PRESEK
List za mlade matematike, fizike, astronome in računalnikarje

ISSN 0351-6652
Letnik **26** (1998/1999)
Številka 6
Stran 342

Matja Lokar:

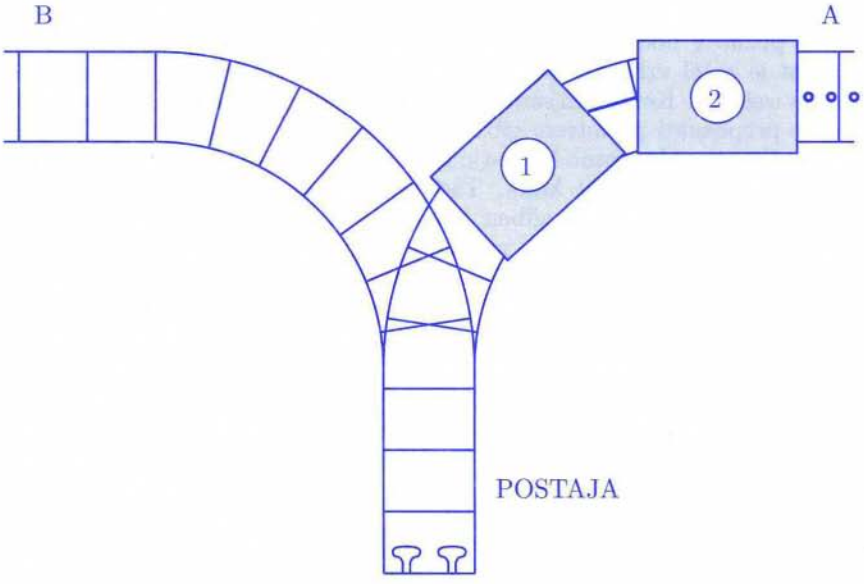
TEŽAVE POSTAJNEGA NAČELNIKA

Ključne besede: računalništvo, kombinatorika, permutacije, razporeditve vagonov, računalniški programi.

Elektronska verzija: http://www.presek.si/26/1384-Lokar.pdf

TEŽAVE POSTAJNEGA NAČELNIKA

V mestu PopPush je znamenita železniška postaja, zgrajena še v prejšnjem stoletju. Žal so bila sredstva za gradnjo takrat zelo omejena. Zato je postaja na slepem tiru, do katerega vodi tir iz smeri A, iz njega pa izhaja tir v smeri B (glej sliko).



Lokalna tradicija je, da vsak vlak, ki pride iz smeri A, nadaljuje pot v smeri B z na določen način preurejenim vrstnim redom vagonov. Predpostavi, da ima vlak, ki prihaja iz smeri A, $N \leq 1000$ vagonov, oštevilčenih naraščajoče: $1, 2, \ldots, N$. Postajni načelnik mora vedeti, ali je vagone, ki prihajajo iz smeri A, možno preurediti tako, da bo njihov vrstni red v smeri B enak $a_1, a_2, \ldots, a_N$. Enega ali več vagonov lahko z vlaka odpnejo in zapeljejo na postajo, od tam pa na izhodni tir v smeri B. V vsakem trenutku je na postaji lahko poljubno mnogo vagonov. Ko vagon pride na postajo, se ne more več vrniti na tir v smeri A. Prav tako se vagon ne more vrniti nazaj na postajo, ko jo enkrat zapusti v smeri B.

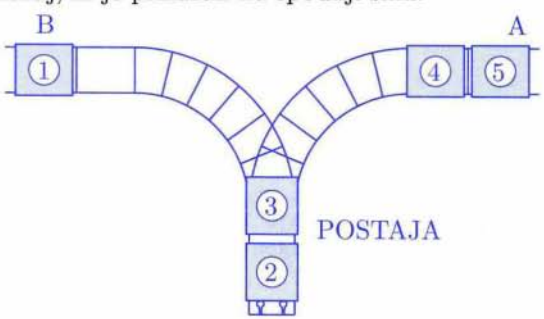
Pomagaj postajnemu načelniku in sestavi program, ki določi, ali je možno dobiti želeni razpored vagonov. Na primer, za podatke 1 3 4 5 2 bo odgovor pritrdilen (Da), za podatke 1 3 5 2 4 pa nikalen (Ne).

