Korešpondenčný seminár z programovania
XXI. ročník, 2003/04
Katedra 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

Milí riešitelia!
KSP vám opäť raz prináša pár riadkov o tom, ako sa dali ľahko, rýchlo a správne vyriešiť zadané úlohy. Dúfame, že si ich so záujmom prečítate a budete ešte múdrejší ako už ste :-) A samozrejme na tých najlepších z vás sa tešíme, stretneme sa na sústredení... A aj vy sa máte na čo tešiť, sústredenie bude ešte lepšie ako obvykle!

Keď to nejde podobrom, musí to ísť nad obrom!

KSPáci

1. Zachariáš bezdomovec

opravovala Monika
(max. 15+1 bodov)

Zachariáš má na začiatku N špakov a z K špakov vie vytvoriť cigaretu, z ktorej následne získa ďalší špak späť. Teda na výrobu jednej cigarety odobral $K-1$ špakov (a žiaden nový špak nezískal). Takto môže odoberať stále $K-1$ špakov, až kým je počet špakov menší ako K. Zachariáš vyfajčí toľko cigariet, koľkokrát odoberie $K-1$ špakov. Potrebujeme teda v celých číslach vydeliť počet špakov N počtom špakov spotrebovaných pri výrobe jednej cigarety $K-1$. Jediná problematická situácia nastáva, ak nám nakoniec zostane $K-1$ špakov, čo sa pri celočíselnom delení započíta ako ďalšia cigareta. To ale, samozrejme, nie je pravda, a preto si na záver skontrolujem zvyšok po delení. Pokiaľ je zvyšok nulový (teda N bolo deliteľné $K-1$), tak treba od výsledku delenia odčítať 1 (a zostane nám $K-1$ špakov, nie 0).

Keďže program má fungovať pre veľmi veľké čísla, na zapamätanie čísla N používam pole celých čísel. Na začiatku načítam číslo N po znakoch (teda cifrách) a tieto cifry uložím do poľa `spaky`. Ošetria sa špeciálne prípady ($K=0$ a $K=1$). Potom predelím číslo N číslom $K-1$ klasickým delením, aké sa učí niekde na prvom stupni ZŠ: cifru predelím, ostane celá časť a zvyšok, za zvyšok pridám nasledujúcu cifru delenca a opakujem, až kým sa delenec neminie. Ostane mi celá časť – počet cigariet (nachádza sa v poli `cigarety`) a zvyšok (premenná `zv`) – počet zvyšných špakov. Skontrolujem, či nie je zvyšok nulový, ak je, tak počet cigariet znížim o jedna (funkcia `odcitaj`). Na záver už len vypíšem výsledný počet cigariet (použijem funkciu `orez`, aby na začiatku neboli zbytočné nuly).

Pamäťová zložitosť je logaritmická od počtu špakov $O(N)$ – alebo jednoduchšie povedané: lineárna od počtu cifier čísla N. Pri orezávaní núl algoritmus prejde pole cigariet najviac raz a pri delení je pole špakov tiež prezerané len raz a počet cigariet nemôže byt väčší ako počet špakov (až na špeciálne prípady), takže časová zložitosť je tiež logaritmická od N (alebo lineárna od počtu cifier N).

Hodnotenie.

- reprezentácia čísla N v poli – maximálne 8 bodov
- odôvodnenie, prečo použiť vzorec N div $(K-1)$ a zamyslenie sa nad zvyškom, prípadne odôvodnenie podobného vzťahu – maximálne 3 body

- použitie viacnásobného delenia (N sa predelí K, zostane celá časť a zvyšok, zvyšok sa sčíta s celou časťou a opakuje sa delenie) alebo použitie vzťahu $N div (K-1)$ alebo jemu podobného (bez odôvodnenia správnosti) – maximálne 2 body
- popis maximálne 2 body
- číslo N uložené v premennej 1 bod

A ešte drobný výber odpovedí na prémiovú otázku:

Matej Pagác: Najprv Zachariáš vyfajčil tých 10 špakov a mal teda vyfajčené 4 cigarety a dva zvyšné špaky, potom jeden špak stratil a o niečo nato našiel úplne celú jednu cigaretu a vyfajčil ju, takže nakoniec dňa mal vyfajčených 5 cigariet a ostali mu 2 špaky.

Michal Holub: To, že mal ráno 10 špakov môže znamenať, že vyfajčil jednu cigaretu a našiel 9 špakov. Potom z 10 špakov dokázal spraviť 4 cigarety, čiže za celý deň vyfajčil cigariet 5.

Peťo Goga: Z posledného delenia (N=123456789123456789 a K=5) mu zostal špak, no a z 11 špakov si môže vyrobiť 5 cigariet.

A ešte skutočné Zachariášovo riešenie: Zachariáš išiel za svojim kamarátom bezdomovcom Zoltánom a požičal si jeden špak. Tak si mohol vyrobiť aj poslednú piatu cigaretu. Keď ju dofajčil, špak vrátil späť Zoltánovi.

