;
  while frac(hermios) > 0 do begin
    hermios := hermios * 2; q := q * 2;
  end;
  writeln(hermios:0:0, '/', q);
end.
```

3. Zle naplánovaný výlet

Opravoval PPershing
(max. 10 bodov)

Poďme sa pozrieť na vzorové riešenie. Ako sa v histórii KSP ukazuje, riešenia, ktoré prezrú všetky možnosti (pre niektorých známe aj ako bruteforce) sú obľúbené medzi riešiteľmi. Tak môžu byť obľúbené aj medzi vedúcimi :-).

Našou úlohou je zistiť optimálny počet jázd na antilope S_A. Ak už máme S_A, počet tričiek sa dá ľahko vypočítať ako $S_B = \lfloor (K - S_A \cdot A)/B \rfloor$. Prečo? Po antilopách nám ostane $K - S_A \cdot A$ peňazí a jedno tričko stojí B. Teda, pre daný počet jázd na antilope máme počet tričiek jednoznačne určený. Stačí, ak teda preskúmame všetky možnosti, koľko mohlo byť jázd na antilope a jednoducho nájdeme, ktorá z nich je najlepšia.

Koľko je týchto možností? Najmenej jázd je samozrejme 0, najviac je ich $\lfloor K/A \rfloor$. Každú z nich vieme dopočítať a porovnať s maximom v konštantnom čase. Spolu bude náš program teda mať časovú zložitosť $O(K/A)$. Pamäťová zložitosť je $O(1)$, nakoľko si potrebujeme pamätať iba konštantne veľa premenných.

Poznámky na záver: naše riešenie sa dá ľahko vylepšiť na čas $O(\min(K/A, K/B))$ – stačí, ak si na začiatku vyberieme, či chceme prechádzať cez antilopy a dopočítavať tričká alebo opačne. Toto však stále nie je optimálne riešenie, prezradíme, že existuje riešenie, ktoré je omnoho rýchlejšie. Je však nad rámec tohto vzoráku.

Ešte nejaký ten pokec o bodovaní – za správne riešenie bolo 10 bodov, -1 za chýbajúci odhad, do -2 za chýbajúce zdôvodnenie. Pomalšie alebo časovo náročnejšie riešenia sa nejako naškálovali. A samozrejme, špeciálny bonus max 3 body dostali nefunkčné riešenia – úloha bola ľahká a máte si testovať svoje riešenia predtým než nám ich posielate.

Listing programu:

```
var a, b, k : integer;
    naj_antilop, naj_triciek : integer;
    antilop, triciek : integer;

begin
  readln(a,b,k);
  naj_antilop := 0;

  for antilop := 1 to k div a do begin
    triciek := (k - antilop * a) div b;
    naj_triciek := (k - naj_antilop *a) div b;
    if ((antilop * a + triciek * b) >=
        (naj_antilop * a + naj_triciek * b)) then naj_antilop := antilop;
  end;
  writeln(naj_antilop, ' ', (k - naj_antilop * a) div b);
end.
```

proofread: mic

4. Zadania Mega kola KSP

opravoval Mišo
(max. 10 bodov)

Stačí ísť postupne cez všetky pozície od 1 po K a pre každú pozíciu p vyberať príklad, ktorého interval obsahuje p a končí čo najskôr (a zároveň sme ho už nevybrali). Dokážme, že ak týmto spôsobom nenájdeme riešenie, tak žiadne riešenie neexistuje.

Dôkaz sporom: Predpokladajme, že sme hore-uvedeným spôsobom nenašli riešenie, ale riešenie existuje (označme ho R). Dajme tomu, že pre pozíciu p sme vybrali zlý, i-ty príklad a v riešení R sme mali namiesto toho vybrať j-ty príklad. Podľa toho, ako sme vyberali príklady interval j-teho príkladu končí neskôr ako interval i-teho. Sú dve možnosti:

1. V riešení R sa i-ty príklad vôbec nepoužije: Riešenie R upravíme tak, že namiesto j použijeme na p-tu pozíciu i. Je to správne riešenie, čím sme sa dostali do sporu.
2. V riešení R sa použije príklad i na pozícií $q > p$. Potom obidva príklady i aj j sa dajú použiť na pozície p aj q, takže sa dajú vymeniť a našim postupom sme teda získali tiež správne riešenie, čím sme sa dostali do sporu.

Postupovať budeme nasledovne nasledovne. Zoberieme všetkých kandidátov. Vyberieme z kandidátov na prvý príklad ten, čo najskôr končí a potom ho z kandidátky vyškrtneme. Potom vyberieme z kandidátov na druhý príklad ten ... Takto budeme pokračovať, až kým sme nevybrali všetky príklady, alebo kým nemáme čo vybrať (a vtedy neexistuje riešenie). Budeme potrebovať efektívnu dátovú štruktúru, ktorá zvláda nasledovné operácie:

- Pridaj príklad (keď sa posúvame na ďalšiu pozíciu, tak príklady, ktorých interval začína na tej pozícii sa stáva vhodným)
- Vyhoď príklad (nepoužitý príklad pri posune na ďalšiu pozíciu, kde už nesiaha jeho interval, vyhodíme)
- Zisti príklad, ktorého interval končí najskôr (chceme to spraviť pre každú pozíciu)

Ak by sme na implementáciu dátovej štruktúry použili pole, tak by prvá operácia trvala v čase $O(1)$ a ostatné operácie by trvali $O(N)$. Utriedené pole by nám tiež nepomohlo a ani spájaný zoznam. Budeme potrebovať niečo lepšie. Ako vhodná štruktúra, ktorá toto zvláda sa ukáže byť halda (popísaná je nižšie). Halda zvláda prvé dve operácie v čase $O(\log N)$ a poslednú operáciu zvládne v čase $O(1)$. Použitím haldy vieme dosiahnuť dobrú časovú zložitosť. Aká bude? Najskôr potrebujeme všetky prvky utriediť v čase $O(N \log N)$ napríklad tak,

že najskôr všetky prvky vložíme do haldy a potom ich povyberáme. Zvyšná časť algoritmu bude s použitím haldy trvať $O((N+K)\log N)$.

Halda: Tí skúsenejší z vás môžu tento text preskočiť a vrhnúť sa rovno na kód. Rozsiahlejšie vysvetlenie haldy tiež nájdete na wikipédii: `http://en.wikipedia.org/wiki/Binary_heap` alebo v `http://ksp.sk/ksp/zadania/prilohy/ps243/6/kp_teoria_1.pdf`.

Halda je binárny strom – vo vrcholoch máme čísla a každý vrchol má najviac dvoch synov. Vždy platí, že v otcovi je uložené menšie číslo, ako v jeho synoch. Navyše je halda úplný binárny strom, to znamená, že všetky listy sú na jednej úrovni, prípadne posledná úroveň nemusí byť úplná, ale listy idú zľava; inými slovami, v strome nie je „diera“.

Haldu vieme ľahko implementovať priamo v poli: políčko 1 bude koreň a vždy ľavý syn i-teho vrcholu bude na políčku $2i$ a pravý na políčku $2i+1$. Teda otec je na políčku $\lfloor i/2 \rfloor$. Vďaka usporiadaniu haldy je najmenší prvok v koreni.

Vkladáme tak, že pridáme prvok na koniec poľa ($(n+1)$-vé políčko, zvýšime n) a „vybubleme“ ho nahor. Môže sa totiž stať, že je menší ako jeho otec, čím je porušené usporiadanie haldy. Vtedy ho vymeníme s otcom. Ak je menší ako jeho nový otec, vymeníme ho aj s ním, a tak ďalej, až kým sa nedostaneme do koreňa alebo nenapravíme usporiadanie haldy.

Vymazať nejaký prvok vieme tak, že ho zameníme za posledný prvok, zmenšíme n a „prebubleme“ ho nadol (kým nenapravíme usporiadanie haldy alebo sa nedostaneme k listom).

Takáto halda sa tiež volá min-halda, pretože do nej vieme rýchlo vkladať a rýchlo z nej vyberať minimum. Veľmi podobne vieme spraviť max-haldu, z ktorej vieme rýchlo vyberať maximum (v skutočnosti stačí zmeniť znamienka $>$ na $<$). Ako rýchlo vieme dané operácie robiť? V oboch prípadoch len prebubleme nejaký prvok nahor alebo nadol, čo je v najhoršom prípade úmerné výške haldy a tá je logaritmická.

Listing programu:

