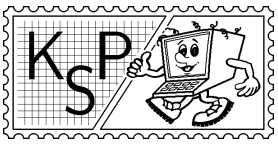


Korepondenčný seminár z programovania
XIX. ročník, 2001/2002

Katedra vyučovania informatiky FMFI UK,
Mlynská Dolina, 842 48 Bratislava

KSP finančne podporuje nadácia Open Society Fund Bratislava a
IUVENTA – zariadenie pre vožný čas detí, mláde«e a dospelých
MICROSTEP spol s.r.o.

Milí riešitelia,

jesenné sústredenie máme úspešne za sebou a vám sa (konečne?) dostávajú do rúk vzorové riešenia prvého kola. Tak si ich pozorne prečítajte, aby ste vedeli, ako sa tie úlohy mali riešiť. A ak sa dočítate až na koniec, nájdete tam... prekvapenie. (psst... je tam samozrejme výsledková listina... ale najskôr povinne prečítať vzorové riešenia!)

Po použití záchodu si umyte ruky.

KSPáci

1. O výsledkovej listine

opravoval Goober
(max. 15 bodov)

Pri opravovaní tohoto príkladu som bol vežmi príjemne prekvapený, lebo takmer vetky prídené rieenia boli správne. Bodovanie sa odvíjalo od kvality popisu (±**1 bod**), správneho odhadu zlo«itosti (**1 bod**), časovej náročnosti a celkovej elegancie rieenia. Za kvadratickú zlo«itos¸ bolo mo«né získa¸ **8 bodov**, za $O(N \log N)$ **11** a lineárne rieenie si vyslú«ilo **13 bodov**. Nesprávne rieenie dostalo nanajvý **3 body**.

A ako sa to dalo urobi¸ lineárne? Na základnú mylienku ste prili vetci – o umiestnení dru«stiev rozhodujú len maximálny a minimálny počet bodov, ktoré mohli jednotlivé krajiny získa¸. Majme teda dve polia – Min bude obsahova¸ minimálne mo«né body jednotlivých krajín, Max naopak maximálne, ktoré mohli dosiahnu¸. Z istých príčin by bolo dobré, keby tieto polia boli usporiadané. Min je usporiadané hneď, lebo také ho dostávame priamo na vstupe. Ako uká«eme neskôr, aj Max mo«no usporiada¸ v lineárnom čase. Preto odteraz budeme predpoklada¸, «e obe polia sú (zostupne) usporiadané.

Najlepie umiestnenie dru«stva I sa dá nájs¸ tak, «e zistíme, kožko krajín nemohlo predbehnú¸ (či«e ich minimálne mo«né body boli väčie ako maximálne body krajiny I). Toto hžadanie stačí spravi¸ úplne najjednoduchou metódou – prechádzame Min od začiatku a zastaneme, keď narazíme na ni«ie číslo, ako to, ktoré hžadáme. Otázkou je, v akom poradí treba jednotlivé krajiny hžada¸. Ak najprv pohžadáme krajinu s najvyím maximálnym počtom bodov (či«e tú, ktorej body sú v Max prvé), vieme, «e ďalie krajiny u« nemô«u dopadnú¸ lepie ako ona v najlepom prípade. To ale znamená, «e pri hžadaní ďalích krajín u« stačí zača¸ od miesta, kde sme predtým skončili. Keď teda budeme krajiny hžada¸ v poradí podža ich maximálnych bodov, je zrejmé, «e prejdeme Max aj Min najviac raz, a preto bude zlo«itos¸ tejto časti $O(N)$. Najhorie umiestnenie sa hžadá analogicky.

Ako usporiada¸ Max lineárne? Na horepopísaný postup väčina z vás prila. Problémy vak robilo usporiadanie poža Max. Klasické metódy ako Quicksort sa ukazujú ako príli pomalé (lebo chceme rieenie v $O(N)$). Za vzor si teda vezmime Mergesort, presnejie jeho spájaciu fázu. Keď si toti« krajiny vhodne rozdelíme do niekožkých skupín a v rámci ka«dej skupiny budú usporiadané, stačí u« tieto skupiny len spoji¸. Ako výhodné rozdelenie sa ukazuje rozdelenie podža toho, o kožkých ich sú¸a«iacich nemáme informácie (takto vznikne 5 skupín). V rámci ka«dej skupiny budú krajiny usporiadané podža maximálneho počtu bodov (lebo poradie v rámci skupiny je rovnaké ako poriadie na vstupe). Spájanie prebieha jednoducho – v ka«dej skupine vieme, «e najväčí prvok je ten prvý. Preto úplne najväčí prvok nájdeme

tak, «e porovnáme najväčie v jednotlivých skupinách. Odstránime ho a postup opakujeme. Vybratie jedného prvku je $O(1)$, a teda celé usporiadanie je $O(N)$. Pamä¸ová zlo«itos¸ celého algoritmu je zjavne $O(N)$.

