Korešpondenčný seminár z programovania
XXII. ročník, 2004/05
Katedra základov a 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 1. kola letnej časti, kat. Z

„Človek, ktorý nedokáže hovoriť tak, aby mu ostatní rozumeli, je idiot. Rozumieš?“
„Nie, tati.“
Tak či onak, kto nečíta vzoráky, je hnilé vajce (veľkonočné).

1. Zažite tú hrôzu v knižnici!

opravoval Peťo
(max. 15 bodov)

Začneme asi tým, ako malo vyzerať vzorové riešenie, ku ktorému sa pár z vás prebojovalo. Algoritmus sa volá *binary search* alebo *binárne vyhľadávanie*.

Myšlienka je nasledovná: Pozrieme sa na knihu v strede poľa. Ak je to tá, ktorú hľadáme, tak si ideme podať športku. Ak to nie je tá, ktorú hľadáme, tak sa pozrieme, či je názov lexikograficky väčší alebo menší od hľadanej knihy. Ak je väčší, tak s istotou vieme, že kniha, ktorú hľadáme, je v prvej polovici nášho zoznamu, pretože je abecedne utriedený. Ak je menší, tak vieme, že kniha, ktorú hľadáme, je v druhej polovici. Teraz opakujeme to isté so zostávajúcou polovicou. Pozrieme sa na stredný prvok polovice, porovnáme a vyberieme vhodnú štvrtinu atď. Takto pokračujeme v delení a porovnávaní, až kým nenastane jedna z dvoch možností. Buď nájdeme našu knihu a vypíšeme jej číslo alebo nám ostane časť poľa dĺžky 2. Ak nastane tá druhá možnosť, tak buď je naša kniha jeden z krajných bodov časti poľa alebo kniha v zozname nie je. Porovnávanie dvoch string-ov pokladám vo vzorovom riešení za operáciu s konštantnou zložitosťou. Tam, kde ste porovnávali po znakoch, som to nehodnotil ako chybu, teda ak priemerná dĺžka slova je m, tak riešenie $O(m \times n) \approx O(n)$.

Časová zložitosť vzorového riešenia je $O(\log n)$. Prečo? V najhoršom prípade sa prepracujeme až k úseku poľa dĺžky 2. V každom kroku nášho cyklu sme skrátili pole na polovicu a na začiatku bolo pole dĺžky n, teda $n/2^k = 2$, kde k je počet krokov cyklu. Po úprave dostaneme $n = 2^{k+1}$; odtiaľ $\log_2 n = k + 1$. Môžeme teda napísať, že časová zložitosť nášho algoritmu je $O(\log n)$. Pamäťová zložitosť je $O(n)$, pretože používame iba pole dĺžky n. Ešte jedna rada na záver: pokiaľ to ide, tak sa skúste vyhýbať rekurzii, skoro som vám za to strhol bodík. Keď existuje elegantné riešenie bez rekurzie, tak ho použite.

Teraz ešte pár slov k hodnoteniu:

- Vzorové riešenie alebo podobné riešenie so zložitosťou $O(\log n)$ – $15b$
- Jednoduché lineárne riešenie, zložitosť $O(n)$ – $8b$
- Nefunkčné riešenia – max. $3b$
- Chýbajúci resp. nepostačujúci popis riešenia – $-2b$ resp. $-1b$
- Zlý alebo žiadny odhad zložitosti – $-1b$

Listing programu:

```
const max=10;

type kniha=record
  meno:string;
  cislo:integer;
```

```
end;

var knihy:array[0..max] of kniha;
  i,z,k,s,n:integer;{zaciatok,koniec,stred}
  hladam:string;{meno hladanej knihy}
  c:char;
  f:text;

begin
  assign(f,'knihy.in'); {nacitanie vstupu}
  reset(f);
  {o nacitanie vstupu neslo;moj vstup vyzera: }
  {2                                          }
  {42 Algebra                                   }
  {47 MatAlyza                                   }
  readln(f,n);
  for i:=0 to n-1 do begin
    read(f,knihy[i].cislo);{nacitam cislo}
    read(f,c); {nacitam medzeru ;)}
    readln(f,knihy[i].meno); {nacitam meno knihy}
  end;
  close(f);
  {teraz ide ta dvolezita cast}
  write('Zadaj knihu:');
  readln(hladam);
  {pozrem ci to nie je prva kniha, aby to nerobilo bordel}
  if knihy[0].meno=hladam then begin
     writeln('Cislo hladanej knihy je: ',knihy[s].cislo);
     exit;
  end;
  z:=0;
  k:=n;
  while(k>z+1)do begin
    s:=trunc((z+k)/2);
    if knihy[s].meno=hladam then z:=k+1
    else if knihy[s].meno>hladam then k:=s
    else z:=s;
  end;
  if knihy[s].meno=hladam then writeln('Cislo hladanej knihy je: ',knihy[s].cislo)
  else writeln('Hladana kniha sa tu nenachadza.');
end.
```

2. Zmenený Zákon

opravoval Tono
(max. 15 bodov)

Tento príklad sa pravdepodobne väčšine z vás zdal ľahký. Svedčí tomu fakt, že takmer všetky vaše riešenia fungovali. Asi aj preto (a bohužiaľ) sa väčšine z vás neunúvalo vymyslieť riešenie lepšie ako kvadratické (samozrejme, česť a body výnimkám). Možno nečítate vzoráky, možno riešite KSP po prvýkrát – v každom prípade odporúčam pustiť sa do nasledujúceho textu.

