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 2. kola zimnej časti, kat. Z

Tož, už je to tak, je po zimnej časti KSP. Všetko opravené, všetko rozhodnuté, najlepší z vás už môžu netrpezlivo očakávať pozvánku na sústredenie :-) A ostatní sa môžu s chuťou pustiť do riešenia letnej časti, začína sa odznova, nová šanca je pred vami! No a pred tým, ako začnete riešiť nové príklady, prečítajte si tieto vzorové riešenia, nech ste múdrejší.

Iný krajec, iný mravec...

KSPáci

1. Zruční artisti II

opravoval Paľo
(max. 15 bodov)

Prísť na nejaké riešenie tohoto príkladu bolo ľahké. Stačilo totiž odpísať vzorák z minulého kola a už ste mohli dostať kopec bodov. Na vymyslenie optimálneho riešenia však bolo treba trochu porozmýšlať. Vaše riešenia som hodnotil takto: Za riešenie s časovou zložitosťou $O(N)$ ste mohli získať maximálne 15 bodov, s časovou zložitosťou $O(N \log N)$ maximálne 12 bodov a s časovou zložitosťou $O(N^2)$ maximálne 9 bodov. Body ste mohli stratiť za nedostatočný alebo chýbajúci popis, chyby v programe alebo za chýbajúci odhad časovej resp. pamäťovej zložitosti.

Tak a teraz rovno na samotný vzorák. Skoro všetci ste si uvedomili, že stačí uvedené pole utriediť a potom ho vypísať. Teda odpíšeme minulý vzorák a máme riešenie s časovou zložitosťou $O(N \log N)$. O čom by bola táto úloha, ak by to nešlo aj lepšie. Vtip spočíva v tom, že my prvky nemusíme triediť. Artisti nám chcú utvoriť dátovú štruktúru nazývanú halda. Haldu si môžeme predstaviť ako pyramídu, kde dvaja artisti držia tretieho, ktorý je od obidvoch ľahší. Pre zjednodušenie označenia budeme volať artistu, ktorého držia nejakí iní artisti ich otcom a tých dvoch artistov jeho synmi. Haldu si môžeme ľahko reprezentovať v poli. Prvého artistu dáme na prvý prvok poľa, jeho synov na druhý a tretí atď. Teda synovia artistu na i-tom prvku budú na miestach $2i$ a $2i+1$. Keď budeme mať vytvorenú celú haldu, stačí vypísať prvky zľava doprava a vyriešime našu úlohu.

Klasický prístup tvorenia haldy spočíva v tom, že začneme s prázdnou haldou a budeme do nej postupne pridávať ďalšie prvky. Majme teda vytvorenú už nejakú haldu s n prvkami a prvok a, ktorý do nej chceme pridať. Haldu si samozrejme budeme pamätať na prvých n prvkoch poľa vyššie spomenutým spôsobom. Pridajme prvok a na $(n+1)$-vé miesto. Jediný prvok, ktorý nám môže porušiť podmienku haldy je práve prvok a, ktorý môže byť ľahší ako jeho otec. V tomto prípade vymeňme v poli prvok a s jeho otcom. Pokiaľ je a znovu ľahší ako jeho otec, tak s ním môžeme bublať smerom na začiatok poľa až kým nebude a ťažší ako jeho otec alebo až kým nebude úplne na vrchu pyramídy, teda na prvom prvku poľa. V tomto prípade nám už prvky tvoria haldu a môžeme tam pridať ďalší prvok. Aká je časová zložitosť? Halda má nanajvýš N prvkov, teda jej výška je nanajvýš $\log N$. Každý prvok prebuble maximálne cez všetky úrovne, čo nám zaberie čas $O(\log N)$. Všetkých prvkov je N, teda celková časová zložitosť je $O(N \log N)$. Posledných $N/2$ prvkov môže naozaj prebublať cez $\log N$ úrovní, teda náš odhad je tesný.

Vyzerá to tak, že sme si príliš nepomohli. Opak je však pravdou. Stačí totiž spraviť klasickú fintu použiteľnú aj inde. Ak to nejde zľava, robme to sprava. Našu haldu sme budovali smerom od horných úrovní k spodným. Skúsme to opačne. Najprv si načítajme všetky prvky do poľa. Všimnime si, že posledných $N/2$ prvkov nám už tvorí spodnú úroveň a nemusíme s nimi nič robiť. Predstavme si, že už máme vytvorených niekoľko spodných úrovní, ktoré nám tvorí k malých háld (pyramíd). Zoberme si ďalších $k/2$ artistov a každého z nich postavme na dve malé haldy. Teda chceli by sme vytvoriť $k/2$ trošku väčších háld. Môže sa nám stať, podobne ako vyššie, že čerstvo postavený artista stojaci na dvoch haldách je ťažší ako niektorý z jeho synov. V tomto prípade ho vymeníme s ľahším z nich a budeme pokračovať vo vymieňaní až kým nebude ľahší od svojich synov alebo nebude až úplne dole. Takto budeme postupne pridávať vrstvy od najspodnejšej po najvrchnejšiu a na konci dostaneme hotovú haldu. Tento algoritmus vieme ľahko implementovať v poli. Pôjdeme postupne sprava doľava. Artistu na i-tom mieste skontrolujeme, či je menší od svojich synov na miestach $2i$ a $2i+1$ a prípadne ho prebubleme smerom doprava.