```
type interval = record
  a,b:longint;
  end;
var A:array[1..100] of interval; { <a,b> je interval}
  i,j,n,k:longint;
  halda:array[1..100] of longint; { halda obsahuje indexy do pola A, porovnava sa
podla koncov
  intervalov}
  dlzka_haldy:longint;

function skor(x,y : interval):boolean; {x,y su intervaly, funkcia zisti, ktory zacina
skor}
  begin
    if x.a<y.a then skor:=true
    else if x.a>y.a then skor:=false
    else skor:=x.b<y.b;
  end;

procedure vymen(x,y:longint); {x,y su indexy v halde}
  var pom:longint;
  begin
    pom:=halda[x];
    halda[x]:=halda[y];
    halda[y]:=pom;
  end;

procedure bubbleUp(x:longint); //x je index v halde
  begin
    if x=0 then Exit;
```

```
    if A[halda[x]].b<A[halda[x div 2]].b then
      begin
        vymen(x,x div 2);
        bubbleUp(x div 2);
      end;
  end;

procedure pridaj(x:longint); //x je index intervalu v poli A
  begin
    inc(dlzka_haldy);
    halda[dlzka_haldy]:=x;
    bubbleUp(dlzka_haldy);
  end;

procedure bubbleDown(x:longint); //x je index v halde
  var skorsiSyn:longint;
  begin
    if dlzka_haldy<2*x then Exit;
    if dlzka_haldy=2*x then
      if A[halda[2*x]].b<A[halda[x]].b then vymen(2*x,x);
    if A[halda[2*x]].b<A[halda[2*x+1]].b
      then skorsiSyn:=2*x
      else skorsiSyn:=2*x+1;
    if A[halda[skorsiSyn]].b<A[halda[x]].b then
      begin
        vymen(x,skorsiSyn);
        bubbleDown(skorsiSyn); //vymenili sa len hodnoty, indexy v halde ostali
      end;
  end;

procedure vyber;
  begin
    halda[1]:=halda[dlzka_haldy];
    dec(dlzka_haldy);
    bubbleDown(1);
  end;

procedure QuickSort(zaciatok, koniec: longint);
{ utriedi intervaly, ktore su v poli A na poziciach
  zaciatok .. koniec podla zaciatku}

var malych, strednych, i : longint;
    pom, pivot : interval;

begin
   { ak je usek kratky, rovno ho utried a skonci }
   if (koniec <= zaciatok) then exit;

   { vyber jeden z prvkov ako pivota }
   pivot := A[zaciatok];

   { presunieme "male" prvky na zaciatok }
   malych := 0;
   for i:=zaciatok to koniec do
      if skor(A[i],pivot) then begin
         pom:=A[i];  A[i]:=A[zaciatok+malych];  A[zaciatok+malych]:=pom;
         Inc(malych);
      end;
   { a mame "male" prvky na poziciach zaciatok .. zaciatok+malych-1 }
```

```
   { za ne presunieme prvky rovne pivotu }
   strednych := malych;
   for i:=zaciatok+malych to koniec do
      if (A[i].a=pivot.a) and (A[i].b=pivot.b) then begin
         pom:=A[i];  A[i]:=A[zaciatok+strednych];  A[zaciatok+strednych]:=pom;
         Inc(strednych);
      end;
   { a mame prvky rovne pivotu na poziciach
    zaciatok+malych .. zaciatok+strednych-1 }

   { teraz samostatne utriedime usek "malych" prvkov ... }
   QuickSort(zaciatok, zaciatok+malych-1);
   { ... a usek "velkych" prvkov }
   QuickSort(zaciatok+strednych, koniec);
end;



begin
  dlzka_haldy:=0;
  readln(n,k);
  for i:=1 to n do
      readln(A[i].a,A[i].b);
  QuickSort(1,n);
  j:=1; { index do pola A }
  for i:=1 to k do
    begin
      while (A[j].a<=i) and (j<=n) do
        begin
          pridaj(j);
          inc(j);
        end;
      while A[halda[1]].b<i do vyber;
      writeln('Na pozicii ',j,' bude priklad cislo ',halda[1]);
      vyber;
    end;
end.
```

Opravoval Laci
(max. 15 bodov)

5. Zamotané dlhy

Podstatu príkladu ste odhalili snáď všetci. Naozaj stačí spočítať, koľko je ktorý káespák celkovo v pluse či v mínuse, čo sa dá spraviť s pomocným poľom o N prvkoch už pri načítaní dlžôb. Potom už len sčítať všetky kladné hodnoty v tomto poli (alebo (-1)-krát záporné :-)) a to je hľadané minimum súčtu nominálnych hodnôt všetkých transakcií potrebných na vyrovnanie.

Určite sa hodí aspoň neformálne toto riešenie dokázať, čo mnoho z vás nespravilo. Skúsme to teraz tak poriadnejšie. Označme vypočítanú sumu ako S. Dôkaz spravíme v dvoch krokoch:

1. Ukážeme, že existujú transakcie, ktorými sa KSPáci vyrovnajú a ich súčet je práve S. Toľko, koľko majú KSPáci kladnú bilanciu (to je práve to naše vypočítané S), toľko majú aj zápornú – to je zrejmé, nakoľko všetky dlžoby sú vrámci množiny KSPákov. Potom zjavne existuje priradenie, ktoré každému „zápornému“ euru pridelí „kladné“ euro. Vykonaním týchto transakcií sa KSPáci vyrovnajú a suma hodnôt transakcií je S. Teda S je riešením.

2. Ukážeme, že žiadne riešenie menšie ako S nemôže existovať. Nech D je suma, ktorú zadĺžení KSPáci v súčte dlhujú. Na to, aby sa vyrovnali, musí byť prevedených aspoň D euro. Z toho, že S je počítané tak ako je D definované, vyplýva $D = S$. Preto je S minimálne.

Z bodov 1 a 2 vyplýva, že S je minimálny súčet nominálnych hodnôt všetkých potrebných transakcií.

K riešeniu už chýbajú len odhady zložitostí. Časová zložitosť je $O(M + N)$, pretože musíme prečítať všetkých M dlžôb a prejsť pomocné pole. Pamäť je $O(N)$ kvôli už spomínanému pomocnému poľu.

Ak sa vyskytli zlé alebo chýbajúce odhady zložitostí, dôkaz alebo popis, za každé putoval dole jeden bodík.

Listing programu:

```
#include <iostream>
#define _MAX 47474747
using namespace std;

int N, M, result, stav[_MAX];

int main(){
    cin>>N>>M;

    for(int i=0; i<M; i++){
        int a,b,c;
        cin>>a>>b>>c;

        stav[a] -= c;
        stav[b] += c;
    }

    //spocitame co je kladne
    for(int i=1; i<=N; i++) result += max(0,stav[i]);

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

6. O šialenej ceste

Opravoval kuko
(max. 20 bodov)

Tento príklad riešilo iba málo ľudí a tí, čo ho riešili, ho vyriešili dobre. Na vstupe máme ohodnotený graf a chceme nájsť cestu s najmenšou pravdepodobnosťou uvidenia. Tento problém je veľmi podobný hľadaniu najkratšej cesty, ktorý sa dá riešiť Dijsktrovým algoritmom[1].

Povedzme si najskôr, ako sa vlastne rátajú tie pravdepodobnosti: majme nejakú cestu v grafe; aká je pravdepodobnosť, že nás na tejto ceste uvidia? Nuž, často sa stáva, že je jednoduchšie pozrieť sa na úlohu z druhej strany – aká je pravdepodobnosť, že nás na nejakej ceste *ne*uvidia? Neuvidia nás zrejme vtedy, keď nás neuvidia na žiadnej hrane na ceste, teda stačí vynásobiť pravdepodobnosti neuvidenia na celej ceste.

[1] Dijkstrov algoritmus si tu nevysvetlíme, keďže všetci, čo úlohu riešili, ho zjavne poznali – kto ho nepozná, môže si pozrieť napríklad článok v liahni – `liahen.ksp.sk`

Namiesto počítania s pravdepodobnosťami uvidenia $p(e)$ budeme počítať s pravdepodobnosťami neuvidenia $q(e) = 1 - p(e)$. No a ak $\pi = e_1 e_2 \dots e_n$ je nejaká cesta, tak pravdepodobnosť, že nás neuvidia je $q(\pi) = q(e_1) \cdot q(e_2) \cdots q(e_n)$. Pravdepodobnosť, že nás uvidia je zrejme opačná, ako že nás neuvidia: $p(\pi) = 1 - q(\pi)$.

Úloha sa teda dá riešiť drobnou úpravou Dijkstrovho algoritmu – namiesto sčítavania dĺžok budeme násobiť pravdepodobnosti a namiesto najkratšej cesty budeme hľadať najväčšiu pravdepodobnosť (že nás neuvidia).

Ak ste úlohu riešili, možno sa pýtate, prečo ste nedostali plný počet bodov – nuž chcel som docieliť, aby ste si prečítali vzorák `:-)`. Ukážeme si totiž ešte jednu fintu.

Finta fň: Jediný problém s vyššie uvedeným algoritmom je, že sa tam násobia veľmi malé čísla. Predstavte si graf, ktorý má tisíce, či desaťtisíce vrcholov a pravdepodobnosti sú malé čísla, dajme tomu 0.01 a menej. Potom výsledná pravdepodobnosť bude niečo ako 10^{-V}, kde V je véééľa. (Rozsah double má exponent do ± 300). Môže sa stať, že stratíme presnosť, alebo dokonca dostaneme nulové hodnoty (ktoré majú byť nenulové).

Preto sa často namiesto pravdepodobností počíta s ich logaritmami! Logaritmus je opak exponenciálnej funkcie, presnejšie $\log_{10} a = b$ práve vtedy, keď $10^b = a$. Napríklad tisíc je 10^3, preto $\log_{10} 1000 = 3$. (Logaritmus je zruba počet cifier čísla -1.) Veľmi dôležitou vlastnosťou logaritmov je

$$\log(a \cdot b) = \log(a) + \log(b),$$

teda logaritmy akoby premieňali násobenie na sčítavanie. (Na tento účel boli vlastne vymyslené a preto staršie generácie používali logaritmické tabuľky a pravítka – ak chceli vynásobiť dve čísla, tak ich zlogaritmovali, logaritmy sčítali a potom výsledok zase „odlogaritmovali“ – za pomoci tabuľky, alebo pravítka, samozrejme; nuž a sčitovať je oveľa menej namáhavé ako násobiť.)

Ak sme predtým počítali pravdepodobnosť neuvidenia $q(\pi)$ ako súčin $q(e_1) \cdot q(e_2) \cdots q(e_n)$, tak

$$\log q(\pi) = \log q(e_1) + \log q(e_2) + \cdots + \log q(e_n).$$

(Uvedomte si, že logaritmy počítame iba úplne na začiatku a na konci „odlogaritmujeme“ výsledok.)

- Výhoda 1: Nedostávame také malé čísla (ani príliš veľké); takže nič nepretečie.
- Výhoda 2: Sčítanie môže byť o chlp rýchlejšie ako násobenie.
- Výhoda 3: Ak namiesto pravdepodobnosti uvidenia $p(e)$ použijeme číslo $-\log(1-p(e))$, môžeme použiť úplne štandardný nijak neupravený Dijkstrov algoritmus na hľadanie najkratšej cesty v grafe – taký sme už možno niekedy programovali, alebo môžeme použiť nejakú knižnicu, kde to už je kvalitne naprogramované a optimalizované.

Posledná poznámka: V zdrojáku sú použité C-čkovské funkcie log a exp. Dôležité je, že sú navzájom inverzné. V skutočnosti to však nie je logaritmus pri základe 10, ale prirodzený logaritmus (pri základe $e \approx 2.718$) a $\exp(x) = e^x$, ale to je detail.