Riešenie: Bolo jednoduché prísť na to, že sa bude triediť. Problém bol jedine v tom, ako pri rovnakých menách zachovať pôvodné poradie. Toto sa dalo jednoducho vyriešiť pridaním ďalšieho čísla k činnosti, rovnajúcemu sa poradiu na vstupe. Potom už (napríklad po prečítaní predošlých vzorákov) nebolo ťažké napísať triedenie fungujúce v čase $O(N \log N)$.

Pozrime sa teraz na riešenie optimálne. Triedime reťazce, poďme teda skúmať ich vlastnosti. Pre jednoduchosť predpokladajme (hoci to v zadaní nie je), že sa skladajú len z malých

písmenok anglickej abecedy. Skúsme teraz vygenerovať všetky možné reťazce dĺžky najviac n. Na prvé miesto môžeme dať všetkých 26 znakov. Za každý znak môžeme potom dať znovu 26 rôznych znakov. Toto si môžeme zakresliť, napríklad do stromu. Začneme koreňom, ktorý reprezentuje prázdne slovo (také, ktoré nemá žiadne písmenko[1]). Z neho potom pôjde 26 hrán (pre každé písmenko jedna) do 26 nových vrcholov. A z každého z nich pôjde znovu 26 hrán do nových 26 vrcholov... až do hĺbky n. V tomto strome si teraz môžeme vyznačiť ľubovoľné slovo dlhé najviac n znakov. Začneme v koreni, zoberieme prvé písmenko a vydáme sa po jeho hrane. Potom zoberieme druhé a pokračujeme... až do n-tého. Slovo si chceme vyznačiť, preto zakrúžkujeme vrchol, do ktorého sme sa dostali (premyslite si, že nám netreba krúžkovať všetky po ceste, len posledný). Slovo teda môžme chápať aj ako cestu v takomto strome.

Presne na tomto princípe je založená dátová štruktúra, ktorá nám pomôže vyriešiť náš problém. Volá sa *trie*[2] (alebo písmenkový, prefixový, lexikografický strom). Je to strom, v ktorom z každého vrcholu môže ísť až 26 (ak uvažujeme len našu abecedu) hrán, postupne ohodnotených písmenkami a až z. Navyše si každý vrchol pamätá, či v ňom končí nejaké slovo (teda či je zakrúžkovaný).

- Hľadanie: To, že je nejaké slovo $w = a_1 a_2 \dots a_{|w|}$ v trie zistíme tak, že pôjdeme po ceste od koreňa, ktorej hrany budú postupne a_1 až $a_{|w|}$. Slovo sme našli, ak je vrchol zakrúžkovaný. Pre každý znak slova sa posunieme o vrchol ďalej – celková časová zložitosť bude $O(|w|)$.
- Pridávanie: Pridávanie bude fungovať podobne. Začneme v koreni. Zoberieme prvé písmenko pridávaného slova a vyberieme sa po hrane s jeho hodnotou. Potom zoberieme ďalšie a vyberieme sa po jeho hrane. Toto budeme robiť dovtedy, kým takáto hrana existuje. Keď už nebude, postupne zoberieme zvyšné písmenka a vytvoríme im vrcholy. Poslednému povieme, že v ňom končí slovo. Takto vlastne vytvoríme cestu pre vložené slovo. Čas bude $O(|w|)$.
- *Trie*denie: Keď trie prehľadáme do hĺbky (a pôjdeme po hranách od a po z), pričom budeme vypisovať reťazce prislúchajúce končiacim vrcholom, dostaneme jej obsah utriedený. Vypíšeme tak vlastne všetky cesty v trie, od najľavejšej po najpravejšiu. Ak sú v trie slová w_1 až w_n, celková zložitosť triedenia bude $O(|w_1| + \dots + |w_n|)$ (rovnakou aj všetky slová nahádžeme do trie), čo je lepšie ako $O(N \log N)$. Je to dokonca optimálne, pretože rovnakú zložitosť má samotné načítanie slov.
- Pamäť: Stačí, keď si budeme pamätať len vrcholy, ktoré ležia po ceste k zakrúžkovaným (vrátane), pretože do iných sa ani pri pridávaní, ani pri hľadaní a triedení nedostaneme. Pamäť, ktorú trie zaberie, je teda maximálne rovná súčtu dĺžok všetkých slov (keď je nejaké slovo začiatkom iného, je menšia) – $O(|w_1| + \dots + |w_n|)$, čo je rovnako, ako keď si slová pamätáme zvlášť v stringoch.