Pár slov k implementácii: Ako si pozorný čitatež iste vimol, hžadanie maxima a minima funguje vežmi podobne a dá sa spoji¸ do jedného a teda namiesto dvoch utriedených polí budeme ma¸ len jedno. Ete pozoronejí čitatež si tie« asi vimol, «e my vlastne ka«dý údaj z tohoto usporiadaného poža pou«ijeme iba raz, a teda toto pole vôbec netreba ma¸ ulo«ené. Preto v ni«ieuvedenom programe polia Max a Min nenájdete. Namiesto toho, v«dy keď prezrieme najväčie prvky v skupinách, rovno zistený údaj spracujeme. Pri spájaní je ete dôle«ité, aby pri rovnosti bodov ili najprv maximá a potom minimá (premyslite si prečo). To je zabezpečené triviálne – keď prvýkrát narazíme na údaj o nejakej krajine, pova«ujeme ho za maximum, druhýkrát za minimum.

Listing programu:

```
const
 MAXN=20;

type
 tkrajina=
  record
   meno:string;
   best,worst:integer;       {worst/best su 0 ak ich este nepozname}
  end;
 tzaznam=
  record
   body,kto:integer;
  end;

var
 bodov,chyba,i,krajin,minbodov:integer;
 kr:array[1..MAXN] of tkrajina;           {udaje o krajinach}
 pom:array[-1..4,1..MAXN+1] of tzaznam;   {pomocne polia pri usporiadani
                                           -1 = minima, 0..4 maxima}
 index:array[-1..4] of integer;

procedure Pridaj(chyba,komu,kolko:integer;typ:boolean);
 var k:integer;
 begin
  if typ then k:=chyba else k:=-1;
  inc(index[k]);
  with pom[k,index[k]] do begin body:=kolko; kto:=komu; end;
 end;

function VrchB(kde:integer):integer;
 begin VrchB:=pom[kde,index[kde]].body; end;

function VrchK(kde:integer):integer;
 begin VrchK:=pom[kde,index[kde]].kto; end;

function Cmp(a,bodya,bodyb:integer):boolean;
 begin
```

```
  Cmp:=(bodya>bodyb) or                    {ak ma viac bodov}
      ((bodya=bodyb) and (kr[a].best=0));  {alebo rovnako, ale je to max}
 end;

procedure SpojARataj;
 var
  i,j,k,m:integer;
  maxim,minim:integer;
 begin
{ nastavime si zarazku pri hladani, aby sme nevybehli z pola }
  for i:=-1 to 4 do pom[i,index[i]+1].body:=-1;
  for i:=-1 to 4 do index[i]:=1;
  maxim:=0; minim:=0;

  for i:=1 to 2*krajin do
   begin
    k:=-1; m:=VrchB(k);   { minim bude vzdy dost }
    for j:=0 to 4 do
     if Cmp(VrchK(j),VrchB(j),m) then begin k:=j; m:=VrchB(j); end;

    with kr[VrchK(k)] do
     if best=0 then begin best:=minim+1; inc(maxim); end
     else           begin worst:=maxim;  inc(minim); end;
    inc(index[k]);
   end;
 end;

 begin
  read(krajin,minbodov); dec(minbodov);
  for i:=1 to krajin do
   begin
    read(bodov,chyba,kr[i].meno);
    Pridaj(chyba,i,bodov,FALSE);
    Pridaj(chyba,i,bodov+chyba*minbodov,TRUE);
   end;

  SpojARataj;
  for i:=1 to krajin do with kr[i] do
   writeln(meno,' ',best,'. - ',worst,'.');
 end.
```

2. O čarovnom strome

opravovala Danka
(max. 15 bodov)

Väčina rieení bola správna. Líili sa najmä spôsobom hžadania koreňa v inorder zápise. Najjednoduchie je prehžadáva¸ od začiatku celý inorder zápis, a« kým nenájdeme koreň. Týmto dostaneme algoritmus s časovou zlo«itostou $O(N^2)$. Vylepenie prináa prehžadávanie len tej časti inorder zápisu, ktorá zodpovedá práve spracovávanému podstromu. Týmto sa zlepí časová zlo«itos¸ v priemernom prípade (keď je strom vyvá«ený) na $O(N \log N)$, ale v najhorom prípade je zlo«itos¸ $O(N^2)$ (keď má ka«dý vrchol len žavú vetvu).

Vo vzorovom rieení vyu«ijeme to, «e vo vrcholoch sú čísla 1 a« N, a na začiatku si pre ka«dý vrchol zapamätáme jeho pozíciu v inorder zápise. Neskôr potom vieme zisti¸ pozíciu prvku v inorder zápise v kontantnom čase.