Odhad časovej zložitosti: $N/2$ artistov v spodnej vrstve nebubleme vôbec, $N/4$ artistov vo vrstve nad ňou bubleme najviac raz, $N/8$ artistov v ešte vyššej vrstve prebubleme maximálne 2 krát atď. To znamená, že celková časová zložitosť je: $O(0.\frac{N}{2} + 1.\frac{N}{4} + \ldots + (\log N - 1).1)$ Dá sa spočítať, že tá suma v zátvorke neprevýši N, takže toto riešenie je naozaj lineárne.

Pre záujemcov aspoň náznak jednej možnosti. Nech $N = 2^k$, potom môžeme našu sumu zapísať ako $\sum_{i=1}^{k}(i-1)2^{k-i} = \sum_{i=2}^{k}\sum_{j=1}^{i-1}2^{k-i} = \sum_{j=1}^{k-1}\sum_{i=j+1}^{k}2^{k-i} = \sum_{j=1}^{k-1}(2^{k-j}-1) < \sum_{j=1}^{k-1}2^{k-j} < 2^k = N$. Fajnšmekri to dokážu dokázať jedným obrázkom :-)

Listing programu:

```
var a : array [1..10000] of integer;
    n, i, j, m, tmp : integer;

begin
  readln (n);
  for i := 1 to n do read (a[i]);

  for j := n div 2 downto 1 do begin
    i := j;
    while 2*i <= n do begin
      m := 2*i;
      if (m+1 <= n) and (a[m+1] < a[m]) then inc (m);
      if a[i] < a[m] then break;
      tmp :=a[i]; a[i] := a[m]; a[m] := tmp;
      i := m;
    end;
  end;

  write (a[1]);
  for i := 2 to n do write (' ', a[i]);
  writeln;
end.
```

opravoval KuK

2. Zábava na kolieskach... (max. 15 bodov)

Nuž nedošlo mi veľmi veľa optimálnych riešení, takže (1) nedôjde vám veľmi veľa bodov, (2) pozrime si šup-šup vzoráky. Najskôr si stručne zopakujeme onu trochu teórie z minulého kola, ktorá bude potrebná aj tu, potom už príde sľúbené riešenie. Zakončím to nejak optimisticky a uvediem chyby, ktoré ste robili a oboznámim vás, ako som hodnotil.

Teória: Poďme si trochu zaspomínať na minulé kolo. Grafom sme nazývali nejakú dvojicu (V, E), kde V bola množina vrcholov a E množina hrán (kde hrana je dvojica vrcholov, ktoré spája). Vo vzoráku z minulého kola sme si ukázali, ako v grafe nájsť najkratšiu cestu medzi dvoma vrcholmi (tj. takú, že musíme prejsť najmenší počet hrán). Ten postup, známy ako prehľadávanie do šírky (alebo *breadth-first search*, skrátene BFS), sa dá zhrnúť do niekoľkých krokov:

1. [*Inicializácia*] Začíname z nejakého vrcholu r. Ten si ofarbíme a vložíme do fronty.
2. [*Test ukončenia*] Ak je fronta prázdna, končíme.
3. [*Výber vrcholu*] Z fronty vyberieme vrchol v—všetky vrcholy vo fronte sú už ofarbené a poznáme ich vzdialenosť od r.
4. [*Postup do šírky*] Každý sused, ktorý ešte nie je ofarbený má vzdialenosť o jedna väčšiu ako je vzdialenosť vrcholu v. Suseda ofarbíme a vložíme na koniec fronty. Pokračujeme krokom 2.

Všimnite si, že každý vrchol iba raz ofarbíme a každú hranu grafu vyskúšame iba dvakrát, preto časová (aj pamäťová) zložitosť je $O(|V|+|E|)$, kde $|V|$ je počet vrcholov a $|E|$ je počet hrán.

Na Jančiho mapu sa môžeme, podobne ako v predchádzajúcom kole, pozerať ako na graf, kde políčka sú vrcholy a hrany sú iba medzi takými vrcholmi, ktoré sú oba prejazdné (tieto hrany si, samozrejme, netreba pamätať—pozriem na mapu a vidím). Jediné, čo sa v porovnaní s minulým kolom zmenilo je, že cez niektoré hrany sa ide pomalšie ako cez iné. Hranám nášho grafu môžeme priradiť číslo, ktoré udáva čas, ktorý trvá jeho prechod (takýto graf voláme *ohodnotený*). Úlohou teraz nie je nájsť cestu, ktorá sa skladá z najmenšieho počtu hrán, ale takú, že súčet časov pre tieto hrany je minimálny.

Riešenie: Na hľadanie najkratšej cesty v ohodnotenom grafe existujú rôzne algoritmy (napr. Dijkstrov, Bellman-Fordov). Všimnime si však, že máme čo dočinenia so špeciálnym grafom: čas prechodu na susedné políčko je kladné celé číslo menšie ako 12. Klasické vyššie-uvedené algoritmy síce najkratšiu cestu nájdu, ale budú pomalšie ako vzorové riešenie—sú vymyslené pre všeobecné grafy.

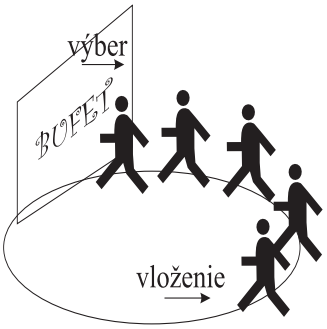
Základná myšlienka riešenia je, že každú hranu, ktorá má pridelený čas t rozdelíme na t jednotkových hrán. Na takto upravený graf už môžeme použiť trochu upravené riešenie z predošlého kola. Keďže $t < 12$, každá hrana sa rozpadne na najviac 11 jednotkových hrán a 10 vrcholov, takže celý graf sa zväčší nanajvýš 11-krát, čo je zhruba to isté. [1] Časová aj pamäťová zložitosť bude lineárna od veľkosti vstupu, tj. $O(MN)$.