Listing programu:

```
program Zacharias_bezdomovec;
uses Crt;
const max=20;
var c:char;
    spaky : array [1..max] of byte;
    cigarety : array [1..max] of byte;
    K,i,zv,dlz_spak,dlz_cig:byte;

procedure orez;    {orezanie zaciatocnych 0 v poli cigarety}
var j:byte;
begin
  j:=0; repeat inc(j); until (cigarety[j]<>0) or (j>dlz_spak); {vyhladavanie pokial su 0}
  dec(j);
  for i:=1 to dlz_cig-j do cigarety[i]:=cigarety[i+j];
  dlz_cig:=dlz_cig-j;
end;

procedure odcitaj;       {odcitanie jednotky od pola cigarety}
var r:byte;
    skonci:boolean;
begin
  r:=dlz_cig; skonci:=false;
  repeat
    if cigarety[r]=0 then begin cigarety[r]:=9; dec(r); end
       else begin dec(cigarety[r]); skonci:=true; end;
  until (skonci) or (r=0);
end;

begin
  for i:=1 to max do begin spaky[i]:=0; cigarety[i]:=0; end;   {vynulovanie poli}
  dlz_spak:=0; dlz_cig:=0;
  repeat                {nacitanie vstupu}
    inc(dlz_spak);      {pocitanie cifier cisla N}
    c:=readkey;         {ocakavam vstup len cisla}
    spaky[dlz_spak]:=ord(c)-ord('0');
  until c=#13;
  dec(dlz_spak); read(K);
```

```
  if (K=0) or (K=1) then begin writeln('Vyrobi nekonecne vela spakov'); exit; end;

  dlz_cig :=dlz_spak;
  for i:=1 to dlz_spak do begin {samotne delenie}
     cigarety[i]:=spaky[i] div (K-1);
     if (i<>dlz_spak) then begin
        zv:=spaky[i] mod (K-1);
        spaky[i+1]:=spaky[i+1]+10*zv;
     end else zv:=spaky[i] mod (K-1);  {nakonci si zapamatam zvysok}
  end;

  if (zv = 0) then odcitaj; {odcitanie jednotky od pola cigarety}
  orez;
  writeln('Pocet cigariet je ');
  if (dlz_cig > 0) then for i:=1 to dlz_cig do write(cigarety[i]) {nejake cigarety vyrobil}
     else write('0.'); {pokial nevyrobi ani jednu cigaretu}
end.
```

2. Zábavka pred obedom

opravovala Julka
(max. 15 bodov)

Milí naši riešitelia. Tak ste sa ponáhľali na obed, že ste poväčšinou tento príklad ani neriešili. A aj z tých riešení fungovala iba polovica. Vzorové riešenie nevymyslel nikto, takže som za menej efektívne riešenie nestrhávala veľa bodov. Bodovanie vyzeralo takto:

- Vzorové riešenie, stačilo spomenúť, že sa to dá v lepšej pamäťovej zložitosti, čiže časová zložitosť $O(M+N)$ a pamäťová $O(M+N)$ mohlo dostať - 15 bodov
- Casová zložitosť $O(M+N)$ a pamäťová $O(N^2)$ - 14 bodov
- Casová aj pamäťová zložitosť $O(N^2)$ - 13 bodov
- Casová zložitosť $O(NM)$ a pamäťová $O(M+N)$ - 12 bodov
- $O(NM)$ a $O(N^2)$ alebo $O(N^3)$ a $O(N^2)$ - 11 bodov
- Horšie riešenia - podľa môjho vlastného uváženia
- Za zlý odhad zložitosti -1 bod, za chýbajúci -2 body, za chýbajúci popis som bola nekompromisná a podľa myšlienky max 6 bodov

Ták a hor sa na vzorák. Takýto príklad sa nazýva topologické triedenie. Väčšina z Vás si uvedomila, že ak sa našiel človek, ktorý nemal na zozname nikoho, tak mohol ihned ísť. Títo mohli ísť už v ľubovoľnom poradí, pretože neboli nijak na seba viazaný. Potom, ak sa našiel niekto, kto mal na zozname jedine ľudí, ktorí už odišli na obed, tak mohol na obed ísť tiež, ale až po týchto. Toto vedie k riešeniu: nájdeme všetkých ľudí, čo nemajú na zozname nikoho a dáme ich do fronty(zachvíľu vysvetlím) a každému ďalšiemu takéhoto človeka vyškrtneme. Problém je, že by bolo treba prehľadať každého človeka, či nemá niekoho z fronty na lístočku a potom mu ho vyškrtnúť. Avšak namiesto toho, aby sme si pre každého človeka pamätali, koho má mať pred sebou, budeme si pamätať iba koľko má ľudí pred sebou a koho má za sebou. Takto stačí v jeho zozname kto má prísť za ním každému človeku zmenšiť počet ľudí pred ním o 1, pretože vždy iba zisťujeme, či je ten počet ľudí 0.

A čo je to teda fronta? Je to taká dátová štruktúra, že vyberať sa dá iba zo začiatku a zapisovať iba na koniec (ako normálna fronta na obed, keď predpokladáte, že sa nikto nebude predbiehať). Môže sa napríklad zapisovať do poľa, pričom v jednej premennej si budete pamätať začiatok a v druhej koniec v poli. A aby ste nemuseli pole posúvať a pritom ho mohli plne využívať, na to sa používa myšlienka cyklického poľa (3.príklad 1.séria). Toto riešenie má zatiaľ iba optimálnu časovú zložitosť, ešte mu treba spraviť optimálnu pamäťovú

zložitosť. V jednom lineárnom poli (lineárnom od M, čo je počet hrán = podmienok) si budeme pamätať všetky hrany a v ďalšom lineárnom poli (od N) si budeme pamätať, od ktorého indexu v tom poli hrán, začínajú jeho hrany. Ale aby som Vás nepoplietla, tak zdrojový kód napíšem s kvadratickou pamäťovou zložitosťou.

Listing programu:

```
const maxN=50;
type Tclovek=record za:array [1..maxN] of integer; predn,zan:integer; end;
var lud:array [1..maxN] of Tclovek;
    n,i,j,pom,pocet,ide:integer;
    fronta,poradie:array [0..maxN] of integer;
    jedia:array [1..maxN] of boolean;
    koniec:boolean;
    fzac,fkon:integer;

procedure finsert(koho:integer);
begin fronta[fkon]:=koho; fkon:=(fkon+1) mod maxN; end;

function fdelete():integer;
begin
  if fzac=fkon then begin koniec:=true; end
     else begin fdelete:=fronta[fzac]; fzac:=(fzac+1) mod maxN; end;
end;

begin
  fzac:=0; fkon:=0; koniec:=true; pocet:=0;
  readln(n);
  for i:=1 to n do begin lud[i].zan:=0; lud[i].predn:=0; jedia[i]:=false; end;
  for i:=1 to n do begin
    pom:=-1;
    repeat
      read(pom);
      if (pom<>0) then begin
        inc(lud[i].predn); inc(lud[pom].zan);
        lud[pom].za[lud[pom].zan]:=i;
      end;
    until pom=0;
  end;
  { vlozime do fronty ludi, ktori uz mozu ist jest }
  for i:=1 to n do if lud[i].predn=0 then begin finsert(i); koniec:=false; end;
  repeat
    ide:=fdelete;
    if koniec then break;
    jedia[ide]:=true; inc(pocet); poradie[pocet]:=ide; { clovek "ide" ide jest }
    for i:=1 to lud[ide].zan do begin
      dec(lud[lud[ide].za[i]].predn);
      if (lud[lud[ide].za[i]].predn=0) then begin
        finsert(lud[ide].za[i]); { vlozime do fronty dalsieho co moze jest }
        if jedia[lud[ide].za[i]] then koniec:=true;
      end;
      if koniec then break;
    end;
  until (pocet=n) or koniec;
  if koniec then writeln('Poradie neexistuje, umru hladom')
     else for i:=1 to n do writeln(poradie[i]);
end.
```

opravoval Tom
(max. 15 bodov)