Výpis postorder zápisu robí rekurzívna procedúra postorder, ktorá dostane pre spracovávaný podstrom začiatok v preorder zápise, začiatok v inorder zápise a veľkos¸. Prvý prvok v preorder zápise je koreň stromu. Z jeho pozície v inorder zápise (máme ju ulo«enú v poli `kdeje[koreň]`) vieme urči¸ veľkos¸ ľavého a pravého podstromu. Potom vieme rozdeli¸ preorder zápis na ľavý a pravý podstrom. Keď poznáme preorder a inorder zápis podstromov, rekurzívne vypíeme najprv ľavý, potom pravý a nakoniec koreň.

Ka«dý prvok v preorder zápise spracovávame práve raz, spracovanie trvá kontantný čas, preto je časová zlo«itos¸ algoritmu $O(N)$.

Listing programu:

```
program O_carovnom_strome;
var
  preorder, kdeje: array[1..50] of Integer;
  i, n, x: Integer;

procedure postorder(zp, zi, velkost: Integer);
var vl, vp: Integer;
begin
  if velkost > 0 then
  begin
    vl := kdeje[preorder[zp]] - zi; {velkost laveho podstromu}
    vp := velkost - vl - 1; {velkost praveho podstromu}
    postorder(zp+1, zi, vl); {vypis lavy podstrom}
    postorder(zp+vl+1, zi+vl+1, vp); {vypis pravy podstrom}
    write(preorder[zp] , ' '); {vypis koren}
  end;
end;

begin
  n := 0;
  while not eoln do
  begin
    inc(n);
    read(preorder[n]);
  end;
  readln;
  for i := 1 to n do
  begin
    read(x);
    kdeje[x] := i;
  end;
  postorder(1, 1, n);
  writeln;
end.
```

3. Ondrejove zápalky

opravoval Martin
(max. 13 bodov)

Tento príklad bol vežmi jednoduchý, preto som pri opravovaní dbal na ka«dú drobnos˛, v ktorej ste mohli urobi˛ chybu. Za správne rieenie úlohy v čase $O(N)$, kde N je počet vetkých zápaliek, s kontantnou pamä˛ovou zlo«itos˛ou ste mohli získa˛ **10 bodov**, s lineárnou pamä˛ovou zlo«itos˛ou **8 bodov**. Za rôzne funkčné backtracky som dával maximálne **3 body** a za zlé, respektíve nedokončené rieenia sa dal pri vežkom ˛astí získa˛ najviac **jeden bod**. K tomu za popis váho rieenia a dôkaz správnosti ste mohli získa˛ od **mínus dvoch** a« po **3 body** a to tak, «e za ka«dú čas˛ popisu som vám dával plus alebo mínus. Za vysvetlenie váho algoritmu ste mohli získa˛ jedno plus, za dôkaz, kedy sa dá mnohouholník postavi˛, ste mohli získa˛ ďalie plus a za dôkaz, «e vami postavený mnohouholník bude naozaj mnohouholníkom, to znamená, «e «iadne zápalky na jeho obvode sa nebudú navzájom pretína˛, ste mohli získa˛ posledné plus. Za tri plus sa vám k celkovému počtu bodov pripočítali 3 body, za dve plus 2 body, za jedno plus 0 bodov a ak sa vám nepodarilo získa˛ «iadne plus, tak ste stratili 2 body. Za chýbajúci, alebo nesprávny odhad časovej alebo pamä˛ovej zlo«itosti váho programu ste mohli strati˛ jeden bod a za ka«dú záva«nú chybu v programe, alebo ka«dý neoetrený okrajový prípad ste mohli strati˛ ďalie body.

Prejdime teda k samotnému rieeniu:

Tvrdenie: Ak máme A modrých, B zelených, C červených a D ru«ových zápaliek, tak mnohouholník sa dá z nich postavi˛ práve vtedy, keď A, B, C a D sú párne a aspoň dve z nich sú nenulové.

Dôkaz: Najprv doká«eme prvú implikáciu: Ak sa mnohouholník postavi˛ dá, potom A, B, C a D sú párne a aspoň dva z nich sú nenulové.

Zaveďme si súradnicovú sústavu tak, aby y-ová os bola rovnobe«ná s modrou zápalkou a dĺ«ka zápalky bola 1. Ka«dú zápalku mnohouholníka reprezentujme ako vektor. Modré zápalky budú reprezentované vektormi $(0,1)$ a $(0,-1)$, zelené $(1,0)$ a $(-1,0)$, červené (q,q) a $(-q,-q)$ a ru«ové $(q,-q)$ a $(-q,q)$; $q=\sqrt{2}/2$. Potom nutne súčet vektorov vetkých zápaliek bude rovný nulovému vektoru:

$$\begin{aligned}(0,0) &= a_1(0,1)+a_2(0,-1)+b_1(1,0)+b_2(-1,0)+c_1(q,q)+c_2(-q,-q)+\\ &\quad +d_1(q,-q)+d_2(-q,q)\\ A &= a_1+a_2\\ B &= b_1+b_2\\ C &= c_1+c_2\\ D &= d_1+d_2\end{aligned}$$