Implementácia: Takýto graf si, samozrejme, nemusíme konštruovať. Iba si pre každý vrchol navyše zapamätáme čas, koľko sa na ňom ešte „zdržíme“, kým prejdeme na ďalšie políčko. Vždy keď v treťom kroku algoritmu vyberieme vrchol z fronty, zmenšíme tento čas o jedna. Ak už je nulový, „rozlejeme“ sa na susedné políčka (krok 4), inak vrchol vložíme späť (ale na

[1] v informatike sa až tak nevzrušujeme pre nejaké konštanty—stačí kúpiť 11-krát rýchlejší počítač a je to... dôležitý je *rád* funkcií: všimnite si, že či už výpočet pre vstup veľkosti N trvá N milisekúnd alebo $11N$ milisekúnd, v oboch prípadoch, ak sa N zdvojnásobí, čas sa tiež zdvojnásobí; ak však výpočet trvá N^2 milisekúnd, potom pri dvojnásobnom N-ku je čas štvornásobný!

koniec!) do fronty. Teda, aby som bol presný, časová zložitosť je vo všeobecnosti $O(HNM)$, kde H je maximálne zdržanie na políčku, pretože každé políčko vložíme do fronty najviac H-krát; avšak v našom príklade je $H < 12$, teda konštanta. Použiť Dijkstrov algoritmus by bolo výhodnejšie, keby bolo H väčšie (rádovo väčšie ako $\lg NM$).

Tu by som sa chcel ešte pristaviť pri implementácii fronty. Máte bufet a prichádzajú vám zákazníci. Treba vedieť povedať, kto je prvý v rade a treba vedieť vložiť nového zákazníka do fronty (do radu). Jedna možnosť, ako spraviť výber je, že sa všetci ľudia o jeden krok posunú—to však trvá $O(N)$—oveľa rýchlejšie je posunúť bufet (lebo ľudí je N, ale bufet iba jeden). Aby sa nám nepoposúvali úplne mimo, stačí keď sa budú posúvať v kruhu. V minulom kole som vám poradil, že *keď viete, že každý vrchol grafu navštívite iba raz*, môžete si spraviť frontu veľkosti $|V|$, čo nezhorší pamäťovú zložitosť a v takejto fronte sa nemusíte točiť. Viacerí z vás to použili v tomto kole, ale na takýto špás si spravili príliš malú frontu, preto, ak si nie ste istí, radšej sa točte (stŕhal som za to 2 body).



Hodnotenie: Za optimálne riešenie v čase $O(MN)$ sa dalo získať 15 bodov. Ako sa dalo očakávať, Dijkstrov[2] fanklub sa nám rozrástol—niektorí sa pochlapili[3] a zimplementovali to aj s haldou (t.j. v čase $O(MN \lg(MN))$), za čo ich neminie pochvala (trochu menšia ako za optimálne riešenie) a 13 bodov (trochu menej ako za optimálne riešenie). Bez haldy je to o dosť pomalšie ($O(M^2N^2)$) a tak som to ocenil 8smimi bodmi. Prišlo aj nemálo takých riešení, kde sa tvrdilo, že sa použilo prehľadávanie do šírky, avšak po výbere z fronty sa ihneď rozliali na susedné vrcholy, ak našli kratšiu cestu. Toto riešenie určite *nie je* prehľadávanie do šírky a už vôbec nie je lineárne (ale kvadratické). Tí, čo sa nedali takto jednoducho zaškatuľkovať tiež dostali body... každému podľa jeho zásluh a podľa jeho potrieb. Body som stŕhal za nedostatočný popis, zlý odhad zložitosti, okraje, frontu a tak...

Listing programu:

```
const max_x = 40;
      max_y = 40;
      max_f = max_x*max_y;
   infinity = maxint;

var n, m, h, x, y, nx, ny, startx, starty, cielx, ciely, i, zdr : integer;
    mapa    : array [0..max_y+1, 0..max_x+1] of char;  { prva suradnica je y }
    d, s, t : array [0..max_y+1, 0..max_x+1] of integer;  { vzdialenost od J }
        { spina na (najkr.) ceste z J a cas, kt. sa este zdrzime na policku }
    z       : array [char] of integer;        { ake zdrzanie symbol oznacuje }
    fx, fy  : array [0..max_f] of integer;                        { fronta }
    fz, fk  : integer;                         { zaciatok a koniec fronty }
    c       : char;

procedure vyznac (x, y : integer);     { rekurzivna procedura, ktora vyznaci }
var nx, ny : integer;                                  { najdenu cestu; }
begin
  if (x = startx) and (y = starty) then exit;
  for nx := x-1 to x+1 do
    for ny := y-1 to y+1 do                          { najdeme policko, }
      if (d[ny, nx]+z[mapa[ny,nx]] = d[y, x])          { odkial sme prisli }
       and (s[ny, nx]+z[mapa[ny,nx]]-1 = s[y, x]) then begin
        mapa[y, x] := '';                              { oznacime ho }
```

[2] tak pozor, tento chlapík sa volá Edsger Wybe Dijkstra, nie Dijkster, ako sa vyskytlo v niektorých riešeniach

