Korešpondenčný seminár z programovania
XX. ročník, 2002/03
Katedra vyučovania informatiky FMFI UK,
Mlynská Dolina, 842 48 Bratislava

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

Vzorové riešenia 2. kola letnej časti, kat. Z

Milí riešitelia!
My KSPáci sme už starí a leniví, navyše sme rozlezeni po celom Slovensku (a okolí) a prázdninujeme. Preto to tak dlho trvalo, kým sa tieto vzorové riešenia dostali až do vašich rúk. Čo dodať na záver jubilejného 20. ročníka? S najlepšími sa stretneme na jesennom sústredení na Dobrej Vode, s najstaršími na matfyze a s ostatnými pri riešení ďalšieho ročníka KSP.

Kto nepracuje, nech sa aspoň naje.

KSPáci

1. Zas. . . chodník

opravoval Dávidko
(max. 15 bodov)

V zadaní sa ticho predpokladalo, že lajná sú usporiadané zľava doprava. To budeme predpokladať aj my.

Označme ľavý koniec (ten s menšou x-ovou súradnicou) i-teho lajna ℓ_i a pravý koniec r_i pre $1 \leq i \leq N$. Označme zatiaľ neznámu dĺžku kroku d. Ak máme prvým krokom prekročiť prvé lajno a zároveň nestúpiť do druhého, tak d musí spĺňať nerovnosť $r_1 \leq d \leq \ell_2$. Uvážme, čo musí platiť, ak chceme spraviť dva kroky. Pochopiteľne musí platiť to, čo doteraz: $r_1 \leq d \leq \ell_2$. Navyše si treba uvedomiť, že po dvoch krokoch sa budeme nachádzať na súradnici $2d$ a to musí byť medzi druhým a tretím lajnom. Takže naviac musí platiť aj $r_2 \leq 2d \leq \ell_3$.

Teraz to skúsime zovšeobecniť. Ak máme po k-tom kroku byť medzi k-tym a $(k+1)$-vým lajnom, tak $r_k \leq kd \leq \ell_{k+1}$. Po predelení k dostaneme $r_k/k \leq d \leq \ell_{k+1}/k$. Ak chceme teda spraviť k-krokov, tak musí platiť týchto k nerovností

$$r_1 \leq d \leq \ell_2,\ \frac{r_2}{2} \leq d \leq \frac{\ell_3}{2},\ \ldots,\ \frac{r_k}{k} \leq d \leq \frac{\ell_{k+1}}{k}.$$

Špeciálne, ak $k = n$ (teda keď chceme prekročiť všetky lajná), tak posledná nerovnosť nemá pravú stranu a vyzerá len ako $r_n/n \leq d$.

Každá z nerovností nám určuje nejaký interval, v ktorom musí ležať dĺžka kroku d. Prvá nerovnosť určuje interval $\langle r_1, \ell_2\rangle$, druhá $\langle r_2/2, \ell_3/2\rangle$, ..., k-ta $\langle r_k/k, \ell_{k+1}/k\rangle$, ... $(n-1)$-vá $\langle r_{n-1}/(n-1), \ell_n/(n-1)\rangle$ až n-tá $\langle r_n/n, \infty)$. Ak máme prejsť k krokov, tak d musí ležať v prieniku prvých k z týchto intervalov.

Zrejme prvý interval je neprázdny, takže jeden krok môžme spraviť. Ale už prienik prvého a druhého môže byť prázdna množina. V tomto prípade sa dva kroky nedajú spraviť. Všeobecne, ak k je také, že prienik prvých k intervalov je neprázdny a už prienik $k+1$ intervalov je prázdny, tak možno spraviť práve k krokov. Dĺžka kroku bude potom ľubovoľné číslo z prieniku prvých k intervalov.

Ako sa to naprogramuje? V prvom rade si treba uvedomiť, že prienik intervalov je zase len interval. Jednoducho budeme zaradom načítavať lajná a rátať prienik prvých k intervalov. Stačí si len pamätať ľavý d_ℓ a pravý koncový bod d_r tohto prieniku. Ak máme vypočítaný prienik $\langle d_\ell, d_r\rangle$ po $k-1$ krokoch, tak prienik po k krokoch bude interval

$$\langle d_\ell, d_r\rangle := \langle d_\ell, d_r\rangle \cap \langle r_k/k, \ell_{k+1}/k\rangle = \langle \max(r_k/k, d_\ell), \min(\ell_{k+1}/k, d_r)\rangle.$$

Ak je ľavý koncový bod väčší ako pravý, tak prienik je prázdna množina. Ak niekedy dostaneme prázdny interval, tak prienik $k-1$ intervalov nám určí dĺžku kroku. V šťastnejšom prípade sa dostaneme až na koniec a dostane neprázdny interval po n krokoch. Ako dĺžku kroku vypíšeme napr. stred posledného neprázdneho intervalu.

Lajná nie je nutné načítať do poľa (tým ušetríme pamäť), stačí si vždy pamätať to, ktoré práve spracúvame. Pamäťová zložitosť bude preto konštantná. Časová zložitosť je zjavne lineárna do počtu lajen t.j. $O(N)$, pretože pre každé lajno potrebujeme spraviť jeden prienik intervalov, čo trvá konštantný čas.

