Žaidimas

Tai yra klasikinis perrinkimo (grįžimo metodo) uždavinys.

Tarkime, kad turime kokią nors situaciją žaidimo lentoje. Pasirenkame bet kurį ėjimą, jį atliekame, įsimename ir taip žaidžiame, kol susidaro situacija lentoje, kai nebeįmanoma nieko nukirsti. Jei lentoje liko tik viena šaškė, sprendinys rastas, jei ne – atstatome paskutinį įsimintą ėjimą ir bandome eiti kitaip. Jei išbandome visus ėjimus ir nei vienas netinka, grįžtame dar per žingsnį atgal.

Darbą baigiame, kai randame sprendinį arba išbandome visus galimus variantus.

Algoritmo sudarymui paimsime dvimatį kvadratinį masyvą, kuriame tilptų lenta o kraštiniai masyvo elementai liktų už lentos ribų. Lentą uždėsime ant kvadrato taip, kad jų centriniai langeliai sutaptų. Masyvo elementus, atitinkančius lentos langelius su šaškėmis užpildysime vienetais (1), o atitinkančius tuščius langelius – nuliais (0). Masyvo langelius, neįeinančius į lentą taip pat užpildysime minus vienetais (-1).

```
const MAX_N = 5;
  MAX_M = 5;
  MAX = MAX_N div 2 + MAX_M+2;  { maksimalaus kvadrato, kuriame telpa }
                                { lenta, kraštinė; lenta apgaubiama tuščiais langeliais}
  MAX_K = 150;  { maksimalus kirtimų skaičius }
type t_lenta = array [-MAX..MAX, -MAX..MAX] of integer;
                                    { 1 – šaškė, 0 – tuščia −1 – už lentos ribų }
     kirtimai = array [0..MAX_K] of
                          record
                            x1, y1, x2, y2: integer
                          end;

procedure šaškės (dydis,  { n div 2 + m + 2}
                  ša: integer;  { šaškių skaičius lentoje }
                  lenta: t_lenta;
                  var nuk: kirtimai;  { kirtimų seka }
                  var rasta: boolean  { ar rastas sprendinys });
  var k: integer;       { nukirstų šaškių skaičius }
```

```
  procedure kirsti (x1, y1, x2, y2: integer);
  { rekursinė procedūra, atliekanti kirtimus }
    var i, j: integer;
  begin
    k := k + 1;   { įsidėmime kirtimą }
    nuk[k].x1 := x1;      nuk[k].x2 := x2 + (x2 - x1);
    nuk[k].y1 := y1;      nuk[k].y2 := y2 + (y2 - y1);
    lenta[x1, y1] := 0;
    lenta[x2, y2] := 0;
    lenta[x2 + (x2 - x1), y2 + (y2 - y1)] := 1;
    if k = ša - 1
       then rasta := true
    else
     for i := -dydis to dydis do
       for j := -dydis to dydis do
         if (lenta[i, j] = 1) and not rasta
            then if (lenta[i,j+1]=1) and (lenta[i,j+2]=0)
                     then kirsti (i, j, i, j+1)
            else if (lenta[i,j-1]=1) and (lenta[i,j-2]=0)
                     then kirsti (i, j, i, j-1)
            else if (lenta[i-1,j]=1) and (lenta[i-2,j]=0)
                     then kirsti (i, j, i-1, j)
            else if (lenta[i+1,j]=1) and (lenta[i+2,j]=0)
                    then kirsti (i, j, i+1, j);
    { grąžiname nukirstą šaškę į vietą }
    if not rasta
       then begin
              k := k - 1;
              lenta[x1, y1] := 1;
              lenta[x2, y2] := 1;
              lenta[x2 + (x2 - x1), y2 + (y2 - y1)] := 0;
            end;
  end;  { kirsti }

begin   { šaškės }
  rasta := false;
  k := -1;
  { pirmasis kirtimas bus fiktyvus }
  kirsti (dydis, dydis, dydis, dydis)
end;  { šaškės }
```