[3] Ivica si môže slovo 'pochlapili' dačím vhodným nahradiť, mne na um nič neprichádza...

```
        vyznac (nx, ny);                          { a oznacujeme od neho }
        exit;
      end;
end;

begin
  { Nacitame mapu }
  readln (n, m);
  readln (h);
  for i := 1 to h do begin
    readln (c); z[c] := i;
  end;
  z['J'] := 1; z['D'] := 1;
  for y := 1 to n do begin
    for x := 1 to m do begin
      read (mapa[y, x]);
      { vzdialenost policok na zaciatku nepozname, nastavime ju na nekonecno }
      d[y, x] := infinity;              { co znamena, ze sme tam este neboli }
      s[y, x] := infinity;      { kym tam prideme, neuveritelne sa zaspinime }
      t[y, x] := z[mapa[y, x]];
      if mapa[y, x] = 'J' then begin                          { najdeme J }
        starty := y; startx := x;
      end;
      if mapa[y, x] = 'D' then begin                          { najdeme D }
        ciely := y; cielx := x;
      end;
    end;
    readln;
  end;

  { Obalime stenami }
  for y := 0 to n+1 do begin
    mapa[y,   0] := '#'; d[y,   0] := infinity;
    mapa[y, m+1] := '#'; d[y, m+1] := infinity;
  end;
  for x := 1 to m do begin
    mapa[ 0, x] := '#'; d[ 0, x] := infinity;
    mapa[n+1, x] := '#'; d[n+1, x] := infinity;
  end;

  { BFS }
  fx[0] := startx; fy[0] := starty;          { zaciatok zaradime do fronty }
  fz := 0; fk := 1;
  d[starty, startx] := 0;                    { vzdialenost zaciatku je 0 }
  s[starty, startx] := 0;                                { ... a sme cisti }
  repeat
    x := fx[fz]; y := fy[fz];                { vyberieme policko z fronty }
    inc (fz); if fz > max_f then fz := 0;

    dec (t[y, x]);
    if t[y, x] > 0 then begin
      fx[fk] := x; fy[fk] := y; inc (fk);       { opaet zaradime do fronty }
    end else
      { t[y, x] = 0, teda zdrzali sme sa tu }
      { akurat dost, mozeme sa rozliat na susedov }
      for ny := y-1 to y+1 do                { vsetkym susedom, v ktorych }
        for nx := x-1 to x+1 do                          { sme este neboli }
          if (mapa[ny, nx] <> '#') then
            if (d[y, x]+zdr < d[ny, nx]) then begin
              d[ny, nx] := d[y, x] + zdr;          { vypocitame vzdialenost }
```

```
            s[ny, nx] := s[y, x] + zdr - 1;      { a ssspinu.. sssedimenty }
            fx[fk] := nx; fy[fk] := ny;          { ...a zaradime do fronty }
            inc (fk); if fk > max_f then fk := 0;
          end else if (d[y, x]+zdr = d[ny, nx])
          and (s[y, x]+zdr-1 < s[ny, nx]) then
            s[ny, nx] := s[y, x] + zdr - 1;  { iba upravime mnozstvo spiny }
  until (fz = fk);

  if d[ciely, cielx] = infinity then writeln ('Skapes hladom, ty zmrd')
  else begin
    { Vyznac cestu spaet }
    vyznac (cielx, ciely);
    { Vystup }
    for y := 1 to n do begin
      for x := 1 to m do write (mapa[y, x]);
      writeln;
    end;
  end;
end.
```

opravovali Monika a KuK

3. Zakódované dáta

(max. 15 bodov)

Riešenia boli roznorodé. Z plného počtu bodov ste mohli stratiť 3 body za kvadratickú časovú zložitosť, pokial ste hladali koncové zátvorky ku každej počiatočnej. Za ukladanie výstupu do stringu a nevypisovanie priamo na výstup ste mohli stratit najviac 5 bodov, pretože vám mohla vzniknúť až exponenciálna pamäťová zložitosť. Za nedostatočný popis sme strhávali najviac 5 bodov a za kvadratickú pamäťovú zložitosť sme strhli najviac 3 body.

Najčítajme si vstup do stringu. Vo vzorovom riešení budeme používať rekurziu. Samozrejme, že všetko ide napísať aj bez nej (stačí si simulovať vlasný zasobník).

Pokiaľ string, ktorý sme dostali neobsahuje '[' a ']', tak ho vieme rovno rozpísať. Pokiaľ ale obsahuje zátvorky, tak by sme intuitívne postupovali takto: Napíšeme si na papier pôvodný string. Pozrieme sa a nájdeme si k sebe prislúchajúce nevnorené zátvorky (všetko, čo je v ich vnútri, vrátane prípadných ďalších párov zátvoriek, zatiaľ ignorujeme). Keby sme mali dekódovaný obsah každého takéhoto páru zátvoriek, tak by stačilo každý niekoľkokrát zopakovať a vložiť do textu. Takže teraz treba zistiť, čo vlastne je v týchto zátvorkách.

Vezmeme si nový papier a napíšeme si naňho text, ktorý bol „uzavretý“ v jednom páre '[' a ']'. Starý papier si odložíme nabok a pokračujeme na novom papieri. Všimnime si, že opäť riešime pôvodnú úlohu, len s kratším textom. No a opäť použijeme ten istý postup: Pokiaľ už obsah nového papieru neobsahuje vnorené '[' a ']', tak ho vieme rozpísať. Pokiaľ obsahuje, tak znova urobíme rovnakú úvahu, vytvoríme nový papier a starý položíme na ten predchádzajúci a tak ďalej... Pokiaľ sme rozpísali všetko, tak máme pred sebou papier s rozpísaným reťazcom. Vyberieme si vrchný papier z našej kopy a dosadíme namiesto '[' a ']' vypočítaný string toľko krát čo treba. Znova budeme mať vypočítaný string a tak ďalej budeme spätne dosadzovať.