Listing programu:

```
Program Zas_chodnik;

var N,k:integer;
    l,r:real;      { lavy a pravy koniec (nejakeho) lajna }
    dl,dr:real;    { minimalna a maximalna dlzka kroku }
    dl_old, dr_old:real;

{ minimum dvoch cisel }
function min(a,b:real):real;
begin if a<b then min:=a else min:=b; end;

{ maximum dvoch cisel }
function max(a,b:real):real;
begin if a>b then max:=a else max:=b; end;

begin
     assign(input,'zprikl1.in');
     reset(input);
     read(N);        { nacitame N }

     if N=1 then begin
          readln(l,r);
          writeln('krok=',r); { vypiseme vysledok }
          halt;               { skoncime }
     end;

     { ak N>1 }
     read(l);        { lavy koniec praveho lajna nas netrapi }
     read(dl,dr);  { pravy koniec 1. lajna a lavy 2. }

     for k:=2 to N-1 do begin
          read(r,l); { nacitame r[k] a l[k+1] }

          { zapamatame si stare hodnoty po (k-1) krokoch }
          dl_old:=dl;
          dr_old:=dr;

          { urobime prienik <dl,dr> a <r[k]/k,l[k+1]/k> }
          dl:=max(r/k,dl);
          dr:=min(l/k,dr);

          { ak sa k-ty krok uz nedal spravit, tak ... }
          if dl>dr then begin
             writeln('krok=',(dl_old+dr_old)/2); { vypiseme vysledok }
             halt;                               { skoncime }
          end;
     end;
```

```
     { prienik <dl,dr> a <r[n]/n, nekonecno) }
     read(r);
     dl_old:=dl;
     dr_old:=dr;
     dl:=max(r/n,dl);

     { ak sa n-ty krok uz nedal spravit }
     if dl>dr then
        writeln('krok=',(dl_old+dr_old)/2) { vypiseme vysledok }
     else
        writeln('krok=',(dl+dr)/2);        { prienik vsetkych intervalov }
end.
```

2. Závažný problém

opravoval FIXME
(max. 18 bodov)

Tento príklad nebol až taký náročný, pokiaľ budeme ignorovať poznámku v zadaní, že by bolo vhodné napísať nejaké rýchlejšie riešenie. Vtedy stačí raz prejsť celý log, postupne extrahovať dáta, podľa podmienok v zadaní počítať každému tímu body a trestné minúty, na konci potom utriediť tímy podľa počtu bodov, resp. trestných minút a vypísať výsledok. Takto postupovali viacerí z vás, riešenia sa líšili v použitom triedení. V súťaži IPSC je na každú úlohu a každú obtiažnosť povolených maximálne 10 submitov. (Táto skutočnosť v zadaní bohužiaľ nebola uvedená.) Keď obmedzíme počet submitov, dostávame horné ohraničenie počtu trestných minút (presnejšie $8.10.10 + 8.10.20 + 16.2\,000 = 34\,400$).[1]

Pri určovaní časovej náročnosti riešenia preto uvažujme obidva prípady. Nech N je počet tímov, S je počet submitov. Pri neobmedzenom počte submitov má optimálne riešenie časovú zložitosť $O(N \log N + S)$ a dá sa dosiahnuť dobrou implementáciou vyššie uvedeného postupu – potrebujeme použiť triedenie v čase $O(N \log N)$ a vedieť vyhľadať tím podľa jeho mena v konštantnom čase. Neskôr si ukážeme, ako na to.

Niektorí z vás aktualizovali poradie tímov v podstate po každom spracovanom submite. Toto riešenie má potom zložitosť $O(S \log N)$ alebo $O(SN)$ podľa implementácie, pri veľkých vstupoch je teda horšie.

Ukážeme si teraz riešenie, ktoré pobeží dokonca v čase $O(N + S)$ za predpokladu, že je počet submitov obmedzený. Od všeobecného riešenia sa bude líšiť použitým triedením.

Zatiaľ mlčky predpokladajme, že vieme vyhľadať tím v zozname tímov v konštantnom čase. Teraz nám stačí načítať postupne všetky submity, pre každý tím si udržiavať informácie o riešených príkladoch a priebežne počítať body a trestné minúty. Po načítaní celého logu pristúpime k triedeniu tímov. Teraz využijeme skutočnosť, že počet submitov je obmedzený. Ako sme si ukázali vyššie, obmedzí sa tým aj počet trestných minút. Tak môžeme použiť triedenie zvané CountSort. Toto triedenie sa dá použiť len vtedy, ak je počet rôznych hodnôt, ktoré môže nadobúdať kľúč, ohraničený a rozumne veľký. Pre každú z týchto hodnôt si vypočítame, koľko kľúčov nadobúda túto hodnotu (to vieme v lineárnom čase).