3. Zima je tu

Väčšina z vás našla správne riešenie tohto príkladu, ale len málokto sa pokúsil dokázať jeho správnosť. Body, ktoré ste mohli získať sa dali rozdeliť na body za zložitosť (max. 9) a body za popis (max. 6). Za časovú zložitosť $O(N \log N)$ bolo 9 bodov, za $O(N^2)$ bolo 7 bodov a za horšiu boli 3 body. Za samotný popis myšlienky sa dali získať 3 body a za jej dôkaz (alebo aspoň zdôvodnenie) bolo ďalších 0 až 3 bodov.

Vzorové riešenie: Nech teda máme N lyžiarov a N lyží. Výšky lyžiarov nech sú $v_1, v_2, \ldots, v_N$, kde $v_i \leq v_{i+1}$ pre $1 \leq i < N$. Dĺžky lyží nech sú $d_1, d_2, \ldots, d_N$, kde $d_i \leq d_{i+1}$ pre $i \leq i < N$. Uvažujme o priradení lyží lyžiarom v ktorom i-temu najnižšiemu lyžiarovi (t.j. tomu s výškou v_i) priradíme p_i-te najkratšie lyže (t.j. tie s dĺžkou d_{p_i}). Chceme nájsť také priradenie, v ktorom súčet $\sum_1^N |v_i - d_{p_i}|$ je minimálny. Tvrdím, že to dosiahnem ak i-temu najnižšiemu lyžiarovi priradím i-te najkratšie lyže. Tvrdenie samozrejme treba dokázať, tak poďme na to. Pre každé priradenie (ktoré i-temu najnižšiemu lyžiarovi priradí p_i-te najkratšie lyže) zrejme existuje číslo k také, že pre $i < k$ je $p_i = i$, ale $p_k > k$ ($p_k < k$ už nemôže byť). Toto k si nazvime stupeň priradenia. Vezmime si teraz najlepšie priradenie (teda také, pre ktoré ten súčet je minimálny). Ak ich je viac vezmime si to s najväčším stupňom. Nech je to priradenie také, že i-temu najnižšiemu lyžiarovi priradí p_i-te najkratšie lyže. Jeho stupeň nech je k. Ak $k = N$ nie je čo dokazovať. Inak nech $p_k = m$, $p_l = k$. Vezmime si teraz priradenie, ktoré že i-temu najnižšiemu lyžiarovi priradí p'_i-te najkratšie lyže, kde $p'_i = p_i$ pre $i \notin \{k, l\}$, $p_k = k$ a $p_l = m$. Zrejme $\sum_1^N |v_i - d_{p'_i}| = \sum_1^N |v_i - d_{p_i}| + (|v_k - d_k| + |v_l - d_m| - |v_k - d_m| - |v_l - d_k|)$, pričom zrejme $v_k \leq v_l$ a $d_k \leq d_m$. Môže nastať 6 prípadov:

- $d_k \leq d_m \leq v_k \leq v_l$, potom $|v_k - d_k| + |v_l - d_m| - |v_k - d_m| - |v_l - d_k| = v_k - d_k + v_l - d_m - v_k + d_m - v_l + d_k = 0$.
- $d_k \leq v_k < d_m \leq v_l$, potom $|v_k - d_k| + |v_l - d_m| - |v_k - d_m| - |v_l - d_k| = v_k - d_k + v_l - d_m - d_m + v_k - v_l + d_k = 2v_k - 2d_m < 0$.
- $d_k \leq v_k \leq v_l < d_m$, potom $|v_k - d_k| + |v_l - d_m| - |v_k - d_m| - |v_l - d_k| = v_k - d_k + d_m - v_l - d_m + v_k - v_l + d_k = 2v_k - 2v_l \leq 0$,
- $v_k < d_k \leq d_m \leq v_l$, potom $|v_k - d_k| + |v_l - d_m| - |v_k - d_m| - |v_l - d_k| = d_k - v_k + v_l - d_m - d_m + v_k - v_l + d_k = 2d_k - 2d_m \leq 0$,
- $v_k < d_k \leq v_l < d_m$, potom $|v_k - d_k| + |v_l - d_m| - |v_k - d_m| - |v_l - d_k| = d_k - v_k + d_m - v_l - d_m + v_k - v_l + d_k = 2d_k - 2v_l \leq 0$,
- $v_k \leq v_l < d_k \leq d_m$, potom $|v_k - d_k| + |v_l - d_m| - |v_k - d_m| - |v_l - d_k| = d_k - v_k + d_m - v_l - d_m + v_k - d_k + v_l = 0$.

Teda nové priradenie má stupeň $k+1$ a je také dobré, alebo dokonca lepšie, ako to, ktoré sme si vybrali. To sa však nemôže stať, teda tvrdenie je dokázané.

Stačí nám teda utriediť si lyže podľa dĺžky, lyžiarov podľa výšky a priradiť i-temu lyžiarovi i-te lyže pre $0 < i \leq N$.

