Korešpondenčný seminár z programovania
XX. ročník, 2002/2003
Katedra vyučovania informatiky FMFI UK,
Mlynská Dolina, 842 48 Bratislava

KSP finančne podporuje MICROSTEP-MIS spol. s r.o.

Kategória Z

Vzorové riešenia 1. kola letnej časti

Milí riešitelia!

Ako už možno viete, v piatok 16. mája 2003 organizovali KSPáci ďalší ročník celosvetovej (nielen) stredoškolskej programátorskej súťaže IPSC. Tento rok sa prihlásilo vyše 600 tímov z celého sveta a nás teší, že sa slovenskí riešitelia v tejto konkurencií nestratili. Zo slovenských tímov obstáli najlepšie tím J2 (15. miesto celkovo) z matfyzu a IdeaZ (16.) z UPJŠ v Košiciach. Najlepší náš stredoškolský tím bol 4Dimensional GJH team, ktorý skončil medzi stredoškolskými tímami na 20. mieste. Všetkým zúčastneným blahoželáme k dosiahnutým výsledkom. A už ostáva len dodať, že najlepšou cestou k úspechu v takejto silnej konkurencií je čítať vzoráky KSP. Hor sa na to.

Gn fvsen wr gh yra gnx, nol fgr fn arahqvyv.

KSPáci

1. Zúfalý poštár

opravoval Paľo
(max. 15 bodov)

Tento príklad patril medzi tie ťažšie. Len niekoľko z vás prišlo na riešenie v čase $O(N)$, za ktoré ste mohli získať maximálne **15 bodov**. Za riešenia v čase $O(N^2)$ ste mohli získať maximálne **12 bodov**, v čase $O(N^3)$ **9 bodov**, za nefunkčné riešenia maximálne **2 body**.

Ostrov s mestami a cestami medzi nimi si nazvime graf. Mestá budeme volať vrcholmi grafu a cesty hranami grafu. Graf, ktorý má N vrcholov a $N-1$ hrán a je spojitý (teda taký, ktorý spĺňa podmienky zadania) nazvime strom. V strome existuje len jedna cesta medzi každými dvomi vrcholmi. Vrcholy, z ktorých vychádza len 1 hrana nazvime listami. Vrchol, kde má byť postavená pošta nazvime centrum grafu. Centrum je taký vrchol, z ktorého je cesta do najvzdialenejšieho vrchola najkratšia možná.

V zadaní bol návod ako centrum nájsť. Spočítame si z každého vrchola vzdialenosť do všetkých vrcholov a zapamätáme si vzdialenosť najvzdialenejšieho vrchola. Nakoniec vypíšeme ten vrchol, ktorý ju má minimálnu. Zistenie vzdialeností medzi nejakým vrcholom a všetkými ostatnými sa dá urobit napr. pomocou prehľadávania do šírky, ktoré je možné naprogramovať v čase $O(N+M)$ – viz. riešenia predchádzajúcich sérií. Náš graf má však iba $N-1$ hrán, takže časová zložitost jedného prehľádávania je $O(N)$. My prehľadáme graf z každého vrchola jedenkrát, čiže celková časová zložitost je $O(N^2)$.

Existuje však viacero spôsobov ako to urobiť lepšie. Pozrime sa na vrcholy, z ktorých vychádza len jedna hrana (na listy). Ukážeme, že ak má graf aspon 3 vrcholy, tak žiadny list nemôže byt centrom. Ukážeme to sporom: Nech je teda nejaký list v centrom grafu. Označme si jeho suseda w. Ak existuje ešte nejaký iný vrchol x, tak cesta z vrchola v do vrchola x prechádza cez w, teda je o 1 dlhšia ako cesta z w do x, teda cesta z w do ľubovolného x je o 1 kratšia ako z v do x (okrem samotného bodu v), preto v nemôže byť centrum grafu. Ak teda žiadny list nie je centrom, tak ich všetky z grafu zmažme a skúmajme graf G_2, ktorý takto vznikne. Tento graf je znovu strom, lebo odstránením listu nemôže vzniknúť cyklus a graf ostane spojitý. Predstavme si, že by sme už vedeli centrum tohoto novovzniknutého grafu G_2. Označme si ho c. Ukážeme, že je aj centrom pôvodného grafu.

Nech x je ľubovoľný vrchol v G_2, najvzdialenejší vrchol od neho v G_2 označíme y. Zjavne y je list, inak by sme vedeli nájsť vzdialenejší vrchol. Ale v G žiaden z vrcholov G_2 nebol listom, lebo by sme ho odstránili. Preto sme určite vrcholu y odstánili nejakého suseda z (ktorý už bol list). Keď pridáme všetky odstránené hrany a vrcholy späť, bude najvzdialenejší vrchol od x práve z.[1] Takže každému vrcholu G_2 sa vzdiali najvzdialenejší vrchol práve o 1. Preto ale centrum grafu G_2 je rovnaké ako centrum pôvodného grafu.