Trie nám v našom príklade takmer úplne stačí. Jediný problém je v opakujúcich sa slovách. Budeme teda koncové vrcholy nielen krúžkovať, ale si v nich aj budeme pamätať hodnoty príjmov. Na tie potrebujeme takú dátovú štruktúru, ktorá nám prvý vložený príjem aj ako prvý vráti. Presne toto robí fronta, s ktorou ste sa všetci už určite stretli (keď nie v minulých kolách KSP, tak aspoň v školskej jedálni). Dopredu ale nevieme, koľko údajov bude v jednotlivých vrcholoch – nemôžme použiť statické pole. Môžme ale použiť spájaný zoznam. Názov a zdrojový kód snáď povedia dosť. Pridávanie a vyberanie z takejto fronty, ak je správne napísaná, trvá len konštantný čas – zložitosť operácii s trie nám to nepokazí.

Riešenie teda bude vyzerať nasledovne. Slova načítame a povkladáme do trie – časová zložitosť $O(|w_1| + \dots + |w_n|)$. Trie prehľadáme do hĺbky od koreňa. V každom vrchole vždy najprv vypíšeme obsah fronty. Spolu s každým príjmom vypíšeme aj slovo prislúchajúce

[1] niečo podobné ako obzerance s makom, ale bez maku

[2] čítaj [tráj]

vrcholu. To si musíme pamätať v globálnom stringu, inač by sme si pokazili pamäťovú zložitosť. Z vrcholu potom pokračujeme hranami od a po z (tými, čo existujú), pričom vždy pridáme písmenko do globálneho stringu.

Celková časová zložitosť je zložitosť pridania slov do trie a jej vypísania. To je, ako sme si povedali, $O(|w_1| + \ldots + |w_n|)$, čo je rovnako ako zložitosť načítania vstupu. Toto riešenie je teda optimálne.

Ešte pre záujemcov spomeňme, že úlohu vieme riešiť aj radix sortom, ktorého zložitosť sa dá použitím drobného triku upraviť na optimálnu. To už je ale iný príbeh ...

Hodnotenie: Za riešenie behajúce v čase $O(N^2)$ ste mohli dosať najviac 9 bodov. Za použitie triedenia so zložitosťou $O(N \log N)$ ste dostali po 12 bodov, za niečo lepšie aj viac. Za optimálne riešenie bolo plných 15. Body ste mohli stratiť za chýbajúci alebo nedostatočný popis a odhad zložitostí. Potrestaní boli aj opisovači, ktorých svedomie, dúfajme, nemá príliš ostré zuby (aj keď niektorí by si zaslúžili).

Listing programu:

```
type pTrie=^TrieVrchol;
     pFronta=^FrontaVrchol;

    TrieVrchol=record
         z,k:pFronta; {zaciatok a koniec fronty}
         dalsi:array['a'..'z'] of pTrie; {synovia}
    end;

    FrontaVrchol=record
        dalsi:pFronta;
        c:integer; {cislo}
    end;

var trie:pTrie;
    n,i,c:integer;
    s:string;

function novy: pTrie; {vytvori novy vrchol}
var tmp:pTrie;
    c:char;
begin
    new(tmp);
    for c:='a' to 'z' do tmp^.dalsi[c]:=nil;
    tmp^.z:=nil;
    tmp^.k:=nil;
    novy:=tmp;
end;

procedure f_zarad(vrchol:pTrie; c:integer); {zaradi do fronty vrcholu}
var v:pFronta;
begin
    if vrchol^.k=nil then begin {ak je prvy}
        new(v);
        v^.dalsi:=nil;
        v^.c:=c;
        vrchol^.z:=v;
        vrchol^.k:=v;
    end
    else begin {inak zaradi na koniec}
        new(v);
        v^.dalsi:=nil;
        v^.c:=c;
```

```
        vrchol^.k^.dalsi:=v; {posunie koniec fronty}
        vrchol^.k:=v;
    end;
end;

procedure vloz(var trie:pTrie; s:string; c:integer); {vlozi do trie}
var x:pTrie;
    i:integer;

begin

    i:=1;
    if trie=nil then trie:=novy;
    x:=trie;
    {najde najdlhsi prefix v trie}
    while (x^.dalsi[s[i]]<> nil) and (i<=length(s)) do begin
        x:=x^.dalsi[s[i]];
        inc(i);
    end;
    if i<length(s) then    {zvysok vytvori}
        while (i<=length(s)) do begin
            x^.dalsi[s[i]]:=novy;
            x:=x^.dalsi[s[i]];
            inc(i);
        end;
    f_zarad(x,c); {zaradi cislo do fronty vrchola}
end;

procedure vypis(trie:pTrie); {vypise trie}
var f:pFronta;
    c:char;
begin
    f:=trie^.z;
    while (f<>nil) do begin {vypise cisla(aj so stringom) vo fronte}
        writeln(s,' ',f^.c);
        f:=f^.dalsi; {dalsi prosim}
    end;

    for c:='a' to 'z' do {potom vypise vsetky nasledujuce retazce}
        if trie^.dalsi[c]<>nil then  begin
           s:=s+c; {prida pismenko do pomocneho stringu}
            vypis(trie^.dalsi[c]);
            dec(s[0]); {ureze pismenko}
         end;
end;

begin
    trie:=nil;
    readln(n);
    for i:=1 to n do begin
        readln(s);
        readln(c);
        vloz(trie,s,c); {povklada do trie}
    end;
    s:=''; {string prisluchajuci prehladavanemu vrcholu v trie}
    vypis(trie);{vypise trie}
end.
```

