PRESEK
List za mlade matematike, fizike, astronome in računalnikarje

ISSN 0351-6652
Letnik **27** (1999/2000)
Številka 1
Strani 16–21

Boštjan Brešar:

KRALJICE NAPADAJO

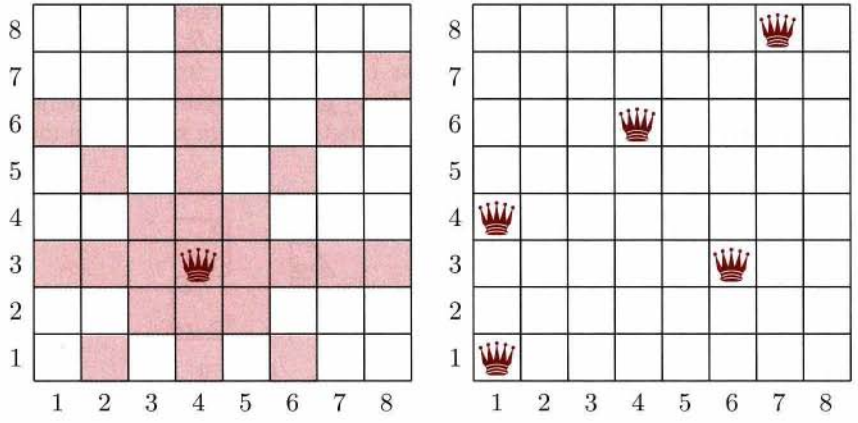
Ključne besede: matematika, računalništvo, programiranje, pascal, šah.

Elektronska verzija: http://www.presek.si/27/1389-Bresar.pdf

KRALJICE NAPADAJO

Situacija, ko sta na šahovski deski več kot dve kraljici, se v šahovski igri ne primeri prav pogosto. Še bolj redko, čeprav ni v nasprotju s pravili, pa imamo na šahovnici pet ali šest kraljic (mimogrede, kaj si mislite o taki igri?). Pustimo si tokrat nekaj svobode in predpostavimo, da od vseh figur polagamo na desko le kraljice in še teh ne ločimo med seboj po barvi.

Spomnimo se, da je velikost šahovske deske 8×8 polj in da se kraljice premikajo v vodoravni, navpični in v obeh diagonalnih smereh. Pravimo tudi, da kraljice *napadajo* določena polja. Zaradi preprostejše računalniške obdelave bomo tako vrstice kot stolpce šahovnice označevali s številkami od 1 do 8. Tako nam polje $(5, 6)$ pomeni polje v peti vrstici in šestem stolpcu. Na levi strani slike 1 vidimo kraljico na polju $(3, 4)$ in označena vsa polja, ki jih kraljica napada. Dogovorimo se, da kot napadeno polje štejemo tudi polje, na katerem kraljica stoji. Če kraljica stoji na enem izmed polj $(4, 4)$, $(4, 5)$, $(5, 4)$ ali $(5, 5)$, napada največje možno število polj (to je 28), najmanj polj (22) pa napada kraljica, ki stoji nekje na robu šahovnice.



Slika 1. Polja, ki jih napada kraljica, in rešitev s petimi kraljicami.

Zastavimo si naslednje vprašanje: *Koliko kraljic je potrebnih, da je napadenih vseh* 64 *polj na šahovnici?*

S tem in podobnimi problemi so se ukvarjali šahovski navdušenci v Evropi že sredi devetnajstega stoletja. Za zastavljeni problem so našli rešitev s petimi kraljicami. Takih rešitev je v resnici veliko, eno vidite

na desni strani slike 1. Seveda nas zanima najmanjše število kraljic, ki rešijo dano nalogo. Že v devetnajstem stoletju so pravilno predvidevali, da rešitve s štirimi kraljicami ni. Tedaj tega ni nihče preveril, saj je različnih postavitev štirih kraljic na šahovnico kar 635376 (to so kombinacije 64 elementov 4. reda). S pomočjo računalnika pa ta naloga za nas ni pretežka. Sestavili bomo torej program, ki bo preveril vse postavitve štirih kraljic na šahovski deski in ugotavljal, ali postavitve napadajo vsa polja. Program bomo zapisali v programskem jeziku pascal, ki ga gotovo pozna veliko bralcev Preseka.