Listing programu:

```
#include <cstdio>
#include <cmath>
#define MAX 10000
#define INFTY 100000000
#define REP(i, n) for (int i=0; i<(n); i++)

int n, m;        // pocet vrcholov a hran
double g[MAX][MAX]; // matica vzdialeností
double dist[MAX]; // dlzka najkratsej cesty do vrcholu
int done[MAX]; // je vrchol uz spracovany?
int back[MAX]; // odkial si prisiel
```

```
void dijkstra (){
  REP(i,n) { dist[i]=INFTY; done[i]=0; }
  dist[0]=0;
  REP(k,n) {
    int min, v;
    // najdeme najblizsi nespracovany vrchol
    min=INFTY;
    REP(i,n) if (!done[i] && dist[i]<min) { min=dist[i]; v=i; }
    // spracujeme ho
    REP(i,n)
      if (dist[i] > dist[v]+g[v][i]) {
        dist[i] = dist[v]+g[v][i];
        back[i] = v;
      }
    done[v] = 1;
  }
}

void output (int v) {
  if (v != 0) output (back[v]);
  printf ("%d ", v+1);
}

int main() {
  scanf ("%d %d", &n, &m);
  REP(a,n) REP(b,n) g[a][b] = INFTY;
  REP(i,m) {
    int a, b; double p;
    scanf ("%d %d %lf", &a, &b, &p); --a; --b;
    g[a][b] = g[b][a] = -log (1-p);
  }
  dijkstra ();
  output (n-1);
  printf ("\n%.2lf\n", 1-exp(-dist[n-1])); // vysledna pravdepodobnost
}
```

7. O dešifrovaní

opravoval Mic
(max. 20 bodov)

Tento príklad bol tak trošku netradičný a o to viac dúfam, že sa vám páčil. Rozšifrovať Vigenèrovu šifru nie je až také zložité, ale dá to nejakú robotu. Tak sa pozrime, ako na to.

Prvá úloha: Stačilo si všimnúť podivných znakov `'N/V++'`, ktoré sa nápadne podobajú na známe 'C/C++'. Čo ak hej? Tak potom by na daných pozíciach boli časti hesla `'K'` a `'S'`. Pozornejší riešiteľ zbystrie a vyskúša, či náhodou heslo nie je KSP. A čuduj sa svete, bolo. A riešením bol odstavec z našich pravidiel, konkrétne 'Požiadavky na riešenia':

```
Poziadavky na riesenia
Zakladom riesenia kazdeho prikladu (ak nie je v zadani uvedene inak) je listing
programu v lubovolnom vyssom programovacom jazyku (najlepsie Pascal, C/C++). Na
programe nas zaujima najma jeho korektnost (dava program pre kazdy vstup
spravny vysledok?) a efektivita (kolko casu a pamate potrebuje na spracovanie
vstupu urcitej velkosti). Doraz kladte na algoritmicku cast programu, pri
nacitavani vstupu a vypisovani vystupu nemusite uplne presne dodrziavat format
zo zadania. Rovnako dolezitou sucastou riesenia je slovny popis riesenia.
Slovny popis by mal obsahovat popis pouziteho algoritmu, pripadne popis
```

```
klucovych datovych struktur. Mal by byt natolko jasny a zrozumitelny, aby bolo
podla neho mozne napisat program rovnako efektivny, ako ten vas. Dalej
vyzadujeme zdovodnenie (dokaz) spravnosti pouziteho algoritmu a odhad casovej a
pamatovej zlozitosti algoritmu (t.j. kolko casu a pamate potrebuje vas program
v zavislosti od velkosti vstupnych dat). Na tejto stranke najdete ukazkovo
vyriesenych niekolko starsich prikladov. Na tej istej stranke sa mozete aj
docitat, co je vlastne casova a pamatova zlozitost (ak vam tieto pojmy vela
nehovoria).
```

Druhá úloha: V prvej úlohe sme vedeli, kde začať (našli sme reťazec `'N/V++'`, kde sme vedeli uhádnuť, čo sa v ňom skrývalo), ale v druhej úlohe nie sú žiadne špeciálne znaky a ani medzery. Keby tam boli medzery, tak by sme mohli nájsť krátke slová (jedno, prípadne dvojpísmenové) a tipnúť si, čo tam bude po rozkódovaní (jednopísmenových a dvojpísmenových slov je v slovenčine málo). Avšak rozšifrovať takúto šifru sa dá aj inak. Skúsme napríklad uhádnuť dĺžku hesla d.

Pozrime sa na nasledovný príklad:

```
Šifra: LAEUT AZLQX TIUTI LLQXW EMKUL AEU
```

Pôvodný text nepoznáme (ani heslo), ale v zašifrovanom texte vidíme dvakrát reťazec `LAEU` (na 1 a 25 pozícii). Ak by obe časti textu boli pôvodne rovnaké a zašifrované rovnakou časťou hesla, tak to naznačuje, že dĺžka hesla d asi delí číslo $25 - 1 = 24$ (24 má zvyšok 0 po delení číslom d).

Ako to vyzerá v skutočnosti?

```
Text:  AHOJA KOSAM ASJAS AMAMD OBREA HOJ
Heslo: KSPKS PKSPK SPKSP KSPKS PKSPK SPK
Šifra: LAEUT AZLQX TIUTI LLQXW EMKUL AEU
```

V tomto prípade je dĺžka hesla 3 a naozaj $24 = 8 \cdot 3$. Vráťme sa teraz k druhej šifre. Najdlhšie reťazce, ktoré sa tam opakujú majú dĺžku 7. Sú to `BWBIZNW`, ktorý je na pozíciach 75 a 219 a reťazec `SPZJOXX`, ktorý je na pozíciach 748 a 802. Ak budeme predpokladať, že oba vznikli zašifrovaním toho istého reťazca, tak potom zrejme d delí $219 - 75 = 144$ a zároveň d delí 54. Aké môže byť najväčšie d? Bude určite menšie alebo rovnaké najväčšiemu spoločnému deliteľu čísiel 54 a 144, čo je 18. Pozrime sa teraz na reťazce dĺžky 6. Sú to: `OPPXYZ` na pozíciach 396 a 504, `QJERHL` na 342 a 552, `RKDBWZ` na 318 a 612 a tiež predchádzajúce reťazce dĺžky 7 (ich podreťazce). Z tohto dostaneme, že d pravdepodobne delí čísla 108, 210 a 294. Výpočtom najväčšieho spoločného deliteľa dostaneme číslo 6 a teda heslo môže byť dlhé 1, 2, 3 alebo 6 znakov.