Označme P_i počet kľúčov, ktoré nadobúdajú hodnotu i. Následne pre každú hodnotu vypočítame, koľko kľúčov nadobúda rovnakú alebo menšiu hodnotu. Tento údaj si označíme S_i. Ľahko vidíme, že $S_i = \sum_{j<i} P_j$ (a to je zároveň aj návod, ako vypočítať hodnoty S_i). Teraz postupne prejdeme všetky prvky odzadu a budeme ich zatrieďovať do ďalšieho poľa. Spracúvaný prvok uložíme na miesto S_i a hodnotu S_i znížime o 1. A to je všetko. Uvedomte si, že keď zatrieďujeme nejaký prvok, hodnota $S_i - 1$ nám určuje, koľko prvkov pred ním má kľúč s rovnakou alebo menšou hodnotou. A nakoľko prvky zo zoznamu spracúvame odzadu, určite by bol spracúvaný prvok po utriedení na pozícii S_i.

[1] Počet submitov by sme obmedzili aj vtedy, ak by sme predpokladali, že v jednej minúte môže jeden tím odovzdať každú úlohu najviac raz.

Teraz už len ostáva otázkou, ako vyhľadávať tímy v konštantnom čase. Názvy si budeme pamätať v špeciálnom strome. Každý vrchol bude mať toľko synov, koľko písmen je v abecede. Hranu od otca k synovi označíme príslušným písmenom. Potom každá konečná cesta od koreňa reprezentuje nejaké slovo. A ako si do tohoto stromu ukladať tímy? Vo vrchole, kde končí cesta z koreňa reprezentujúca názov daného tímu, si uložíme pointer na dáta onoho tímu. V opačnom prípade tam dáme nil. Tento strom zaberie pamäť lineárne závislú od počtu tímov. Vyhľadať tím znamená nájsť vrchol, kde končí cesta z koreňa, zodpovedajúca jeho menu. (Ak v našom strome taký vrchol nie je, máme nový tím, pridáme do stromu cestu z koreňa zodpovedajúcu jeho menu.) Keďže mená tímov majú zhora ohraničenú dĺžku, keď vieme meno tímu, dostaneme sa k jeho dátam v konštantnom čase.

K bodovaniu:

- Za použité triedenie v čase $O(N \log N)$ – 15 bodov
- Za použité triedenie v čase $O(N^2)$ – 11 bodov
- Za aktualizovanie poradia po každom spracovanom submite – 9 bodov
- Za chýbajúce utriedenie – maximálne 6 bodov
- Za zlé riešenia – bodík až dva

Ďalej som strhával body za chyby pri počítaní bodov a trestných minút, vypisovaní poradia, chýbajúci alebo zlý odhad zložitosti, či nedostatočný popis.

Listing programu:

```
program IPSC; {program nebol odladeny, asi su v nom chyby ;) }

const   maxt = 100;
        max  = 34400;

type    TTim = record
                body, minuty: integer;
                priklady: array['A'..'H',1..2] of integer;
               end;

type    PVrchol = ^TVrchol;
        TVrchol = record
                a: array['a'..'z'] of PVrchol;
                t: integer;
                end;

var     K: PVrchol;
        T: array[1..maxt] of TTim;
        ID: array[1..maxt] of integer;
        cas, i, j: integer;
        n: integer;
        zn: char;
        meno, hlaska: string;

procedure NovyVrchol(var p: PVrchol);
  var zn: char;
begin
  New(p);
  for zn:='a' to 'z' do begin p^.[zn]:=nil; p^.t:=0; end;
end;

procedure NewTeam(zvysok: string; p: PVrchol);
  var j: integer;
begin
  j:=1;
```

```
  while j <= Length(zvysok) do begin
    NovyVrchol(p^.a[zvysok[j]]); p:=p^.a[zvysok[j]]; Inc(j);
  end;
  Inc(n);
  p^.t:=n; T[n].body:=0; T[n].minuty:=0;
  FillChar(T[n].priklady, sizeof(T[n].priklady),0);
end;

function FindTeam(meno: string): integer;
  var i: integer;
      zn: char;
      p: PVrchol;
      res: integer;
begin
  res:=0; p:=K; i:=1;
  while true do begin
    if i>Length(meno) then begin
      if p^.t<>0 then FindTeam:=p^.t
                 else begin
                      inc(n); T[n].body:=0; T[n].minuty:=0;
                      FillChar(T[n].priklady, sizeof(T[n].priklady),0);
                      FindTeam:=n;
                      exit;
                      end;
      end;
    if p^.a[meno[i]] = nil then begin
      NewTeam(Copy(meno,i,Length(meno)-i+1),p);
      FindTeam:=n+1;
      exit;
      end;
    p:=p^.a[meno[i]];
    end;
end;

procedure Aktualizuj(cas: integer;meno: string;uloha:char;cislo:integer;hlaska:string);
  var i: integer;
begin
  i:=FindTeam(meno);
  if hlaska = ' OK' then begin
    if T[i].priklady[uloha,cislo] <> -1 then begin
      Inc(T[i].body,cislo);
      Inc(T[i].minuty,cas+10*cislo*T[i].priklady[uloha,cislo]);
      T[i].priklady[uloha,cislo]:=-1;
      end;
    end;
end;


procedure CountSort;
  var i,j: integer;
      T1: array[1..maxt] of TTim;
      A: array[1..max] of integer;
      B: array[1..24] of integer;
begin
  for i:=1 to max do A[i]:=0;
  for i:=1 to n do Inc(A[T[i].minuty]);
  for i:=2 to max do A[i]:=A[i]+A[i-1];
  for i:=n downto 1 do begin T1[A[i]]:=T[i]; dec(A[i]); end;
  for i:=1 to 24 do B[i]:=0;
  for i:=1 to n do Inc(B[T[i].body]);
```

```
  for i:=2 to 24 do B[i]:=B[i]+B[i-1];
  for i:=n downto 1 do begin T[B[i]]:=T1[i]; dec(B[i]); end;
end;

procedure Done(p: PVrchol);
  var zn: char;
begin
  for zn:='a' to 'z' do if P^.a[zn] <> nil then Done(p^.a[zn]);
  Dispose(p);
end;

begin
  n:=0;
  NovyVrchol(K);
  assign(input,'ipsc.log');
  reset(input);
  read(cas); read(zn); read(zn); meno:='';
  while(zn <> ' ') do begin meno:=meno+zn; read(zn); end;
  read(zn); read(i);
  readln(hlaska);
  Aktualizuj(cas,meno,zn,i,hlaska);
  close(input);
  CountSort;
  for i:=1 to n do begin
    if (i>1) then
      if (T[i].body = T[i-1].body) and (T[i].minuty = T[i-1].minuty)
         then write(' ':5) else write(i:5);
    write(' ',T[i].meno,' ',B[i].body,' ',B[i].minuty,' ');
    for zn:='A' to 'H' do
      for i:=1 to 2 do
        if T[i].priklady = -1 then write(zn,i,' ');
    writeln;
    end;
  Done;
end.
```