Najprej se dogovorimo, na kakšen način bomo predstavili podatke. Šahovska deska s stolpci in vrsticami nas napeljuje k temu, da izberemo dvorazsežno tabelo. Ker nas bo o vsakem polju šahovnice zanimalo le, ali je napadeno ali ne, lahko kot elemente tabele izberemo logične vrednosti. V pascalu bomo torej uporabili podatkovno strukturo: `array`[1..8, 1..8] `of boolean`. Postavitev kraljice je povsem opisana z dvema podatkoma, to je stolpcem in vrstico. Zapišimo najprej podprogram `kaj_napada`, ki bo za dano postavitev kraljice določil napadena polja. Za 'navpična' in 'vodoravna' polja bomo uporabili preprosto zanko *for*, za 'diagonalna' polja pa bolj prefinjeni stavek *while*. Uporabili bomo štiri zanke *while* za vse štiri diagonalne smeri in pri tem posnemali pot kraljice po diagonali. Zanka se tako konča, ko pride kraljica do roba šahovnice. Predhodno moramo vrednosti tabele napadenih polj nastaviti na *false*. Za to poskrbi podprogram `nastavi`.

```
{Nastavi vsa polja šahovnice velikosti n × n kot nenapadena.}
procedure nastavi(var a: sahovnica);
  var i,j: integer;
begin
  for i := 1 to n do
    for j := 1 to n do a[i,j] := false;
end; {nastavi}

{Popravi tabelo napadenih polj ob postavitvi kraljice na polje (x,y).}
procedure kaj_napada(x,y: integer; var a: sahovnica);
  var i,j: integer;
begin
  for i := 1 to n do begin
    a[x,i] := true; a[i,y] := true
  end; {for}
```

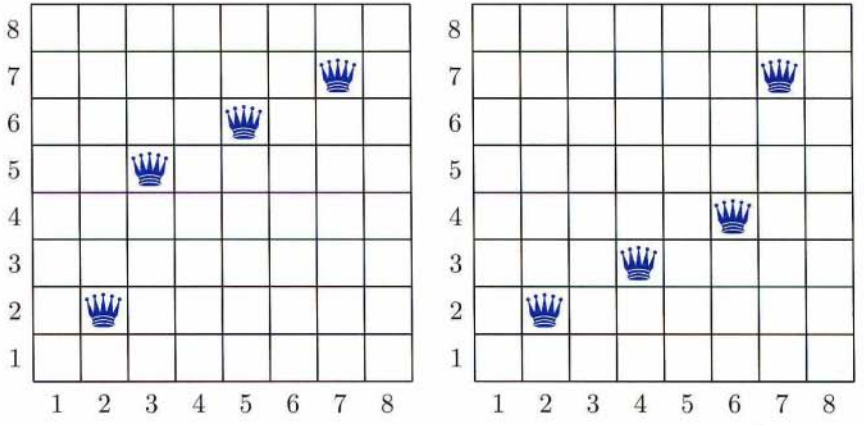
```
  i := x-1; j := y-1; { levo navzdol }
  while (i>=1) and (j>=1) do begin
    a[i,j] := true; i := i-1; j:= j-1;
  end; {while}
  i := x-1; j := y+1; { desno navzdol }
  while (i>=1) and (j<=n) do begin
    a[i,j] := true; i := i-1; j := j+1;
  end; {while}
  i := x+1; j := y-1; { levo navzgor }
  while (i<=n) and (j>=1) do begin
    a[i,j] := true; i := i+1; j := j-1;
  end; {while}
  i := x+1; j := y+1; { desno navzgor }
  while (i<=n) and (j<=n) do begin
    a[i,j] := true; i := i+1; j := j+1;
  end {while}
end; {kaj_napada}
```

Zdaj pa k osnovnemu konceptu programa. Za vsako postavitev štirih kraljic bomo določili polja, ki so napadena. To naredimo kar s podprogramom `kaj_napada`, ki ga pokličemo štirikrat zaporedoma. Na koncu bodo napadena polja v tabeli označena z logično vrednostjo *true*. Zatem bomo s funkcijo `preveri` ugotovili, ali so napadena vsa polja, in v tem primeru izpisali rešitev (podprogram `izpis`).

```
{Preveri, ali so napadena vsa polja šahovnice a.}
function preveri(a: sahovnica): boolean;
  var i,j: integer;
      test: boolean;
begin
  test := true;
  for i := 1 to n do
    for j := 1 to n do
      if a[i,j]=false then test := false;
  preveri := test;
end; {preveri}
```

```
{Izpiše rešitev (če ta obstaja).}
procedure izpis(i1,j1,i2,j2,i3,j3,i4,j4: integer);
begin
  writeln('Resitev:',i1,j1,', ',i2,j2,', ',i3,j3,', ',i4,j4);
end;
```

Preden zapišemo glavni program še nekaj opomb, zaradi katerih lahko skrajšamo čas izvajanja programa. Najprej o vrtenju šahovnice ali zvijanju vratu. Povsem jasno je namreč, da postavitve kraljic napadajo enako število polj, če šahovnico zavrtite za 90, 180 ali 270 stopinj (primer vidimo na sliki 2, kjer smo šahovnico zavrteli za 180°). To tudi pomeni, da stolpce in vrstice lahko oštevilčimo tako, da bo vsaj ena kraljica ležala na polju z vrstico in stolpcem od 1 do 4. Poleg tega v programu (delno) upoštevamo, da so si kraljice enakovredne, torej naj ne bi ponavljali postavitev dveh kraljic, ki le zamenjata svoji mesti. Ker želimo napisati čim preprostejši program, je nekaj takih ponavljanj vendarle ostalo (ali znaš ugotoviti, katera?).