Tak, a máme funkčné riešenie. Tento postup ale nie je optimálny, pretože časová zložitosť je kvadratická. Keď máme ľavú zátvorku, na zistenie, kde je zodpovedajúca pravá zátvorka treba prezrieť celý string. Najhorší prípad ale vyzerá tak, že zátvorky sú vnorené v sebe (napr. '[[[ ]]]').

Skúsime naše riešenie zlepšiť do lineárneho času. Riešenie bude vyzerať rovnako, iba vynecháme napísanie celého stringu vždy, keď si zoberieme nový papier. Stačí nám vedieť, že:
– ak sme narazili na '[', zoberieme nový papier a pokračujeme na ňom

– ak sme narazili na ']', s aktuálnym papierom sme skončili, string, ktorý sme dostali, doplníme príslušný počet krát na miesto na predchádzajúcom papieri

Časová zložitosť je lineárna od veľkosti výstupu. Lepšie ju nevieme urobiť, pretože každý znak musíme niekedy vypísať. Pamäťová zložitosť je lineárna od veľkosti vstupu, pretože si vstup musíme nejak pamätať a na základe neho pracovať.

Listing programu:

```
var s : string;
    i : integer;

procedure opakuj (od, poc : integer);
var j, pom : integer;
begin
  for j := 1 to poc do begin
    i := od;
    while s[i] <> ']' do
      if ('0' <= s[i]) and (s[i] <= '9') then begin
        pom := 0;
        while ('0' <= s[i]) and (s[i] <= '9') do begin
          pom := 10*pom + ord(s[i])-ord('0');
          inc (i);
        end;
        opakuj (i+1, pom);
      end else begin
        write (s[i]);
        inc (i);
      end;
  end;
  inc (i);
end;

begin
  readln (s);
  s := s + ']';
  opakuj (1, 1);
  writeln;
end.
```

4. Zimný spánok

opravoval Tono
(max. 15 bodov)

Vaše riešenia tohto príkladu boli rozmanité, ale väčšinou nie optimálne, aj keď si to veľa z vás myslelo. Ale pekne poporiadku. Správne ste pochopili, že sa bude riešiť úloha na grafe. Jeho vrcholy reprezentujú miestnosti, hrany sú chodby medzi nimi. Zo zadania navyše vyplývalo, že graf bude orientovaný – jeho chodby vedú len jedným smerom, a acyklický – nedá sa chodiť dokola medzi nejakými dvoma vrcholmi (to by mohol Boris nazbierať nekonečne veľa čerešní).

Navyše si môžeme povedať, že Boris konzumuje čerešne rovno z chodby vedúcej do miestnosti. Dostávame teda orientovaný acyklický ohodnotený graf. V ňom hľadáme najdlhšiu cestu (alebo najčerešňovejšiu, len sa to jednoduchšie skloňuje) začínajúcu vo vchode. Ako ste mnohí skúsili, šlo by to napríklad postupným prehľadávaním, či už do hĺbky alebo do šírky, pri ktorom by sme si pre každý vrchol pamätali vzdialenosť od vchodu. Problém je, že pri takomto prehľadávaní niektoré vrcholy navštívime viackrát. Ak sa nám podarí predĺžiť

cestu do nejakého vrcholu, musíme predĺžiť aj cesty do všetkých vrcholov za ním. To nám pokazí zložitosť, ktorá už nebude lineárna od počtu vrcholov a hrán.

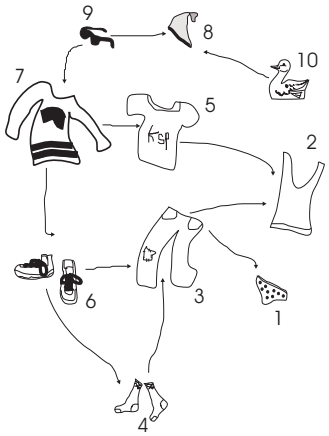
Pozrime sa na takú najdlhšiu cestu. Tá nech ide postupne cez vrcholy v_1 (vchod) až v_n. Je zrejmé, že pre každý vrchol v_i ($i \leq n$) platí, že neexistuje cesta do nejakého vrcholu dlhšia ako cesta cez v_{i+1} až v_{n-1} do v_n. Ak by totiž existovala, bola by táto cesta napojená na cestu v_1 až v_i dlhšia ako najdlhšia cesta, čo je spor.

Podobne platí, že ak sme v ľubovoľnom vrchole a, ktorý má susedov a_1 až a_s, tak najdlhšia cesta z a musí prechádzať cez ten z vrcholov a_1 až a_s, z ktorého najdlhšia cesta je maximálna. Ak máme teda vyrátané dĺžky najdlhších ciest z a_1 až a_s, vieme jednoznačne určiť kadiaľ bude pokračovať najdlhšia cesta z neho. Ak a nemá žiadneho suseda, tak v ňom jeho najdlhšia cesta začína aj končí. Toto nám už poskytuje návod ako pozisťovať najdlhšie cesty postupne od koncových miestností až ku vchodu. Po krátkom zamyslení sa ale zistíme, že našťastie všetko ide spraviť pekne odpredu, prehľadaním grafu do hĺbky. To upravíme tak, že nám pre vrchol vráti dĺžku najdlhšej cesty z neho. Ako už vieme, tú nájde tak, že vyberie suseda s maximálnou touto dĺžkou (ktorú vyráta takisto) a priráta k nej dĺžku hrany doňho. Keďže chceme najdlhšiu cestu aj vypísať, zapamätáme tohto suseda ako nasledovníka v najdlhšej ceste. Navyše vidno, že výsledky prehľadávania sú pre vrchol vždy rovnaké. Preto, pred spustením prehľadávania z nejakého vrcholu, skontrolujeme, či už výsledok náhodou nemáme vyrátaný. Prehľadáme graf od vchodu a nakoniec ešte postupne prejdeme po nasledovníkoch a vypíšeme tak vrcholy nájdenej najdlhšej cesty.

