Korešpondenčný seminár z programovania
XXII. ročník, 2004/05
Katedra základov a vyučovania informatiky FMFI UK,
Mlynská Dolina, 842 48 Bratislava

KSP finančne podporujú: aSc – Applied Software Consultants spol. s r.o.
MICROSTEP-MIS spol. s r.o.

Vzorové riešenia 1. kola zimnej časti, kat. Z

A sú tu vzoráky! Kto nevie čo s nimi, nech si naplní ústa vodou a môže začať čítať.

A pozor, mokrá dlážka!!!

KSPáci

1. Zruční artisti

opravovali Anička a Rasťo
(max. 15 bodov)

O čo v tomto príklade išlo ste pochopili všetci. Bolo treba usporiadať váhy artistov zostupne, teda použiť nejaký sort. Aj to ste všetci zvládli. Niektorí z vás však sortili v čase $O(N^2)$, napríklad BubbleSortom, InsertSortom a podobne. Ide to však aj rýchlejšie. Napríklad triedením zvaným QuickSort. Povedzme si niečo o ňom.

Predstavme si pole váh artistov A. Vyberme z neho nejakú hodnotu H. Radi by sme dostali všetky prvky väčšie (rovné) ako H napravo od H a všetky prvky menšie (rovné) ako H naľavo od neho. To sa nám nejako podarí. Pozrieme sa na ľavú časť nášho poľa A (tu sú všetky prvky menšie, rovné ako H). Vyberieme z nej nejakú hodnotu H_2. Opäť chceme všetky prvky väčšie ako H_2 z tejto časti dostať napravo od neho a menšie naľavo. To isté spravíme pre pravú časť (prvky väčšie, rovné ako H).A tak ďalej...

Toto je podstata triedenia prvkov v poli QuickSortom: vyberieme z pola prvok. Všetky väčšie prvky hádžeme napravo od neho, menšie naľavo. Rozdelíme pole na ľavú a pravú časť a pre obe zopakujeme tento istý postup. To znamená, že na konci každú z nich znovu rozdelíme na dve časti a pre ne zase zopakujeme ten istý postup. Ale niekedy by sme asi mali aj skončiť ;) Je myslím zrejmé, že jedno číslo samé o sebe triediť nemusíme a takisto dve čísla už ľahko utriedime.

Ako sa to píše? Kamarátom nám bude rekurzia. Majme pole. Vyberme si našu hodnotu H, ktorá sa väčšinou nazýva pivot, teda vodiaci prvok. Bežne sa pivotovi priradí súčet hodnôt prvého a posledného prvku poľa div 2. Ďalej majme dve ukazovátka. Jedno z nich ide po prvkoch poľa zľava a kontroluje, či niektorí prvok nie je väčší ako pivot. Ak taký nájde, zastaví. Druhé ukazovátko ide sprava a kontroluje, či niektorí prvok, ktorý stretne, nie je menší ako pivot. Ak taký nájde, zastaví. Potom si ľavé a pravé ukazovátko vymenia prvky, na ktorých zastali. Takto pokračujú, kým sa nepretnú, lebo potom už je jasné, že boli všetky prvky prezreté. Potom sa zavolá tá istá procedúra na ľavú a pravú časť poľa. (Tomuto volaniu toho istého postupu sa hovorí rekurzia. Procedúrke treba samozrejme pri každom volaní povedať nové parametre, ktoré označujú začiatok a koniec časti poľa, na ktorú procedúrku práve voláme), ale iba v prípade, že majú dĺžku aspoň dva.

Časová zložitosť: V priemernom prípade $O(N \log N)$, kde N je počet prvkov poľa. Prečo? V každom kroku rozdelíme každú časť poľa na dve časti. V priemernom prípade budú približne rovnako dlhé, teda budú mať približne polovičnú dĺžku ako pôvodné pole. Budeme teda raz spracúvať celé pole, dvakrát polovicu poľa, štyrikrát štvrtinu, atď. až N-krát jeden prvok. Zjavne spracovať k-krát k-tinu poľa nám trvá rovnako ako spracovať raz celé pole – lineárne dlho, teda $O(N)$. Aby sme dostali z poľa dĺžky N polia dĺžky 1, potrebujeme ho deliť $\log N$ krát. Preto celková časová zložitosť bude $\log N \cdot O(N) = O(N \log N)$.

Pamäťová zložitosť: Je $O(N)$. Prvky máme uložené v jednorozmernom poli, ktorého dĺžka je závislá od počtu triedených prvkov, teda lineárna.