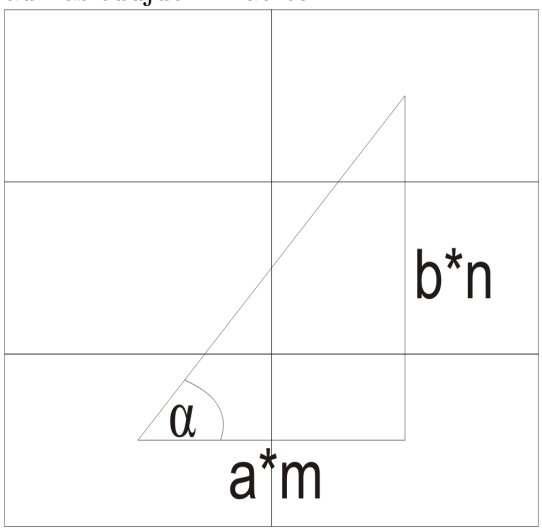
3. Zelený stôl

opravoval MMx
(max. 15 bodov)

Veľká väčšina z vás prišla na správne riešenie tohto príkladu, za ktoré sa dalo získať 15 bodov. Po bode sa dalo stratiť za nedostatočný popis, odhad zložitosti alebo za chybu v programe. Najčastejšou chybou bolo, že ste zabudli ošetriť situáciu, keď $m = 0$, pretože vtedy dochádzalo k deleniu nulou. Býva dobrým zvykom zakaždým, keď píšete do programu delenie zamyslieť sa nad tým, kedy bude menovateľ nulový a ošetriť to.

Všimnime si najprv pohyb gule v horizontálnom smere (teda zmeny vzdialenosti od strany b). Od stredu sa bude hýbať k niektorej zvislej stene (prekoná vzdialenosť $\frac{a}{2}$), odrazí sa a pôjde naspäť do stredu (opäť vzdialenosť $\frac{a}{2}$). Toto bude opakovať m-krát, teda v horizontálnom smere prejde vzdialenosť $m \cdot a$. Podobnou úvahou pre vertikálny smer zistíme, že v tomto smere prejde vzdialenosť $n \cdot b$.

Teraz sa zamyslime nad nasledujúcim náčrtom:



Zakaždým, keď sa má guľa odraziť od okraja stola, plynule prejde na vedľajší stôl. Tu sa pohybuje po dráhe, ktorá je osovo symetrická s jej pôvodnou dráhou. To znamená, že si zachová počet odrazov aj vzdialenosť od stredu v každom okamihu. Z toho vyplýva, že z tohoto obrázku môžeme vypočítať počiatočný uhol, pod ktorým ju treba odpáliť. Použijeme funkciu arctg, ktorá z pomeru protiľahlej odvesny ku priľahlej odvesne vypočíta veľkosť uhla. Špeciálne treba ošetriť prípad, keď $m = 0$, pretože nulou deliť nemôžeme. V takomto prípade je hľadaný uhol 90 stupňov.

Listing programu:

```
program Zeleny_stol;
var a,b,u  :real;
    n,m    :integer;

begin
 writeln('Zadaj rozmery stola:'); readln(a,b);
 writeln('Zadaj pocty odrazeni:'); readln(n,m);
 if m=0 then u:=90 else u:=arctan(b*n/(a*m)*180/pi);
 writeln('Do gule udri pod uhlom ',u:0:3,' stupnov.');
end.
```

opravovali Paľo a Julka
(max. 15 bodov)

4. Zaujímavá permutácia

Tento príklad riešila väčšina z vás. To asi preto, lebo existovalo aj triviálne riešenie. To ale zďaleka nestačilo na plný počet bodov. Veď sami usúďte: Riešenia s časovou zložitosťou $O(N^2)$ dostali maximálne 15 bodov. A za časovú zložitosť $O(N^3)$ bolo možné získať 9 bodov. Pomalšie funkčné riešenia sa našťastie nevyskytli ;-) Za popis sme sa chystali strhnúť maximálne 4 body a chýbajúce odhady časových zložitosti boli penalizované dvoma bodmi.