Listing programu:

```
program ksp_z_21_2_3;
const Vela=1000;
type Pole=array[1..Vela] of integer;
var
  V,D,Perm,iPerm:Pole;
  N,i:integer;

procedure Tried(var A:Pole;l,h:integer;per:Boolean);
var i,j,p,x:integer;
begin
  p:=A[(l+h) div 2]; i:=l; j:=h;
```

```
  repeat
    while A[i]<p do inc(i);  while A[j]>p do dec(j);
    if i<=j then begin
      x:=A[i]; A[i]:=A[j]; A[j]:=x;
      if per then begin x:=Perm[i]; Perm[i]:=Perm[j]; Perm[j]:=x; end;
      inc(i); dec(j);
    end;
  until i>j;
  if l<j then Tried(A,l,j,per);
  if i<h then Tried(A,i,h,per);
end;

begin
  ReadLn(N);
  for i:=1 to N do begin Read(V[i]); Perm[i]:=i; end; {vysky lyziarov}
  for i:=1 to N do Read(D[i]);
  Tried(V,1,N,True);
  Tried(D,1,N,False);
  for i:=1 to N do iPerm[Perm[i]]:=i;
  for i:=1 to N do
     writeLn(i,'-ty lyziar (vysky ',V[iPerm[i]],') dostane lyze dlzky ',D[iPerm[i]]);
end.
```

opravoval WSX
(max. 15 bodov)

4. Za rohom je kôň

Gary bol z vašich riešení sklamaný, väčšina z nich nenašla optimálne rozostavenie koní pre šachovnicu rozmerov $8x8$. Tieto riešenia boli založené na všemožných heuristických metódach. Ich bodovanie záležalo najmä na optimalite riešení a takéto riešenia mohli získať najviac 8 bodov. Ďalšia skupina riešení, založená na métode backtrackingu, sa vyznačovala tým, že vždy vrátila optimálne rozostavenie koní. Tieto riešenia delíme na 2 skupiny:

- veľmi pomalé (max. 10 bodov) – skúšame zaradom všetky umiestnenia $1, 2, \ldots$ koní, keď nájdeme riešenie pokrývajúce celú šachovnicu, vypíšeme ho a skončíme
- pomalé (max. 15 bodov) – prechádzame políčka šachovnice, keď nájdeme neohrozené, skúsime ho ohroziť jedným z 9 možných spôsobov a pokračujeme v prechádzaní

Samozrejme, body sa strhávali za chýbajúci popis, zdôvodnenie správnosti alebo neprehľadnosť kódu.

A teraz k samotnému vzorovému riešeniu:

Budeme používať pole *x_kone*, v ktorom si pre každé políčko šachovnice budeme pamätať, či je to prázdne políčko, kôň alebo okraj (mimo šachovnice), pole *x_ohroz*, v ktorom bude počet ohrození daného políčka a pole *x_best*, v ktorom je najlepšie doteraz nájdené rozloženie koní. Po inicializácií týchto polí zavoláme hlavnú funkciu algoritmu – *backtrack*, so súradnicami $[0, 0]$. Vstupné súradnice $[pos_x, pos_y]$ tejto funkcie znamenajú, že všetky políčka $[x, y]$ také, že platí $pos_y < y \vee (pos_y = y \wedge pos_x < x)$ sú ohrozené aspoň jedným koňom. Funkcia bude po zavolaní postupne prechádzať políčka až kým nenájde také, ktoré nie je ohrozené. Týmto maximalizujeme hodnoty pos_x, pos_y, pre ktoré platí vstupná podmienka. Pokiaľ sme vyšli von z šachovnice, znamená to, že všetky políčka sú ohrozené, teda sme našli možného kandidáta na najlepšie riešenie. V opačnom prípade je políčko $[pos_x, pos_y]$ neohrozené. Uvažujme všetkých 9 (vrátane políčka $[pos_x, pos_y]$, ktorým políčko obsadíme) políčok, kde umiestnením koňa by sme ohrozili políčko $[pos_x, pos_y]$. Pokiaľ sa takéto políčko nachádza na šachovnici, určite sa na ňom kôň nenachádza (v opačnom prípade by bolo $[pos_x, pos_y]$

ohrozené). Teda môžeme postupne na každé z týchto 9 políčiek skúsiť položiť koňa, čím ohrozíme $[pos_x, pos_y]$, a rekurzívne sa zavolať so súradnicami $[pos_x, pos_y]$.

Tento algoritmus určite raz skončí. V každom rekurzívnom zavolaní určite zväčšíme súradnice $[pos_x, pos_y]$, lebo toto políčko sme ohrozili práve v predchádzajúcom volaní. Ďalej v každom volaní funkcie *backtrack* sa rekurzívne zavoláme maximálne 9 krát. Teda v najhoršom prípade skončíme vykonávanie po 9^{n^2} volaniach funkcie *backtrack*, čo je zároveň aj horný odhad časovej zložitosti.

Po skončení algoritmu vyberieme spomedzi kandidátskych riešení najlepšie a tvrdíme, že je to optimálne riešenie, čo aj dokážeme:

Predpokladajme, že najlepšie riešenie má nejaké rozostavenie koní. Pre každé políčko si vyberme práve jedeného koňa, ktorý toto políčko ohrozuje. Teraz predpokladajme volania funkcie *backtrack* a vždy, keď sa rozhodujeme, ako ohroziť dané políčko umiestnime práve vybraného koňa z predpokladaného optimálneho riešenia. Kandidátske riešenie, ktoré vzniklo, pozostáva z podmnožiny koní najlepšieho riešenia, teda je aspoň také dobré ako najlepšie. Teda riešenie nájdené popísaným algoritmom je optimálne.

Popísané riešenie bude ešte stále veľmi pomalé a ukážeme si, ako ho zlepšiť: Počas chodu algoritmu si budeme pamätať počet neohrozených políčok. Ďalej si budeme najlepšie riešenie počítať priebežne namiesto počítania kandidátskych riešení. Po každom zavolaní funkcie *backtrack* si overíme, či ešte máme šancu dosiahnuť lepšie riešenie ako doteraz najlepšie nájdené. Na ohrozenie z neohrozených políčok potrebujeme aspoň $\lceil \frac{z}{9} \rceil$ koní. Pokiaľ sme už použili k koní, máme z neohrozených políčok, počet koní v najlepšom riešení je bk a platí $k + \lceil \frac{z}{9} \rceil \geq bk$, nemá zmysel v danej vetve pokračovať, lebo lepšie riešenie určite nedosiahneme.