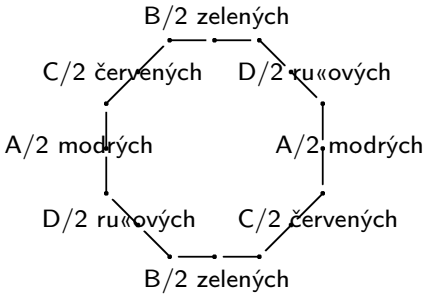
Pre x-ovú zlo«ku platí: $b_1-b_2+q.(c_1-c_2+d_1-d_2)=0$, pre y-ovú zlo«ku: $a_1-a_2+q.(c_1-c_2-d_1+d_2)=0$. Číslo q je iracionálne, či«e «iaden jeho racionálny násobok (okrem nuly) nebude racionálny, potom ale nutne platí: $a_1-a_2=0 \Rightarrow A=2.a_1 \Rightarrow A$ je párne; $b_1-b_2=0 \Rightarrow B=2.b_1 \Rightarrow B$ je párne; $c_1-c_2=d_2-d_1 \wedge c_1-c_2=d_1-d_2 \Rightarrow c_1=c_2, d_1=d_2 \Rightarrow C=2.c_1, D=2.d_1 \Rightarrow C$ aj D sú párne.

Zostáva nám ete dokáza˛, «e ak sa mnohouholník postavi˛ dá, tak aspoň dve z A, B, C alebo D sú nenulové. Doká«eme to sporom. Predpokladajme, «e máme zápalky len jedného (resp. «iadneho) druhu (z ostatných druhov máme po nula zápaliek), potom vak mô«eme z nich postavi˛ len úsečku (resp. nič), čo vak nie je mnohouholník. Doli sme k sporu, či«e pôvodné tvrdenie musí plati˛.

Teraz doká«eme aj opačnú implikáciu: Ak A, B, C a D sú párne a aspoň dva z nich sú nenulové, potom sa mnohouholník dá postavi˛.

Polo«me zápalky v poradí $\frac{A}{2}$ modrých, $\frac{D}{2}$ ru«ových, $\frac{B}{2}$ zelených, $\frac{C}{2}$ červených a opačne orientovaných $\frac{A}{2}$ modrých, $\frac{D}{2}$ ru«ových, $\frac{B}{2}$ zelených a $\frac{C}{2}$ červených zápaliek. Takto vzniknutý

útvar je uzavretý: $\frac{A}{2}(0,1)+\frac{D}{2}(q,-q)+\frac{B}{2}(1,0)+\frac{C}{2}(q,q)+\frac{A}{2}(0,-1)+\frac{D}{2}(-q,q)+\frac{B}{2}(-1,0)+\frac{C}{2}(-q,-q)=\frac{A}{2}((0,1)+(0,-1)+\frac{B}{2}((1,0)+(-1,0))+\frac{C}{2}((q,q)+(-q,-q))+\frac{D}{2}((q,-q)+(-q,q))=(0,0)$ a nutne konvexný (8 krát meníme smer o 45° „dožava“) mnohouholník (osem, es˛ alebo tvoruholník).



V samotnom programe vyu«itím práve dokázaného tvrdenia zistíme, či sa zo zadaných zápaliek dá mnohouholník postavi˛ a ak áno, podža postupu opísaného v druhej časti dôkazu mnohouholník vypíeme. Časová zlo«itos˛ algoritmu je $O(N)$, kde $N=A+B+C+D$, hoci test zápaliek, či sa z nich dá posklada˛ mnohouholník, prebehne v kontantnom čase; výpis samotného mnohouholníka trvá $O(N)$. Pamä˛ová zlo«itos˛ algoritmu je kontantná.

Listing programu:

```
#include <stdio.h>

int main(void)
{
  char *f[4] = {"modra, ", "ruzova, ", "zelena, ", "cervena, "};
  int i, j, z[4];
  scanf("%d %d %d %d", &z[0], &z[2], &z[3], &z[1]);

  if (z[0]%2 || z[1]%2 || z[2]%2 || z[3]%2 || !z[0]+!z[1]+!z[2]+!z[3] > 2)
  {
    printf("Z tychto zapaliek sa mnohouholnik postavit neda.");
  }
  else
  {
    for (i=0; i<8; i++) for (j=0; j<z[i%4]/2; j++) printf(f[i%4]);
  }

  printf("\n");
  return 0;
}
```

4. O menovej reforme

opravoval Poko
(max. 17 bodov)

U« som sa teil na vožný víkend, keď tu vám príde neskutočne veža rieení tvrtého príkladu. S povzbudením *„Tak a« ten ďalí...“* som teda začal opravova˛ :-)

Spolu ste poslali niečo cez 40 rieení, čo síce nie je málo, ale ani najviac. Niekožko z nich bolo chybných - buď neodhalili vetky nevhodné sady mincí, alebo nejakú vhodnú vyhlásili za nevhodnú. Tie ostatné sa líili – v čom? No predsa v zlo«itosti. A ete kde-tu chýbal popis, ako funguje vá algoritmus, zdôvodnenie niektorých vaich tvrdení (vežmi často ste nezdôvodnili, po ktorú hodnotu stačí testova˛ sumy) a „nezabudli“ ste ani na chyby vo vaich programoch :-(Tie nabudúce radej vynechajte!