Celý algoritmus bude fungovať nasledovne: Zmažeme všetky listy v pôvodnom grafe. V takto vzniknutom grafe znovu zmažeme všetky listy a pokračujeme dovtedy až kým nedostaneme graf s jedným alebo dvomi vrcholmi. Ak nám ostane len jeden alebo dva vrcholy, tak tieto sú centrami (centrom) grafu, ktorý nám ostal. A teda sú aj centrami pôvodného grafu. Tento algoritmus implementujeme takto: Pre každý vrchol si pamätáme jeho susedov v spájanom zozname. Pre každý vrchol si pamätáme aj počet jeho susedov. Zistíme, ktoré vrcholy majú len jedného suseda a vložíme ich do fronty. Potom do fronty vložíme špeciálny znak (napr. -1), ktorým oddelíme listy, ktoré sme do fronty pridali z pôvodného grafu od listov, ktoré nám vznikli až po ich odstránení z grafu. Potom budeme z fronty postupne vrcholy vyberať. Pre každý vybraný vrchol z fronty sa pozrieme na jeho susedov a každému susedovi znížime pocet susedov o jedna. Ak nám vznikol nový list, tak tento nový list vložíme do fronty. Ak vyberieme z fronty špeciálny znak, tak sa pozrieme, aké vrcholy máme vo fronte. Ak už tam máme iba posledný vrchol alebo posledné dva vrcholy grafu, tak tieto vrcholy sú centrami pôvodného grafu a teda ich vypíšeme. Inak znovu vložíme do fronty špeciálny znak, ktorým oddelíme listy, ktoré sú už vo fronte, od listov, ktoré vzniknú až po ich odstránení z grafu.

Každý vrchol pridáme do fronty len raz. Pri vybratí vrcholu z fronty sa pozrieme na hrany, ktoré z neho vychádzajú, ale na každú sa pozrieme maximálne dvakrát (jedenkrát z každého z vrcholov, ktoré hrana spája) a hrán je dokopy $N - 1$, takže časová zložitost algoritmu je $O(N)$. Pre každý vrchol si pamätáme jeho susedov v spájanom zozname. Teda aj pamäťová zlotitosť je $O(N)$.

Listing programu:

```
const MAXN = 1000;
type PMESTO=^MESTO;
     MESTO = record  next: PMESTO; sus: integer; end;

var n,a,b,i: integer;
    akt: PMESTO;
    poczmaz: integer;
    mesta: array[1..MAXN] of PMESTO; { susedia miest }
    pocsus: array[1..MAXN] of integer; { pocty susedov }
    fronta: array[1..MAXN] of integer;
    frzac, frpoc: integer;

procedure cesta(odkial,kam: integer);
var akt: PMESTO;
begin
    new(akt);
    akt^.sus := kam; akt^.next := mesta[odkial];
    mesta[odkial] := akt;
end;

procedure uvolni(co: integer);
var tmp, tmp2: PMESTO;
begin
```

[1] Presnejšie: najvzdialenejších vrcholov od x môže byť viac, y a následne z je ľubovoľný z nich.

```
    tmp := mesta[co];
    while (tmp <> nil) do begin tmp2 := tmp^.next; dispose(tmp); tmp := tmp2; end;
end;

procedure pridaj(co: integer);
begin fronta[(frzac+frpoc-1)mod MAXN +1] := co; inc(frpoc); end;

function zober: integer;
var r: integer;
begin r := fronta[frzac]; frzac := frzac mod MAXN + 1; dec(frpoc); zober := r; end;

begin
    {nacitanie}
    read(n);  for i := 1 to n do begin mesta[i] := nil; pocsus[i] := 0; end;

    if (n = 1) then begin  writeln('Postu treba postavit v meste 1.'); exit; end else
      if (n = 2) then begin writeln('Postu treba postavit v meste 1 alebo 2.'); exit; end;

    for i := 1 to n - 1 do begin
        read(a,b); cesta(a,b); cesta(b,a); inc(pocsus[a]); inc(pocsus[b]);
    end;

    frzac := 1; frpoc := 0;
    for i := 1 to n do if (pocsus[i] = 1) then pridaj(i);
    pridaj(-1); poczmaz := 0;

    while true do begin
        i := zober;
        if (i = -1) then begin
            if (poczmaz >= n-2) then break else pridaj(-1);
        end else begin
            pocsus[i] := 0;   akt := mesta[i];
            while akt <> nil do begin
                dec(pocsus[akt^.sus]);
                if (pocsus[akt^.sus] = 1) then pridaj(akt^.sus);
                akt := akt^.next;
            end;
            inc(poczmaz);
        end;
    end;

    if (frpoc = 1) then begin writeln('Postu treba postavit v meste ', fronta[frzac], '.');
    end else begin
        write('Postu treba postavit v meste ', fronta[frzac]);
        writeln(' alebo v meste ', fronta[frzac mod MAXN + 1], '.');
    end;
    for i := 1 to n do uvolni(i); {uvolnenie pamati}
end.
```

Opravovali Monika & WSX :)

2. Záh(r)adná párty

(max. 15 bodov)

Na začiatku začíname s poľom dĺžky N, v ktorom je vstup. Najskôr si spočítame počet písmen a čísel (urobíme to jedným prechodom cez pole, to nám nemôže ovplyvniť zložitosť). Týmto prechodom si trocha uľahčíme program.[2] Ďalej presunieme všetky písmena na ľavú stranu poľa, potom rovnakým spôsobom čísla na pravú stranu poľa a v strede nám potom

[2] Program sa dá upraviť tak, aby tento prechod nebol potrebný.

zostanú bodky. Presúvať písmena budeme asi takto: Na začiatku ukazujeme s premennou i na ľavý kraj poľa a druhou premennou j do poľa za miesto, kde majú končiť písmená. Premennou i sa posúvam v poli smerom doprava a hľadám prvé miesto, kde nie je písmeno. Potom sa začneme posúvať premennou j, ktorá bude hľadať všetky písmená, ktoré sú mimo časti, kam patria. Keď nejaké nájdeme, tak znaky na pozíciách i a j vymeníme. Takýmto spôsobom budeme pokračovať znova v posúvaní premennej i doprava, až kým neprejdeme všetkými miestami, kde majú byť písmená. To vieme presne, lebo sme si to na začiatku spočítali. Týmto prechodom sme presunuli všetky písmená na začiatok poľa a urobili sme maximálne dva prechody poľom (raz premennou i, raz j). Podobne teraz presunieme čísla na koniec poľa. Premenná i bude ukazovať do poľa na miesto, odkiaľ majú začínať čísla (to sme si tiež na začiatku spočítali). Premenná j bude ukazovať do poľa tesne pred i a budeme ju časom posúvať doľava. Posúvajme i doprava, kým je na príslušnom políčku poľa číslo. Akonáhle nájdeme bodku (písmená tu už nemôžu byť), začneme posúvať j doľava a hľadat číslo. (Doprava od j už nie sú žiadne čísla, ktoré by bolo treba presunúť.) Keď nájdeme nejaké číslo, tak znaky na pozíciách i a j vymeníme. Takto prejdeme celým zvyškom poľa a presunieme čísla napravo. Teraz budú všetky čísla napravo a všetký písmená naľavo, no a bodky budú v strede.

