

H2.

MENGE-UNIQUE (M, q)

let points be a new List of arrays of int

IF MENGE-UNIQUE-BACKTRACKING (M, q, points, 0, 0)

RETURN points

ELSE

ERROR

MENGE-UNIQUE-BACKTRACKING (M, q, points, s, a)

IF points.length > 1

IF SIND-VERSCHIEDEN (points)

RETURN TRUE

ELSE

RETURN FALSE

ELSE

WHILE points.length < q

IF S < M.length

points.add (M[S])

ELSE

points.remove (points.length - 1)

a++

S = a

IF a == M.length

RETURN FALSE

ELSE

points.add (M[a])

IF NOT MENGE-UNIQUE-BACKTRACKING (M, q, points, ~~S~~^{S+1}, a)

IF points.length > 0

points.remove (points.length - 1)

ELSE

RETURN FALSE

S++

RETURN TRUE

SIND - VERSCHIEDEN (points)

IF points.length < 1

RETURN FALSE

FOR i = 0 TO points[0].length - 1

FOR j = 0 TO points.length - 1

FOR k = j + 1 TO points.length - 1

IF points[j][i] == points[k][i]

RETURN FALSE

RETURN TRUE

H3. BERGSTEIGERALGORITHMUS

i)

BERGSTEIG(panel, x, y, zx, zy)

ax = x

ay = y

dist = 2 * panel[0].length // maximaler Abstand.

adist = f(x, y, zx, zy)

WHILE adist < dist

dist = adist

FOR i = 1 DOWN TO -2 // mit 2 Sprung; also i = 1 und -1

IF ((i == 1 AND y < 4) OR (i == -1 AND y > 0)) AND panel[x][y+i]

IF f(x, y+i, zx, zy) < adist

ax = x

ay = y+i

adist = f(ax, ay, zx, zy)

IF ((i == -1 AND x < 4) OR (i == 1 AND x > 0)) AND panel[x-i][y]

IF f(x-i, y, zx, zy) < adist

ax = x-i

ay = y

adist = f(ax, ay, zx, zy)

x = ax

y = ay

return [x, y]

f(x1, y1, x2, y2)