Bodovanie:

- QuickSort alebo iný sort so zložitosťou $O(N \log N)$: 15 bodov
- BubbleSort, InsertSort alebo iné triedenie so zložitosťou $O(N^2)$: 10 bodov
- Chýbajúci/zlý odhad časovej zložitosti: -2 body / -1 bod
- Chýbajúci/nedostatočný popis: -2 body / -1 bod

Poznámka pre tých, ktorým je QuickSort jasný.
QuickSort má síce v priemernom prípade zložitosť $O(N \log N)$, no ak máme nešťastie na pivota, jeho zložitosť sa môže vyšplhať až na $O(N^2)$. Napríklad, ak sa nám vždy podarí vybrať takmer najväčší prvok z časti, ktorú práve spracovávame, triedenie sa vlastne zmení na MaxSort. Ako tomu pomôcť? Poznáme triedenie zvané MergeSort, ktorého zložitosť je vždy $O(N \log N)$. Ako to funguje? Povedzme, že máme dve utriedené postupnosti a chceme z nich spraviť jednu. Predstavme si ich ako dve kôpky kariet číslami navrch. Pozrieme sa na obe kôpky a vyberieme z dvoch kariet, ktoré vidíme tú menšiu. To je najmenší prvok našej zlúčenej postupnosti. Takto pokračujeme, kým neprehádžeme všetky prvky pôvodných postupností do tej zlúčenej. Toto je základ MergeSortu, samotný merge, teda zlúčenie. Na začiatku sortenia teda rozdelíme naše pole prvkov na veľmi malé kúsočky a potom ich zlučujeme. Jednoprvkové postupnosti zlúčime do dvojprvkových, tie do štvorprvkových a tak ďalej. Veľa šťastia pri kódení;)

Listing programu:

```
program QuickSort;

const max = 50;
var n:integer;
    a: array[1..max] of integer;

  procedure nacitaj;
  var f:text;
      i:integer;
  begin
    assign(f,'artisti.in'); reset(f);
    readln(f,n);
    for i:= 1 to n do read(f,a[i]);
    close(f);
  end;

  procedure Qsort(i,j:integer);
  var pivot,lavy,pravy,pom: integer;
  begin
    pivot:=(a[i] + a[j]) div 2;
    lavy:=i;
    pravy:=j;
    repeat
      while a[lavy] < pivot do inc(lavy);
      while a[pravy] > pivot do dec(pravy);
      if lavy <= pravy then begin
        pom:=a[lavy]; a[lavy]:=a[pravy]; a[pravy]:=pom;
        inc(lavy);
        dec(pravy);
      end;
    until lavy > pravy;
    if i < pravy then Qsort(i,pravy);
    if lavy < j then Qsort(lavy,j);
```

```
  end;

  procedure vypis;
  var f: text;
      i: integer;
  begin
    assign(f,'artisti.out'); rewrite(f);
    for i:= 1 to n do write(f,a[i],' ');
    close(f);
  end;

Begin
  nacitaj;
  Qsort(1,n);
  vypis;
End.
```

2. Zábava na kolieskach...

opravoval Kubo
(max. 15 bodov)

Hodnotenie: Za optimálne riešenie v čase $O(MN)$ sa dalo získať 15 bodov, o čosi horšie riešenia v čase $O(MN^2)$ boli ocenené 12 bodmi, vyskytol sa aj Dijkstrov fanklub, ktorý vyfasoval 8 bodov. Backtrackisti dostávali 4 body. Tí, čo sa nedali takto jednoducho zaškatuľkovať tiež dostali body... priam niekoľko... Body som stŕhal za nedostatočný popis, zlý odhad zložitosti a tak... Tu sa pristavím ešte k jednej veci a tou bol okraj mapy. Uvedomujem si, že príklad vstupu v zadaní nebol najšťastnejší, ale v zadaní nebolo napísané, že mapa je obkolesená znakmi '#' a viacerým z vás mohlo prehľadávanie vyskočiť z mapy – za to som stŕhal bod.[1]

Teória: Aby ste si z tohoto príkladu niečo odniesli, pozrieme sa aj na trochu teórie za týmto príkladom. Vzorové riešenie sa zakladá na prehľadávaní do šírky z teórie grafov. Formálne je graf $G = (V, E)$ množina vrcholov V a množina hrán E (hrana je dvojica vrcholov, ktoré spája). Grafy nám rečou matematiky popisujú bežné situácie ako napr. mestá (vrcholy) pospájané cestami (hranami), routery pospájané linkami, priateľstvá (dva vrcholy sú spojené, ak sú príslušní dvaja ľudia priatelia)... a vôbec, je to silná zbraň a oplatí sa ju poznať.

Ako funguje prehľadávanie do šírky? Máme daný graf a nejaký začiatok z; budeme chcieť graf nejak systematicky prehľadať a "nájsť" každý vrchol, do ktorého sa vieme zo z dostať. Na začiatku máme iba vrchol z a vieme, že sa doňho vieme dostať na 0 krokov. Majme teraz množinu vrcholov M_k, do ktorých sa vieme dostať (najkratšou cestou) na k krokov, potom množinu vrcholov M_{k+1}, do ktorých sa vieme dostať na $k+1$ krokov zostrojíme jednoducho: sú to vrcholy, v ktorých sme ešte neboli a vieme sa do nich dostať na jeden krok z vrcholov, ktoré sú v M_k. Inak povedané: majme vrchol $v \in M_k$ teda taký, že najkratšia cesta zo z do v má dĺžku k, potom do všetkých susedov v, v ktorých sme ešte neboli, sa vieme dostať najkratšou cestou dĺžky $k+1$ (nemôže existovať kratšia cesta, pretože potom by sme museli toho suseda navštíviť už skôr). Môžeme si to predstaviť ako takú potopu. Dajme tomu, že v z zabudli vypnúť vodu—najskôr zatopí susedov z; potom všetkých susedov týchto susedov atď.

