Algoritmusok és adatszerkezetek II

Sakk

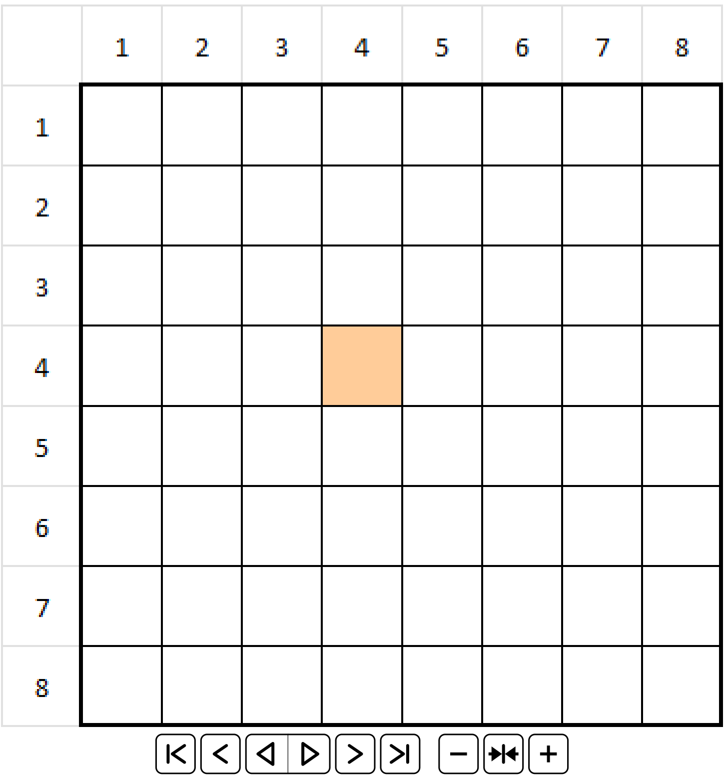
Boda Bálint

2022. őszi félév

A sakkban a huszár kétféleképpen tud mozogni:

- függőlegesen két mezőt és vízszintesen egyet
- függőlegesen egy mezőt és vízszintesen kettőt

Ha elkezdjük beszínezni a huszár által n $(n \in 0, 1, 2 \ldots)$ lépésből elérhető mezőket (melyet a következő animáció szemléltet),



megfigyelhetjük, hogy a beszínezés eljárása sok tekintetben hasonlít a szélességi bejárás algoritmusára, gyakorlatilag annak egy olyan módosítása mely az algoritmussal egyidőben építi fel a gráfot.

Könnyű meggondolni, hogy bizonyos táblaméreteknél a huszár nem tudja az összes mezőt elérni, illetve kellően kicsi tábla esetén még mozogni sem tud, azaz előfordulhat olyan eset, hogy nem találunk utat a keresett mezőbe.

A csúcsok reprezentálására vezessük be a következő típust:

Vertex
+ int i + int j
+ Vertex(r,c) $\{i = r;\ j = c\}$

Készítsünk egy segédeljárást, mely megadja egy csúcsból a huszár által 1 lépéssel elérhető mezők halmazát.

getNeighbours$(n, m : \mathbb{N}^+, i, j : \mathbb{N}) : Vertex\,\{\}$

- $V : Vertex\,\{\}$ //set of vertices
- $V := V \cup \{Vertex(i-2, j-1), Vertex(i-2, j+1)\}$
- $V := V \cup \{Vertex(i-1, j-2), Vertex(i-1, j+2)\}$
- $V := V \cup \{Vertex(i+1, j-2), Vertex(i+1, j+2)\}$
- $V := V \cup \{Vertex(i+2, j-1), Vertex(i+2, j+1)\}$
- $\forall v \in V$
  - if $v.i < 1 \lor v.i > n \lor v.j < 1 \lor v.j > m$
    - then: $V := V \setminus \{v\}$
    - else: SKIP
- **return** V

shortestKnightPathLength$(n, m : \mathbb{N}^+, i_1, j_1, i_2, j_2 : \mathbb{N}) : \mathbb{N}$

- if $i_1 = i_2 \land j_1 = j_2$
  - then: **return** 0
  - else: SKIP
- $d : \mathbb{N}[n][m]$//distance of a given vertex from source
- $i := 1..n$
  - $j := 1..m$
    - $d[i][j] := \infty$
- $s := Vertex(i_1, j_1);\ d[s.i, s.j] = 0$
- $Q : Queue;\ Q.add(s)$
- $\neg Q.isEmpty()$
  - $u := Q.rem()$
  - $neighbours := getNeighbours(n, m, u.i, u.j)$
  - $\forall v \in neighbours$
    - if $d(v.i, v.j) = \infty$
      - then:
        - $d[v.i][v.j] = d[u.i][u.j] + 1$
        - if $v.i = i_2 \land v.j = j_2$
          - then: **return** $d[v.i][v.j]$
          - else: SKIP
        - $Q.add(v)$
      - else: SKIP
- **return** ∞