3. Zase neporiadok

opravoval Tom
(max. 15 bodov)

Všetky riešenia, ktoré som videl, vyzerali byť funkčné. Ale zďaleka nie všetky boli optimálne. Najčastejšie sa vyskytovala prostá simulácia toho, čo robí Riško. Takéto riešenia mali časovú zložitosť $O(t)$, alebo $O(tN)$ a mohli získať (za myšlienku) 5 bodov. Riešenia s časovou zložitosťou $O(N)$ mohli získať 10 alebo 9 bodov podľa toho, či ich pamäťová zložitosť bola konštantná, alebo $O(N)$.

Ako ste si mnohí všimli, úloha, ktorú mal Riško vyriešiť, bol vlastne problém hanojských veží.[2] V zadaní bol popísaný spôsob, ako ho vyriešiť na najmenší možný počet ťahov. (Až na jeden detail, a to, že podľa postupu v zadaní sa pre párne N veža presunie na druhú tyč.) Na problém hanojských veží sa však dá pozerať aj nasledovne:

Ak máme presunúť N diskov z tyče a na tyč b s pomocnou tyčou c tak pre $N = 1$ ju jednoducho prenesieme, pre $N > 1$ najprv prenesieme vrchných $N - 1$ diskov z a na c (pomocou b), potom presunieme najväčší disk z a na b a nakoniec presunieme tých $N - 1$ diskov z c na b (pomocou a). Týmto spôsobom na presun N diskov potrebujeme $2^N - 1$ ťahov. Dokážeme, že pre párne N je presun veže z tyče 1 na tyč 3 týmto spôsobom taký istý ako

[2] Pre tých, čo ho nepoznajú: Máme 3 tyče, na prvej z nich je N deravých diskov, usporiadaných odspodu od najväčšieho po najmenší. Úlohou je presunúť ich na tretiu (druhá je pomocná), pričom sa smie presúvať vždy len jeden disk naraz a nesmie sa položiť väčší disk na menší.

presun Riškovým spôsobom a pre nepárne N je presun veže z 1 na 2 týmto spôsobom taký istý ako presun Riškovým spôsobom. Oboje dokážeme indukciou. Vezmime si párny ťah t. Ukážeme, že nepohne najmenším diskom. Pre $N = 1, 2$ tvrdenie zjavne platí. Nech tvrdenie platí pre $N - 1 > 0$. Ak $t < 2^{N-1}$, tak je to jeden z ťahov, ktorými sa presúva menšia veža a tvrdenie podľa indukčného predpokladu platí. Ak $t = 2^{N-1}$, tak sa presúva najväčší disk, ktorý nie je najmenší, lebo $N > 1$. Inak sa presúva menšia veža druhýkrát a tvrdenie analogicky platí. Podobne sa dá dokázať druhá podmienka zo zadania (tá o nepárnych ťahoch a najmenšom disku).

A na čo nám bude tento nový popis toho istého postupu? No predsa na ňom už vidno, že tie presuny majú štruktúru, ktorá sa dá využiť na rýchlejšie riešenie. Nech $f(a, b, c, N, t)$ je funkcia, ktorá nám povie, ako to bude vyzerať po vykonaní t ťahov, ak presúvame N diskov z a na b pomocou c. Zrejme ak $t < 2^{N-1}$ tak sa ani nedotkneme najväčšieho disku, teda N-tý disk bude po dokončení t ťahov na tyči a a polohy ostatných nám povie $f(a, c, b, N - 1, t)$. Ak $t \geq 2^{N-1}$, tak je najväčší disk už presunutý a začalo sa druhé presúvanie menšej veže – N-tý disk bude na tyči b a polohy ostatných nám povie $f(c, b, a, N - 1, t - 2^{N-1})$. Toto sa dá jednoducho implementovať pomocou jedného cyklu. Priama implementácia vedie k algoritmu s časovou zložitosťou $O(N)$ a pamäťovou zložitosťou $O(N)$. Dá sa to ešte zlepšiť tak, že výsledky si nebudeme pamätať, ale budeme ich rovno vypisovať – pre každú vežu pustíme náš algoritmus a vypíšeme čísla tých diskov, ktoré budú po t krokoch na nej.