Ešte skôr ako si povieme, ako sa to napíše, si musíme povedať, čo je to **fronta**. Frontu ste už určite všetci zažili. Predstavte si rad ľudí, čo sa tlačia do bufetu; vložiť prvok do fronty

[1] Ten príklad určite nebol mienený, aby vás nachytal, ale opravujem to ja, takže tak; treba sa s tým zmieriť; poučenie: treba vždy dobre čítať zadania.

znamená, že prišiel ďalší človek a postavil sa do radu (a čaká, kým sa dostane na rad); vybrať prvok z fronty znamená, že sme človeka na začiatku vybavili (kúpil si čo chcel a odišiel). Ako takú frontu naprogramujeme? Ako jednoduché pole.[2] Budeme si pamätať, **kde** v poli je začiatok a kde koniec fronty. Vloženie prvku do fronty spravíme tak, že posunieme koniec o 1 a prvok tam uložíme; vybranie prvku urobíme tak, že vrátime ten na začiatku a začiatok posunieme o 1. Problém je, že takto by sme mohli vyskočiť z poľa—ideálne by bolo, keby to pole bolo "kruhové" (koniec by bol napojený na začiatok); to ale nie je žiadny problém: keď sa dostaneme na koniec poľa, jednoducho pokračujeme na začiatku poľa.

Konečne k implementácii: pre každý vrchol si budeme pamätať, ako ďaleko je od z. Na začiatku budú všetky vzdialenosti ∞, iba vzdialenosť z bude 0. Ďalej si budeme pamätať vrcholy, ktoré ešte treba prehľadať vo fronte (na začiatku tam vložíme iba z, kde "nevypli vodovodný kohútik"). Vždy vyberieme vrchol z fronty (odteraz bude zatopený); všetkých susedov, ktorých vzdialenosť je ∞ (teda ešte sme tam neboli) zaradníme do fronty (nech si počkajú na vytopenie) a upravíme im vzdialenosť—toto opakujeme, kým je čo prehľadávať (kým nevyprázdnime frontu), alebo kým nenatrafíme na cieľ.

Všimnite si, že každý vrchol iba raz zatopíme (keď je zatopený, už ho viac netopíme) 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.

Riešenie: Na Jančiho mapu sa môžeme tiež pozerať ako na graf, kde políčka sú vrcholy a hrany sú iba medzi takými vrcholmi, ktoré sú oba označené znakmi '.' (tieto hrany si, samozrejme, netreba pamätať—pozriem na mapu a vidím). Použijeme prehľadávanie do šírky, ktoré nájde najkratšiu cestu. Tú potom vyznačíme na mape tak, že ideme od domu naspäť po políčkach, ktorých vzdialenosť je vždy o jedna menšia. Časová aj pamäťová zložitosť bude lineárna od veľkosti vstupu, tj. $O(MN)$ (všimnite si ešte, že počet hrán je lineárny $< 4MN$).

Tipy/triky: Veľa programov, čo som dostal bolo dosť grcných, tak si prečítajte, ako sa to dá napísať jednoducho a celkom elegantne. Do pozornosti dávam ešte zopár trikov, ktoré sa dali použiť:

- Jeden z problémov, s ktorým ste sa mohli stretnúť, bol, aby vám prehľadávanie nevyskočilo z mapy. Jedna možnosť je testovať, či $1 \le x \le N$ a $1 \le y \le M$; podľa mňa elegantnejšia[3] možnosť je celú mapu na začiatku obkolesiť znakmi '#'.
- Pre každý vrchol bolo treba prehľadať jeho susedov; mnohí ste to riešili tak, že ste napísali kód pre jeden smer, sedemkrát skopírovali a pomenili konštanty. Toto je však veľmi častý zdroj chýb, ktoré sa blbo hľadajú, pretože zdroják sa navyše stane neprehľadným. Výborná alternatíva je napísať dva vnorené for-cykly:

```
for nx := x-1 to x+1 do
  for ny := y-1 to y+1 do dačo;
```

- Niekoľkí z vás pri písaní fronty spravili pole dosť malé (dokonca menšie ako $2(M+N)$[4]. Na frontu síce nepotrebujeme $M \cdot N$ políčok, ale keď si takú vytvoríme, nepokazí nám to pamäťovú zložitosť a navyše nemusíme testovať, či už sme nakonci, lebo až na koniec sa nedostaneme.
- `String` je pole symbolov (`char`); mapa sa dala čítať do stringov po riadkoch. Ohraničenie zľava a zprava sa potom dalo spraviť jednoducho ako `s := '#'+s+'#';`, hľadať sa potom dalo pomocou `pos` a.p.
- Posledná rada: keď píšete frontu, alebo aj hocičo podobné, je dobre si najprv rozmyslieť niečo, čo sa volá **invariant**—to je vlastnosť, ktorá sa počas behu programu nebude

[2] Priamočaro by sme pri vložení uložili prvok na koniec poľa a pri výbere vybrali prvý prvok a zvyšné posunuli—to je ale pomalé!