Vzorové rieenie: Je zrejmé, «e sada, ktorá obsahuje jednu mincu, je vhodná. Preto ďalej budem predpoklada˛, «e sada sa skladá aspoň z dvoch mincí. Optimálnym platením budem ďalej chápa˛ platenie minimálnym počtom mincí. Nech K je počet mincí v testovanej sade, a_i ich hodnoty a nech pre $1 \le i < j \le K$ platí $a_i < a_j$. Je zrejmé, «e sumu $S=1$ zaplatíme optimálne pomocou mince s hodnotou 1, no takto to bude zaplatené zároveň aj kiribatským

spôsobom. Predpokladajme teraz, «e sumy 1 a« nejaké N ($N \geq 1$) kiribatským spôsobom zaplatíme zároveň aj optimálne. Poďme zisti ̨, kožkými mincami zaplatíme sumu $N+1$ optimálne a či pri platení kiribatským spôsobom nepou«ijeme viac mincí.

Pri zis ̨ovaní vyu«ijeme metódu dynamického programovania. Postupne budeme vypĺňa ̨ tabužku B (B ako Best). V tabužke si do i-teho políčka zapíeme, kožkými mincami sa suma i dá zaplati ̨ optimálne. Vieme, «e $B[0] = 0$, lebo na zaplatenie sumy $S = 0$ nepotrebujeme «iadnu mincu. Predpokladajme, «e máme vyplnené aj políčka 1 a« N a budeme sa sna«i ̨ vyplni ̨ políčko $N+1$. Určite bude ka«dé políčko 1 a« $N+1$ nenulové. Inak by sme vedeli sumu $S > 0$ zaplati ̨ pomocou 0 mincí, čo je jasný spor. Tým pádom sumu $S > 0$ platíme aspoň jednou mincou, a preto pre ka«dé $S > 0$ existuje minca, ktorej hodnota nebude väčia ako S a zároveň bude maximálna mo«ná. Nech pre sumu $S = N+1$ je to minca a_j. To znamená, «e pri platení S pou«ijeme len mince $a_1, \ldots, a_j$. Postupne si pre vetky i; $1 \leq i \leq j$ zistíme, pomocou kožkých mincí by sme sumu $S = N+1$ vedeli zaplati ̨, keby sme pri platení sumy $N+1$ pou«ili mincu a_i. To vieme pomocou $B[N+1-a_i]+1$ mincí. Potom

$$B[N+1] = \min_{1 \leq i \leq j} \{B[N+1-a_i]\} + 1$$

Hodnota $B[N+1]$ nemô«e by ̨ väčia, lebo by suma $S = N+1$ nebola zaplatená optimálne, nakožko pomocou $B[N+1]$ mincí ju isto vieme zaplati ̨. Takisto nemô«e by ̨ ani menia, lebo potom by musela existova ̨ minca a_l, ktorú keby sme pou«ili pri platení, a teda by sme sumu $N+1-a_i$ vedeli zaplati ̨ na menej ako $B[n+1-a_i]$ mincí, čo je spor. Tak«e pomocou $B[N+1]$ mincí sa suma $N+1$ zaplatí optimálne.

Pri platení kiribatským spôsobom by sme isto pou«ili mincu a_j, preto na zistenie, či je suma $S = N+1$ zaplatená kiribatským spôsobom optimálne, stačí porovna ̨ hodnoty $B[N+1]$ a $B[N+1-a_j]+1$. Ak sa rovnajú, tak je suma $N+1$ zaplatená kiribatským spôsobom zároveň optimálne. V opačnom prípade sme nali sumu, ktorá sa dá zaplati ̨ menej mincami ako kiribatským spôsobom.

Tabužku nám stačí vyplni ̨ po políčko $B[a_K + a_{K-1} - 1]$ vrátane[1]. Ak nenájdeme sumu $S < a_K + a_{K-1}$, tak mô«eme vyhlási ̨ sadu za vhodnú. Prečo, to si povieme a« po reklame.

Časová zlo«itos ̨ tohoto algoritmu je $O(Ka_K)$, pre ka«dú sumu 1 a« $a_K + a_{K-1} - 1$ vieme urči ̨ počet mincí potrebných na optimálne zaplatenie v čase $O(K)$. Pamä ̨ová zlo«itos ̨ je $O(K + a_K)$, ale keď«e v«dy platí $K \leq a_K$, lepí odhad je $O(a_K)$.