Matija Lokar

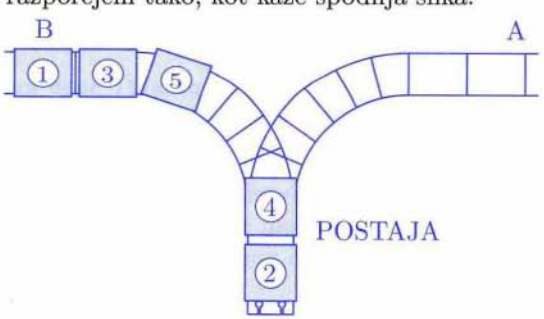
TEŽAVE POSTAJNEGA NAČELNIKA – Rešitev s str. 342

Ideja rešitve je, da simuliramo sestavljanje preurejenega vrstnega reda vagonov. Denimo, da na izhodnem tiru (smer B) potrebujemo vagon številka k. Na postaji so čakajoči vagoni vedno urejeni padajoče. Ker moramo s postaje najprej premakniti prvi čakajoči vagon, pogledamo samo tega. Če to ni vagon številka k, je možno, da je iskani vagon še vedno sestavni del vlaka na vhodnem tiru (smer A). Če vagona tam ni, želene razporeditve ni moč dobiti. Ker so vagoni na tiru A urejeni po vrsti, tudi tam pogledamo le prvega. Če ima številko manjšo ali enako k, prepeljemo vse vagone do k-tega na postajo.

Poglejmo primer. Denimo, da bi radi dosegli razporeditev 1 3 4 5 2. Najprej prepeljemo vagon številka 1 na izhodni tir. Na naslednjem koraku potrebujemo vagon številka 3. Zato vagona 2 in 3 zapeljemo na postajo in dobimo položaj, ki je prikazan na spodnji sliki.



Vagon številka 3 odpeljemo na izhodni tir, nato pa nanj prepeljemo še vagona številka 4 in 5, prav nazadnje pa s postaje odpeljemo še vagon številka 2. Če pa bi bila želena razporeditev 1 3 5 2 4, pač ne bi šlo. Začeti bi morali tako kot prej. Ko bi na tir B prepeljali vagon številka 5, bi bili vagoni razporejeni tako, kot kaže spodnja slika.



Iz tega položaja pa vagona številka 2 ni mogoče dobiti pred številko 4.

```
#include <stdio.h>
/* Ugotovi, ali je s pomočjo slepega tira vagone mogoče
   preurediti v želeni vrstni red.                          */
#define MAXN 1000 /* največja dolžina vlaka */
int main(void)
{
  int N;              /* dejanska dolžina vlaka - število vagonov */
  int vagon[MAXN];    /* želena razporeditev vagonov na tiru B */
  int postaja[MAXN]; /* vagoni, ki čakajo na postaji */
  int na_tiru_A;      /* številka prvega vagona na tiru A */
  int na_postaji;     /* število vagonov na postaji */
  int i, j;
  printf("\nVnesi dolzino vlaka: "); scanf("%d", &N);
  printf("Vnesi zeleno razporeditev vagonov na izhodnem tiru:\n");
  for (i = 0; i < N; i++) {
    printf(" st. %d. vagona: ", i + 1);
    scanf("%d", &vagon[i]);
  };
  na_tiru_A = 1;
  /* Na začetku so vsi vagoni na tiru A, */
  na_postaji = 0;          /* postaja pa je prazna. */
  for (i = 0; i < N; i++) {   /* Na tir B moramo dati vagon[i]. */
    if (na_tiru_A <= vagon[i]) {
      /* Vagoni do vagon[i] morajo iti na postajo. */
      for (j = na_tiru_A; j < vagon[i]; j++) {
        postaja[na_postaji] = j;
        na_postaji++;
      };
      /* vagon[i] "prepeljemo", prvi na tiru A je vagon[i] + 1. */
      na_tiru_A = vagon[i] + 1;
    }
    else if (vagon[i] == postaja[na_postaji - 1])
      na_postaji--;              /* Vagon gre s postaje na tir B. */
    else break;          /* želene razporeditve ni mogoce dobiti! */
  } /* for */
  if (i == N) printf("\nGre!\n"); else printf("\nNe gre!\n");
  return 0;
}
```

Pri pisanju programa smo privzeli, da bodo vhodni podatki vnešeni pravilno in da bodo res predstavljali neko permutacijo števil od 1 do N. Dopolnite program tako, da bo pri napačnem vnosu "protestiral".

Matija Lokar