Celým poľom prejdeme najviac 5 krát: prvýkrát pri počítaní písmen a čísel, dvakrát pri presúvaní písmen na správne miesto a dvakrát pri presúvaní čísel. Preto casová zložitost je $O(N)$. Nižšiu časovú zložitosť už nevieme docieliť, pretože vždy musíme prejsť aspoň raz celým poľom, aby sme zistili, či sú prvky správne usporiadané. Pamäťová zložitosť je tiež $O(N)$, okrem vstupného poľa však používame len konštantný počet premenných.

Hodnotenie:

- časová zložitosť $O(N)$ **15 bodov**
- časová zložitosť $O(N \log N)$ **13 bodov**
- časová zložitosť $O(N^2)$ **10 bodov**
- pri využití pomocného poľa **-7 bodov**
- pokiaľ nebol priložený popis alebo bol program nedostatočne popísaný, vášmu riešeniu boli odčítané body (od 1 do 7 v závislosti od popisu).
- a tí, ktorí opisovali, majú počet bodov predelený počtom členov „kolektívu“

Poznámka. Síce to tak možno nevyzerá, ale to, čo ste v tejto úlohe naprogramovali, má veľa spoločného s triedením. Predstavme si nasledovný spôsob triedenia. Vyberieme si náhodný prvok X z poľa. Prvky menšie od neho budú predstavovať „písmená“, rovnako veľké budú „bodky“ a väčšie budú „čísla“. Teraz spustíme práve vysvetlený algoritmus a preusporiadame (v lineárnom čase) pole. Na začiatku budú prvky menšie ako X, potom prvky rovné X a za nimi prvky väčšie ako X. To už je na dobrej ceste k utriedenému poľu. Len ešte potrebujeme utriediť prvý aj tretí úsek. Tak sa na každý z nich rekurzívne zavoláme. T.j. spravíme s nimi presne to isté – zvolíme si prvok, preusporiadame úsek poľa a ak ešte máme nejakú neutriedenú časť, opäť sa na ňu zavoláme, atď.

A ajhľa, na svete je triedenie. A **rýchle** triedenie. Volá sa QuickSort a v priemernom prípade má časovú zložitosť $O(N \log N)$. Lepšia sa ani nedá dosiahnuť!

Listing programu:

```
Uses Crt;
Var A       : Array [1..1000] of char; {tu je nacitany vstup}
    cis,pis : integer;                 {pocet cisel a pismen}
    i,j,N   : integer;
    c       : char;

begin
```

```
  cis:=0; pis:=0;
  readln(N); for i:=1 to N do read(A[i]);
  for i:=1 to N do begin         {vypocet poctu pismen a cisel v poli}
    if ( (A[i]>='0') and (A[i]<='9')) then Inc(cis);
    if ( (A[i]>='A') and (A[i]<='Z')) then Inc(pis);
  end;

  i:=1;j:=pis+1;
  repeat       {tento repeat je pre presuvanie pismen nalavu stranu pola}
    while (  (A[i]>='A') and (A[i]<='Z') and (i<=pis) ) do inc(i);
    while (not ( (A[j]>='A') and (A[j]<='Z') ) and (j<=N) ) do inc(j);
    if (i<=pis) and (j<=N) then begin c:=A[i]; A[i]:=A[j]; A[j]:=c; end; {vymena znakov}
  until (i>pis);

  i:=N-cis+1;j:=N-cis;
  repeat          {tento repeat je na presuvanie cisel na pravu cast pola}
    while ( (A[i]>='0') and (A[i]<='9') and (i<=N) ) do inc(i);
    while (not ( (A[j]>='0') and (A[j]<='9') ) and (j>=1) ) do dec(j);
    if (i<=N) and (j>=1) then begin c:=A[i]; A[i]:=A[j]; A[j]:=c; end; {vymena znakov}
  until (i>N);
  writeln(A);
end.
```

3. Zamotaný Santo

opravovala Julka
(max. 15 bodov)

Bola som veľmi potešená, že väčšina riešení bola správna (aj keď nie vždy optimálna), ale veľa ich neprišlo. Avšak našlo sa aj pár nefunkčných riešení. A teraz prejdime k bodovaniu:

- Časová zložitosť $O(N)$ a pamäťová zložitosť $O(1)$ – max **15 bodov**
- Časová zložitosť $O(N)$ a pamäťová zložitosť $O(N)$ – max **14 bodov**
- Časová zložitosť $O(N \log N)$ a pamäťová zložitosť $O(N)$ – max **13 bodov**
- Časová zložitosť $O(N^2)$ a pamäťová zložitosť $O(N)$ – max **12 bodov**
- Časová zložitosť $O(N^3)$ a pamäťová zložitosť $O(N)$ – max **11 bodov**
- Chýbajúci popis **-2 body**
- Chýbajúce odhady časovej a pamäťovej zložitosti **-2 body**

Vzorové riešenie: Aby mala záhradka tvar konvexného mnohouholníka, musela spĺňať dve podmienky:

- Každý vnútorný uhol musí byť menší ako 180° (konvexnosť).
- Žiadne dve strany sa nesmú pretínať (existencia mnohouholníka).