Listing programu:

```
var N,tah,i:integer;

procedure xchg(var a,b:integer);
var t:integer;
begin t:=a; a:=b; b:=t; end;

procedure Veza(k:integer);
var z,na,pom,i,d,poz,t:integer;
begin
  t:=tah; z:=1;
  if (odd(N)) then begin na:=2; pom:=3; end
     else begin na:=3; pom:=2; end;
  d:=1; for i:=1 to N do d:=d*2; d:=d-1; {d:=2^N-1;}
  for i:=N downto 1 do begin
    d:=(d-1) div 2;
    if t<=d then begin poz:=z; xchg(na,pom); end
       else begin poz:=na; xchg(z,pom); t:=t-d-1; end;
    if poz=k then write(' ',i);
  end;
end;

begin
  readln(N,tah);
  for i:=1 to 3 do begin
    write(i,'.veza:'); Veza(i); writeLn;
  end;
end.
```

Opravovala Monika
(max. 15 bodov)

4. Zo sna hovoriaci čarodej

Na to, aby program potreboval čo najmenej pamäte, nebudeme si ukladať celé slovo, ktoré chceme otestovať – naraz si budeme pamätať len jedno písmeno. Ľahko overíme, či to nie je znak f ani c.

Na to, aby malo slovo dĺžku deliteľnú tromi, stačí si počítať počet písmen a na záver sa táto podmienka poľahky overí. Alebo ešte lepšie, nebudeme si pamätať počet už prečítaných písmen (môže byť veľký), ale iba zvyšok, aký dáva po delení tromi.

Aby dve samohlásky nenasledovali po sebe, stačí si pamätať, či bol posledný načítaný znak samohláska. Ak je načítané písmeno samohláska, tak sa pozrieme, či bolo aj predchádzajúce samohláska. Ak áno, tak testované slovo určite nie je zaklínadlo. Zapíšeme si, či bolo práve načítané písmeno samohláska a pokračujeme ďalej.

Posledná podmienka je najzložitejšia. Chceli by sme overiť, či sa nachádza vo vstupnom slove kúzelná formulka „abraka“. Overovať to budeme nasledovne:

Budeme si pamätať číslo od 0 do 6, udávajúce, aký najdlhší začiatok slova „abraka“ sa nachádza na konci reťazca, ktorý sme doteraz prečítali. Toto číslo budeme volať *stav*. T.j. ak sme doteraz prečítali „dgarawabraabra“, budeme v stave 4 – na konci máme „abra“. Slovo „abraka“ sa v nádejnom zaklínadle vyskytovalo práve vtedy, ak sme niekedy počas jeho čítania boli v stave 6.

Otázkou ostáva, ako meniť stav po prečítaní ďalšieho písmena. Zjavne ak sme napr. v stave 4 a prečítame k, zmeníme stav na 5. Keď hocikedy prečítame iné písmeno ako a,b,r,k, stav sa zmení na 0. Ale treba si uvedomiť, že napr. ak v stave 4 prečítame písmeno b, stav sa zmení na 2! Prechody medzi stavmi teda budú vyzerať nasledovne:

Stav 0: Tento stav sa zmení na stav 1 až vtedy, keď načítame znak a, teda začiatok formulky „abraka“

Stav 1: Teraz vieme, že posledný znak bol „a“. Ak načítame znak „b“, tak zmeníme stav na 2. Ak načítame znak „a“, zostávame v stave 1. Inak ideme do stavu 0.

Stav 2: Vieme, že posledné dva znaky boli „ab“, teda ak načítame „r“ presunieme sa do stavu 3. Ak však načítame „a“, ideme späť do stavu 1 (reťazec končí „aba“, so začiatkom reťazca „abraka“ sa zhoduje len jeden znak – „a“). V inom prípade zmeníme stav na 0 – nič sa nezhoduje.

Stav 3: Ak načítame „a“, ideme do stavu 4, inak 0.

Stav 4: Ak načítame „k“, ideme do stavu 5. Ak načítame „b“, ideme do stavu 2 (načítaný reťazec končí na „abrab“). Ak načítame „a“, ideme do stavu 1. Inak ideme do stavu 0.

Stav 5: Ak je práve načítané písmeno „a“, prejdeme do stavu 6, inak do stavu 0.

Stav 6: Ľahšie ako si pamätať, že sme tu už boli je jednoducho zostať v stave 6, akonáhle sa doň raz dostaneme. Takže nech čítame čokoľvek, ostávame v stave 6.

2cm (Kvôli prehľadnosti chýbajú na obrázku šípky vedúce do 0 aj ostávajúce v 6.)

Časová zložitosť je lineárna od dĺžky vstupného slova, teda $O(N)$, pretože každé písmenko načítame a spracujeme v konštantnom čase. Pamäťová zložitosť je konštantná, teda $O(1)$.