Keďže už máme dostatočne málo kandidátov na dĺžku hesla a vyberieme si toho najväčšieho. Pozrime na slovenčinu. Najčastejšie sa vyskytujúce znaky v slovenčine sú `'A'`, `'O'`, `'E'`, `'S'` a `'N'`. Teraz budeme postupne zisťovať jednotlivé znaky hesla. Ukážeme to len na jednom a konkrétne druhom znaku hesla. Vypíšme si iba tie znaky, ktoré sa zašifrujú druhým znakom hesla a spočítajme, koľko krát sa tam ktorý nachádza. Vidíme, že sa tam nachádza znak `J` 26-krát. Najčastejším písmenom v slovenčine je A, tak vyskúšame, či by to nemohlo byť A. Preto nastavíme druhý znak hesla na I a vypočítame príslušné výskyty jednotlivých znakov po rozkódovaní. Po takomto posune bude znak `Q` v texte 21-krát a `F` tam bude 14-krát, čo asi príliš nezodpovedá slovenčine. Druhé najčastejšie písmeno je tam `Z`. Žeby to bolo prísmeno `A`? V takom prípade by bolo potom v texte písmeno `W` 16-krát, čo je zase nejaké divné. Skúsme teda `V`, ktoré je v texte 16-krát. Nastavíme teda druhé písmeno hesla na `U` a pozrieme sa, koľkokrát sa tam ktorý znak nachádza. Najčastejšie znaky sú `A`, `E`, `O`, `S`, `T`, ktoré majú ako jediné dvojciferný počet výskytov. Navyše znaky `F` a `Q` sa tam nenachádzajú a nachádza sa tam iba raz znak `X` a `W`, takže by to mohlo byť ono. Takýmto spôsobom môžeme postupne určiť všetky znaky hesla a rozšifrovať text. Heslo bolo `MULICA` a text bol vybraný z Príručky mladého cowboya, ktorú si môžete stiahnuť a prečítať na adrese `http://www.ksp.sk/ksp2.0/sustredenia/2007dobra_voda/main.pdf`.

Ak by ste radi videli len daný text, tak tu ho máte:

```
VTEJT OCAST ISADO ZVIET EAKOP OSTAV ITPRA VUWES TERNO VUBUD OVUPR EDPOK LADOM
NASTA VBUJE STAVE BNEPO VOLEN IEKTO RESAD AZISK ATVHO DNYML OBING OMUNI EKTOR
EHOST AVEBN EHOUR ADNIK AANIC OKOLA DNIKD YNIEJ EDOST POZIS KANIS TAVEB NEHOP
OVOLE NIAJE POTRE BNEZO HNATS TAVEB NYMAT ERIAL NASTA VBUJE POTRE BNYCH NIEKO
LKODO SIEKK LINCE KLADI VOPIL ASTYR IPANT YLANO ANAPL ASTEN ADOTL CENEP RSTYV
PRIPA DENED OSTAT KUDOS IEKDO BREPO SLUZI AJLEP ENKAN AJSKO RSINA ZEMNA CRTNE
METVA RBUDO VYVZI VOTNE JVELK OSTIP RIPOH LADES PREDU NASLE DNENA CELYO BRAZO
KPOUK LADAM EDOSK YAOBR EZEME ICHTA KABYK OPIRO VALIO BRYSB UDOVY POMOC OUKLI
NCOVA KLADI VAPRI PADNE ESTEN EJAKY CHDOS IEKVS ETKYD OSKYP OZBIJ AMEDO HROMA
DYAZA ROVEN SIOBN APLAS TUVAV AMEPR STYPO KTORY CHSIT RIESK AMEPO TOMNA CELUD
OMYSE LNUKO NSTRU KCIUP RIBIJ EMEPO DPERY TAKAB YBUDO VAPOV ZTYCE NIDOV ERTIK
ALNEJ POLOH YNESP ADLAN ASLED NEBUD OVUPO MOCOU LANAP RESUN IEMED OVERT IKALN
EJPOL OHYVY REZEM EDVER EAPOM OCOUP ANTOV ICHPR IPEVN IMEBU DOVUN AMALU JEMEA
POVYS CHNUT IUMIE STNIM ENAPO ZADOV ANEMI ESTON IEKTO RYBUR ZUJIZ VYKNU OSTAT
NESTE NYADO KONCA AJSTR ECHUP ODLAZ IEPOS CHODI EAPOD OBNEA LEKON STRUK CIATA
KYLUX USNYC HKOMP LEXOV JENAD RAMEC TOHTO TEXTU
```

Tretia úloha: Tento zašifrovaný text bol najťažší, čo sa aj prejavilo na počte riešení. Pre nedočkavcov poviem, že heslo bolo `CICERO` a jazyk bola latinčina. Výsledný text bola pôvodná verzia Lorem ipsum (`http://wikipedia.org/wiki/Lorem_ipsum`). Po rozšifrovaní by ste mali dostať niečo takéto:

```
Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium
doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore
veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim
ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia
consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque
porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur,
adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et
dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis
nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex
ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea
voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem
eum fugiat quo voluptas nulla pariatur?
```

Ako sa to malo rozšifrovať? Určenie dĺžky hesla šlo ľahko. Nachádza sa tam slovo `ERQME-WJWJE` dvakrát (slovo, oddelené medzerami) na pozíciach 43 a 289, z čoho dostaneme, dĺžka hesla delí číslo 246. Keďže $246 = 2 \cdot 3 \cdot 41$ a 41 je prvočíslo, tak sa dá ľahko odvodiť, že dĺžka hesla bude asi deliť číslo 6. Ak si nájdeme ďalšie opakujúce sa reťazce, tak sa nám teto odhad len potvrdí.

Ako hádať heslo, keď nevieme jazyk? Budeme musieť spraviť tip. Čo ak je povodný jazyk angličtina? Ak budeme využívať výskyty jednotlivých písmen, tak sa nám podarí odvodiť heslo CISERO[2]. Keď si to pomocou toho hesla rozšifrujeme, tak dostaneme na latinčinu sa podobajúci text. Zoberme si prvých 5 slov `SEN UT PERCPICIADIS UNDO OMNIS`. Slovo percpiciandis vyzerá trošku zvláštne (obsahuje rcp). Ak si ho skúsime vyhľadať (Google), tak zistíme, že správne to má byť `PERSPICIATIS`. Teraz nám stačí podľa toho upraviť heslo a dostaneme heslo `CICERO`.

V poslednom prípade sme síce jazyk tipli zle, ale napriek tomu sme dostali dobrý výsledok, lebo angličtina prebrala z latinčiny veľa slov. Ak by nám tip s angličtinou nevyšiel, najlepšie by bolo potom skúsiť napríklad nemčinu a potom nejaké dalšie typy jazykov (napríklad ugrofínske a iné). Ako vidíme, na dešifrovanie sme skoro vôbec nepotrebovali vidieť dešifrovaný text, jediné na čo sa nám stačilo pozerať, boli počty výskytov jednotlivých písmen.

[2]Trafili sme sa až na jeden znak.

8. O božskej svadbe

Opravoval Lukáš
(max. 25 bodov)

Ukážeme si riešenie s časovou zložitosťou $O(N)$. Zo zadania vyplýva, že každá bytosť má božskosť najmenej 0 a najviac 1. Tento fakt využijeme v našom riešení. Najprv pôjdeme od najstarších ku najmladším a budeme si pre každú bytosť značiť minimálnu a maximálnu možnú božskosť, určenú iba za pomoci starších bytostí. Keďže tieto dve hodnoty budeme určovať len za pomoci predkov, tak ešte nedostaneme celkom presné hodnoty minima a maxima pre každú bytosť. K tomuto sa však ešte vrátime neskôr.