Slika 2. Šahovnico zavrtimo za polkrog.

```
program kraljice(input,output);
  const n=8; { velikost šahovnice }
  type sahovnica=array[1..n,1..n] of boolean;
  var a: sahovnica; { tabela napadenih polj }
      x1,y1,x2,y2,x3,y3,x4,y4: integer; { položaji kraljic }
            .
            .
            .
```

```
begin
  for x1 := 1 to (n+1 div 2) do
    for x2 := x1 to n+1-x1 do
      for x3 := x2 to n+1-x1 do
        for x4 := x3 to n+1-x1 do
          for y1 := 1 to (n+1 div 2) do
            for y2 := 1 to n do
              for y3 := 1 to n do
                for y4 := 1 to n do begin
                  nastavi(a);
                  kaj_napada(x1,y1,a); kaj_napada(x2,y2,a);
                  kaj_napada(x3,y3,a); kaj_napada(x4,y4,a);
                  if preveri(a)=true then
                    izpis(x1,y1,x2,y2,x3,y3,x4,y4);
                end;
  writeln('Pregledovanje postavitev kraljic je koncano.'); readln;
end.
```

V manjkajoči del programa je seveda treba prepisati podprograme, ki smo jih zapisali prej, in dobimo program, ki naj bi pravilno deloval. To pomeni, da naj ne bi vrnil nobene rešitve, preden izpiše 'Pregledovanje postavitev kraljic je končano'. Če spremenimo vrednost konstante n, dobimo program, ki bo tekel tudi na šahovnicah drugih dimenzij. Bralec lahko brez veliko dela spremeni program tudi tako, da bo preverjal rešitve z večjim številom kraljic (v glavnem programu je treba dodati ustrezno število zank *for* in popraviti podprogram `izpis`). Kot zanimivost povejmo, da je pet kraljic dovolj tudi za šahovnice velikosti 9×9, 10×10 in 11×11, vendar je število rešitev ustrezno manjše. Za te primere lahko naš program spremenimo tako, kot smo maloprej opisali. A opozorilo: na mnogo večjih šahovnicah tega ne bomo počeli, saj problem sodi v skupino, ki ji v računalništvu rečemo NP-težki problemi. Kdor bo na primer poskusil pognati program za šahovnico velikosti 11×11 s petimi kraljicami, si bo lahko med izvajanjem programa privoščil kosilo. Morda bo celo čas za popoldanski spanec.

Naš program bi ob spretnem preprogramiranju lahko postal tudi malce hitrejši. Namreč namesto dvorazsežne tabele bi za vodenje evidence o napadenih poljih vzeli stolpce, vrstice in diagonale šahovnice (upoštevaj, da je posamezno polje napadeno natanko tedaj, ko je v njegovem stolpcu, vrstici ali eni od diagonal kaka kraljica). Teh je skupaj $8 + 8 + 26 = 42$, saj imamo dve vrsti diagonal ('naraščajoče' in 'padajoče'), pri čemer kotne diagonale z enim samim poljem izpustimo. Izvajanje podprograma

`kaj_napada` bi se tako precej skrajšalo, saj bi za postavitev kraljice na določeno polje morali spremeniti le štiri podatke (vrstico, stolpec in diagonali, ki jih kraljica napada). Ob tem bi bilo seveda potrebno spremeniti tudi podprograma `nastavi` in `preveri`, pri čemer bi za slednjega potrebovali nekaj več vrstic programske kode kot v gornji različici.

Naloge (rešitve boste našli v naslednji številki Preseka):

1. Na desnem delu slike 1 imamo eno izmed mnogih rešitev s petimi kraljicami. Ali znaš poiskati tako rešitev, pri kateri se bodo vse kraljice medsebojno napadale (recimo rešitev, pri kateri vse kraljice ležijo na najdaljši diagonali šahovnice)? In še: poišči tako rešitev, da se kraljice medsebojno ne bodo napadale. Obeh nalog se lahko lotiš s pomočjo računalnika ali s poskušanjem na šahovnici.

2. Podoben problem lahko zastavimo za trdnjave, vendar je rešitev zelo preprosta. Vsaj 8 trdnjav potrebujemo, da so vsa polja na šahovnici velikosti 8×8 napadena. Vendar ni vsaka postavitev z osmimi trdnjavami prava. Ali znaš povedati, kakšno lastnost imajo postavitve osmih trdnjav, ki niso prave (ki torej ne napadajo vseh polj)? Kaj pa isti problem na šahovnicah večjih dimenzij?

3. Koliko kraljev potrebujemo, da so napadena vsa polja šahovnice 8×8? Komur se to vprašanje zdi prelahko, naj poskusi najti formulo za najmanjše število kraljev, ki napadajo vsa polja šahovnice z n vrsticami in n stolpci.

Boštjan Brešar

ISKRICA

Po težkem kolokviju iz teorije mere so se študenti usidrali v bližnjem lokalu, da s kavo poplaknejo naporno delo in preverijo rešitve. Podajali so si Cauchy-Riemannove enačbe, pa "u po dv" in "v po du" ter se niso mogli zediniti. Tako vso uro. Ko so odhajali, je pristopil natakar in vsakemu ponudil bonbon. Prijetno presenečeni so vprašali, s čim so si to zaslužili.

In odgovor: "Da ne boste ves čas mislili le na DDV!"

Tatjana Hernaus