Uvedieme dve možné riešenia. Prvé spočíva v nasledovnej myšlienke: Vyberieme sa po obvode mnohouholníka. Aby bol konvexný, musíme stále zatáčať do tej istej strany. Stačí? Nie! Ešte totiž môže nastať prípad, kedy prejdeme „po slučke“:

2cm

Potrebujeme ešte overiť, že ani takýto prípad nenastal. Uvedomte si, že na to nám stačí počítať si uhol, o ktorý sme sa už otočili. Keď obídeme celý mnohouholník, určite to bude násobok 360° (lebo sa vrátime, odkiaľ sme začínali). Pritom ak sme naozaj išli po obvode konvexného mnohouholníka, bude to presne 360°, inak viac (podľa počtu „slučiek“, ktoré sme prešli).

Pre zaujímavosť uvedieme aj druhé riešenie, ktoré viacerí z vás poslali ale nik poriadne nedokázal. Je známe, že v každom n-uholníku je súčet vnútorných uhlov $180(N-2)°$ čiže $\pi(N-2)$ radiánov. (Každý n-uholník vieme rozdeliť na $n-2$ trojuholníkov s vrcholmi v jeho vrcholoch. Súčet uhlov v trojuholníku je 180°.)

Prejdeme po našej lomenej čiare, v každom vrchole spočítame veľkosť menšieho z uhlov (t.j. veľkosť toho, ktorý je konvexný) a sčítame ich. Pre konvexný mnohouholník zjavne dostaneme práve $\pi(N-2)$ radiánov. Potrebujeme ukázať, že pre každú inú lomenú čiaru to vyjde ináč, ukážeme, že to bude menej.

Ak čiara tvorí obvod nekonvexného mnohouholníka, je to zjavné – namiesto konkávneho vnútorného uhlu φ zarátame menší uhol $2\pi-\varphi$, preto dostaneme výsledok menší ako $\pi(N-2)$. Ostáva nám prípad ak čiara sama seba pretína. Uvážte, že sa nemusíme zaoberať prípadmi, keď sa pretína vo vrchole alebo viackrát na tom istom mieste – vhodným drobným posunutím jednej hrany sa nezmenia ani uhly, ani skutočnosť, že čiara sama seba pretína. Máme teda situáciu ako na nasledujúcom obrázku.

3cm

Čierne bodky sú vrcholy čiary, biele sú priesečníky, oblúčikom sú vyznačené uhly, ktoré nameriame. Všimnime si, že rovina sa nám rozdelila na niekoľko oblastí – mnohouholníkov (a jednu nekonečnú časť). Čo vieme o týchto mnohouholníkoch povedať?

Máme 4 typy bodiek. Pôvodné vrcholy ležia na obvode jedného alebo dvoch mnohouholníkov. Počty takýchto bodiek označíme v_1 a v_2. Podobne p_3 bude počet priesečníkov, ktoré ležia na obvode 3 a p_4 takých, ktoré ležia na obvode 4 mnohouholníkov. Všetky oblasti dokopy majú $v_1+2v_2+3p_3+4p_4$ vrcholov. Všimnime si teraz len biele bodky a čiary medzi nimi. Platí nasledovný vzťah: počet bielych bodiek + počet oblastí (vrátane nekonečnej) je o 2 väčší ako počet čiar medzi nimi. (Toto je známe tvrdenie z teórie grafov, zvané Eulerova veta, ľahko sa dá dokázať indukciou. V matematickejšej podobe hovorí, že keď máme súvislý graf, ktorý sa dá nakresliť do roviny bez križovania hrán, tak počet vrcholov + počet oblastí = počet hrán + 2.) Počet bielych bodiek poznáme, tých je p_3+p_4. Z každej vychádzajú 4 čiary, teda čiar je $4(p_3+p_4)/2$. (To /2 preto, že každú čiaru sme zarátali na každom jej konci.) Naša lomená čiara nám teda vyznačila $2(p_3+p_4)+2-(p_3+p_4)=p_3+p_4+2$ častí, a teda p_3+p_4+1 mnohouholníkov.

Ľahko nahliadneme, že súčet vnútorných uhlov všetkých týchto mnohouholníkov je π-krát (súčet počtov vrcholov − dvakrát počet mnohouholníkov), teda $\pi(v_1+2v_2+p_3+2p_4-2)$. O koľko menší výsledok dostaneme my sčítaním vyznačených uhlov? Nezarátame uhly pri priesečníkoch. Takže za každý z p_4 priesečníkov vo vnútri budeme mať o 2π menej, za každý z p_3 priesečníkov na okraji o vyše π menej. Takisto pri každom z v_2 vrcholov vo vnútri zarátame len menší z uhlov, takže opäť o aspoň π menej. Nehovoriac o tom, že výsledok sa ešte zmenší, ak bude niektorý z uhlov pri vonkajších v_1 vrcholoch konkávny.

Takže výsledný súčet, ktorý dostaneme, bude menší ako $\pi(v_1+2v_2+p_3+2p_4-2)-2\pi p_4-\pi p_3-\pi v_2=\pi(v_1+v_2-2)=\pi(N-2)$, čo sme chceli ukázať.

Teraz ešte k implementačným detailom. Skoro nikto si neuvedomil, že celý čas počas výpočtu spočítané uhly zaokrúhľoval, takže vznikla nejaká nepresnosť. Počítanie s reálnymi číslami principiálne nemôže byť presné. Za to som síce nestrhávala body, ale nabudúce si to treba uvedomiť.