Ak nejaká bytosť nemá rodičov, tak buď božskosť je jedno číslo x a vtedy je interval možných hodnôt $\langle x, x\rangle$, alebo božskosť nepoznáme a vtedy je interval hodnôt $\langle 0, 1\rangle$. Ak chceme vyrátať interval pre bytosť, ktorej rodičov poznáme, potrebujeme poznať intervaly rodičov tejto bytosti. Nech zodpovedajúce intervaly rodičov sú $\langle m_a, m_b\rangle$ a $\langle o_a, o_b\rangle$, potom interval pre tento vrchol je $\langle (m_a + o_a)/2, (m_b + o_b)/2\rangle$. Ak sa stane, že božskosť je známa dopredu a nie je z tohto nami vyrátaného intervalu, určite nevieme správne určiť božskosť všetkých predkov, preto v takejto situácii prehlásime, že Mircos má nesprávne informácie. V opačnom prípade postupujeme rovnako, ako pri bytostiach bez rodičov – ak x je daná božskoť, potom zodpovedajúci interval je $\langle x, x\rangle$.

Predstavme si teraz, že sme už vyrátali tieto predbežné odhady minima a maxima pre celý rodokmeň. Platí, že pre každú bytosť A sme minimum a maximum odhadli tak, že jeho predkom sa dajú priradiť božskosti tak, aby bytosť A mohla mať ľubovoľnú hodnotu božskosti medzi svojim minimom a maximom. Špeciálne si všimnime koreň. Jeho minimum a maximum sme vypočítali podľa celého stromu, takže to znamená, že vieme celý strom ohodnotiť tak, aby mal koreň božskosť z toho nami vyrátaného intervalu.

Teraz budeme strom prechádzať od najmladších ku najstarším. Začneme tým, že koreňu priradíme ľubovoľnú hodnotu z jeho intervalu. Pre jednoduchosť si zvolíme minimum. Predpokladajme, že bytosti B sme priradili hodnotu x a predtým vyrátané intervaly rodičov sú $\langle m_a, m_b\rangle$ a $\langle o_a, o_b\rangle$. Nech hodnoty rodičov sú $m_a + y$ a $o_a + z$, pričom $0 \le y \le m_b - m_a$ a $0 \le z \le o_b - o_a$. Potom $x = (m_a + y + o_a + z)/2$, z čoho úpravami dostaneme $y + z = 2x - m_a - o_a$. Zaujíma nás hocijaké riešenie, preto ak na pravej strane je hodnota menšia ako maximum pre y (teda $m_b - m_a$), jedno z riešení je $y = 2x - m_a - o_a$ a $z = 0$. V druhom prípade je riešením napríklad $y = m_b - m_a$, $z = 2x - m_a - o_a - y = 2x - m_a - o_a - (m_b - m_a) = 2x - o_a - m_b$.

Týmto spôsobom vieme z hodnoty potomka vyrátať hodnoty rodičov a opakovaním tohto postupu pre obidvoch rodičov rekurzívne dorátame hodnoty celého stromu. Celý algoritmus sa bude skladať z dvoch rekurzívnych funkcií. Tá prvá bude zodpovedať prvej časti nášho algoritmu, kde zostrojíme intervaly pre bytosti. Najprv funkcia vyráta intervaly pre rodičov bytosti a z nich vyráta interval pre samotnú bytosť. Druhá funkcia bude zodpovedať druhej časti, kde dorátame božskosti jednotlivých bytostí. Zo známej hodnoty božskosti bytosti doráta božskosť jej rodičov. Zložitosť obidvoch častí algoritmu je lineárna, lebo každý vrchol spracujeme presne raz. Takisto pamäťová zložitosť je lineárna, lebo si pamätáme len pár čísel pre každý vrchol stromu.

Listing programu:

```
#include <string>
#include <cstdio>
#include <cstdlib>
#include <vector>
using namespace std;

struct Bytost
{
    double mi, ma, res;
    int r[2];
```

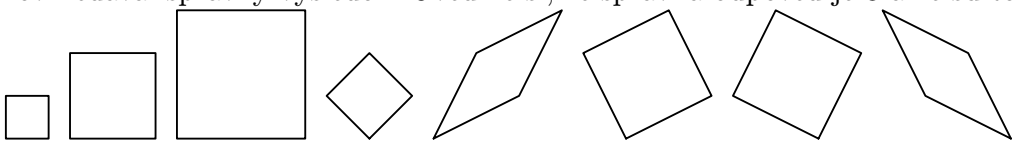
```
    Bytost(double _mi, double _ma, int _r0, int _r1) :
        mi(_mi), ma(_ma)
    {
        r[0] = _r0;
        r[1] = _r1;
    }
    void zrataj1();
    void zrataj2(double);
};
vector<Bytost> a;
bool zle = false;
void Bytost::zrataj1()
{
    if (r[0] != -1)
    {
        a[r[0]].zrataj1();
        a[r[1]].zrataj1();
        mi = max(mi, (a[r[0]].mi + a[r[1]].mi) / 2);
        ma = min(ma, (a[r[0]].ma + a[r[1]].ma) / 2);
        if (mi > ma)
            zle = true;
    }
}
void Bytost::zrataj2(double result)
{
    res = result;
    if (r[0] != -1)
    {
        double zostalo = 2 * res - a[r[0]].mi - a[r[1]].mi;
        double y = min(a[r[0]].ma - a[r[0]].mi, zostalo);
        a[r[0]].zrataj2(a[r[0]].mi + y);
        a[r[1]].zrataj2(a[r[1]].mi + max(0., zostalo - y));
    }
}
int main()
{
    int n; scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        int x; scanf("%d", &x);
        int l, p;
        if (x == 1)
            scanf("%d %d", &l, &p);
        char b[100];
        scanf("%s", b);

        double e = 0, f = 1;
        if (string(b).find("-1") == string::npos)
        {
            int s, t;
            sscanf(b, "%d/%d", &s, &t);
            e = f = double(s) / t;
        }

        if (x == 1)
            a.push_back(Bytost(e, f, l - 1, p - 1));
        else
            a.push_back(Bytost(e, f, -1, -1));
    }
```

```
    a[0].zrataj1();
    if (zle)
        printf("Mircos ma nespravne udaje\n");
    else
    {
        a[0].zrataj2(a[0].mi);
        for (int i = 0; i < n; i++)
            printf("Bytost %d: %lf\n", i + 1, a[i].res);
    }
}
```

Opravoval Zemčo
(max. 25 bodov)

9. Taká stanovačka...

Veru, tento príklad vyzeral kamarátsky ako farebná kvetinka a vy ste sa na neho pozlietali ako hladné včielky vo vidine chutných bodov. Treba však povedať, že také jednoduché to nebolo a bohužiaľ všetky riešenia, ktoré prišli, boli nekorektné. Už pre $N = 3$ žiadny z vašich programov nedával správny výsledok. Uveďme si, že správna odpoveď je 8 a že sú to tieto:



Príklad teda nikto korektne nevyriešil. Keďže sme usúdili, že je ho škoda, tak si ho môžete nájsť v ďalšej sérii ako príklad číslo 8. Veríme, že to skúsite aj nabudúce a že vaše riešenia budú všetky korektné (okrem obrázku, nech je vám nápomocná aj informácia, že pre 4 je odpoveď 16 a pre 6 je to 39).