[3] avšak proti gustu žiaden dišputát

[4] a takýto stav sa dá dosiahnuť, ak sú na mape samé bodky a začína sa v strede

meniť. Ja som si povedal, že `fz` bude vždy ukazovať na prvý prvok vo fronte a `fk` bude vždy ukazovať za posledný prvok (potom bude fronta prázdna, keď $fz = fk$) a podľa toho som sa zariadil. Nehovorím: "dvakrát meraj, raz rež"—tu sa nič nereže, ale je lepšie najskôr si trochu rozmyslieť, ako sa bezhlavo pustiť do kódenia.

Listing programu:

```
const max_x = 90;
      max_y = 90;
   infinity = maxint;

var n, m, x, y, nx, ny, startx, starty, cielx, ciely : integer;
    mapa   : array [0..max_y+1, 0..max_x+1] of char;   { prva suradnica je y }
    d      : array [0..max_y+1, 0..max_x+1] of integer;   { vzdialenost od J }
    fx, fy : array [0..max_x*max_y] of integer; { fronta; x-ova a y-ova sur. }
    fz, fk : integer;                          { zaciatok a koniec fronty }

procedure vyznac (x, y : integer);     { rekurzivna procedura, ktora vyznaci }
var nx, ny : integer;                                  { najdenu cestu; }
begin
  for nx := x-1 to x+1 do
    for ny := y-1 to y+1 do                      { najdeme policko s mensou }
      if d[y, x]-1 = d[ny, nx] then begin                  { vzdialenostou }
        mapa[y, x] := '*';                               { oznacime ho }
        vyznac (nx, ny);                          { a oznacujeme od neho }
        exit;
      end;
end;

begin
  { Nacitame mapu }
  assign(input, 'ksp.in'); reset (input);
  readln (n, m);
  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 }
      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 }
  repeat
    x := fx[fz]; y := fy[fz]; inc (fz);      { vyberieme policko z fronty }
    for ny := y-1 to y+1 do                  { vsetkym susedom, v ktorych }
      for nx := x-1 to x+1 do                          { sme este neboli }
        if (d[ny, nx] = infinity) and (mapa[ny, nx] <> '#') then begin
          d[ny, nx] := d[y, x] + 1;          { vypocitame vzdialenost }
          fx[fk] := nx; fy[fk] := ny; inc (fk);    { a zaradime do fronty }
        end;
  until (fz = fk) or (d[ciely, cielx] < infinity);

  { Vyznac cestu spaet }
  if d[ciely, cielx] = infinity then writeln ('Skapes hladom, ty zmrd')
  else vyznac (cielx, ciely);

  { Vystup }
  for y := 1 to n do begin
    for x := 1 to m do write (mapa[y, x]);
    writeln;
  end;
end.
```

3. Zaľúbený čaj

opravoval MMx
(max. 15 bodov)

Hodnotenie: Vaše riešenia tohoto príkladu sa dajú podľa časovej a pamäťovej zložitosti rozdeliť do štyroch skupín. Zložitosti budem odhadovať v závislosti od M. Vzorové riešenie má lineárnu časovú a konštantnú pamäťovú zložitosť a bolo za neho 12 bodov. Väčšina z vás si nevšimla, že číslo L môžeme prevádzať do trojkovej sústavy aj od najvyšších mocnín a výsledky môžeme rovno vypisovať, čím sa zhoršila pamäťová zložitosť na lineárnu. Takéto riešenia boli hodnotené 10 bodmi. Niektorí ste používali rádovo M konštánt (mocniny trojky, súčty mocnín trojek), ktoré ste buď zakaždým znova počítali alebo ste ich predpočítavali nešikovne, čím ste si zhoršili časovú zložitosť na horšiu ako lineárnu, ale stále polynomiálnu a dostávali ste 8 bodov. Našlo sa aj niekoľko riešení, ktoré skúšali všetky možnosti, teda mali exponenciálnu zložitosť. Boli ohodnotené 6 bodmi. Zvyšné 3 body sa dali zarobiť kvalitným popisom programu a odhadom zložitosti.

Riešenie s lineárnou pamäťou: Najprv popíšem riešenie, ktoré potrebuje lineárnu pamäť a potom ho zlepším na vzorové. Označme si číslo, ktoré má byť na výstupe na i. mieste a_{m-i}. Keď Mirko pridá prvý deň hubu do príslušného čaju, rozdiel chutí bude a_{m-1}. Táto huba sa do druhého dňa rozmnoží na trojnásobok, teda druhý deň po pridaní huby bude rozdiel chutí $a_{m-1} * 3 + a_{m-2}$. Tento súčet sa bude podobne pre každý deň násobiť tromi a bude sa k nemu pripočítavať príslušné a, až posledný deň bude rozdiel chutí $L = a_{m-1} * 3^{m-1} + a_{m-2} * 3^{m-2} + \ldots + a_1 * 3 + a_0$ (1). Z tohoto vidíme, že L je súčet nejakých násobkov trojky a čísla a_0, preto musí platiť $a_0 = L \bmod 3$. Lenže mi potrebujeme, aby a_0 bolo -1, 0 alebo 1, ale $L \bmod 3$ môže byť 0, 1 alebo 2. Preto spravíme malú fintu: k obidvom stranám rovnice (1) pripočítame jednotku. Potom bude platiť $a_0 + 1 = (L + 1) \bmod 3$ a teda $a_0 = (L + 1) \bmod 3 - 1$. Teraz už bude a_0 nadobúdať správne hodnoty a teda máme výsledok pre posledný deň. Teraz celú rovnicu celočíselne vydelíme trojkou. Na pravej strane sa všetky exponenty trojky znížia o jedna a zvyšok po delení $a_0 + 1$ sa odreže. Z popisu, prečo platí (1) vidno, že na pravej strane teraz máme rozdiel chutí o deň skôr a môžeme na neho použiť podobnú úvahu ako

na posledný deň. Takýmto spôsobom zistíme všetky výsledky v opačnom poradí, priebežne si ich zapisujeme do poľa a na konci ich vypíšeme v opačnom poradí.