Veta: Ak vetky sumy 1 a« $a_K + a_{K-1} - 1$ zaplatia Kiribatčania optimálne, potom je testovaná sada mincí pre Kiribati vhodná.[2]

Dôkaz: Ten prevedieme sporom. Predpokladajme, «e platí negácia tvrdenia z vety, teda: „Vetky sumy 1 a« $a_K + a_{K-1} - 1$ zaplatia Kiribatčania optimálne a zároveň sada nie je pre Kiribati vhodná.“ Označme $K(N)$ počet mincí, akým by Kiribatčania zaplatili sumu N, a $O(N)$ počet mincí potrebný na optimálne zaplatenie sumy N. Nakoniec označme S najmeniu sumu, ktorú by Kiribatčania neplatili optimálne. Taká suma podža predpokladu («e sada nie je vhodná) určite existuje. Teda platí $K(1) = O(1)$, $K(2) = O(2)$, …, $K(S-1) = O(S-1)$, $K(S) > O(S)$, kde $S \geq a_K + a_{K-1}$. Suma $S = a_K + a_{K-1}$ sa ale nedá zaplati ̨ lepie ako dvoma mincami (rozmyslite si prečo!), a pomocou dvoch mincí ju vieme zaplati ̨, preto u« túto sumu nemusíme uva«ova ̨. Teda stačí uva«ova ̨ $S > a_K + a_{K-1}$.

Vieme, «e pre $N \geq a_K$ platí $K(N) = K(N-a_K)+1$, pričom $K(0) = 0$, lebo pre $N \geq a_K$ Kiribatčania určite začnú plati ̨ mincou s hodnotou a_K. A nakožko $S > a_K + a_{K-1} > a_K$, pre sumu S potom platí $O(S) < K(S) = K(S-a_K)+1 = O(S-a_K)+1$. Keby sme pri optimálnom platení sumy S pou«ili mincu a_K, tak $S - a_K$ by sme vedeli zaplati ̨ na $O(S)-1$

[1] Niektorí z vás správne zistili, «e si stačí pamäta ̨ posledných a_K hodnôt a tie cyklicky prepisova ̨. Pri správnom určení hranice, pokiaž treba sumy testova ̨, to vak nebolo nutné a pamä ̨ovú zlo«itos ̨ to nepokazilo.

[2] Tu platí dokonca ekvivalencia, ale implikácia opačným smerom („Ak je sada vhodná, tak Kiribatčania zaplatia sumy 1 a« $a_K + a_{K-1} - 1$ optimálne.“) platí triviálne.

mincí, a tým pádom by sa suma S zaplatila kiribatským spôsobom zároveň optimálne, čo by bol spor. Z čoho vyplýva, «e mincu a_k nepou«ijeme, a teda nutne platíme aspoň jednou mincou s menou hodnotou ako a_K. Označme ju a_i, $1 \leq i < K$.

Ďalej si treba uvedomi¸ vežmi dôle«itú vec, a síce, «e keď vieme, «e na optimálne zaplatenie sumy S nutne musíme pou«i¸ nejaké mince, tak je jedno, v akom poradí ich budeme plati¸. Inými slovami, ak vieme, «e na zaplatenie sumy S optimálnym spôsobom nutne pou«ijeme mince a_i a a_j, tak $O(S - a_i) = O(S - a_j)$. To vyplýva zo spôsobu, akým počítame hodnotu $O(S)$ (tento spôsob je popísaný a dokázaný vyie).

Ostáva nám ete zaplati¸ suma $S - a_i$. Ale, nakožko $0 < S - a_i < S$, sumu $S - a_i$ mô«eme smelo zaplati¸ kiribatským spôsobom, lebo vieme, «e sumy menie ako S sú kiribatským spôsobom zaplatené zároveň aj optimálne. Keď«e $a_i < a_k$, tak $S - a_i \geq S - a_{K-1} > a_K$. To ale znamená, «e pri platení $S - a_i$ kiribatským spôsobom (a teda aj optimálnym) pou«ijeme mincu a_K, čo je spor. Čím je naa veta dokázaná.

A nakoniec, za čo ste mohli získa¸ body?

- za rieenie pracujúce na rovnakom alebo podobnom princípe ako vzorové rieenie – maximálne **17 bodov**
- za rieenie, ktoré na vypočítanie počtu mincí pre optimálne platenie skúalo rozklada¸ sumu na vetky mo«ne súčty – maximálne **16 bodov**
- za exponenciálne rieenie – maximálne **8 bodov**
- za nesprávne rieenie – maximálne **3 body**

Body ste mohli strati¸ za chýbajúci zdrojový kód (5 bodov), za záva«nejie chyby v implementácii (najviac 2 body), za chýbajúci popis a/alebo zdôvodnenie správnosti váho rieenia (najviac 3 body), za zlý odhad časovej a pamä¸ovej zlo«itosti (najviac 1 bod) a za zhorenú časovú a/alebo pamä¸ovú zlo«itos¸ (napr. kvôli väčej hranici, po ktorú testujete) (najviac 5 bodov).