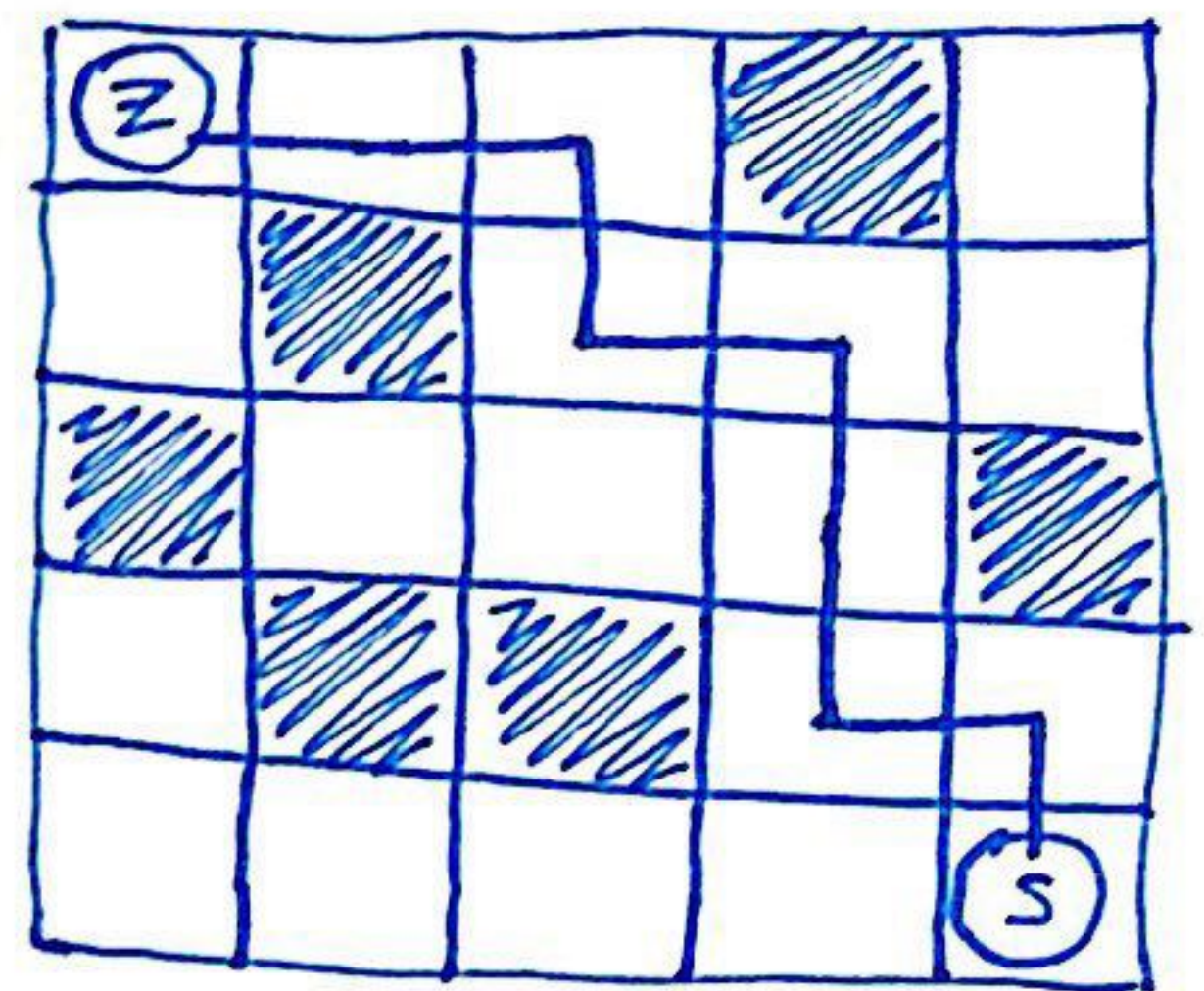
RETURN |x1 - x2| + |y1 - y2|

ii) wir betrachten hier die Positionen von 0 bis 4 und nicht von 1 bis 5 wie auf dem Bild, weil unser Algorithmus so arbeitet. Wir könnten aber die Lösung einfach ändern, um die gewünschten Positionen zwischen 1 und 5 zu bekommen. Es ist Analog.

wir fangen also in $[4, 0]$ an (analog zu $[5, 1]$)

$[4, 0] \rightarrow [4, 1] \rightarrow [3, 1] \rightarrow [3, 2] \rightarrow [3, 3] \rightarrow [2, 3]$

$\rightarrow [2, 4] \rightarrow [1, 4] \rightarrow [0, 4] \rightarrow$ Ziel wird erreicht!