Pre zaujímavosť, optímálne počty koní, ktoré sú známe sú:

rozmer	1	2	3	4	5	6	7	8	9	10	11	12	13	14
koní	1	4	4	4	5	8	10	12	14	16	21	24	28	32

Pre väčšie šachovnice sú nájdené riešenia, ktoré sa považujú sa veľmi dobré, ale zistiť, či sú optimálne je náročnejšie ako nájsť dobré riešenie.

Listing programu:

```
#include <stdio.h>
#include <string.h>

#define MAX_N 100
#define NIC 0
#define KON 1
#define OKRAJ 2

int smer_x[9]={-1, 1, 2, 2, 1, -1, -2, -2, 0};
int smer_y[9]={2, 2, 1, -1, -2, -2, -1, 1, 0};
int x_kone[MAX_N+4][MAX_N+4], x_ohroz[MAX_N+4][MAX_N+4];
int x_best[MAX_N+4][MAX_N+4], k_now, k_best, neohroz;
int n;


void backtrack(int pos_x, int pos_y) {
  int kon_x, kon_y;
  int i, j;

  if(k_now+(neohroz+8)/9>=k_best) return;
  while(x_ohroz[pos_x+2][pos_y+2]) {
    pos_x++;
    if(pos_x==n) {
```

```
      pos_x=0; pos_y++; // a ak mame mame lepsie riesenie, koniec:
      if(pos_y==n) { k_best=k_now; memcpy(x_best, x_kone, sizeof(x_best)); return; }
    }
  }

  k_now++;
  for(i=0; i<9; i++) {
    kon_x=pos_x+smer_x[i]; kon_y=pos_y+smer_y[i];
    if(x_kone[kon_x+2][kon_y+2]==OKRAJ) continue;
    x_kone[kon_x+2][kon_y+2]=KON; // umiestnime kona
    for(j=0; j<9; j++) { // spocitame ohrozene policka
      if(x_kone[kon_x+smer_x[j]+2][kon_y+smer_y[j]+2]!=OKRAJ &&
         !x_ohroz[kon_x+smer_x[j]+2][kon_y+smer_y[j]+2]) neohroz--;
      x_ohroz[kon_x+smer_x[j]+2][kon_y+smer_y[j]+2]++;
    }
    backtrack(pos_x, pos_y); // rekurzivne umiestnime dalsie kone
    for(j=0; j<9; j++) { // vratime vsetko do povodneho stavu
      x_ohroz[kon_x+smer_x[j]+2][kon_y+smer_y[j]+2]--;
      if(x_kone[kon_x+smer_x[j]+2][kon_y+smer_y[j]+2]!=OKRAJ &&
         !x_ohroz[kon_x+smer_x[j]+2][kon_y+smer_y[j]+2]) neohroz++;
    }
    x_kone[kon_x+2][kon_y+2]=NIC;
  }
  k_now--;
}

int main() {
  int i, j;

  scanf("%d", &n);
  memset(x_ohroz, 0, sizeof(x_ohroz)); memset(x_kone, 0, sizeof(x_kone));
  for(i=0; i<n+4; i++) {
    x_kone[i][0]=OKRAJ; x_kone[i][1]=OKRAJ;
    x_kone[i][n+2]=OKRAJ; x_kone[i][n+3]=OKRAJ;
    x_kone[0][i]=OKRAJ; x_kone[1][i]=OKRAJ;
    x_kone[n+2][i]=OKRAJ; x_kone[n+3][i]=OKRAJ;
  }
  k_best=n*n+1;   k_now=0;   neohroz=n*n;

  backtrack(0, 0);
  printf("best solution: %d\n", k_best);
  for(i=0; i<n; i++) {
    for(j=0; j<n; j++) if(x_best[i+2][j+2]==KON) printf("K"); else printf(".");
    printf("\n");
  }
  return 0;
}
```

5. Zaopravujte si

opravoval Poko
(max. 15 bodov)

Milí riešitelia. Máte pravdu, bol to netradičný príklad. Ale aj veľmi poučný, a nielen pre vás. Podľa vašich riešení sa dá totiž zistiť, kto (už) pochopil, ako sa má písať riešenie príkladu do KSP. Keď totiž pochopíte (objavíte), ako také riešenie treba opraviť, bude vám jasné, ako ho treba napísať a nebyť prekvapený, že za toto a toto mi strhli body. Nebude vám viac chýbať popis algoritmu, zmienka o jeho správnosti, či aspoň pokus o odhad zložitosti.

Ďalej sme mali nápad, že podľa vašich opravovaní by sme mohli opravovať vás. Každý vedúci by dostal kópiu vášho riešenia tohoto príkladu, resp. inštrukcie, ako vás hodnotiť ... No nechajme to radšej tak :-)

A ako by riešenia opravil niekto z KSP (napríklad ja :-)? Problém, ktorý treba vyriešiť, je známy problém nájdenia najkratšej cesty v grafe. Mestá na planéte Diks budeme reprezentovať vrcholmi, potrubia hranami ohodnotenými časom, ktorý treba na presun medzi dvoma mestami. Za optimálne riešenie budeme považovať Dijkstrov algoritmus, ktorý má časovú aj pamäťovú zložitosť $O(N^2)$, kde N označuje počet vrcholov.

Riešenie 1: Procedúra `Load` načíta graf do matice susednosti, t.j. $A[i,j]$ bude čas potrebný na presun medzi mestami i, j. Potrubím sa dá cestovať oboma smermi v rovnakom čase, takže graf je neorientovaný a správne bude matica symetrická (t.j. $A[i,j] = A[j,i]$). Navyše na najkratšiu cestu z nejakého vrchola do toho istého nepotrebujeme žiaden čas, teda správne $A[i,i] = 0$ pre všetky i. Dvojica príkazov vo vnútri cyklov sa vykoná $N-1+N-2+\ldots+1+0 = \frac{1}{2} \cdot N \cdot (N-1) = \frac{1}{2} \cdot (N^2 - N)$ krát, teda procedúra má časovú zložitosť $O(N^2)$.