Listing programu:

```
Program O_Menovej_Reforme;

Const Max = 100;

Var A: array [1..Max+1] of integer; {mince A[1]=1 < A[2] < ... < A[K] <=100}
    B: array [0..2*Max] of integer; {pocet minci pre opt. platenie sumy i}
    K, Kiribatcan, i, j, ptr: integer;

begin
  readln(K);                          {pocet minci}
  for i:=1 to K do read(A[i]);        {mince}

  A[K+1]:=A[K]*3;
  B[0]:=0; ptr:=1;                  {A[ptr] je hodnota najvacsej mince <= i}
  for i:=1 to 2*A[K]-1 do           {testujeme sumy 1..2*A[K]-1, co zrejme}
                                    {        staci a zlozitost to nepokazi}
  begin
    if i=A[ptr+1] then Inc(ptr);
    Kiribatcan:=B[i-A[ptr]]+1;      {Kiribatcan pouziva najvacsiu mincu, }
                                    { akou moze platit}
    B[i]:=Kiribatcan;               {Kiribatcan to nemoze zaplatit lepsie }
                                    { ako optimalne}
    for j:=1 to ptr-1 do
```

```
      if B[i]>B[i-A[j]]+1 then B[i]:=B[i-A[j]]+1;
    if B[i]<Kiribatcan then
    begin
      writeln('Nie: ',i); exit;
    end;
  end;
  writeln('Sada je pre Kiribati vhodna.');
end.
```

5. O čiernych krabičkách

opravovala Meri
(max. 15 bodov)

Tento príklad bol pomerne jednoduchý, rieila ho väčina z vás. A poväčine aj dobre. Tí, ktorí ho mali úplne dokonale, v čase $O(N)$, mohli dosta¸ **15 bodov**. Trochu horie rieenia v čase $O(N^2)$ mohli ma¸ maximálne **7 bodíkov**. Za nefunkčné rieenie ste mohli ma¸ najviac **2 body**. Za nepresný odhad časovej zlo«itosti alebo úplne chýbajúci popis programu ste mohli strati¸ **2–3 body**.

Vzorové rieenie vyu«íva dve čierne krabičky, jednu vkladaciu a jednu vyberaciu. Keď potrebujem do fronty vlo«i¸ prvok, vlo«ím ho, ako inak, do vkladacej krabičky. Tak«e, keď príde prvá «iados¸ o vrátenie prvku, budem ma¸ k dispozícii prázdnu vyberaciu krabičku a vkladaciu krabičku s nejakými prvkami. Vtedy presuniem vetky prvky z vkladacej krabičky do vyberacej. Vo vykladacej krabičke teraz budú vetky v opačnom poradí, ako boli vkladané a teda pri ka«dej «iadosti o vrátenie prvku mo«em jednoducho vybera¸ prvky z vyberacej krabičky. Keď sa vetky minú, jednoducho opä¸ presuniem prvky z vkladacej krabičky do vyberacej a je.

e to má zlo«itos¸ $O(N)$? Nu«, povedzme, «e potrebujeme vlo«i¸ a potom vybra¸ nejakých N prvkov. Nevieme, v akom poradí ich budeme vklada¸ a vybera¸. Zato vieme, «e s ka«dým prvkom budeme robi¸ práve tri operácie. Raz ho vlo«íme do vkladacej čiernej krabičky, raz ho presunieme do vyberacej krabičky a raz ho z nej vyberieme. Celkovo teda vykonáme $O(N)$ operácií.

Listing programu:

```
program O_ciernych_krabickách;

uses crt,krabicky;

const
  vkladacia=1;
  vyberacia=2;

procedure Put(val:integer);
begin
  Push(vkladacia,val);
end;

function Get:integer;
begin
  if (Empty(vyberacia)) then
    while (not Empty(vkladacia)) do Push(vyberacia,Pop(vkladacia));
  Get:=Pop(vyberacia);
end;
```

```
function Emp:boolean;
begin
  Emp:=Empty(vkladacia) and Empty(vyberacia);
end;

begin
end.
```