Opravoval Nik
(max. 25 bodov)

10. Tanečná

Tento príklad neriešil nik a pritom nebol až taký ťažký. 0 odovzdaných riešení si vysvetľujeme tým, že sa to nikto ani nepokúsil riešiť. Aby nebola toho príkladu škoda, tak sme sa rozhodli, že ho presunieme do ďalšej série (pričom bude pod číslom 9), aby ste neboli ukrátení o riešenie takého pekného príkladu.

Výsledková listina po 2. kole kategórie KSP-Z

	Meno a priezvisko	Škola	Trieda		1	2	3	4	5	Σ
1	Horňák Marián	Gym. Párovská Nitra	1	58	10	10	12	12	15	117
2	Balog Matej	Gym. Grösslingová BA	2	59	10	10	10	12	15	116
3	Korbaš Rafael	Gym. Hronská BA	2	47	10	9	10	7	16	99
4	Kekely Michal	Gym. Varšavská Žilina - Vlčince	2	51	10	10	10		15	96
5	Kováč Ondrej	Gym. sv. Cyrila a Metoda Nitra	2	44	10	9	6	12	14	95
6	Porubsky Martin	Gym. sv. Cyrila a Metoda Nitra	2	43	9	10	10	2	14	88
7	Mariš Andrej	Gym. Piaristická Nitra	1	37	10	8	3	7	14	79
7	Mrocková Mária	Gym. Jura Hronca BA	2	29	10	8	10	8	14	79
9	Varga Mátyás	Gym. H. Selyeho Komárno	2	29	10	7	10	7	14	77
10	Jurenka Vladimír	Gym. Grösslingová BA	3	38	10	6	10	12		76
11	Cuc Bruno	Gym. Grösslingová BA	3	49	10	4	7		3	73
12	Dresslerova Anna	Gym. Jura Hronca BA	2	29	10	10	9		14	72
12	Stančiaková Katarína	Gym. Jura Hronca BA	3	34	10		3	12	13	72
12	Vecerik Matej	Gymnázium	2	38	10	4		6	14	72
15	Rohár Pavol	Gym. M. R. Štefánika, Košice	3	42				15	14	71
16	Anderle Michal	Gym. Haličská Lučenec	2	26	10	9	10	12		67
17	Trebichavský Richard	Gym. Jura Hronca BA	1	41	10	0	3		12	66
18	Bachratý Martin	Gym. Veľká Okružná Žilina	3	32	10	0	4	4	15	65
18	Čačko Marek	Gym. M. Galandu T. Teplice	3	35	10	8	..	2	10	65
20	Kuchyňár Martin	Gym. Jura Hronca BA	3	35	7	8	2		12	64
21	Kučera Martin	Gym. Golianova Nitra	2	35	8	10	10			63
22	Tóthová Tatiana	Gym. Jura Hronca BA	3	17	10	9	10		13	59
23	Novella Tomáš	Gym. Alejová Košice	3	28		10	10		10	58
24	Šimko Stanislav	Gym. Jura Hronca BA	3	33	10	6	8			57
25	Pokorný Fridolín	Gym. Haličská Lučenec	3	28	1	7	6	1	13	56
26	Matuš Juraj	Gym. Jura Hronca BA	1	16	10	7	8		13	54
27	Phuong Mariana	Gym. Jura Hronca BA	2	29	9	10				48
28	Sabo Matus	Gym. Jura Hronca BA	2	41						41
29	Strapko Martin	Gym. Jura Hronca BA	2	30	10					40
30	Takács Gabriel	Gym. Fándlyho Šaľa	1	6	7	7	3	1	14	38
31	Jurových Jakub	Gym. Okružná Zvolen	2	30	7					37
31	hruby tomas	Gym. Jura Hronca BA	2	27	10					37
33	Vozarova Zuzana	Gym. Jura Hronca BA	1	28			8			36
34	Mutný Mojmír	Gym. Jura Hronca BA	1	0	6	1	7	9	12	35
35	Kunec Sebastián	Gym. Konštantínova Prešov	4	33						33
36	Šeliga Adam	Súkromná SOŠ Humanus Via	1	15	6	2	3	1	4	31
37	Anderko Maroš	Gym. Konštantínova Prešov	3	13	7	7	3			30
38	Lačok Róbert	Gym. Jura Hronca BA	1	20	6					26
38	Pitoňák Martin	Gym. Tajovského B. Bystrica	3	0		10		3	13	26
40	Plavák Dušan	Gym. Trstená	2	22			3			25
41	Kuncová Alexandra	Gym. Alejová Košice	4	24						24
42	Machac Juraj	Gym. Jura Hronca BA	2	13			9			22
43	Lami Jozef	Gym. Poštová Košice	1	21						21
44	Bartošová Eva	Gym. Poštová Košice	3	20						20
44	Sládek Filip	Gym. Námestovo	3	20						20
44	Součková Kamila	Evanjelické lýceum Bratislava	0	20						20
47	Novák Ivo	Gym. Bernolákova V.n. Topľou	3	19						19
47	Toman Viktor	Gym. Golianova Nitra	2	19						19
49	Pinter Martin	Gym. Jura Hronca BA	2	18						18
50	Bohoš Peter	SPŠ Handlová	3	16						16

	Meno a priezvisko	Škola	Trieda		1	2	3	4	5	Σ
50	Iring Andrej	Gym. Jura Hronca BA	2	16						16
50	Jeník Dominik	Gym. Poštová Košice	3	16						16
53	Quoc Trung Dang	Gym. Jura Hronca BA	1	6	6		3			15
53	Václavík Lukáš	SSOĽ Via Humana	1	0	6	2	2	1	4	15
55	Dzurilla Bohus	Gym. Alejová Košice	3	11						11
55	Uchnár Matúš	Gym. Alejová Košice	3	11						11
57	Chynoransky Rastislav	Gym. Jura Hronca BA	0	4			6			10
57	Galovičová Soňa	Gym. Veľká Okružná Žilina	1	0	10					10
57	Toth Robert	Gym. Alejová Košice	3	10						10
57	Vasko Martin	Gym. Alejová Košice	4	10						10
57	Špano Marek	Gym. Jura Hronca BA	2	0	10					10
62	Hruška Eugen	Gym. Hlohovec	3	0	9					9
63	Zatrochová Zuzana	Gym. Alejová Košice	3	8						8
64	Holly Matej	Gym. Jura Hronca BA	0	4			3			7
64	Sevela Richard	Gym. Jura Hronca BA	3	0		7				7
66	Hulicka Matej	Gym. Jura Hronca BA	1	6						6
66	Stora Michal	Gym. Jura Hronca BA	2	6						6
66	Štajer Andrej	Gym. Dolný Kubín	3	0	6					6
66	Šulík Peter	Gym. Jura Hronca BA	2	6						6
70	Kreuzerová Veronika	Gym. Jura Hronca BA	1	5						5
70	Mikulaj Pavol	Gym. Školská Spiš. Nová Ves	1	5						5
72	Minda Marek	Gym. Jura Hronca BA	1	4						4
72	Szomolai Patrik	Gym. Jura Hronca BA	2	4						4
74	Albert Matej	Gym. Jura Hronca BA	1	0			3			3
74	Bogyai Filip	Gym. Jura Hronca BA	2	3						3