Prehľadávanie sa spustí z každého vrcholu práve raz, každú hranu prejde práve raz a teda bude mať zložitosť $O(N + M)$, kde M je počet hrán. Vypísanie najdlhšej cesty bude pri najhoršom trvať $O(N)$, preto je celková časová zložitosť $O(N + M)$. Pre každý vrchol si musíme pamätať jeho najdlhšiu cestu. Musíme si pamätať aj celý graf – hrany a stupne vrcholov. Celková pamäť je teda $O(N + M)$.

Na tomto mieste by asi bolo vhodné spomenúť aj spôsob, ktorým naozaj pojdeme pekne od konca. K tomu nám poslúži topologické triedenie. Topologické usporiadanie grafu je taká postupnosť jeho vrcholov, v ktorej sa každý vrchol nachádza až po všetkých, do ktorých sa dá z neho dostať. Dá sa jednoducho ukázať, že (aspoň jedno) takéto usporiadanie existuje práve vtedy, keď v grafe nie je cyklus.

Ak už graf máme takto zotriedený, stačí použiť techniku dynamického programovania a princíp z predošlého riešenia. Totiž, ako sme si povedali, vrcholu vieme určiť dĺžku maximálnej cesty z neho podľa maximálnych ciest z jeho susedov. Keďže ale postupne prechádzame topologicky utriedené vrcholy, môžme si byť istý, že tieto hodnoty máme pre susedov už vyrátané. Ostáva už len povedať, ako sa graf takto triedi. Stačí si len uvedomiť, že vrchol do poradia môžme zaradiť až potom, ako zaradíme všetky vrcholy, do ktorých sa vieme z neho dostať **hranou**. To ale znamená, že len drobne pozmeníme prehľadávanie do hĺbky – keď spracujeme všetkých potomkov daného vrcholu, zaradíme ho do postupnosti. (Pozor! Ak **nevieme**, či je graf acyklický, treba na konci postupnosť, ktorú týmto postupom dostaneme, ešte overiť, či naozaj funguje!)



Hodnotenie Riešenia behajúce v čase lineárnom od veľkosti vstupu dostali plných 15 bodov. Iné funkčné riešenia dostali 8 až 12 bodov, v závislosti od časovej zložitosti. Nie úplne funkčné riešenia mohli tiež dostať nejaký ten bodík. Body ste mohli stratiť za nedostatočný popis, zlý odhad zložitostí a samozrejme za odpisovanie.

Listing programu:

```
const maxn = 100;
type miest = record
       cer, deg : integer;  { pocet ceresni, stupen vrchola }
       zac, kon : integer;  { zaciatok a koniec v zozname hran }
       max, next : integer; { najdlhsia cesta, nasledovnik }
     end;
var m : array [1..maxn] of miest;
    h : array [1..maxn*maxn] of integer; { zoznam hran }
    n, nh, c, p, i, j : integer;

function maxcer (v : integer) : integer;
var maxi, i : integer;
begin
  if (m[v].max = -1) then begin { ak este nepozna vysledok}
    m[v].max := m[v].cer;
    if (m[v].zac <= m[v].kon) then begin { vyberie suseda s max. najdlhsou cestou }
      maxi := h[m[v].zac];
      for i := m[v].zac to m[v].kon do
        if (maxcer(h[i])>m[maxi].max) then maxi := h[i];
      m[v].next :=maxi;
      inc (m[v].max, m[maxi].max);
    end;
  end;
  maxcer := m[v].max;
end;

procedure vypiscestu;
var i : integer;
begin
  i := 1;
  while (i > 0) do begin write (i, ' '); i := m[i].next; end;
  writeln;
end;

begin
  readln (n); nh := 1;
  for i := 1 to n do begin
    read (c, p);
    m[i].cer := c; m[i].deg := p;
    m[i].zac := nh; m[i].kon := nh + p - 1;
    m[i].max := -1; m[i].next := 0;
    for j := 1 to p do begin read (h[nh]); inc (nh); end;
  end;
  writeln ('Nazbiera ceresni: ', maxcer(1));
  vypiscestu;
end.
```

5. Z blochou

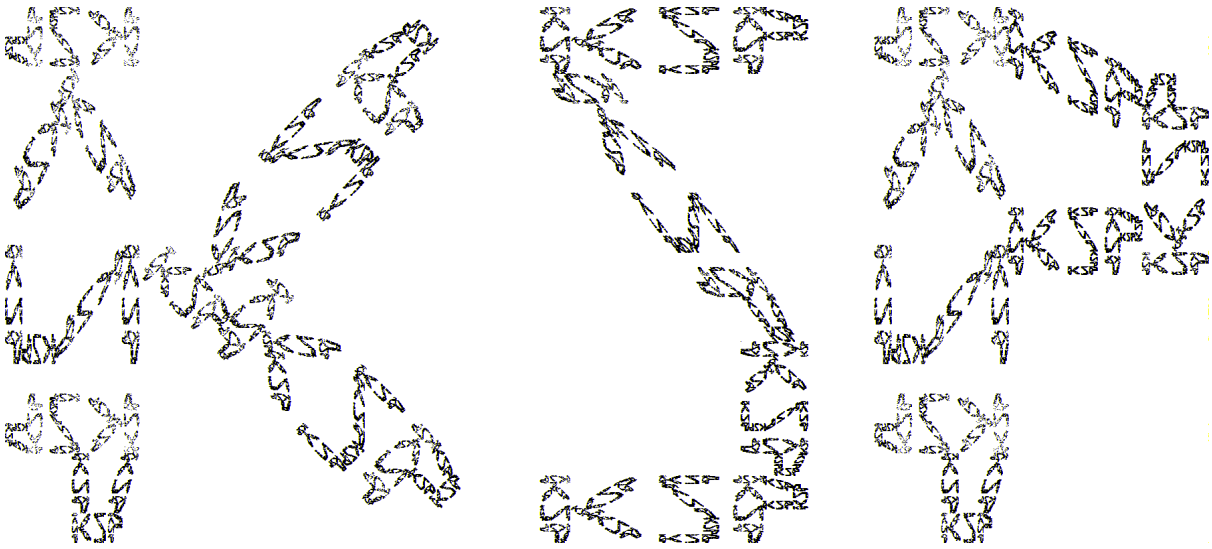
opravovala Majka
(max. 15 bodov)