Výsledková listina po 1. sérii kategórie KSP

	Meno a priezvisko	kola	Trieda	11	12	13	14	15	Σ
1.	Peter Bella	Gym. Jura Hronca BA	4	15	15	13	15	15	73
2.	Pavol Juhos	Gym. Grsslingov BA	4	13	15	12	16	15	71
	Jn Katreni	Gym. kolsk Spi. Nov Ves	4	13	15	13	15	15	71
4.	Michal MalÀ	Gym. iar nad Hronom		13	15	10	17	15	70
5.	Pavol Mravec	Gym. . tra Modra	4	14	15	13	10	15	67
	Vladimr RepiskÀ	Gym. iar nad Hronom	4	13	11	12	16	15	67
7.	Radovan Bauer	Gym. Potov Koice	4	15	15	13	17	6	66
8.	Jakub Teke	Gym. Jura Hronca BA	2	9	11	11	16	14	61
9.	Peter TruchlÀ	Gym. Provsk Nitra	4	15	15	1	14	15	60
10.	Jaroslav Klma	Gym. Jura Hronca BA	3	15	15	13		15	58
	Jozef VeselÀ	Gym. Golianova Nitra	3	15	15	13		15	58
12.	Mat Harvan	Gym. Jura Hronca BA	4	13	10	12	6	15	56
	Oto Vozr	Gym. Palackho BA	3	15	11	13	2	15	56
	Tom Sghy	SPE Nov Zmky	4	12	11	9	9	15	56
15.	Michal Miku	Gym. Jura Hronca BA	4	13	11	9	15	6	54
	Tom Vrbel	Gym. Mal Hora Martin	4	12	15	11	15	1	54
17.	Juraj Sloboda			9	10	7	14	13	53
18.	Pavol Mller	Gym. Grsslingov BA	4	13	11	12		15	51
	Peter Glaus	Gym. Jura Hronca BA	3	13	11	12		15	51
	Jn Mazk	Gym. Potov Koice	4	13	7	13	11	7	51
	Monika Steinov	Gym. Einsteinova BA	4	13	14	8	2	14	51
	Miroslav Cicko		1	10	11	12	5	13	51
23.	Peter Libi	Gym. udovta tra Trenn	3	8	11	10	6	14	49
24.	Marek Ludha	Gym. Tajovskho B. Bystrica	2	13	10	12		13	48
	Tom Dzetkuli	Gym. P. Horova Michalovce	4	12	11	12		13	48
	Ivan omlo	Gym. Mlde«ncka ahy	2	9	10	10	6	13	48
27.	Anton tefanek	Gym. Jura Hronca BA	2	8	9	11	11	6	45
	Marek TesaŠ	Gym. Halisk Luenec	4	13		12	14	6	45
29.	Jozef Tvaro«ek	Gym. Jura Hronca BA	4		15	12	17		44
30.	Michal Teke	Gym. Kontantnova Preov	4	13		13	1	15	42
31.	Ras¸o rmek	Gym. Tilgnerova BA	4	9	8	11	6	6	40
	Matej Max	Gym. Alejov Koice	4	8	11	5	11	5	40
33.	Peter Dzurjanin	Gym. Grsslingov BA	4	10	15	7		7	39
34.	Tom Zhorec	Gym. Jura Hronca BA	4	15	11	10		1	37
35.	Lucia Tiererov	Gym. Jura Hronca BA	4	9	9	11	2	5	36
36.	Anna Hanulov	Gym. Jura Hronca BA	2	9	11			15	35
	Juliana Lipkov	Gym. Jura Hronca BA	3		11	12	5	7	35
38.	Radovan Berta	Gym. Daxnera V.n. Topou	3	9	11		1	13	34
39.	Peter Macko	Gym. LiptovskÀ Hrdok	1	7	2	9	6	6	30
40.	Juraj Andris	Gym. Sere	4	7		7	10	5	29
41.	Jn Palenr	Gym. Mal Hora Martin	2	13			14	1	28
42.	Pavol Rusnk	Gym. Kontantnova Preov	4	10		10	1	6	27
43.	Martin Choma	Gym. Star ubova	3	3	12	8		1	24
44.	Marek Hanes		1	7	11	0	4	1	23
45.	Oliver Jank	Gym. L. Stckela Bardejov	2	9			11	1	21
	Marek Tomacha	Gym. sv. Urule BA	4	10		10		1	21
47.	Ladislav Vadkerti	SPE Pie¸any		9	11				20
48.	Jana ikov	Evanjelick gym. B. Bystrica	4	10		2		7	19
	Marek Cifra	Gym. A. Merici Trnava	2	10	8			1	19
	Jn VeselÀ	Gym. Golianova Nitra	2	7		7	5		19

	Meno a priezvisko	kola	Trieda	11	12	13	14	15	Σ
51.	Marin Schmotzer	Gym. Horvtha BA	2	10			2	5	17
	Peter Grekovi	Gym. Svidnk	1	9	8				17
53.	Milan Burda	Gym. Golianova Nitra	2	2	3	7	2	1	15
54.	Peter KopanskÀ	Gym. Krompachy	4	8				5	13
	Peter ufliarsky		2	9	4				13
56.	tefan KonenÀ	Gym. Jura Hronca BA	3	3		1	8		12
	Michal erea	Gym. Nov Zmky	2	9	2			1	12
58.	Radovan ulk	Gym. Golianova Nitra	2	9		0	1	1	11
	Ladislav HodermarskÀ	SPE Star Tur	4	10				1	11
60.	ubo Bednrik	Gym. udovta tra Trenn	4	9					9
61.	Filip Karas	Gym. Golianova Nitra	2	2	3	0	2	1	8
62.	Miroslav Hudk	Gym. Golianova Nitra	4					1	1
	Jn Hutr	SPE Pie¸any	3					1	1