Vzorové riešenie: V predchádzajúcom riešení sme vlastne pripočítali k číslu L číslo, ktorého zápis v trojkovej sústave je M jednotiek (označme si ho k), súčet sme previedli do trojkovej sústavy a výsledné cifry sme vypisovali zmenšené o jedna. Tento postup sa však dá spraviť aj od najvyšších mocnín trojky. V takom prípade si nemusíme vypočítané cifry pamätať, ale môžeme ich rovno vypisovať, čo nám zlepší pamäťovú zložitosť. Ostáva nám vyriešiť, ako dopredu zistiť, či sa dá rozdiel chutí L dosiahnuť za M dní. Najväčší rozdiel, aký vieme za M dní dosiahnuť dostaneme tak, že každý deň pridáme hubu do Majkinho čaju. Hodnota tohoto čísla je k. Číslo k je vlastne súčet mocnín trojky od 0 po $M-1$, teda súčet geometrického radu, ktorého prvý člen je 1, kvocient je 3 a má M členov, teda $k = \frac{3^M}{2}$.

Odhady zložitostí: Časová zložitosť je $O(M)$, pretože najprv pomocou M krokov vypočítam 3^M a potom pre každý z M dní v konštantnom čase určím, do ktorého čaju treba pridať hubu. Pamäťová zložitosť je $O(1)$, pretože používam iba počet premenných, ktorý je nezávislý od veľkosti vstupu.

Listing programu:

```
var m,l,mocnina,i :integer;
begin
  readln(m,l);
  mocnina:=1;
  for i:=1 to m do mocnina:=mocnina*3;
  if l>mocnina div 2 then writeln('Prepáč Mirko, ale čaje už nestihneš pripraviť.') else
  begin
    inc(l,mocnina div 2);
    repeat
      mocnina:=mocnina div 3;
      if mocnina=1 then writeln(l-1)
        else write((l div mocnina)-1,' '); {na konci riadku nechcem mať medzeru}
      l:=l mod mocnina; {a chcem ho mať ukončený znakom konca riadka}
    until mocnina=1;
  end;
end.
```

opravovala Danka
(max. 15 bodov)

4. Zzzzzz alebo imatrikulácia

Tento príklad bol asi príliš ľahký, pretože nám prišlo veľa riešení a dosť z nich bolo správnych. Bohužiaľ bolo veľa z vašich riešení totožných, alebo veľmi podobných (napr. sa proti svetlu líšili len v tom, že jedna premenná mala iné meno). Čiže nečudujte sa, ak je vaše riešenie takmer správne a napriek tomu máte len tri, štyri body. KSP je súťaž jednotlivcov, preto body za „kolektívne“ riešenie sme rozdelili medzi jednotlivcov tvoriacich daný kolektív. V budúcnosti neodpisujte, riešenia píšte samostatne – inak uškodíte len sami sebe.

Za najlepšie riešenia v čase $O(N)$ máte 15 bodov. Ďalej za riešenia v čase $O(N \log N)$ 13 bodov, v čase $O(N^2)$ 11 bodov, $O(N^2 \log N)$ 9 bodov a za ostatné funkčné riešenia 7 bodov. Niektorí z vás úplne nepochopili zadanie, takže nabudúce čítajte poriadne a všetko. (Ak neviete, odkiaľ v tej časovej zložitosti vzniklo to $\log N$, príliš sa tým netrápte, nie je to také ľahké spočítať. Naše riešenie bude lepšie, takže sa pri tom nebudeme zastavovať.)

Čo vlastne bolo najlepšie riešenie? Predstavme si, ako by sme to asi robili ručne. Napísali by sme všetky mená na papieriky. Hodili by sme ich všetky do klobúka alebo vrecka a jeden z nich by sme vybrali. Pri losovaní ďalšieho by tam už nebolo N papierikov ale $N-1$.

Preto aj pri programovaní musíme na toto zmiznutie myslieť. Dá sa to celkom jednoducho. Dáme čísla (N) do klobúka, to znamená, že ich postupne naukladáme do poľa tak, že v $A[i]$ je číslo i. Vylosovanie nejakého čísla robíme pomocou funkcie $random(K)$, ktorá vráti náhodne jedno z celých čísiel od 0 po $(K-1)$.