Riešenie, ktoré funguje určite, je vyskúšať všetky trojice čísel a zistiť, či tvoria aritmetickú postupnosť (čiže či $a_j - a_i = a_k - a_j$). Avšak toto riešenie bude mať časovú zložitosť $O(n^3)$. Ako to teda vylepšiť? Jednoducho. Keď už máme vybratú dvojicu čísel, tak vieme povedať, ako vyzerá tretie číslo $a_k = 2 \cdot a_j - a_i$. Ak sa číslo a_k nachádza za a_j, tak sme našli aritmetickú postupnosť a ak nie, tak môžme zobrať ďalšiu dvojicu.

Ešte si potrebujeme rozmyslieť, ako zistíme v konštantnom čase, kde sa nachádza dané číslo v postupnosti. Na to nám stačí pri čítaní postupnosti si rovno zapisovať do ďalšieho poľa INDEXY, kde sa dané číslo nachádza v zadanej postupnosti. Tu sme využili to, že zadaná postupnosť je permutácia čísel $1 \ldots n$, takže každé číslo z intervalu je tam práve raz.

Pre každú dvojicu čísel teda spravíme výpočet v konštantnom čase, takže výsledná časová zložitosť bude $O(n^2)$.

Listing programu:

```
const
   MAXN=100;
var
   i,j,n: integer;
   inv, perm: array[1..MAXN] of integer;

begin
   read(n);
   for i := 1 to n do begin
      read(perm[i]);
      inv[perm[i]]:=i;
   end;

   for i := 1 to n-1 do
      for j := i+1 to n do
         if (2*perm[j]-perm[i] >= 1) and (2*perm[j]-perm[i]<=n) then
            if (inv[2*perm[j]-perm[i]]>j) then begin
               writeln('Ano');
               exit;
            end;

   writeln('Nie');
end.
```

opravoval Kewo
(max. 15 bodov)

5. Zvítanie so zelenými mužíkmi...

Riešenia sa našli všelijaké, väčšina so správnou časovou zložitosťou – lineárnou od počtu kvietkov, tvoriacich čiaru (ono moc nebolo čo iné vymýšlať). Optimálna pamäťová zložitosť je konštantná, teda $O(1)$. Väčšina pracovala viac či menej s myšlienkou vyberať si body na priamke, zaokrúhliť ich a podla toho zaradiť do hriadky.

Toto sa dá ľahko spočítať pomocou goniometrických funkcií (sínus a jeho kamaráti), prípadne aj bez nich. Možností, ako presne to spraviť, je veľa. Jedna z nich: pozrieme sa, v

smere ktorej osi sa súradnice menia rýchlejšie, keď hýbeme bodom po úsečke. Teraz budeme po úsečke posúvať vykresľovaný bod. Začneme na jednom konci, potom ho vždy posunieme o 1 v smere „rýchlejšej“ osi, o príslušný (menší) kúsok v smere druhej osi a vykreslíme ho (pričom pri vykreslení druhú súradnicu berieme zaokrúhlenú na celé číslo).

V počítačovej grafike je rýchlosť výpočtu stále veľmi dôležitá. Už pred „zopár rokmi“ (asi tak 50-60?) si pán menom Bresenham uvedomil, že počítanie s reálnymi číslami je zbytočne pomalé a navyše aj nepresné. Jeho algoritmus robí presne to isté, ako sme si práve naznačili, len si pri tom vystačí s celými číslami a väčšinu času dokonca používa len sčítanie a porovnávanie.

Pre vysvetlenie algoritmu uvádzam výsek z publikácie **Rasterizácia (by Peter Gejguš and Marek Zimányi)**: [3]

Bresenhamov algoritmus

Je to jeden z najstarších známych algoritmov v oblasti počítačovej grafiky. Začneme predpokladom, že úsečka je v implicitnom tvare (využije sa to iba pre odvodenie a nie je explicitne vypočítaná v algoritme). Máme úsečku z bodu $q = (q_x, q_y)$ do $r = (r_x, r_y)$, priamka, na ktorej leží, je určená rovnicou $f(x, y) = ax + by + c = 0$.

Predpokladajme, že smer našej úsečky je medzi „východným“ a „severovýchodným“. (Teda keby sme ju presunuli tak, že q bude v začiatku sústavy súradníc, tak r bude v prvom kvadrante pod priamkou $y = x$.) Ostatné smery sú len otázkou „správneho preklápania osí“, viac viď zdroják.