Pozrime sa na procedúru `Skus`. Keď už k programu nemáme popis, tak aspoň z názvov procedúr a premenných si tipneme, že premenná `kam` môže znamenať vrchol, do ktorého sme skúsili prísť. Predpokladajme, že je to tak. Keď sme teda skúsili prísť do vrchola N, porovnáme hodnotu `naj` s `dlzka`. Z toho môžeme usúdiť, že dĺžka cesty sa nejakým spôsobom bude ukladať do premennej `dlzka` a dĺžka zatiaľ najkratšej nájdenej cesty z vrcholu 1 do vrcholu N bude uchovaná v premennej `naj`. Keď sme skúsili prísť do mesta i, $i \neq N$, zaznačíme si, že sme tam už boli[1] a skúsime ísť do iného mesta j, $j \neq i$, (rekurzívne sa zavoláme), pričom doterajšiu dĺžku zväčšíme o $A[i,j]$, t.j. o čas potrebný na presun z mesta i do j. Následné odznačenie, že sme vo vrchole boli, a úvodné volanie procedúry `Skus(1,0)` nám dopovie význam použitého algoritmu. Uvidíme, že toto riešenie skúša všetky možné cesty medzi vrcholmi 1 a N, a keď nájde kratšiu, než je najkratšia dovtedy nájdená cesta, aktualizuje si príslušný údaj. Všetkých možných ciest je však až exponenciálne veľa, teda riešenie má exponenciálnu časovú zložitosť. Je to pomalé, ale správne, preto by som ho hodnotil 8 bodmi.

Riešenie 2: Venujme sa rovno procedúre `Skus`. Predpokladajme, že premenné `kam` a `dlzka` majú ten istý význam, ako v predošlom riešení. Ak `kam <> N`, v následnom cykle nájdeme číslo vrchola, ktorý nie je označený (zo zatiaľ neznámych dôvodov) a je najbližšie k vrcholu `kam` a rekurzívne sa zavoláme na tento vrchol, pričom zväčšíme `dlzka` o vzdialenosť medzi tými vrcholmi. Môže sa nám zdať divné, že sa bavíme o nejakých označených vrcholoch, ale nikde sme zatiaľ nič neoznačovali. A skutočne, ide o chybu v riešení. Totiž v prvom zavolaní `Skus(1,0)` sa neznemia globálne hodnoty a `Skus` sa opäť zavolá s tými istými parametrami, čiže sa zacyklí. Dobre, tu by sme mohli s opravovaním skončiť a prehlásiť riešenie za chybné, ale čo keď to riešiteľ písal na poslednú chvíľu a z nejakých príčin sa iba drobne v programe pomýlil? Bolo by nespravodlivé[2] dať mu za to 0 bodov, keď to môže mať správne, teda až na ten bug. Do poľa `bol` si riešiteľ zrejme chcel označiť vrcholy, ktoré už navštívil. Zrejme teda pred príkazom `max:=30000` mal byť ešte príkaz `bol[kam]:=true;`. Potom sa riešenie nezacyklí, ale... Ešte to nie je ono. Použitý algoritmus teraz začne vo vrchole 1 a pokračuje vždy do najbližšieho ešte nenavštíveného vrcholu. Ide o greedy prístup, ale nie je správny. Ako protipríklad môže poslúžiť príklad zo zadania, keď zmeníme 13 na 21 alebo väčšie. Riešenie je teda tak či tak nesprávne. Ale bez toho bugu správne vyrieši nejakú skupinu vstupov, preto by som dal 1 bodík.

Riešenie 3: Ako prvú si možno položíme otázku, aký význam má premenná `naj`. Čo si do nej ten riešiteľ ukladá? V procedúre `Skus` do `naj[i]` pričítavame nejakú vzdialenosť, takže to bude mať niečo spoločné s nájdenými vzdialenosťami. Predpokladajme, že `naj[i]`

[1] opäť iba usudzujeme z názvu premennej `bol`

[2] i keď riešenia sa do KSP nemajú písať na poslednú chvíľu

chceme mať na konci programu najkratšiu vzdialenosť z vrchola 1 do vrcholu i a podľa tohoto predpokladu skúmajme, ako pracuje algoritmus. Na začiatku inicializujeme celé pole `naj[i]` na hodnotu 30000, čo zvyčajne značí nejaký horný odhad možných hodnôt, teda akési nekonečno. Navyše `naj[1] = 0`, to vieme určite povedať, takže zatiaľ predpoklad sedí. Teraz sa zopár ráz[3] zavolá `Skus` a tá niečo porobí s poľom `naj`. Posvieťme si na to: ak zatiaľ najkratšia nájdená cesta z 1 do i plus vzdialenosť z i do j je kratšia ako dosiaľ najkratšia nájdená cesta z 1 do j, tak aktualizujeme `naj[j]`. To dáva zmysel, lebo po takejto úprave máme v `naj[j]` uloženú kratšiu existujúcu cestu, ako sme doteraz našli. Uvedomte si, že pokiaľ by cesta do i neexistovala, muselo by byť `naj[i]=30000` (prečo?), a teda by sme `naj[j]` neaktualizovali.