Ďalší problém bol ako vypočítať uhol medzi dvoma priamkami. Ten, ak máte základy z analytickej geometrie, ste ľahko mohli vypočítať pomocou skalárneho súčinu smerových vektorov týchto dvoch priamok tvorených bodmi A, B, C:

$$p_x=A_x-B_x,\ p_y=A_y-B_y,\ q_x=C_x-B_x,\ q_y=C_y-B_y$$

$$\cos\alpha=\frac{p_xq_x+p_yq_y}{\sqrt{p_x^2+p_y^2}\sqrt{q_x^2+q_y^2}}$$

Takže každý tento uhol sa vypočítal v konštantnom čase a uhlov bolo N. Čiže časová zložitosť je $O(N)$ a stačí nám si pamätať si v každom kroku 3 body, čo je konštanta. Takže pamäťová zložitosť je $O(1)$.

Listing programu:

```
type TPoint=record x,y:integer; end;
const epsilon=0.00001;
var a1,a2,a3,b1,b2:TPoint;
    i,N:integer;
    suma:real;

function uhol(a,b,c:TPoint):real;
var p,q: Tpoint; {smerovy vektor}
    cosi:real;
begin
  p.x:=a.x-b.x;   q.x:=c.x-b.x;   p.y:=a.y-b.y;   q.y:=c.y-b.y;
  {cosinus vypocitame zo skalarneho sucinu}
  cosi:=(p.x*q.x-p.y*q.y)/(sqrt(sqr(p.x)+sqr(p.y))+sqrt(sqr(q.x)+sqr(q.y)));
  if cosi=0 then uhol:=PI/2
   else if cosi=1 then uhol:=0 else if cosi=-1 then uhol:=PI else
     uhol:= ArcTan (sqrt (1-sqr (cosi)) /cosi);
end;

begin
  readln(N); if N<3 then begin writeln('nie je to mnohouholnik'); exit; end;
  suma:=0;
  readln(a1.x,a1.y); readln(a2.x,a2.y); readln(a3.x,a3.y);
  suma:=suma+uhol(a1,a2,a3);
  b1:=a1; b2:=a2;
  for i:=4 to N do begin
    a1:=a2; a2:=a3; readln(a3.x,a3.y);
    suma:=suma+uhol(a1,a2,a3);
  end;
  suma:=suma+uhol(a2,a3,b1);    suma:=suma+uhol(a3,b1,b2);
  if (suma<PI*(N-2)+epsilon) and (suma>PI*(N-2)-epsilon) then
    writeln('je to mnohouholnik') else writeln('nie je to mnohouholnik');
end.
```

4. Zaujímavý mrakodrap

opravoval Tom
(max. 15 bodov)

Väčšna z vás vyriešila tento príklad správne, ale nie úplne optimálne. Skoro všetci ste prišl na to, že iba výťahom (bez toho, aby sme šľapali pešo) sa dá dostať len na poschodia tvaru $c = ai + bj; i, j \geq 0$. Niektorí to využili priamo a hľadali minimum výrazu $|c - (ai + bj)|$ cez všetky možné hodnoty i a j.[3] To ale zďaleka nie je optimálne. Jedeno zo zlepšení, ktoré ste použili bolo to, že ak mám dané i, viem j, pre ktoré bude mať výraz $|c-(ai+bj)|$ minimálnu hodnotu vypočítať. Je to jedno z $j_1 = (c - ai) \bmod b$, alebo $j_2 = ((c - ai) \bmod j) + 1$. Druhý zlepšovák bol založený na jednuduchej myšlienke „keď som dole, idem hore, keď som hore, idem dole“. Ešte k tomu bolo treba dávať pozor aby sme sa nezacyklili. Oba tieto prístupy viedli k riešeniu s časovou zložitosťou $O(a + b)$. A našlo sa aj (jediné) riešenie fungujúce tak ako vzorové, ktoré si teraz vysvetlíme.

Na ktoré poschodia sa vlastne vieme dostať priamo výťahom? Predsa práve na tie, ktoré sú násobkom spoločného deliteľa čísel a a $-b$[4]. Dokážeme to takto: Zrejme ak sa na to poschodie dostať dá, tak je deliteľné všetkým, čím je deliteľné a i b (a aj $-b$). Ďalej ak sa vieme dostať na poschodie $d = nsd\ (a, -b)$ tak sa vieme dostať aj na každý násobok d, lebo ak $d = ai+bj$ tak $kd = k(ai+bj) = a(ki)+b(kj)$. Stačí sa už len vedieť dostať na poschodie d t.j. treba nájsť také $i, j \geq 0$ také, že $d = ai + bj$. Také sa dajú nájsť Euklidovym algoritmom

[3] Chcelo si to ešte uvedomiť, aké hodnoty i, j sa nám oplatí skúšať a aké už nie.

[4] predpokladáme, že $b < 0 < a$

popri hľadaní $nsd\ (a, -b)$. Jeho základná myšlienka je, že $nsd\ (x, y) = nsd\ (y, x \bmod y)$. Pomocou tohto vzťahu vieme rekurzívne nájsť hodnotu $nsd\ (x, y)$. Nech teraz navyše vieme, že $x = i_x a + j_x b$ a $y = i_y a + j_y b$. Potom $x \bmod y = x - y(x\ div\ y) = i_x a + j_x b - (x\ div\ y)(i_y a + j_y b) = (i_x - (x\ div\ y)i_y)a + (j_x - (x\ div\ y)j_y)b$.