Ak zvolíme $d_x = r_x - q_x$, $d_y = r_y - q_y$, ľahko sa ukáže (pomocou substitúcie), že $a = d_y$, $b = -d_x$ a $c_x = -(q_x r_y - r_x q_y)$. Všimnime si, že všetky koeficienty sú celočíselné. Tiež si všimnime, že $f(x, y) > 0$ pre body, ktoré ležia pod čiarou a $f(x, y) < 0$ pre body nad čiarou. Pre dôvody, ktoré budú neskôr jasné, budeme používať ekvivalentnú reprezentáciu vynásobenú 2: $f(x, y) = 2ax + 2by + 2c = 0$.

Teraz nasleduje intuitívne vysvetlenie Bresenhamovho algoritmu. Pre každú celočíselnú hodnotu x chceme určit, ktorá celočíselná hodnota y je najbližšie k našej čiare. Predpokladajme, že sme práve vykreslili pixel (p_x, p_y) a pýtame sa, ktorý pixel ďalej vykreslíme. Pretože sklon je medzi 0 a 1, z toho vyplýva, že ďalší pixel, ktorý bude vykreslený je buď pixel na východ ($E = (p_x + 1, p_y)$) alebo pixel na severovýchod ($NE = (p_x + 1, p_y + 1)$). Nech q označuje presnú y-hodnotu (reálne číslo) čiary v bode $x = p_x + 1$. Nech $m = p_y + 1/2$ označuje hodnotu medzi E a NE. Ak $q < m$, ako ďalší bod vyberieme E, inak vyberieme NE. Ak $q = m$, tak môžeme vybrať napr. E. Pre určenie bodu, ktorý vyberieme, použijeme rozhodovaciu premennú D, ktorá bude predstavovať hodnotu f v strednom bode m. Preto $D = f(p_x + 1, p_y + (1/2)) = 2a(p_x + 1) + 2b(p_y + 1/2) + 2c = 2ap_x + 2bp_y + (2a + b + 2c)$.

Ak $D > 0$, tak m je pod čiarou a preto bod NE je bližšie k čiare. Na druhej strane, ak $D < 0$, tak m je nad čiarou a preto bod E je bližšie k čiare. (Poznámka: Teraz je jasné, prečo sme vynásobili f(x,y) dvoma. D bude mať celočíselnú hodnotu). Dobrá správa je, že D je celočíselná veličina. Zlá správa je, že potrebujeme najmenej 2 násobenia a 2 sčítania pre výpočet D. Jeden zo šikovných trikov za Bresenhamovým algoritmom je počítať si D inkrementálne. Predpokladajme, že poznáme aktuálnu hodnotu D a chceme určiť hodnotu D pre ďalší pixel. Ďalšia hodnota D závisí od týchto dvoch okolností:

Pokračuj ďalej na E: Potom ďalší stredný bod bude mať súradnice $(p_x + 2, p_y + (1/2))$ a preto nová hodnota D bude $D_{new} = f(p_x + 2, p_y + (1/2)) = 2a(p_x + 2) + 2b(p_y + 1/2) + 2c = 2ap_x + 2bp_y + (4a + b + 2c) = 2ap_x + 2bp_y + (2a + b + 2c) + 2a = D + 2a = D + 2d_y$, čo je vlastne aktuálna hodnota D plus $2d_y$.

Pokračuj ďalej na NE: Potom další stredný bod bude mať súradnice $(p_x + 2, p_y + 1 + (1/2))$ a preto nová hodnota D bude $D_{new} = f(p_x + 2, p_y + 1 + (1/2)) = 2a(p_x + 2) +$

[3] `http://fractal.dam.fmph.uniba.sk/~zimanyi/`

$2b(p_y + 3/2) + 2c = 2ap_x + 2bp_y + (4a + 3b + 2c) = 2ap_x + 2bp_y + (2a + b + 2c) + (2a + 2b) = D + 2(a + b) = D + 2(d_y - d_x)$, čo je vlastne aktuálna hodnota D plus $2(d_y - d_x)$. Všimnime si, že v tomto prípade musíme vykonať iba jedno sčítanie (ak predpokladáme, že máme predpočítané hodnoty $2d_y$ a $2(d_y - d_x)$). Preto vnútorný cyklus algoritmu je veľmi efektívny.

Jediná vec ktorá nám ešte ostáva, je vyrátať začiatočnú hodnotu D. Kedže začíname v bode (q_x, q_y), začiatočný stredný bod je v $(q_x + 1, q_y + 1/2)$ a začiatočná hodnota D je $D_{init} = f(q_x + 1, q_y + (1/2)) = 2a(q_x + 1) + 2b(q_y + 1/2) + 2c = (2aq_x + 2bq_y + 2c) + (2a + b) = 0 + 2a + b = 2d_y - d_x$.