Hodnotenie:

- za každú zabudnutú možnosť pri overovaní, či je slovo zaklínadlo mínus 2 body (maximálne mínus 8 bodov)
- program s pamäťovou zložitosťou lineárnou od dĺžky vstupu mohol získať maximálne 4 body
- pokial nebol priložený popis, alebo bol program nedostatočne popísaný, vášmu riešenie stratilo 1 až 3 body podľa závažnosti tohto zločinu

Chuťovky na záver.

Podobne ako sme vyhľadávali slovo „abraka“ sa dá vyhľadávať aj ľubovoľný iný podreťazec. A skutočne, existujú vyhľadávacie algoritmy, ktoré si najskôr podľa vyhľadávaného slova zostroja niečo podobné nášmu obrázku a potom len raz prebehnú reťazec, v ktorom

majú hľadať. Za domácu úlohu si rozmyslite, ako pre daný podreťazec (napr. „babbabb“) čo najrýchlejšie zostrojiť náš obrázok.

No a pri riešení našej úlohy sme skončili pri tom, že nám stači konštantná pomocná pamäť. Ide to ale ešte o niečo lepšie: Nepotrebujeme premenné ani na pamätanie si stavu, zvyšku počtu písmen po delení tromi a samohlásky, vystačíme si s **jedinou premennou typu char**, do ktorej budeme čítať vstup! Ako na to? Všetko ostatné si budeme pamätať pozíciou v programe: Stav, zvyšok a informáciu o samohláske vieme zakódovať do čísla od 0 do 41. No a náš program by mal 42 labelov (čísla od 0 do 41), po prečítaní písmena by sme vždy pomocou goto skočili na miesto, ktoré zodpovedá novému stavu, zvyšku, atď. Ale teraz už vzorový program, má pár premenných navyše, ale je prehľadnejší.

Listing programu:

```
uses Crt;
var stav,poc:byte;
    zaklinadlo,samohlaska:boolean;
    c:char;

begin
  stav:=0; poc:=0; zaklinadlo:=true; samohlaska:=false;
  c:=ReadKey; write(c); c:=Upcase(c);
  repeat
    Inc(poc); if (poc=3) then poc:=0; {dlzka}

    case stav of     {test na abraka}
       0 : case c of 'A': stav:=1;  else stav:=0;  end;
       1 : case c of 'A': stav:=1;  'B': stav:=2;  else stav:=0;  end;
       2 : case c of 'A': stav:=1;  'R': stav:=3;  else stav:=0;  end;
       3 : case c of 'A': stav:=4;  else stav:=0;  end;
       4 : case c of 'A': stav:=1;  'B': stav:=2;
                     'K': stav:=5;  else stav:=0;  end;
       5 : case c of 'A': stav:=6;  else stav:=0;  end;
    end;

    if c in ['A','E','I','O','U','Y'] then {test na samohlasku}
      if samohlaska then zaklinadlo:=false else samohlaska:=true
    else samohlaska:=false;

    if (c = 'C') or (c='F') then zaklinadlo:=false; {test na c,f}
    c:=ReadKey; write(c); c:=Upcase(c);
  until c=#13;
  if (zaklinadlo) and (stav=6) and (poc=0)
    then writeln(' JE ZAKLINADLO.') else writeln(' NIE JE.');
end.
```

5. ZJEDZ OBED HARRY

opravoval Poko, pokec MišoF.
(max. 15 bodov)

Weasley cannot save a thing,
He cannot block a single ring,
That's why Slytherins all sing:
Weasley is our King.

Weasley was born in a bin
He always lets the Quaffle in
Weasley will make sure we win
Weasley is our King.

Tak, keď sme sa už správne naladili,[3] môžeme sa pustiť do riešenia jeho úlohy. Tá sa veru prílišnej popularite u vás netešila, len niekoľkí z vás sa odhodlali prísť mu na pomoc. A niet divu, úloha to bola vskutku náročná. Nakoniec ostalo všetko na pleciach starnúceho Albusa Dumbledora.[4] A vy sa teraz započúvajte do rozprávania o tom, ako si s Harryho úlohou poradil on.

Rovnosť, v ktorej sú rôzne číslice nahradené rôznymi písmenami sa zvykne nazývať *algebrogram*. Zamyslime sa najskôr nad tým, ako konkrétny algebrogram vyriešiť. Ako prvé si uvedomme, že ak máme viac ako 10 rôznych písmen, riešenie neexistuje – máme totiž len 10 cifier. (No, jedine že by sme dovolili riešenia aj v iných sústavách... ale nekomplikujme si zbytočne úlohu.)

Máme teda najviac 10 rôznych písmen, chceme im priradiť číslice. Rôznych možných priradení číslic písmenám je najviac $10! = 3\,628\,800$ a vo všeobecnosti neexistuje lepšie riešenie ako ich všetky vyskúšať. Pre každé možné priradenie dosadíme za písmená číslice a overíme, či rovnosť platí.

Samozrejme v konkrétnych prípadoch si môžeme odpustiť skúšanie niektorých možností, o ktorých vieme, že nepovedú k riešeniu. Riešme napríklad algebrogram OKO+OKO=KUK. Keď sa napr. rozhodneme že K je 2, pre O už nemá zmysel skúšať iné možnosti ako 1 a 6.