Tento príklad ste takmer všetci riešili správne a posielali pekné obrázky fraktálov. Za správne riešenie s obrázkom a popisom svojho fraktálu ste mohli získat 15 bodov. 10 bodov ste dostali ak ste popísali svoj fraktál, ale neposlali ste obrázok. Ten kto neposlal obrázok ani popis svojho fraktálu dostal najviac 6 bodov a za iné fraktály bez skákania blochy (spôsobu aký bol popísaný v zadaní) ste mohli získat najviac 3 body. Bod sa strhával aj vtedy, keď nebol

k programu žiaden popis. Samotná implementácia bola ľahučká, ťažšie asi bolo hranie sa s konštantami, aby to vyrobilo pekný fraktál.

Časová zložitosť takéhoto programu bude lineárna od počtu vygenerovaných bodov, pamätová zložitosť je úmerná počtu transformácií a veľkosti bitmapy, do ktorej kreslíme.

A najkrajší fraktál? Ten sme si vyrobili sami. Ľaľa, tu je:



Listing programu:

```
uses graph;

const MaxTransfor = 10;
      PocSkok = 10000;

var
  grDriver,grMode : integer;
  a: array [1..6,1..MaxTransfor] of integer;
  n,i,j : integer;
  x,y,pom: real;

begin
  Randomize;
  grDriver:= Detect;
  InitGraph(grDriver,grMode,'');
  assign(input,'Blocha.in'); reset(input);
  readln(n);
  x:=0;y:=0;
  putpixel(round(x),round(y),white);
  for i:=1 to n do
   readln(a[1,i],a[2,i],a[3,i],a[4,i],a[5,i],a[6,i]);
  for i:=1 to 10000 do
  begin
    j:=random(n)+1;
    pom:=x;
    x:=a[1,j]*x+a[2,j]*y+a[3,j];
    y:=a[4,j]*pom+a[5,j]*y+a[6,j];
    putpixel(round(x),round(y),white);
  end;
  close(input);
end.
```

Výsledková listina po 2. kole kategórie KSP-Z

	Meno a priezvisko	Škola	Trieda		21	22	23	24	25	Σ
1	Bílka Ondřej	Gymnázium	3	57	11	14	15	15	15	127
1	Rampášek Ladislav	Gym. Jura Hronca BA	2	67	12	8	15	10	15	127
3	Danko Juraj	Gym. P. de Coubertina Piešťany	2	63	12	8	14	9	15	121
4	Jerguš Ján	Gym. Alejová Košice	2	69	12	12	15	10		118
5	Paulovicová Ivica	Gym. Jura Hronca BA	4	65	12	12	12	15		116
6	Králik Martin	Gym. Grösslingová BA	3	61	11	0	15	8	15	110
6	Okruhlica Adam	Gym. Jura Hronca BA	2	60	12	5	10	8	15	110
8	Sudolský Michal	Gym. Tajovského B. Bystrica	2	52	12	4	15	10	15	108
9	Buštor Ivan	Gym. Jura Hronca BA	2	55	11	12	5	8	15	106
10	Petruchová Zuzana	Gym. Grösslingová BA	3	50	15	10		10	15	100
11	Herman Peter	Gym. Jura Hronca BA	2	53	13	13	9	8		96
12	Pančík Andrej	Gym. Tajovského B. Bystrica	2	54	11	3	10	8		86
13	Besenyi Libor	Gymnázium	4	38	12	3	7	7	15	82
14	Petrucha Michal	Gym. Metodova BA	0	52	14				15	81
15	Tříska Martin	Gym. P. de Coubertina Piešťany	2	41	11	4	15	6	3	80
16	Kekely Lukáš	Gym. Varšavská Žilina - Vlčince	1	44	12	6	7	7	2	78
17	Smolka Tobiáš	Gym. L. Stöckela Bardejov	4	36	11	4	11	9	3	74
18	Takáč Slavomír	Gym. Nové Zámky	3	72						72
19	Halamíček Radovan	Gym. Jura Hronca BA	3	50	12		8			70
19	Mikuláš Ondrej	Gym. Haličská Lučenec	2	38	11		7	8	6	70
21	Kováčovský Tomáš	Gym. Jura Hronca BA	1	40	11	7	10			68
22	Kováč Michal	Gym. Grösslingová BA	3	30	11		6	5	15	67
23	Korcsok Peter	Gym. Mládežnícka Šahy	1	42	12	2		7		63
24	Szabó Tibor	Gym. Šuleka Komárno	2	25	10	7	7	7	3	59
25	Lachata Adrián	Gym. Svidník	3	33	12		10			55
26	Kajan Peter	Gym. Jura Hronca BA	2	25	9		10	9		53
27	Dadová Janka	Gym. Jura Hronca BA	4	50						50
28	Bálint Farkaš	Gym. maďarské Šahy	4	33	8		1			42
29	Miklian Peter	Gym. Haličská Lučenec	8	41						41
29	Sábo Jozef	Gym. Školská Spiš. Nová Ves	2	41						41
31	Beňo Fratišek	SPŠ Humenné	1	22	8		1	0	9	40
31	Pagáč Matej	Gym. Školská Spiš. Nová Ves	4	40						40
33	Baumann Martin	Gym. Jura Hronca BA	2	23			6		10	39
33	Dzurňák Tomáš	Gym. Školská Spiš. Nová Ves	3	39						39
35	Rajčan Šimon	Gym. Ľ. Štúra Zvolen	4	38						38
36	Betík Roman	SPŠ Levice	3	22	11	4				37
36	Žubrietovský Tomáš	Gym. Ľ. Štúra Zvolen	4	37						37
38	Novák Ján	Gym. Ľudovíta Štúra Trenčín	3	24	12					36
39	Dillinger Viliam	Gym. Jura Hronca BA	3	31	4					35
40	Blaho Pavol	Gym. Jura Hronca BA	1	34						34
40	Hegedušová Monika	Gym. P. Horova Michalovce	3	23	11					34
42	Cimba Martin	Gym. Stropkov	4	33						33
43	Brada Miroslav	Gym. Varšavská Žilina - Vlčince	1	32						32
44	Vlček Tomáš	Gym. Šk. Bratov BA	2	21	11					32
45	Oľhava Rastislav	Gym. Alejová Košice	3	30						30
46	Kysel Ondrej	Gym. Žiar nad Hronom	2	27						27
46	Matušov Izidor	Gym. A.Sládkoviča B. Bystrica	1	12	5		7	3		27
48	Šrámek Martin	Gym. Tilgnerova BA	1	10	14					24
49	Morvay Peter	Gym. Jura Hronca BA	2	23						23
49	Takács Michal	Gym. Tajovského B. Bystrica	3	23						23