Predstavme si, že už máme na prvých i miestach poľa A náhodne vylosované čísla. Teraz chceme vylosovať ďalšie a umiestniť ho na miesto $A[i+1]$. Na výber máme čísla na pozíciách $i+1$ až N. Zistíme si teda hodnotu $x = random(N-i)$. Tým sme akoby vylosovali číslo, ktoré sa momentálne nachádza na mieste $i+x+1$. Aby sme ho dali na správne miesto, vymeníme ho s číslom na mieste $i+1$.

A je to správne? Overíme si to nasledovne: Aká je pravdepodobnosť že ako prvé bude číslo K? Pravdepodobne už viete, že je to $\frac{1}{N}$ (počet všetkých správnych čísiel deleno počet všetkých možných čísiel). Potom pravdepodobnosť, že prvá dvojica bude K, L je $\frac{1}{N} \cdot \frac{1}{N-1}$. Pravdepodobnosť, že bude vylosovaná konkrétna N-tica je $\frac{1}{N!}$. Čo je pravdepodobnosť, ktorú sme chceli dosiahnuť.

Podľa nášho očakávania by mala byť zložitosť programu $O(N)$. To skutočne je. V programe máme (okrem načítavania a výpisu) len jeden cyklus, ktorý se vykoná $(N-1)$ krát. V ňom je jeden upravený random a výmena dvoch prvkov (tri priradenia). Obsah cyklu nám teda nezhoršuje zložitosť algoritmu, ktorá je teda naozaj $O(N)$.

Keďže používame jedno pole, pamäťová zložitosť je tiež $O(N)$.

Listing programu:

```
program Ksp_Z_4;
uses Crt;
const MaxN=100;
var A:array[1..MaxN]of integer; {pole poradia studentov}
    i,pom :integer;             {pomocne premenne}
    index :integer;             {index vylosovaneho cisla}
    N     :integer;             {pocet studentov}

BEGIN
  {nacitanie poctu studentov  }
  writeln('Zadaj cislo od 1..100');
  write('N=');
  readln(N);
  {inicializacia funkcie random}
  randomize;

  {do pola a nacitame vsetky cisla reprezentujuce studentov}
  for i:=1 to N do A[i]:=i;

  {vytvorenie poradia }
  {pricom N-ty krat uz nemusime cyklus vykonavat}
  {(zostalo len 1 cislo}
  for i:=1 to N-1 do
  begin
    {random(n-i+1) dava cislo z intervelu 0..(N-i)}
    {index je teda cislo z intervalu i..N}
    index:=random(N-i+1)+i;
    {A[index] dam na zaciatok zostavajucich cisiel}
    pom:=A[index];
    A[index]:=A[i];
    A[i]:=pom;
  end;

  {vypis}
  for i:=1 to N do write(A[i],' ');
```

```
  writeln;
END.
```

5. Zase škola

opravoval Rejdi
(max. 15 bodov)

V tomto príklade bolo úlohou „nakresliť“ čo najkrajší a prehľadný rozvrh. Väčšina z vás to zvládla a výsledné rozvrhy boli pekné. Možno sa to nezdalo, ale v tomto príklade bolo potrebné vyriešiť niekoľko problémov. Okrem toho, že ste mali priložiť rozvrh, ktorý váš program vygeneroval, tak aj vyriešiť ako väčšie množstvo informácií vtesnať na obrazovku terminálu.

Bodovanie. Ak ste zabudli priložiť programom vygenerovaný rozvrh, alebo ste priložili len ceruzkou stroho načrtnutý rozvrh, tak som strhával hneď 5 bodov, keďže táto podmienka bola explicitne napísaná v zadaní. Týchto 5 bodov bolo súčtom bodov za prehľadnosť rovrhu (2b) a bodov za riešenie prekrývania predmetov (3b). Ďalší bod som strhol ak ste skrátili názov predmetu odseknutím zvyšku názvu po niekoľkých znakoch a nedali ich do daľšieho riadku, prípadne ste sa ani o skrátenie nepostarali. Ako bolo na príklade vidieť, tak ste nemuseli vypisovať tú hodinu dňa, ktorá nebola počas celého týždňa obsadená nejakým predmetom. Čiže ak ste sa snažili vypisovať rozvrh podobný tomu v zadaní a vypisovali ste zbytočne aj prázdne hodiny (typicky okolo 19:00 už väčšina ľudí nemá školu), tak som strhol 1 bod. 3 body sa dali získať (stratiť :) za popis algoritmu, ktorý generuje rozvrh. Mali ste popísať hlavne ako ste riešili vyššie spomínané prípady. Posledných 5 bodov bolo za program. Ak ste spravili program, ktorý len nakreslil nevyplnenú tabuľku (našli sa aj takí), tak ste dostali 1 bod.

Väčšina riešení sa snažila vypisovať rozvrh, podobný tomu v zadaní. Našli sa aj prípady, kedy ste vypisovali dni vodorovne. Dalšie riešenia vypisovali predmety do troch stĺpcov (v zadaní bolo, že Monike sa môžu prekrývať až 3 predmety naraz, čo sa dalo pochopiť aj ako maximum súbežných predmetov). Ak ste už takto vypisovali, tak ste používali zástupné znaky a pod rozvrh ste vypísali legendu. Nevýhoda takéhoto riešenia spočíva v menšej prehľadnosti, keďže zo začiatku sa musíte vždy pozrieť do legendy, čo to vlastne za hodinu je, čím sa znižuje efektnosť. Posledný typ riešení sa mi zapáčil myšlienkou, kde ste namiesto rozvrhu vypísali pre každý deň v riadku: začiatok, koniec (resp. počet hodín), predmet, typ a miestnosť. Samozrejme v rámci dňa ste utriedili predmety podľa času, kedy začínali. Takto ste dosiahli aj prehľadnosť aj ste vyriešili prekrývanie predmetov, lebo bolo hneď vidiet, že prvý predmet končí neskôr ako ďalší (ďalšie) v poradí.