Ako to naprogramovať? Majme teda jednoduchý algebrogram (nejaké to sčítanie, možno trochu odčítania a násobenia). Za písmená budeme postupne rekurzívne dosádzať číslice, a to tak, aby sme najskôr doplnili každému číslu číslicu na mieste jednotiek, potom desiatok, atď. Akonáhle vidíme, že už rovnosť určite nebude platiť, vrátime sa o krok späť a vyskúšame naposledy vybranému písmenu priradiť inú číslicu. Takémuto postupu sa hovorí *backtracking*. (Zriedkavo môžete počuť slovenské preklady ako „prehľadávanie s návratom“ a pod.)

A ako nájsť nejaké pekné algebrogramy, keď už máme program, ktorý ich vie riešiť? Nuž, napíšeme si druhý program. Ten nakŕmime nejakým pekným tematickým slovníkom (HARRY, POTTER, MAGIA, VECERA, KSP, atď.) a necháme ho vygenerovať kopu náhodných algebrogramov z týchto slov. Tými potom nakŕmime riešiaci program a vyberieme tie, ktoré majú (pokiaľ možno jednoznačné) riešenie.

Ešte slovo na záver: Uvedomte si, že jeden program, ktorý by mal robiť obe činnosti – generovať aj riešiť algebrogramy – by dosť pravdepodobne vyzeral omnoho humusnejšie ako dva samostatné programy. Takisto by sa v ňom napríklad ťažšie hľadali chyby a pod. O tomto hovorí princíp KISS[5]: každý program by mal mať len jednu funkciu, ale zato ju vykonávať dokonale.

Tí, čo ste sa už stretli s operačným systémom Linux, pravdepodobne viete, o čom hovorím. Na všetko, čo potrebujete, existuje malý šikovný nástroj, ktorý práve to robí. Príklad zo života: máme HTML stránku s výsledkami KSP, potrebujeme zoznam GJHákov, ktorí nás riešia. Preženieme ju filtrom, ktorý nechá len riadky kde škola je GJH (grep), zoradíme ich podľa abecedy (sort) a následne z každého riadku umažeme všetko okrem mena (sed). Toto všetko vieme spraviť z príkazového riadku bez toho, aby sme dotyčný súbor museli otvoriť v editore a čokoľvek s ním ručne robiť. Toť sila malých šikovných nástrojov.

[3] úryvkom z piateho dielu o Harrym Potterovi

[4] Všetkých 5 kníh som čítal po anglicky, nehnevajte sa, že neviem, ako sa volá v preklade.

[5] Keep It Small and Simple

Výsledková listina po 2. kole kategórie KSP-Z

	Meno a priezvisko	Škola	Ročník		21	22	23	24	25	Σ
1	Cicko Miroslav	Gym. Tajovského B. Bystrica	2	73	15	14	15	15	12	144
2	Plžík Milan	Gym. Ľ. Štúra Zvolen	2	65	15	16	10	13	5	124
3	Sedlák Milan	Gym. Liptovský Hrádok	3	63	15	15	12	14		119
4	Poláčik Michal	GLN Tomášikova BA	3	59	15	14	15	7		110
5	Cermak Michal	Gym. Jura Hronca BA	3	53	15	10	10	15		103
6	Ilavský Milan	SPŠE Liptovský Hrádok	3	53	15	9	10	3		90
6	Morihladko Peter	Gym. Školská Spiš. Nová Ves	2	53	12	9	10	4	2	90
8	Gottweis Martin	Gym. Jura Hronca BA	1	49	12		10	7	7	85
9	Krchniak Matej	Gym. Jura Hronca BA	1	44	15	5	12	3		79
10	Lackovič Matúš	Gym. Mudroňova Prešov	2	49	10	8	10			77
11	Kuštárová Tamara	Bilingválne gym. Sučany	2	33	15		15	13		76
12	Mikuláš Ján	Gym. Haličská Lučenec	1	43	9	9	10	4		75
12	Trnovcová Zuzana	Gym. Jura Hronca BA	2	43	15		14	3		75
14	Bundala Daniel	Gym. Jura Hronca BA	1	30	15	14	10	4		73
15	Schlosáriková Eva	Gym. SNP Piešťany	2	48	3	6	10	4		71
16	Imriška Jakub	Gym. Jura Hronca BA	1	8	15	11	15	13		62
16	Krčah Marcel	Gym. Párovská Nitra	???	38	10		10	4		62
16	Struhár Pavel	Gym. Jura Hronca BA	2	46	3		10	3		62
19	Zaťko Tomáš	Gym. 17. novembra Topoľčany	3	29	9	4	10	3		55
20	Ďuďák Juraj	Gym. Golianova Nitra	1	31			10	9		50
21	Goga Peter	Gym. 17. novembra Topoľčany	3	16	9	9	10	3	2	49
21	Legény Jozef	Gym. M. R. Stefanika, Košice	2	49						49
23	Kolibiarova Hanka	Gym. Jura Hronca BA	1	39				9		48
23	Labuda Tomáš	Gym. Grösslingová BA	3	48						48
25	Andruška Igor	SPŠ Levice	2	14	12		10	2		38
26	Chamila Sergius	Gym. Dilongova Prešov	-3	37						37
26	Dravecky Pavol	Gym. Jura Hronca BA	1	37						37
28	Pančík Andrej	Gym. Š. Moyzesa B. Bystrica	0	23	9	1		3		36
29	Halamíček Radovan	Gym. Jura Hronca BA	1	10	15		5	4		34
29	Pekár Jaroslav	SPŠE Stará Turá	2	21	9			4		34
31	Bielik Peter	Gym. SNP Piešťany	2	20			10	3		33
32	Krištofík Štefan	SPŠ Levice	4	10	9		10	3		32
33	Ambróz Peter	Gym. Jura Hronca BA	3	0	12		9	10		31
34	Vitásek Matej	Gym. Grösslingová BA	2	30						30
34	Ľudvík Martin	Gym. Školská Spiš. Nová Ves	2	30						30
36	Panák Pavol	Gym. Jura Hronca BA	1	15	12			1		28
37	Hajnala Jozef	Gym. Golianova Nitra	???	0	12		10	4		26
38	Mačura Martin	SPŠE Michalovce	2	25						25
38	Trubenová Barbora	Gym. Jura Hronca BA	3	25						25
38	Vanderka Peter	Gym. Trstená	???	25						25
41	Kottman Michal	Gym. Jura Hronca BA	2	24						24
42	Kačur Tomáš	SPŠE Michalovce	2	22						22
43	Štefanec Richard	Gym. Jura Hronca BA	3	21						21
44	Hlinková Jana	Gym. Grösslingová BA	3	7	9			4		20
44	Šimko Jakub	Gym. Jura Hronca BA	2	20						20
46	Hruda Ivan	ZS Saratovská	0	18						18
47	Puskajler Ján	SPŠE Michalovce	???	17						17
48	Gallusová Barbora	Gym. Grösslingová BA	2	13				3		16
48	Kuna Matej	Gym. Golianova Nitra	2	16						16
48	Nguyen Tomáš	Gym. Jura Hronca BA	1	3	12			1		16