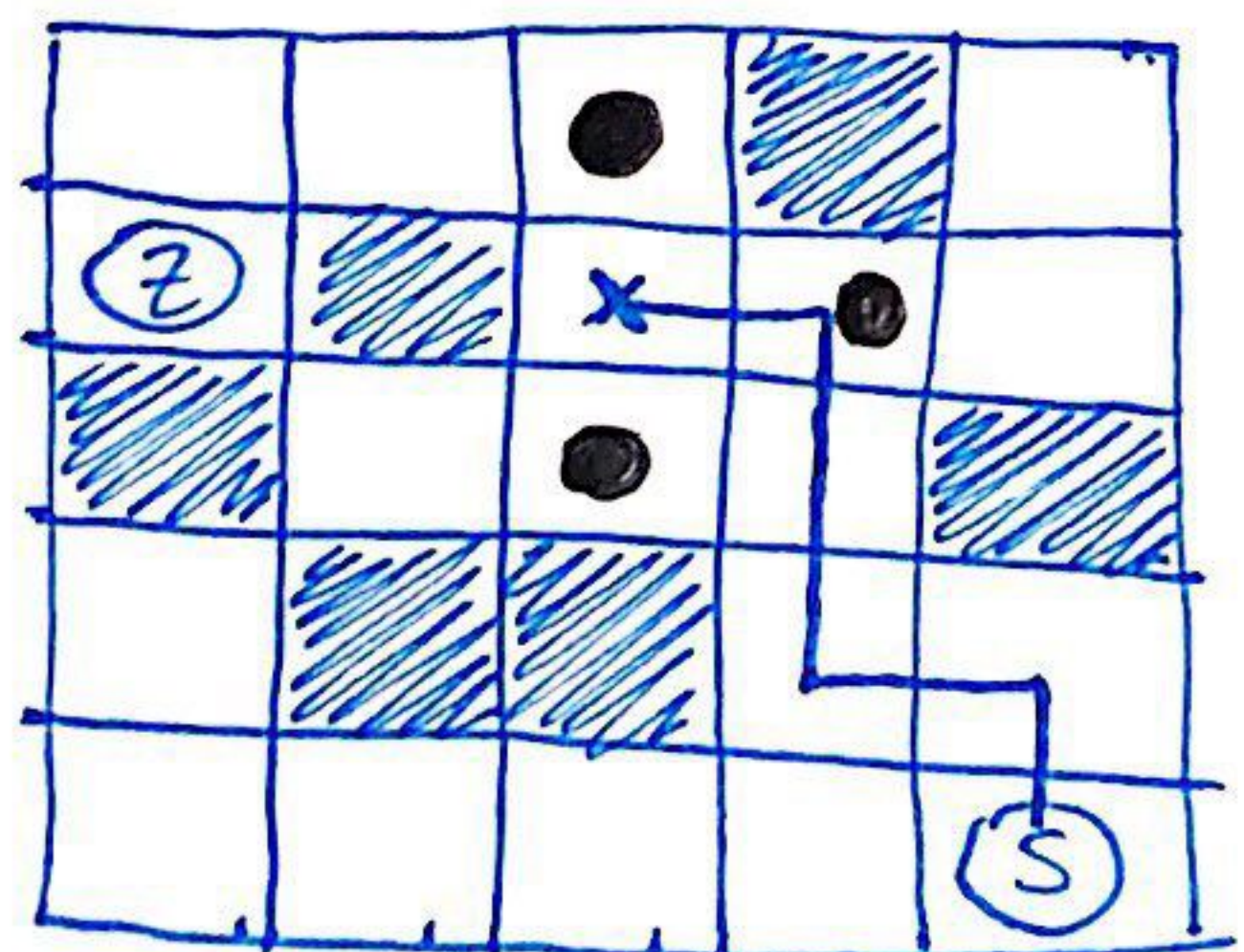


iii)

wir müssen nichts anpassen, einfach nur einen neuen Punkt z_x, z_y geben, in diesem Fall $z_x = 0$ ~~$z_x = 0$~~ $z_y = 3$.
panel bleibt unverändert, weil es gleich wie links ist.

Wir erreichen in diesem Fall das Ziel nicht, wir bleiben in $[2, 3]$ gesperrt.

von $[2, 3]$ aus gibt es keine Position \neq entfernt, die näher ist zu z .
ich habe die mögliche Positionen schwarz markiert \rightarrow



(i)

Für $S_0 = (-2, -4) \rightarrow (-1, -3) \rightarrow (0, -2) \rightarrow (0, -1) \rightarrow \boxed{(0, 0)}$

Für $S_1 = (4, 4) \rightarrow (3, 3) \rightarrow (2, 2) \rightarrow \boxed{(1, 2)}$

Warum sind sie verschieden? Weil es zwei lokale Maxima gibt. Wenn wir die Funktion zeichnen:

Es ist entscheidend, wo wir anfangen, um eines oder anderes Maximum zu finden