A na záver, ako sa hodnotilo: základ plný počet = $15b$. Zlý/chýbajúci časový odhad = $-1b$, pamäťový = $-1b$. Efektivita riešenia = $-0b$ až $-7b$. Chýbajúci popis = $-1b$ až $-7b$, podľa toho, ako veľmi chýbal. Odpisovanie / nápadne podobné riešenia = (výsledný počet bodov)/(počet ludí s takým riešením). Do záporu ste sa dostať nemohli ;). Nefunkčnosť riešenia sa kladne hodnotiť nedá.

Ako chuťovka na záver: Takto vykreslená čiara bude ešte stále na obrazovke „zubatá“. Preto sa používa finta známa ako *antialiasing*. Trik je jednoduchý: Namiesto toho, aby sme vykreslili jeden pixel čiernou, vykreslíme oba pixle, „pomedzi“ ktoré úsečka prechádza. Ale nie čiernou, ale tým tmavším odtieňom sivej, čím bližšie k danému pixlu úsečka ide. Skúste si upraviť program, ktorý ste nám poslali, aby kreslil antialiasované úsečky.

Listing programu:

```
{Lubos Kubon 1.D GJH}
uses crt;
const n=20;
var a:array [0..n] of string;
    i,j:integer;
    s:string;


procedure Line(x1, y1, x2, y2 : integer);
var i, deltax, deltay, numpixels,
    d, dinc1, dinc2,
    x, xinc1, xinc2,
    y, yinc1, yinc2 : integer;
begin
  {vypocitanie dlzku rozdielu na x a y osi}
  deltax := abs(x2 - x1);
  deltay := abs(y2 - y1);

  { vyberanie smeru }
  if deltax >= deltay then
    begin
      { podla osi x }
      numpixels := deltax + 1;
      d := (2 + deltay) - deltax;
      dinc1 := deltay Shl 1;
      dinc2 := (deltay - deltax) shl 1;
      xinc1 := 1;
      xinc2 := 1;
      yinc1 := 0;
      yinc2 := 1;
    end
  else
    begin
      { podla osi y }
      numpixels := deltay + 1;
```

```
      d := (2 * deltax) - deltay;
      dinc1 := deltax Shl 1;
      dinc2 := (deltax - deltay) shl 1;
      xinc1 := 0;
      xinc2 := 1;
      yinc1 := 1;
      yinc2 := 1;
    end;

  if x1 > x2 then
    begin
      xinc1 := - xinc1;
      xinc2 := - xinc2;
    end;
  if y1 > y2 then
    begin
      yinc1 := - yinc1;
      yinc2 := - yinc2;
    end;

  x := x1;
  y := y1;

  { vykreslenie }
  for i := 1 to numpixels do
    begin
      a[y,x]:='*';
      if d < 0 then
        begin
          d := d + dinc1;
          x := x + xinc1;
          y := y + yinc1;
        end
      else
        begin
          d := d + dinc2;
          x := x + xinc2;
          y := y + yinc2;
        end;
    end;
end;

begin
  clrscr;
  {naplnenie}
  s:='';
  for i:=1 to n do s:=s+'.';

  for i:=1 to n do
   a[i]:=s;
  {Vykreslovaci cyklus}

  Line(1,1,5,6);
  {vypis}
  for i:=1 to n do
    writeln(a[i]);
end.
```

