Linguistic Geometry

Project 1

Problem Definition

Given the dimensions of a board space (width height), the position of an item, the reachability for that item (how it moves), and possibly a list of unreachable spaces on the board space, find the minimum number of moves needed for the piece to move to every possible space.

Results

```
> (distance Pawn '(8 8) '(15 15))
'((                              )
  (                              )
  (                              )
  (                              )
  (                              )
  (                              )
  (                              )
  (              0               )
  (              1               )
  (              2               )
  (              3               )
  (              4               )
  (              5               )
  (              6               )
  (              7               ))
> (distance Knight '(8 8) '(15 15))
'((6 5 4 5 4 5 4 5 4 5 4 5 4 5 6)
  (5 4 5 4 3 4 3 4 3 4 3 4 5 4 5)
  (4 5 4 3 4 3 4 3 4 3 4 3 4 5 4)
  (5 4 3 4 3 2 3 2 3 2 3 4 3 4 5)
  (4 3 4 3 2 3 2 3 2 3 2 3 4 3 4)
  (5 4 3 2 3 4 1 2 1 4 3 2 3 4 5)
  (4 3 4 3 2 1 2 3 2 1 2 3 4 3 4)
  (5 4 3 2 3 2 3 0 3 2 3 2 3 4 5)
  (4 3 4 3 2 1 2 3 2 1 2 3 4 3 4)
  (5 4 3 2 3 4 1 2 1 4 3 2 3 4 5)
  (4 3 4 3 2 3 2 3 2 3 2 3 4 3 4)
  (5 4 3 4 3 2 3 2 3 2 3 4 3 4 5)
  (4 5 4 3 4 3 4 3 4 3 4 3 4 5 4)
  (5 4 5 4 3 4 3 4 3 4 3 4 5 4 5)
  (6 5 4 5 4 5 4 5 4 5 4 5 4 5 6))
```

```
> (distance Bishop '(8 8) '(15 15))
'((1   2   2   2   2   2   2   1)
  (  1   2   2   2   2   2   1  )
  (2   1   2   2   2   2   1   2)
  (  2   1   2   2   2   1   2  )
  (2   2   1   2   2   1   2   2)
  (  2   2   1   2   1   2   2  )
  (2   2   2   1   1   2   2   2)
  (  2   2   2   0   2   2   2  )
  (2   2   2   1   1   2   2   2)
  (  2   2   1   2   1   2   2  )
  (2   2   1   2   2   1   2   2)
  (  2   1   2   2   2   1   2  )
  (2   1   2   2   2   2   1   2)
  (  1   2   2   2   2   2   1  )
  (1   2   2   2   2   2   2   1))
> (distance Rook '(8 8) '(15 15))
'((2 2 2 2 2 2 2 1 2 2 2 2 2 2 2)
  (2 2 2 2 2 2 2 1 2 2 2 2 2 2 2)
  (2 2 2 2 2 2 2 1 2 2 2 2 2 2 2)
  (2 2 2 2 2 2 2 1 2 2 2 2 2 2 2)
  (2 2 2 2 2 2 2 1 2 2 2 2 2 2 2)
  (2 2 2 2 2 2 2 1 2 2 2 2 2 2 2)
  (2 2 2 2 2 2 2 1 2 2 2 2 2 2 2)
  (1 1 1 1 1 1 1 0 1 1 1 1 1 1 1)
  (2 2 2 2 2 2 2 1 2 2 2 2 2 2 2)
  (2 2 2 2 2 2 2 1 2 2 2 2 2 2 2)
  (2 2 2 2 2 2 2 1 2 2 2 2 2 2 2)
  (2 2 2 2 2 2 2 1 2 2 2 2 2 2 2)
  (2 2 2 2 2 2 2 1 2 2 2 2 2 2 2)
  (2 2 2 2 2 2 2 1 2 2 2 2 2 2 2)
  (2 2 2 2 2 2 2 1 2 2 2 2 2 2 2))
> (distance Queen '(8 8) '(15 15))
'((1 2 2 2 2 2 2 1 2 2 2 2 2 2 1)
  (2 1 2 2 2 2 2 1 2 2 2 2 2 1 2)
  (2 2 1 2 2 2 2 1 2 2 2 2 1 2 2)
  (2 2 2 1 2 2 2 1 2 2 2 1 2 2 2)
  (2 2 2 2 1 2 2 1 2 2 1 2 2 2 2)
  (2 2 2 2 2 1 2 1 2 1 2 2 2 2 2)
  (2 2 2 2 2 2 1 1 1 2 2 2 2 2 2)
  (1 1 1 1 1 1 1 0 1 1 1 1 1 1 1)
  (2 2 2 2 2 2 1 1 1 2 2 2 2 2 2)
  (2 2 2 2 2 1 2 1 2 1 2 2 2 2 2)
  (2 2 2 2 1 2 2 1 2 2 1 2 2 2 2)
  (2 2 2 1 2 2 2 1 2 2 2 1 2 2 2)
  (2 2 1 2 2 2 2 1 2 2 2 2 1 2 2)
  (2 1 2 2 2 2 2 1 2 2 2 2 2 1 2)
  (1 2 2 2 2 2 2 1 2 2 2 2 2 2 1))
```

```
> (distance King '(8 8) '(15 15))
'((7 7 7 7 7 7 7 7 7 7 7 7 7 7 7)
  (7 6 6 6 6 6 6 6 6 6 6 6 6 6 7)
  (7 6 5 5 5 5 5 5 5 5 5 5 5 6 7)
  (7 6 5 4 4 4 4 4 4 4 4 5 6 7)
  (7 6 5 4 3 3 3 3 3 3 4 5 6 7)
  (7 6 5 4 3 2 2 2 2 3 4 5 6 7)
  (7 6 5 4 3 2 1 1 1 2 3 4 5 6 7)
  (7 6 5 4 3 2 1 0 1 2 3 4 5 6 7)
  (7 6 5 4 3 2 1 1 1 2 3 4 5 6 7)
  (7 6 5 4 3 2 2 2 2 2 3 4 5 6 7)
  (7 6 5 4 3 3 3 3 3 3 3 4 5 6 7)
  (7 6 5 4 4 4 4 4 4 4 4 4 5 6 7)
  (7 6 5 5 5 5 5 5 5 5 5 5 5 6 7)
  (7 6 6 6 6 6 6 6 6 6 6 6 6 6 7)
  (7 7 7 7 7 7 7 7 7 7 7 7 7 7 7))
> (distance Rook '(8 8) '(10 13) '((7 6) (7 7) (7 8) (3 1) (5 5) (9 12)))
'((3 3 X 2 2 2 2 1 2 2)
  (2 2 2 2 2 2 2 1 2 2)
  (2 2 2 2 2 2 2 1 2 2)
  (2 2 2 2 2 2 2 1 2 2)
  (3 3 3 3 X 2 2 1 2 2)
  (3 3 3 3 3 3 X 1 2 2)
  (3 3 3 3 3 3 X 1 2 2)
  (3 3 3 3 3 3 X 0 1 1)
  (2 2 2 2 2 2 2 1 2 2)
  (2 2 2 2 2 2 2 1 2 2)
  (2 2 2 2 2 2 2 1 2 2)
  (2 2 2 2 2 2 2 1 X 2)
  (2 2 2 2 2 2 2 1 2 2))
> (distance Bishop '(2 2) '(10 10) '((7 6) (7 7) (7 8) (3 1) (5 5) (9 12)))
'((1   X   2   2   4  )
  (  0   2   2   3   3)
  (1   1   2   3   3  )
  (  2   1   3   3   3)
  (2   2   X   3   3  )
  (  2   2   3 X 3   3)
  (2   3   2   X   3  )
  (  3   3   2 X 3   3)
  (3   3   3   2   3  )
  (  3   3   3   2   4))
```

```
> (distance King '(5 7) '(10 10) '((7 4) (7 5) (6 6) (7 7) (7 8)))
'((6 6 6 6 6 6 6 6 7)
  (5 5 5 5 5 5 5 5 6)
  (4 4 4 4 4 4 4 5 5)
  (4 3 3 3 3 3 X 4 4 5)
  (4 3 2 2 2 2 X 3 4 5)
  (4 3 2 1 1 X 2 3 4 5)
  (4 3 2 1 0 1 X 3 4 5)
  (4 3 2 1 1 1 X 3 4 5)
  (4 3 2 2 2 2 2 3 4 5)
  (4 3 3 3 3 3 3 3 4 5))
>
```