Ok, teraz nás trápi, prečo sa `Skus` volá $N-2$ krát. Nie je to málo? Nie je to veľa? Riešiteľ neuviedol dôkaz správnosti, zatiaľ teda neveríme, že to má správne. Napriek tomu opäť nebudeme krutí, a skúsime to zistiť za riešiteľa (a potom mu za to strhneme pár bodov :-) Ak $N = 1$, dostaneme určite správne riešenie. Ak $N = 2$, nastáva malý problém, lebo `Skus` sa nezavolá ani raz a algoritmus vráti nesprávny výsledok. Toto považujeme za malú chybu, za ktorú strhneme bod. Nech teraz $N \geq 3$. Uvažujme, pre ktoré i sa hodnota `naj[i]` nebude meniť po prvom zavolaní `Skus`. Zrejme pre $i = 1$. Ďalej máme isté, že existuje vrchol, ktorý je k vrcholu 1 najbližšie. Označme ho v. Potom určite `naj[v] = A[1,v]` (prečo?). Ale hodnotu `naj[v]` vypočítame ešte v prvom prevedení vnútorného cyklu v `Skus`. Pred druhým opakovaním bude pre $i \geq 2$ `naj[i] = A[1,i]`. Než skončí prvé volanie `Skus`, určite sa pre všetky $i, i \neq v$ overí podmienka `naj[i] < naj[v] + A[v,i]`. Potom pre vrchol w, $w \neq 1, w \neq v$, taký, že `naj[w]` je tretie najmenšie (t.j. po `naj[1]` a `naj[v]`), sa `naj[w]` už dokonca nebude meniť. A podobne, v každom ďalšom volaní `Skus` bude pre niektoré ďalšie u hodnota `naj[u]` definitívna. Dôvod je obdobný ako pri zdôvodnení správnosti postupu Dijkstrovho algoritmu. Máme nejakú množinu vrcholov, ktoré majú svoju hodnotu `naj` definitívnu. Nech u je vrchol, ktorý sme do tejto množiny pridali ako posledný (na začiatku je to vrchol 1). Aktualizujeme všetky vrcholy mimo množiny, ak hodnota `naj[i] < naj[u] + A[u,i]`. Po tejto aktualizácii vieme vybrať vrchol v mimo množiny, ktorý má spomedzi ostatných vrcholov mimo množiny minimálnu hodnotu `naj`. Tento vrchol pridáme do množiny, lebo sa nám určite neoplatí ísť do v ešte cez nejaký vrchol w mimo množiny (bude to určite dlhšie).

Preto nám $N-2$ volaní `Skus` určite stačí. A ešte príklad, kedy potrebujeme `Skus` zavolať $N-2$ krát. Nech $N = 5$, `A[1,4] = A[2,3] = A[2,5] = A[3,4] = 1`, ostatné vzdialenosti nech sú rovné 10. Potom dĺžku najkratšej cesty z vrchola 1 do vrcholu 5 nájdeme až v poslednom volaní `Skus`.

Procedúra `Skus` sa zavolá $N-2$ krát, samotná procedúra má časovú zložitosť $O(N^2)$, teda celková časová zložitosť bude $O(N^3)$ (vidíme, že procedúra `Load` pracuje v čase $O(N^2)$, takže nám výslednú zložitosť nezmení). Za toto riešenie by som dal 11 bodov.

A ako som hodnotil vás? Za každé opravené riešenie ste mohli získať maximálne 5 bodov. Ak ste neuviedli dôvod vášho hodnotenia, strhával som 4 body. Ak bol dôvod nesprávny (nepochopili ste, príp. zle pochopili riešenie) alebo neúplný, strhol som najviac 4 body. Ak bolo vaše hodnotenie príliš neprimerané vzhľadom na váš dôvod, strhol som bod. Udeľoval som aj nejaké bonusy: Ak ste si spomenuli, že riešeniam vlastne chýbal popis, zmienka o ich správnosti a odhad zložitosti, pridal som 1 bod. Ak ste riešenie č. 2 rozanalyzovali až do konca, dostali ste navyše 1 bod. Ak ste najlepšiemu riešeniu z tých troch (bez ohľadu na to, aké je zlé) udelili 15 (maximum) bodov, strhával som 1 bod . No a výsledok som ešte uťapkal zdola na 0 a zhora na 15 bodov.

[3] prečo toľkokrát, to nás zatiaľ príliš netrápi, predpokladáme, že to bude dostatočne veľa

Výsledková listina po 2. kole kategórie KSP-Z

	Meno a priezvisko	Škola	Ročník		21	22	23	24	25	Σ
1	Morihladko Peter	Gym. Školská Spiš. Nová Ves	3	66	5	13	12	5	13	114
2	Trnovcová Zuzana	Gym. Jura Hronca BA	3	54	12	13	15		13	107
3	Hrinčár Peter	ZŠ Jarná Poprad	0	54	12	1	12	11	15	105
4	Jerguš Ján	Gym. Alejová Košice	1	44	12	11	6	10	13	96
5	Ilavský Milan	SPŠE Liptovský Hrádok	4	55	6	11	10		12	94
5	Ľudvík Martin	Gym. Školská Spiš. Nová Ves	3	48	5	10	12	7	12	94
7	Mindek Peter	Škola pre mim. nadané deti BA	2	50	15	10	10		8	93
8	Herman Peter	Gym. Jura Hronca BA	1	46	12	10	10	8	6	92
9	Fedák Matúš	Gym. Stará Ľubovňa	2	30	15	10	10	10	10	85
9	Horník Jozef	Gym. SNP Galanta	4	41	5	13	10	8	8	85
11	Buštor Ivan	Gym. Košická BA	1	40	11		12	8	13	84
12	Ďuďák Juraj	Gym. Golianova Nitra	2	51	8	0	12	4	6	81
13	Dubás Jakub	Gym. Humenné	-1	42	13	1	9	6	9	80
13	Špalek Lukáš	Gym. Čadca	3	36	12	11	14		7	80
15	Dzurenko Miroslav	Gym. Ľ. Štúra Zvolen	3	37	4	10	10	9	8	78
16	Goga Peter	Gym. 17. novembra Topoľčany	4	39	5	10	12	5	3	74
16	Tomori Rudolf	Gymnázium	2	36	5	9	10	6	8	74
16	Villaris Vojtech	Gym. Jura Hronca BA	3	45	6	13	10			74
19	Bundala Daniel	Gym. Jura Hronca BA	2	57		1			14	72
19	Domány Dušan	Gym. P. Horova Michalovce	3	34	16	10	12			72
21	Mikuláš Ján	Gym. Haličská Lučenec	2	19	12	11	12	5	10	69
22	Dobiaš Michal	Gym. Jura Hronca BA	2	39	6		10		13	68
23	Holub Michal	Gym. Jura Hronca BA	3	29	6	13	10		9	67
24	Kuchár Richard	SPŠE Prešov	3	18	6	13	10	5	8	60
25	Zagora Martin	SPŠ Komenského Trnava	3	44	4	1	10			59
26	Pagáč Matej	Gym. Školská Spiš. Nová Ves	3	26	9		12		10	57
27	Kajan Peter	Gym. Jura Hronca BA	1	31	10	4	7	4		56
27	Sábo Jozef	Gymnázium	1	30	4	10	12			56
29	Sopoliga Radoslav	Gym. Svidník	4	23	12	10	10			55
30	Mazák Tomáš	Gym. Jura Hronca BA	1	53						53
31	Koman Pavol	Gym. Daxnera V.n. Topľou	4	34	4		12			50
31	Sivák Michal	Gym. Ľudovíta Štúra Trenčín	2	29	5		16			50
33	Ambrož Peter	Gym. Jura Hronca BA	4	48						48
33	Balga Jozef	Gym. maďarské Šahy	4	32	6	0	10			48
35	Korcsok Peter	Gym. Trstená	0	25		10	10			45
35	Varga Peter	Gym. Jura Hronca BA	4	45						45
37	Kottman Michal	Gym. Jura Hronca BA	3	44						44
38	Jirásek Jozef	Gym. Zbrojničná Košice	1	43						43
39	Žubrietovský Tomáš	Gym. Ľ. Štúra Zvolen	3	17	4	6	7		7	41
40	Sivák Vladimír	Gym. Ľudovíta Štúra Trenčín	2	25	5		10			40
40	Zápotocký Radoslav	Gym. Alejová Košice	4	40						40
42	Pančík Andrej	Gym. Tajovského B. Bystrica	1	24	5	9	1			39
43	Belan Tomáš	Škola pre mim. nadané deti BA		37						37
44	Mikuláš Ondrej	Gym. Haličská Lučenec	1	21			10	5		36
45	Vanderka Peter	Gym. Trstená	3	35						35
45	Černo Lukáš	Gym. Ľudovíta Štúra Trenčín		35						35
47	Kompuš Patrik	Gym. Šrobárova Košice	1	14	10		10			34
47	Ragány Peter	SPŠE Prešov		29	5					34
47	Šimják Patrik	Gym. Párovská Nitra	2	11	6	2	11	4		34
50	Fzeši Štefan	Gym. maďarské Šahy	4	21	4		7			32