Výsledková listina po 2. kole kategórie KSP-O

	Meno a priezvisko	Škola	Trieda		4	5	6	7	8	Σ
1	Rohár Pavol	Gym. M. R. Štefánika, Košice	3	71	15	14	19	20	24	163
2	Belan Tomáš	Škola pre mim. nadané deti BA	3	60	12	14	19	20		125
3	Hozza Ján	Gym. Jura Hronca BA	2	67		15	19	20		121
4	Balog Matej	Gym. Grösslingová BA	2	30	12	15	19	20		96
5	Hudec Tobiáš	Gym. Partizánske	3	79						79
5	Pitoňák Martin	Gym. Tajovského B. Bystrica	3	0	3	13	19	20	24	79
7	Csiba Peter	Škola pre mim. nadané deti BA	4	55						55
7	Horňák Marián	Gym. Párovská Nitra	1	28	12	15				55
9	Stančiaková Katarína	Gym. Jura Hronca BA	3	18	12	13		7		50
10	Korbaš Rafael	Gym. Hronská BA	2	23	7	16				46
11	Novella Tomáš	Gym. Alejová Košice	3	0		10		7	24	41
12	Kováč Ondrej	Gym. sv. Cyrila a Metoda Nitra	2	14	12	14				40
13	Kekely Michal	Gym. Varšavská Žilina - Vlčince	2	24		15				39
13	Mariš Andrej	Gym. Piaristická Nitra	1	11	7	14		7		39
13	Ziman Michal	Gym. Haličská Lučenec	3	18	..	14		7		39
16	Habovštiak Martin	Gym. Tvrdošín	3	12	5			20		37
17	Dresslerova Anna	Gym. Jura Hronca BA	2	0		14		20		34
17	Gandžala Jozef	Gym. Tajovského B. Bystrica	3	0		14		20		34
17	Šimko Stanislav	Gym. Jura Hronca BA	3	14				20		34
20	Varga Mátyás	Gym. H. Selyeho Komárno	2	12	7	14				33
21	Porubsky Martin	Gym. sv. Cyrila a Metoda Nitra	2	16	2	14				32
22	Matuš Juraj	Gym. Jura Hronca BA	1	18		13				31
22	Miklovič Tomáš	Gym. Nové Zámky	3	18		13				31
22	Tóthová Tatiana	Gym. Jura Hronca BA	3	18		13				31
25	Mutný Mojmír	Gym. Jura Hronca BA	1	0	9	12		7		28
25	Vecerik Matej	Gymnázium	2	8	6	14				28
25	Čačko Marek	Gym. M. Galandu T. Teplice	3	16	2	10				28
28	Trebichavský Richard	Gym. Jura Hronca BA	1	14		12				26
29	Cuc Bruno	Gym. Grösslingová BA	3	22		3				25
29	Pokorný Fridolín	Gym. Haličská Lučenec	3	11	1	13				25
31	Jurenka Vladimír	Gym. Grösslingová BA	3	12	12					24
31	Kuchyňár Martin	Gym. Jura Hronca BA	3	12		12				24
33	Bachratý Martin	Gym. Veľká Okružná Žilina	3	4	4	15				23
33	Machac Juraj	Gym. Jura Hronca BA	2	23						23
35	Mrocková Mária	Gym. Jura Hronca BA	2	0	8	14				22
35	Novák Ivo	Gym. Bernolákova V.n. Topľou	3	22						22
37	Fulla Peter	SPŠ strojnícka Spišská Nová Ves	4	20						20
37	Kováč Jakub	Gym. sv. Cyrila a Metoda Nitra	4	20						20
39	Vozarova Zuzana	Gym. Jura Hronca BA	1	19						19
40	Kunec Sebastián	Gym. Konštantínova Prešov	4	17						17
41	Rabatin Ladislav	Gym. Jura Hronca BA		15						15
41	Takács Gabriel	Gym. Fándlyho Šaľa	1	0	1	14				15
43	Sabo Matus	Gym. Jura Hronca BA	2	14						14
44	Anderle Michal	Gym. Haličská Lučenec	2	0	12					12
45	Bohoš Peter	SPŠ Handlová	3	11						11
45	Jurových Jakub	Gym. Okružná Zvolen	2	11						11
47	Lačok Róbert	Gym. Jura Hronca BA	1	8						8
47	Toman Viktor	Gym. Golianova Nitra	2	8						8
49	Holas Juraj	Gym. Jura Hronca BA	2	0				7		7
49	Kučera Martin	Gym. Golianova Nitra	2	7						7

	Meno a priezvisko	Škola	Trieda		4	5	6	7	8	Σ
49	Plavák Dušan	Gym. Trstená	2	7						7
52	Lami Jozef	Gym. Poštová Košice	1	5						5
52	Václavík Lukáš	SSOĹ Via Humana	1	0	1	4				5
52	Šeliga Adam	Súkromná SOŠ Humanus Via	1	0	1	4				5

Výsledková listina po 2. kole kategórie KSP-T

	Meno a priezvisko	Škola	Trieda		8	9	10	Σ
1	Fulla Peter	SPŠ strojnícka Spišská Nová Ves	4	70				70
2	Hudec Tobiáš	Gym. Partizánske	3	67				67
3	Rohár Pavol	Gym. M. R. Štefánika, Košice	3	34	24			58
4	Kováč Jakub	Gym. sv. Cyrila a Metoda Nitra	4	36				36
5	Novella Tomáš	Gym. Alejová Košice	3	0	24			24
5	Pitoňák Martin	Gym. Tajovského B. Bystrica	3	0	24			24
7	Matuš Juraj	Gym. Jura Hronca BA	1	9		1		10
7	Miklovič Tomáš	Gym. Nové Zámky	3	10				10
9	Tóthová Tatiana	Gym. Jura Hronca BA	3	9				9
10	Hozza Ján	Gym. Jura Hronca BA	2	7				7
10	Šimko Stanislav	Gym. Jura Hronca BA	3	7				7
12	Habovštiak Martin	Gym. Tvrdošín	3	5				5
13	Mariš Andrej	Gym. Piaristická Nitra	1	0		1		1
13	Quoc Trung Dang	Gym. Jura Hronca BA	1	0		1		1
13	Vozarova Zuzana	Gym. Jura Hronca BA	1	0		1		1
13	Šeliga Adam	Súkromná SOŠ Humanus Via	1	0		1		1