Budeme si pamätať x, y, i_x, j_x, i_y, j_y a v cykle ich meniť podľa predošlého vzťahu. Až bude $y = 0$ tak x je hľadané $nsd\ (a, -b)$ Ešte by sa mohlo stať, že nám jedna z hodnôt i, j vyjde záporná. To ale nevadí, lebo keď zväčšíme hodnotu i o $-kb$ a hodnotu j o ka (pre vhodné dosť veľké k), hodnotu výrazu $ai + bj$ tým nezmeníme.

Aha, ešte by vás asi zaujímalo bodovanie, tak tu je: Počet bodov závisel hlavne od časovej zložitosti vašich programov a to takto:

- kubická $O((a+b)^3)$ – **3 body**
- kvadratická $O((a+b)^2)$ – **5 bodov**
- lineárna $O(a+b)$ – **7 až 9 bodov**
- logaritmická $O(\log(a+b))$ – **10 bodov**

a ešte k tomu sa dalo získať **5 bodov** za popis.

Listing programu:

```
program mrakodrap;
var
  a,b,c, {udaje zo vstupu}
  x,ix,jx,
  y,iy,jy,  {na spocitanie nsd(a,b)}
  z,iz,jz:integer;
  t:integer;
begin
  readln(a,b,c); {precitame vstup}
  {ratame nsd(a,-b)}
  x:=a;ix:=1;jx:=0;   {na zaciatku x=a=1*a+0*b}
  y:=-b;iy:=0;jy:=-1; {            y=b=0*b+1*b}
  while y<>0 do begin
    z:=x mod y;
    iz:=ix-(x div y)*iy;
    jz:=jx-(x div y)*jy;
    x:=y; ix:=iy; jx:=jy;
    y:=z; iy:=iz; jy:=jz;
  end;
    {teraz mame nsd(a,-b)=x=a*ix+b*jx}
  if jx<0 then begin  {ak vysli zaporne koeficienty tak ich upravime}
    t:=(-jx)div a+1;
    ix:=ix-t*b;
    jx:=jx+t*a;
  end;

  t:=c div x; {najblizsie poschodie zdola}
  if c-t*x>(t*x-c+x) then inc(t);  {alebo zhora ak je lepsie}
  {a uz len vypiseme}
  writeln('treba stlacit ',ix*t,' krat prve, ',jx*t,' krat druhe a vyslapat ',c-t*x);
end.
```

opravoval Žužuliačik
(max. 15 bodov)

5. Zaneprázdnené múzy

Všeobecne. Z veľkého množstva došlých riešení som bol nepríjemne rozčarovaný a preto si najprv všeobecne objasnime, o čo by ste sa mali pri riešení takýchto (a ostatných) úloh pokúšať. Stalo sa to neraz, po prečítaní zadania úlohy mi hneď v mysli prebleskло jej riešenie.

Bolo také jednoduché, že som sa touto úlohou ďalej už ani nezaoberal. No vtom prišiel za mnou kamarát a spýtal sa ma na moje úžasné riešenie. Lámane som sa mu ho pokúšal prerozprávať, ničomu nerozumel. Viete, myšlienky strácajú na hĺbke, keď ich premietame do slov. Dorozprával som a zistil som, že tomu ani nemal šancu porozumieť, pretože sám som si veľa častí len idealisticky predstavoval a opisoval som mu ich vzletmo a neurčito. Vedel som, že ja, autor všetkých tých myšlienok, by som tomu na jeho mieste nebol býval porozumel. Kde som spravil chybu? Bol som nedôsledný.

Pri riešení úloh v tomto seminári preto od vás vyžadujeme dôslednosť. Každé riešenie sa musí(!) začínať popisom myšlienok, ako ste úlohu riešili, dostatočne zrozumiteľným na to, aby si zručný programátor vedel podľa toho už samotný program vytvoriť sám. Predchádzajúca veta je kľúčová. Posielať program bez popisu je zbytočné(!). Posielať popis bez toho aby bol zrozumiteľný a podrobný je rovnako naničˇ. Skúsme predtým ako riešenie (popis s programom) zašleme, podstrčiť riešenie úlohy s našim popisom ale bez programu staršiemu kamarátovi, ktorý je zbehlý v programovaní (a nerieši KSP :-). Ak bez námahy naprogramuje funkčný program riešiaci danú úlohu, náš popis je dostatočný a riešenie pokojne pošlime. V opačnom prípade popis prepracujeme a experiment s kamarátom opakujeme až pokým nebude schopný realizovať funkčný program. Presne takto kvalitný popis od vás očakávame.

Konkrétne. Väčšine z vás neušla skutočnosť, že predkladaná úloha bola netradičná. Už dlho sa počítačoví inžinieri pokúšajú zostrojiť počítač schopný dôstojne diskutovať s človekom. Tvoriť poéziu je zjavne ešte nedosiahnuteľnejšia méta a preto sme tu od vás neočakávali riešenia ašpirujúce na Nobelovu cenu (aj keď ani to sa nevylučuje...), ale skôr sme chceli, aby ste sa zamysleli nad náročnou úlohou a navrhli nejaký rozumný spôsob, ako na ňu. Oceňovali sa všetky pionierske pokusy a všetky z tejto kategórie boli naozaj zaujímavé. Niektorí to však, pre mňa nepochopiteľne, považujú len za hlúpu naháňačku za bodmi. Veľmi ma potešilo, že táto skupinka tvorila drvivú menšinu a väčšina riešiteľov tejto úlohy sa riešeniu skutočne dôkladne venovala.