Implementation

Not that much complex here, except I changed how reachability works. Instead of simply checking to see if a space is valid, it checks to see if the path is reachable, so on top of making sure the movement pattern is valid, it checks to make sure all spaces beforehand (that must be traveled through) are clear of obstacles.

```
(define (movement x)
  (cond
    [(equal? x 0) 0]
    [(< x 0) (+ x 1)]
    [(< 0 x) (- x 1)]))
(define (King xc yc p)(and (<= (abs xc) 1) (<= (abs yc) 1)))
(define (Knight xc yc p) (or (and (equal? 2 (abs xc)) (equal? 1 (abs yc)))
(and (equal? 1 (abs xc)) (equal? 2 (abs yc)))))
(define (Bishop xc yc p)
  (if (and (equal? xc 0) (equal? yc 0))
      #t
      (and
       (equal? (abs xc) (abs yc))
       (not (member (list (+ (first p) xc) (+ (second p) yc)) blocked))
       (Bishop (movement xc) (movement yc) p))))
(define (Rook xc yc p)
  (if (and (equal? xc 0) (equal? yc 0))
      #t
      (and
       (or (equal? yc 0) (equal? xc 0))
       (not (member (list (+ (first p) xc) (+ (second p) yc)) blocked))
       (Rook (movement xc) (movement yc) p))))
(define (Pawn xc yc p) (and (equal? yc 1) (zero? xc)))
(define (Queen xc yc p) (or (Rook xc yc p) (Bishop xc yc p)))
```