Výsledková listina po 1. kole kategórie KSP-Z

	Meno a priezvisko	Škola	Trieda	11	12	13	14	15	Σ
1	Sucha Martin	Gym. Jura Hronca BA	1	15	12	15	14	13	69
2	Králik Martin	Gym. Grösslingová BA	3	15	12	14	14	12	67
3	Danilák Michal	Gym. Hubeného BA	2	15	10	14	15	11	65
4	Danko Juraj	Gym. P. de Coubertina Piešťany	2	15	13	15	9	12	64
5	Kováč Michal	Gym. Grösslingová BA	3	15	12	13	11	11	62
6	Korcsok Peter	Gym. Mládežnícka Šahy	1	15	9	14	8	11	57
7	Mikuláš Ondrej	Gym. Haličská Lučenec	2	14	12	14	15		55
7	Pavlíček Tomáš	SPŠE Piešťany	2	15	9	13	8	10	55
9	Košdy Martin	Gym. Jura Hronca BA	0	15	12	12	6	7	52
10	Okruhlica Adam	Gym. Jura Hronca BA	2	15	15	0	7	11	48
11	Petrucha Michal	Gym. Metodova BA	0	15	12		9	11	47
12	Tříska Martin	Gym. P. de Coubertina Piešťany	2	15	9	1	9	12	46
13	Brada Miroslav	Gym. Varšavská Žilina - Vlčince	1	8	9	7	9	11	44
14	Zajacová Danica	Gym. Jura Hronca BA	2	15	9		8	11	43
15	Uherčík Tomáš	Gym. P. de Coubertina Piešťany	2	7	8	12	5	9	41
16	Takács Michal	Gym. Tajovského B. Bystrica	3	9	9		9	13	40
17	Kucharík Marcel	Gym. Športová Nové Mesto n.V.	3	15	8		7	9	39
18	Mazal Tomáš	Gym. Jura Hronca BA	2	13	8	12			33
19	Švec Marcel	Gym. Jura Hronca BA	2	14	9			9	32
20	Hlavatý Peter	Gym. Jura Hronca BA	1	7	8		7	9	31
20	Orihel Lukas	Gym. Jura Hronca BA	3	9	2	13	5	2	31
20	Panák Pavol	Gym. Jura Hronca BA	3	15	12		4		31
20	Synak Peter	Gym. Jura Hronca BA	1	7	8		7	9	31
24	Kollár Jakub	Gym. Jura Hronca BA	3	7	9	13			29
24	Nagy Kristián	Gym. Jura Hronca BA	0	14	8		7		29
26	Beňo Fratišek	SPŠ Humenné	1	7	8		7	6	28
26	Nikodem Peter	Gym. Školská Spiš. Nová Ves	1		9		7	12	28
28	Betík Roman	SPŠ Levice	3	15				11	26
28	Fedáková Dominika	Neznama skola	1	15	9		2		26
28	Vojtko Martin	Gym. Jura Hronca BA	2	6	8	12			26
31	Dobiaš Michal	Gym. Jura Hronca BA	3	15	9				24
31	Lachata Adrián	Gym. Svidník	3	6	9		9		24
33	Berthoty Adam	Gym. Ľudovíta Štúra Trenčín	3	6	9		6		21
33	Kuboň Luboš	Gym. Jura Hronca BA	0	2	8		7	4	21
35	Gálik Marian	Gym. Jura Hronca BA	1		8	12			20
36	Jarabek Tomas	Gym. Jura Hronca BA	1	6			12		18
36	Vojtko Jakub	Gym. Jura Hronca BA	0				8	10	18
38	Pavlovský Martin	Gym. Jura Hronca BA	2	8	9				17
38	Trnovec Matúš	Gym. Jura Hronca BA	1	7	3		7		17
40	Sabo Michal	Gym. Jura Hronca BA	1		8		7		15
40	Šuranyi Peter	Gym. Jura Hronca BA	3	7	8				15
40	Vlček Tomáš	Gym. Šk. Bratov BA	2	7			8		15
43	Beňo Miroslav	Gym. Jura Hronca BA	3	6	8				14
43	Fojtík Michal	Gymnázium	3		8		6		14
43	Holla Kamila	Gym. Jura Hronca BA	3	5	8		1		14
43	Jarabek Michal	Gym. Jura Hronca BA	3				14		14
43	Mošný Vladimír	Gym. Jura Hronca BA		6	8				14
43	Schuster Vladimír	Gym. Jura Hronca BA	1				14		14
43	Struhár Filip	Gym. Jura Hronca BA	2			14			14
50	Janoskova Michaela	Gym. Jura Hronca BA	1	7			6		13

	Meno a priezvisko	Škola	Trieda	11	12	13	14	15	Σ
50	Martinovic Miroslav	Gym. Jura Hronca BA	1				13		13
50	Morvay Peter	Gym. Jura Hronca BA	2	9	2			2	13
53	Kostolányi Peter	Gym. Jura Hronca BA	0		8		4		12
54	Baumann Martin	Gym. Jura Hronca BA	2	3			7		10
54	Meszaros Milan	Gym. Jura Hronca BA	1		3		7		10
56	Dillinger Viliam	Gym. Jura Hronca BA	3	8					8
56	Hegedušová Monika	Gym. P. Horova Michalovce	3		8				8
56	Mego Marek	Gym. Jura Hronca BA	3	6	2				8
59	Dittrich Andrej	Gym. Jura Hronca BA	1				7		7
59	Doležalková Andrea	Gym. Jura Hronca BA	2	7					7
59	Malícková	Gym. Jura Hronca BA	1				7		7
59	Pilák Otto	Gym. Jura Hronca BA	0				7		7
59	Slobodník	Gym. Jura Hronca BA	1				7		7
59	Travniček Bohuš	Gym. Jura Hronca BA	2	7					7
59	Zalesakova Dada	Gym. Jura Hronca BA	1				7		7
66	Gruska Jonáš	Gym. Jura Hronca BA	0				6		6
66	Truben Martin	Gym. Jura Hronca BA	1				6		6
66	Zvac Peter	Gym. Jura Hronca BA	2	6					6
69	Svoboda Boris	Gym. Jura Hronca BA	1					5	5
70	Koval Michal	Gym. Jura Hronca BA	1		4				4
70	Lesnák Stefan	Gym. Jura Hronca BA	1		4				4
70	Vido Rudolf	Gym. Jura Hronca BA	2	4					4
73	Hamid	Gym. Jura Hronca BA	0				3		3
73	Kvasnička Martin	Gym. Grösslingová BA					3		3
75	Petro Juraj	Gym. Jura Hronca BA	1				1		1