Spôsoby riešenia: Naučiť program štruktúru a krásy nášho jazyka je, v týchto časoch, nie celkom realizovateľné. Preto sa jediný spôsob riešenia zakladá na tom, že náš program „naučíme“ nejakú malú konkrétnu množinu informácií. Určite to bude slovník slov jazyka, ale náhodné postupnosti slov ešte netvoria verše, preto k slovníku musia pribudnúť ešte akési vzťahy a väzby medzi slovami v slovníku. Práve na tomto mieste sa vaše riešenia rozštiepili na viacero vetiev. Jednoduchšie (na realizáciu výsledných básničiek) je pamätanie si celých veršov. Básnička sa potom generuje ako viac-menej premyslené vyberanie týchto zapamätaných veršov. Výsledné diela sa potom pekne rýmujú, jednotlivé verše majú pekný básnický nádych, ale zväčša sú susedné verše myšlienkovo od seba odtrhnuté. Druhý spôsob využíva väzby medzi dvojicami (alebo skupinami) slov. Verše sa potom generujú po slovách tak, že vždy nadviažeme slovom, ktoré je s predchádzajúcimi nejako späté. Výsledkom budú myšlienkovo súvislé diela, ktoré sa budú rýmovať, ale stratí to poetický nádych a môže sa stať, že to skončí pri jednoduchých oznamovacích vetách, ktoré určite nie sú pre nás zaujímavé. Tretí spôsob spočíva v rozbore štruktúry veršu. Kde do veršu sa môžu aké typy slov podosadzovať, aby nakoniec vznikol ucelený celok. V tomto prípade môže byť význam susedných slov až nepríjemne nesúvisiaci. V praxi sa najlepší výsledok dosiahne správnym vyvážením týchto troch postupov. Myšlienky sú to v princípe jednoduché a prakticky všetci riešitelia, ktorý sa reálne nad úlohou zamysleli, ich vo svojich implementáciach viac-menej úspešne realizovali. Nejaké detailnejšie riešenie nie je vhodné sem, ale skôr na rozsiahlejší projekt.

Vaše riešenia, bodovanie: Vaše riešenia možno, podľa typov väzieb medzi slovami uvedenými vyššie, rozdeliť do týchto skupín.

- Generuje náhodnú postupnosť znakov, konce veršov sa upravia tak, aby sa rýmovali. Takýmto spôsobom vznikajú zhluky nič nehovoriacich písmen, ktoré zjavne netvoria básničky. (max. **2 body**).

- Databáza veršov, ktoré sa nejakým premysleným spôsobom používajú na generovanie básničky. (max. **8 bodov**).
- Databáza syntaktickej skladby veršov, do ktorej sa dajú flexibilne dopĺňať slová podľa slovných druhov alebo rodov. (max. **14 bodov**)

K týmto bodom sa dalo za popis a kvalitu veršíkov čo to získať alebo stratiť, najmenej sa však dalo získať 0 bodov a najviac 15 bodov. 15 bodov nezískal nikto, najviac – 14 bodov – získal Miro Cicko z Banskej Bystrice, ktorý si slová v databáze delil na slovné druhy, ktoré potom skladal do veršov v duchu riešení z tretej skupiny.

Vaše verše: Namiesto nejakého nášho zdrojového kódu sme sa rozhodli zverejniť niektoré zaujímavé verše od vás. Verše uvádzame v pôvodnom znení spolu s menom autora.

Matej Kuna
už nám fúka do komína
chladná pani Meluzína
na jeseň si vietor píska
mocne fučí na strniská
kváče káčer rapoták
že už nemá ani mak
ježko, ježo ježatý
čo ti treba na šaty?

Milan Sedlák
Sviečka sviečka otvor viečka
Tma nech stráži v poli škrečka
Prostred kruhu šťastie spinká
V poli krásna nezábudka
Pozor veď vám horí metla
Včera som zas jeho stretla
Tešila sa stará kára
Čo ten môj vnuk doma stvára

Andrej Pančík
Pihol ma veď viete kam
Nevyskakuj z voza
Padol na mňa nový krám
Vidím maďarského mroža

Milan Pĺžik
Dievča išlo k vode
Aj tak to bola ich chyba
Našli ho v záhrade
A hneď padla prvá žaba

Evička Schlosáriková
Janko na lúku išiel
Ráno kvetinku našiel
Ako zajačik išiel
Večer Janíčko prešiel

Miro Cicko
Zelený Kuko presýpa nízke dvierka.
Volá pre šťastie čerstvého vrabčeka,
no i tmavý medvedík skúša vôkol.
Drahá srnka pečie sťa by orol.