	Meno a priezvisko	Škola	Ročník		21	22	23	24	25	Σ
51	Pavlovský Martin	Gym. Jura Hronca BA	1	0	11	5	7	8		31
51	Varga Ľubomír	SPŠ Nitra	4	27					4	31
53	Sedlák Štefan	Gym. Daxnera V.n. Topľou	3	30						30
54	Halamíček Radovan	Gym. Jura Hronca BA	2	29						29
54	Hlavačiková Jana	Gym. Einsteinova BA	3	29						29
54	Hrubý Martin	Gym. P. Horova Michalovce	3	29						29
57	Zachar Lukáš	Gym. A.Sládkoviča B. Bystrica	3	21	4	0	0	3		28
58	Bálint Farkaš	Gym. Mládežnícka Šahy	3	11	5	1	10			27
58	Hegedušová Monika	Gym. P. Horova Michalovce	2	8	5		10		4	27
60	Golier Richard	Gym. Alejová Košice	4	26						26
60	Petrezsél Tibor	Gym. maďarské Šahy	4	8	5	11	2			26
62	Barna Jozef	Gym. Humenné	3	25						25
62	Lukáč Matej	Gym. Šrobárova Košice		15			10			25
64	Kováčovský Tomáš	Neznama skola		0	10	4	7			21
64	Račko Tibor	Gym. Mládežnícka Šahy	3	6	5		10			21
64	Urminský Tomáš	SPŠ Nové Mesto nad Váhom	2	21						21
67	Macík Tomáš	Gym. A.Sládkoviča B. Bystrica	3	20						20
68	Baumann Martin	Gym. Jura Hronca BA	1	19						19
68	Dzurňák Tomáš	Gym. Školská Spiš. Nová Ves	2	19						19
70	Paulis Peter	Gym. Cyrila a Metoda Nitra	3	0	5		10		3	18
71	Herich Andrej	Gym. Golianova Nitra	2	17						17
72	Hrabčák Ján	Gym. Alejová Košice	4	16						16
72	Kolesár Štefan	Gym. Alejová Košice	2	9			7			16
72	Kubek Vojtech	Gym. Jura Hronca BA	1	16						16
75	Jacko Vladimír	Gym. Alejová Košice		8			7			15
75	Kováč Ivo	Gym. Alejová Košice		15						15
75	Škrovinová Katarína	Gym. Párovská Nitra	2	15						15
78	Svonava Daniel	Gym. Jura Hronca BA	2	14						14
79	Buday Tomáš	Gym. Golianova Nitra	3	13						13
79	Kučera Dmitrij	Gym. Šrobárova Košice	1	2	4	0	5	0	2	13
81	Mižáková Katarína	Gym. Alejová Košice		12						12
82	Bažík Martin	Gym. Jura Hronca BA		0	4		7			11
82	Gálik Maroš	Gym. Daxnera V.n. Topľou	2	11						11
82	Nižňanský	Gym. Alejová Košice		11						11
85	Kramárik Matúš	Gym. Ľudovíta Štúra Trenčín	2	10						10
85	Mondoková Denisa	Gym. Šrobárova Košice	1	2	3		5			10
87	Godány Robert	Gym. Grösslingová BA	2	9						9
87	Hornáková Anna	Gym. Alejová Košice		9						9
87	Kováč Peter	Gym. Alejová Košice		9						9
90	Hulitka Edmund	Gym. Fiľakovo	2	7						7
90	Jokel Tomáš	Gym. Šrobárova Košice	1	2	5					7
90	Kertész Tomás	Gym. Fiľakovo	2	7						7
90	Varga Marek	SPŠ Nitra	4	7						7
94	Kollár Martin	Gym. Alejová Košice		6						6
95	Králová Diana	Gym. Šrobárova Košice	1	3		0		2		5
95	Lopata Ladislav	Gym. Alejová Košice		5						5
97	Kostolanský Rastislav	SPŠ Nitra	4	4						4
97	Mačaj Martin	SPŠE Prešov	3	4						4
97	Tinajová Andrea	Gym. Malá Hora Martin	3	4						4
97	Vida Jozef	SPŠ Nitra	4	4						4
	...	...								