	Meno a priezvisko	Škola	Trieda		21	22	23	24	25	Σ
49	Zajacová Danica	Gym. Jura Hronca BA	1	0	11		12			23
52	Nikodem Peter	Gym. Školská Spiš. Nová Ves	1	21						21
52	Pavlovský Martin	Gym. Jura Hronca BA	2	0	13		8			21
52	Renčo Peter	Gym. A.Sládkoviča B. Bystrica	1	13	5			3		21
52	Vido Rudolf	Gym. Jura Hronca BA	2	21						21
56	Kolesár Štefan	Gym. Alejová Košice	3	2	12		6			20
57	Nosko Slavomil	SPŠ Myjava	1	19						19
58	Kán Peter	Gym. Einsteinova BA	4	18						18
58	Schuster Vladimír	Gym. Jura Hronca BA	1	8	10					18
60	Hromuľák Matúš	Gym. Školská Spiš. Nová Ves	2	16						16
60	Švec Marcel	Gym. Jura Hronca BA	2	16						16
62	Hrebíček Martin	Gym. Ľudovíta Štúra Trenčín	2	15						15
62	Krupel Matej	Gym. Ľudovíta Štúra Trenčín	1	15						15
64	Hornak Michal	Gym. Alejová Košice	3	13						13
64	Jančár Michal	Gym. Alejová Košice	1	13						13
66	Bodnár Jozef	Gym. Fiľakovo	4	12						12
66	Krištof Peter	Gym. Haličská Lučenec	4	12						12
68	Kopil Tomáš	Gym. Humenné	4	0	9		2	0		11
69	Hanuska Norbert	Gym. Ľ. Štúra Zvolen	4	10						10
69	Mižáková Katarína	Gym. Alejová Košice	1	10						10
69	Schwarz Erik	Gym. Alejová Košice	1	10						10
72	Kurpel Matej	Gym. Ľudovíta Štúra Trenčín	3	0	9					9
73	Jakubek Maroš	Gym. Ľudovíta Štúra Trenčín	2	8						8
73	Račko Tibor	Gym. Mládežnícka Šahy	4	8						8
75	Kovac Zolo	Gym. Alejová Košice	1	7						7
75	Matečný Alexander	Gym. Ľudovíta Štúra Trenčín	2	7						7
75	Petro Jakub	Gym. Alejová Košice	1	7						7
78	Melo Damián	Gym. Ul. 1. mája Trenčín	2	6						6
78	Melo Jakub	Gym. Ľudovíta Štúra Trenčín	1	6						6
78	Novacek Milos	Gym. Jura Hronca BA	3	0	6					6
78	Orihel Lukas	Gym. Jura Hronca BA	3	0	6					6
78	Sukuba Ivan	Gym. Alejová Košice	1	6						6
78	Szönyi Iwan	Gym. Ľudovíta Štúra Trenčín	2	6						6
84	Andreansky Marek	Gym. Alejová Košice	3	5						5
84	Kovac Michal	Gym. Alejová Košice	3	5						5
84	Kubejova Lucia	Gym. Alejová Košice	3	5						5
84	Riganová Ivana	Gym. Alejová Košice	1	5						5
88	Holla Kamila	Gym. Jura Hronca BA	3	0	4					4
88	Mazal Tomáš	Gym. Jura Hronca BA	2	4						4
88	Travniček Bohuš	Gym. Jura Hronca BA	2	4						4
91	Beran Jakub	Gym. Alejová Košice	3	1						1