Výsledková listina po 1. kole kategórie KSP-Z

	Meno a priezvisko	Škola	Trieda	11	12	13	14	15	Σ
1	Cicko Miroslav	Gym. Tajovského B. Bystrica	2	15	15	15	14	14	73
2	Plžík Milan	Gym. Ľ. Štúra Zvolen	2	10	15	14	14	12	65
3	Sedlák Milan	Gym. Liptovský Hrádok	3	15	12	10	12	11	60
4	Poláčik Michal	GLN Tomášikova BA	3	12	10	14	15	6	57
5	Cermak Michal	Gym. Jura Hronca BA	3	12	15	14	11	1	53
	Ilavský Milan	SPŠE Liptovský Hrádok	3	11	10	14	12	6	53
	Morihladko Peter	Gym. Školská Spiš. Nová Ves	2	12	15	11	9	6	53
8	Gottweis Martin	Gym. Jura Hronca BA	1	7	12	12	12	6	49
	Lackovič Matúš	Gym. Mudroňova Prešov	2	8	15	12	12	2	49
	Legény Jozef	Gym. M. R. Stefanika, Košice	2	9	10	12	12	6	49
11	Labuda Tomáš	Gym. Grösslingová BA	3	11	13	7	12	5	48
	Schlosáriková Eva	Gym. SNP Piešťany	2	2	15	12	10	9	48
13	Struhár Pavel	Gym. Jura Hronca BA	2	12	15	7	12		46
14	Krchniak Matej	Gym. Jura Hronca BA	1	11	15	7	8	3	44
15	Mikuláš Ján	Gym. Haličská Lučenec	1	11	13	10	9		43
	Trnovcová Zuzana	Gym. Jura Hronca BA	2	9	14	10	10		43
17	Kolibiarova Hanka	Gym. Jura Hronca BA	1	12	15		12		39
18	Krčah Marcel	Gym. Párovská Nitra		11	15		12		38
19	Chamila Sergius	Gym. Dilongova Prešov		6	15	2	10	4	37
	Dravecky Pavol	Gym. Jura Hronca BA	1		15	6	10	6	37
21	Kuštárová Tamara	Bilingválne gym. Sučany	2		11	12	8	2	33
22	Ďuďák Juraj	Gym. Golianova Nitra	1	8	11	0	12		31
23	Bundala Daniel	Gym. Jura Hronca BA	1	1	15		10	4	30
	Vitásek Matej	Gym. Grösslingová BA	2	2	14		14	0	30
	Ľudvík Martin	Gym. Školská Spiš. Nová Ves	2	8	10		12		30
26	Zaťko Tomáš	Gym. 17. novembra Topoľčany	3	8	13		8		29
27	Mačura Martin	SPŠE Michalovce	2	7	8		10		25
	Trubeňová Barbora	Gym. Jura Hronca BA	3	8	8	8	1	0	25
	Vanderka Peter	Gym. Trstená		11	14				25
30	Kottman Michal	Gym. Jura Hronca BA	2	11	13				24
31	Pančík Andrej	Gym. Š. Moyzesa B. Bystrica	0		10		7	6	23
32	Kačur Tomáš	SPŠE Michalovce	2		15		7		22
33	Pekár Jaroslav	SPŠE Stará Turá	2		9		12		21
	Štefanec Richard	Gym. Jura Hronca BA	3		15			6	21
35	Bielik Peter	Gym. SNP Piešťany	2		8		10	2	20
	Šimko Jakub	Gym. Jura Hronca BA	2		12	8			20
37	Hruda Ivan	ZS Saratovská	0		7	2	9		18
38	Puskajler Ján	SPŠE Michalovce		6	8		1	2	17
39	Goga Peter	Gym. 17. novembra Topoľčany	3	1	10	2	1	2	16
	Kuna Matej	Gym. Golianova Nitra	2		10			6	16
	Varga Ľubomír	SPŠ Nitra	3	2	14				16
42	Antonič Michal	Gym. Grösslingová BA	1		6			9	15
	Drábik Juraj	Gym. Jura Hronca BA	1		8		7		15
	Fiala Martin	Gym. Grösslingová BA	3		15				15
	Panák Pavol	Gym. Jura Hronca BA	1		8		7		15
46	Gallusová Barbora	Gym. Grösslingová BA	2		13				13
47	Andruška Igor	SPŠ Levice	2	11	*3				11
	Borsuk Andrej	Gym. Grösslingová BA	1				11		11
	Hlavatá Ivana	Gym. Jura Hronca BA	2	11					11
	Špalek Jozef	Gym. Čadca			3	2		6	11

	Meno a priezvisko	Škola	Trieda	11	12	13	14	15	Σ
51	Halamíček	Gym. Jura Hronca BA	1		1	4	5		10
	Krištofík Štefan	SPŠ Levice	4		10				10
	Táborský Roman	Gym. Jura Hronca BA	1	2	8				10
54	Dillinger Vialiam	Gym. Jura Hronca BA	1		3		6	0	9
55	Imriška Jakub	Gym. Jura Hronca BA	1	8					8
	Parák Jakub	Gym. Jura Hronca BA	1		8				8
	Polák Ján	Gym. Jura Hronca BA	1		8				8
	Trembulák Michal	SPŠE Michalovce	2		8				8
	Valiska Ján	SPŠE Michalovce	2		8				8
60	Budáč Ondrej	Gym. Haličská Lučenec	1	3		4	0		7
	Hlinková Jana	Gym. Grösslingová BA	3		7				7
62	Holla Kamila	Gym. Jura Hronca BA	1		3		1	0	4
63	Nguyen Tomáš	Gym. Jura Hronca BA	1		3				3
	Soha Miroslav	Gym. Liptovský Hrádok	1		3			0	3
65	Sabo Matej	Gym. Jura Hronca BA	1		2				2
	Sopinka Ondrej	SPŠE Michalovce	1				1	1	2
67	Bíziková Ivona	Gym. Jura Hronca BA	1		1				1
	Demuthová Eva	Gym. Jura Hronca BA	1				1	0	1
	Foltýnová Monika	Gym. Jura Hronca BA	1		1				1
	Galla Martin	Gym. Jura Hronca BA	1				1		1
	Harton Eugen	Gym. Jura Hronca BA	1		1				1
	Helebrandt Pavol	Gym. Jura Hronca BA	1				1		1
	Holec Juraj	Gym. Jura Hronca BA	1		1				1
	Kalivoda Lukáš	Gym. Jura Hronca BA	1		1				1
	Kollár Jakub	Gym. Jura Hronca BA	1		1				1
	Nagyová Iveta	Gym. Jura Hronca BA	1		1				1
	Oremus Vladimír	Gym. Jura Hronca BA	1		0		1		1
	Polák Matúš	Gym. Kežmarok	2				1		1
	Thurzo Martin	Gym. Jura Hronca BA	1				1		1
	Šafár Števo	Gym. Jura Hronca BA	1		1				1
81	Cielontko Tomáš	Gym. Jura Hronca BA	1		0				0
	Polievka	Gym. Jura Hronca BA	1		0				0
	Števíková Tereza	Gym. Jura Hronca BA	1					0	0