Vzorové riešenie. Popíšem tu, ako ste mohli riešiť jednotlivé podproblémy na to, aby ste získali plný počet bodov. Rozvrh mohol byť vykreslený aj na výšku aj na šírku, prípadne metódou zoznamu predmetov v danom dni. Použitím jedného z týchto typov rozvrhov ste získali body za prehľadnosť (v poslednom type rozvrhu, ste ešte museli zotriediť hodiny podľa začiatku). Ďalší problém bol v prekrytí jednotlivých hodín. Posledný typ rozvrhu to rieši už zo svojho princípu, kde sú vypísané všetky hodiny. Inak som uznával riešenia, ktoré vypísali viac rozvrhov, ktoré sa líšili obmenou hodín, ktoré sa prekrývali. Ideálna metóda by bola vypísať rozvrh, ktorý bude obsahovať tú kombináciu hodín, pri ktorej by sa Monika učila najviac hodín (keďže je tak snaživá :). Zvyšné hodiný, ktoré sa prekrývajú s tými v rozvrhu, by ste vypísali ako utriedený zoznam. Ďalší problém bol skracovanie názvov predmetov resp. ich vtesnanie do buniek. Riešenie spočíva buď rozdelení názvu na viacej riadkov (niektorí ste to robili aj s pomlčkami). Asi ideálne riešenie bolo skracovanie vždy najdlhšieho slova v názve až dovtedy, kým by sa celý názov nezmestil do políčka. Skracovanie by urezávalo konce slov vždy na najbližšiu spoluhlásku. Posledná, dôležitá vec (čo sa týka priamo riešenia) bola nevypisovanie tých hodín, ktoré sú počas celého týždňa voľné. Toto je pomerne zaujímavá

vec, keďže môže ušetriť dosť miesta, ktoré sa dá využiť na niečo iné. Samozrejme pridanie programu, popisu obsahujúceho vysvetlenie použitých optimalizácií a vygenerovaného rovrhu bolo rovnako dôležité.

Výsledková listina po 1. kole kategórie KSP-Z

	Meno a priezvisko	Škola	Trieda	11	12	13	14	15	Σ
1	Takáč Slavomír	Gym. Nové Zámky	3	15	15	15	15	12	72
2	Jerguš Ján	Gym. Alejová Košice	2	15	15	13	11	15	69
3	Rampášek Ladislav	Gym. Jura Hronca BA	2	15	15	12	13	12	67
4	Paulovicová Ivica	Gym. Jura Hronca BA	4	15	14	11	15	10	65
5	Danko Juraj	Gym. P. de Coubertina Piešťany	2	15	8	13	15	12	63
6	Králik Martin	Gym. Grösslingová BA	3	15	11	8	15	12	61
7	Okruhlica Adam	Gym. Jura Hronca BA	2	15	7	11	15	12	60
8	Bílka Ondřej	Gymnázium	3	15	15	12	15		57
9	Buštor Ivan	Gym. Jura Hronca BA	2	15	14	11	15		55
10	Pančík Andrej	Gym. Tajovského B. Bystrica	2	15	14		15	10	54
11	Herman Peter	Gym. Jura Hronca BA	2	15	14	9	15		53
12	Petrucha Michal	Gym. Metodova BA	0	15	15	10	12		52
12	Sudolský Michal	Gym. Tajovského B. Bystrica	2	15		12	11	14	52
14	Dadová Janka	Gym. Jura Hronca BA	4	10	4	12	11	13	50
14	Halamíček Radovan	Gym. Jura Hronca BA	3	15	13	9	2	11	50
14	Petruchová Zuzana	Gym. Grösslingová BA	3	15	15	13	7		50
17	Kekely Lukáš	Gym. Varšavská Žilina - Vlčince	1	7	7	11	9	10	44
18	Korcsok Peter	Gym. Mládežnícka Šahy	1	15	4	8	15		42
19	Sábo Jozef	Gym. Školská Spiš. Nová Ves	2	15	6	13	7		41
19	Tříska Martin	Gym. P. de Coubertina Piešťany	2	8	2	6	15	10	41
21	Kováčovský Tomáš	Gym. Jura Hronca BA	1	11	6	9	2	12	40
21	Pagáč Matej	Gym. Školská Spiš. Nová Ves	4	15	4	10	11		40
23	Dzurňák Tomáš	Gym. Školská Spiš. Nová Ves	3	15	14	9	1		39
24	Besenyi Libor	Gymnázium	4	8	4	6	8	12	38
24	Mikuláš Ondrej	Gym. Haličská Lučenec	2	15	4	12	7		38
24	Rajčan Šimon	Gym. Ľ. Štúra Zvolen	4	10		11	9	8	38
27	Žubrietovský Tomáš	Gym. Ľ. Štúra Zvolen	4	9		8	13	7	37
28	Smolka Tobiáš	Gym. L. Stöckela Bardejov	4	13		8	15		36
29	Blaho Pavol	Gym. Jura Hronca BA	1	13	10	11			34
30	Bálint Farkaš	Gym. maďarské Šahy	4	7	0	6	15	5	33
30	Cimba Martin	Gym. Stropkov	4	9		9	15		33
30	Lachata Adrián	Gym. Svidník	3	8		11	13	1	33
33	Brada Miroslav	Gym. Varšavská Žilina - Vlčince	1	9.5		5	13	5	32
34	Dillinger Viliam	Gym. Jura Hronca BA	3	13	3		15		31
35	Kováč Michal	Gym. Grösslingová BA	3	10		11	9		30
35	Oľhava Rastislav	Gym. Alejová Košice	3	7		12	11		30
37	Kysel Ondrej	Gym. Žiar nad Hronom	2	9		9	9		27
38	Kajan Peter	Gym. Jura Hronca BA	2	11	3	11			25
38	Szabó Tibor	Gym. Šuleka Komárno	2	8	1	8	8		25
40	Novák Ján	Gym. Ľudovíta Štúra Trenčín	3	15			9		24
41	Baumann Martin	Gym. Jura Hronca BA	2	10	13				23
41	Hegedušová Monika	Gym. P. Horova Michalovce	3	15		1	6	1	23
41	Morvay Peter	Gym. Jura Hronca BA	2	8			15		23
41	Takács Michal	Gym. Tajovského B. Bystrica	3	10			13		23
45	Beňo Fratišek	SPŠ Humenné	1	8			14		22
45	Betík Roman	SPŠ Levice	3	15	7				22
47	Nikodem Peter	Gym. Školská Spiš. Nová Ves	1	10			11		21
47	Vido Rudolf	Gym. Jura Hronca BA	2	6			15		21
47	Vlček Tomáš	Gym. Šk. Bratov BA	2	6		6	9		21
50	Nosko Slavomil	SPŠ Myjava	1	7			8	4	19