	Meno a priezvisko	Škola	Ročník		21	22	23	24	25	Σ
48	Varga Ľubomír	SPŠ Nitra	3	16						16
52	Antonič Michal	Gym. Grösslingová BA	1	15						15
52	Drábik Juraj	Gym. Jura Hronca BA	1	15						15
52	Fiala Martin	Gym. Grösslingová BA	3	15						15
55	Demuthová Eva	Gym. Jura Hronca BA	1	1	12			1		14
56	Kalivoda Lukáš	Gym. Jura Hronca BA	1	1	12					13
56	Oremus Vladimír	Gym. Jura Hronca BA	1	1	12					13
58	Lebedová Barbora	Gym. Jura Hronca BA	1	0	12					12
59	Borsuk Andrej	Gym. Grösslingová BA	1	11						11
59	Hlavatá Ivana	Gym. Jura Hronca BA	2	11						11
59	Wertlen Andrej	Gym. Jura Hronca BA	2	0	9			2		11
59	Špalek Jozef	Gym. Čadca	???	11						11
63	Táborský Roman	Gym. Jura Hronca BA	1	10						10
64	Dillinger Vialiam	Gym. Jura Hronca BA	1	9						9
65	Parák Jakub	Gym. Jura Hronca BA	1	8						8
65	Polák Ján	Gym. Jura Hronca BA	1	8						8
65	Trembulák Michal	SPŠE Michalovce	2	8						8
65	Valiska Ján	SPŠE Michalovce	2	8						8
69	Budáč Ondrej	Gym. Haličská Lučenec	1	7						7
69	Svonava Daniel	Gym. Jura Hronca BA	1	0			5	2		7
71	Willmann Marek	Gym. Jura Hronca BA	2	0				6		6
72	Aleksandrov Vladimír	Gym. Jura Hronca BA	2	0			5			5
73	Holla Kamila	Gym. Jura Hronca BA	1	4						4
74	Satinský Ján	Gym. Jura Hronca BA	1	0				3		3
74	Soha Miroslav	Gym. Liptovský Hrádok	1	3						3
76	Foltýnová Monika	Gym. Jura Hronca BA	1	1				1		2
76	Sabo Matej	Gym. Jura Hronca BA	1	2						2
76	Sopinka Ondrej	SPŠE Michalovce	1	2						2
79	Bíziková Ivona	Gym. Jura Hronca BA	1	1						1
79	Galla Martin	Gym. Jura Hronca BA	1	1						1
79	Harton Eugen	Gym. Jura Hronca BA	1	1						1
79	Helebrandt Pavol	Gym. Jura Hronca BA	1	1						1
79	Holec Juraj	Gym. Jura Hronca BA	1	1						1
79	Kollár Jakub	Gym. Jura Hronca BA	1	1						1
79	Nagyová Iveta	Gym. Jura Hronca BA	1	1						1
79	Polák Matúš	Gym. Kežmarok	2	1						1
79	Thurzo Martin	Gym. Jura Hronca BA	1	1						1
79	Šafár Števo	Gym. Jura Hronca BA	1	1						1
89	Cielontko Tomáš	Gym. Jura Hronca BA	1	0						0
89	Polievka	Gym. Jura Hronca BA	1	0						0
89	Števíková Tereza	Gym. Jura Hronca BA	1	0						0