	Meno a priezvisko	Škola	Trieda	11	12	13	14	15	Σ
51	Kán Peter	Gym. Einsteinova BA	4	9			9		18
52	Hromuľák Matúš	Gym. Školská Spiš. Nová Ves	2	9			7		16
52	Svec Marcel	Gym. Jura Hronca BA	2	10	6				16
54	Hrebíček Martin	Gym. Ľudovíta Štúra Trenčín	2	6			9		15
54	Krupel Matej	Gym. Ľudovíta Štúra Trenčín	1	8			7		15
56	Hornak Michal	Gym. Alejová Košice	3	1			12		13
56	Jančár Michal	Gym. Alejová Košice	1	1			12		13
56	Renčo Peter	Gym. A.Sládkoviča B. Bystrica	1	8			5		13
59	Bodnár Jozef	Gym. Fiľakovo	4			12			12
59	Krištof Peter	Gym. Haličská Lučenec	4	10			2		12
59	Matušov Izidor	Gym. A.Sládkoviča B. Bystrica	1	7			5		12
62	Hanuska Norbert	Gym. Ľ. Štúra Zvolen	4	8		2			10
62	Mižáková Katarína	Gym. Alejová Košice	1	1		9			10
62	Schwarz Erik	Gym. Alejová Košice	1	1			9		10
62	Šrámek Martin	Gym. Tilgnerova BA	1	10					10
66	Jakubek Maroš	Gym. Ľudovíta Štúra Trenčín	2	8					8
66	Račko Tibor	Gym. Mládežnícka Šahy	4	8					8
66	Šuster Vladimír	Gym. Jura Hronca BA	2	8					8
69	Kovac Zolo	Gym. Alejová Košice	1	1			6		7
69	Matečný Alexander	Gym. Ľudovíta Štúra Trenčín	2				7		7
69	Petro Jakub	Gym. Alejová Košice	1	1			6		7
72	Melo Damián	Gym. Ul. 1. mája Trenčín	2	3			3		6
72	Melo Jakub	Gym. Ľudovíta Štúra Trenčín	1	3			3		6
72	Sukuba Ivan	Gym. Alejová Košice	1	2			4		6
72	Szönyi Iwan	Gym. Ľudovíta Štúra Trenčín	2	3			3		6
76	Andreansky Marek	Gym. Alejová Košice	3	1			4		5
76	Kovac Michal	Gym. Alejová Košice	3	1			4		5
76	Kubejova Lucia	Gym. Alejová Košice	3	1			4		5
76	Riganová Ivana	Gym. Alejová Košice	1	1			4		5
80	Mazal Tomáš	Gym. Jura Hronca BA	2	4					4
80	Travniček Bohuš	Gym. Jura Hronca BA	2	4					4
82	Kolesár Štefan	Gym. Alejová Košice	3	1.5			1		2
83	Beran Jakub	Gym. Alejová Košice	3	1					1
84	Beleš Lukáš	Gym. Čadca	4						0
84	Domány Dušan	Gym. P. Horova Michalovce	4						0