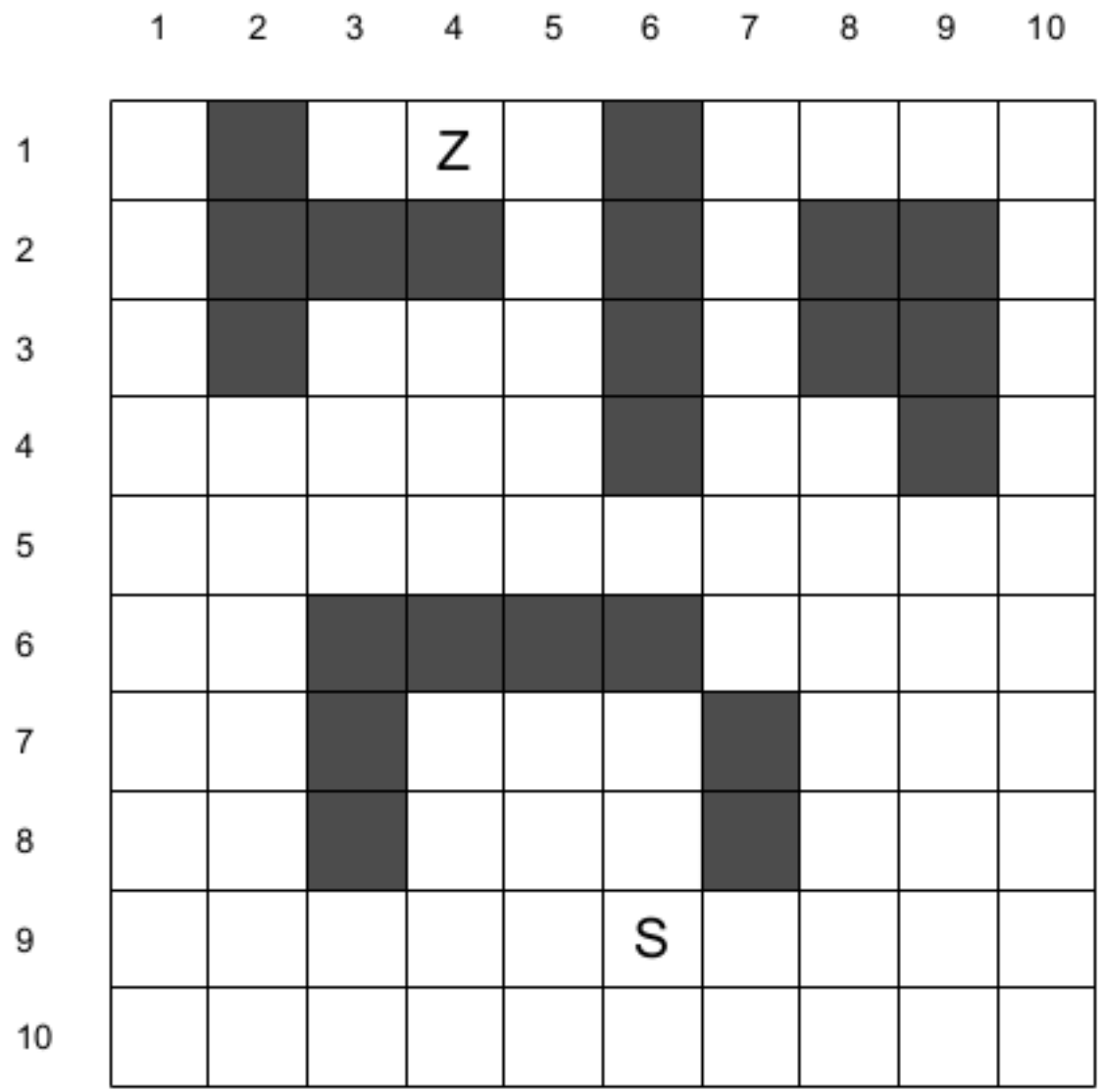
A-Stern Programm

Analyse eines

Programms zu A*

für das einfache Labyrinth

A-Stern Programm



A-Stern Programm

Module

- prioritaetswarteschlange.scm
  - Prädikat vor?
- Feld
  - Assoziationsliste der Hindernisse *(gesperrte)*
  - Prädikate **im-feld?** **ist-hindernis?** **zulaessig?**
- Kosten
  - **kosten** und **restkosten** (aktuelle → Länge Weg)
- Expansion der Ebenen
  - **expandiere** und **nachfolger**
- Aufrufhülle

A-Stern Programm

Prädikat vor?

```
(define
  (vor? weg-1 weg-2)
  (< (kosten weg-1 ziel) (kosten weg-2 ziel)))
```

zur Funktion **kosten** → *weiter hinten*

A-Stern Programm

Prädikat im-feld?

```
(define
 (im-feld? ort)
 (and
  (> (first ort) 0)
  (< (first ort) 11)
  (> (second ort) 0)
  (< (second ort) 11)))
```

A-Stern Programm

Assoziationsliste der Hindernisse

```
(define
 hindernisse-assoziationsliste
 '((1 2 6)          ; (zeilenindex spaltenindex-1 ...)
  (2 2 3 4 6)
  (3 2 6 8 9)
  (4 6 8 9)
  (5 9)
  (6 3 4 5 6)
  (7 3 7)
  (8 3 7)
  (9)
  (10)))
```

A-Stern Programm

Prädikat ist-hindernis?

```
(define
 (ist-hindernis? ort)
 (member
  (second ort)
  (rest
   (assoc
     (first ort)
      hindernisse-assoziationsliste)))
```

A-Stern Programm

Prädikat zulaessig?

```
(define
 (zulaessig? ort)
 (and
  (im-feld? ort)
  (not (ist-hindernis? ort))))
```

A-Stern Programm

Funktion nachfolger

```
(define
 (nachfolger ort)
 (append
  (if (zulaessig? (list (sub1 (first ort)) (second ort)))
      (list (list (sub1 (first ort)) (second ort))) '())
  (if (zulaessig? (list (add1 (first ort)) (second ort)))
      (list (list (add1 (first ort)) (second ort))) '())
  (if (zulaessig? (list (first ort) (sub1 (second ort))))
      (list (list (first ort) (sub1 (second ort)))) '())
  (if (zulaessig? (list (first ort) (add1 (second ort))))
    (list (list (first ort) (add1 (second ort)))) '())))
```

A-Stern Programm

Funktion restkosten (*Manhattandistanz*)

```
(define
 (restkosten ort ziel)
 (+
  (abs
   (- (first ort) (first ziel)))
  (abs
   (- (second ort) (second ziel)))))
```

A-Stern Programm

Funktion kosten

```
(define
  (kosten weg ziel)
  (+ (length weg) (restkosten (first weg) ziel)))
```

A-Stern Programm

Funktion expandiere

```
(define
 (expandiere weg ziel)
 (let lokal ((nachfolger (nachfolger (first weg))) (akku'()))
  (cond
   ((null? nachfolger) akku)
   ((member ziel nachfolger)
    (list (cons (first (member ziel nachfolger)) weg)))
   ((member (first nachfolger) weg)
    (lokal (rest nachfolger) akku))
   (else
    (lokal (rest nachfolger)
     (cons (cons (first nachfolger) weg) akku))))))
```

A-Stern Programm

Aufrufhülle a*

```
(define
 (a* start ziel wege)
 (cond
  ((null? wege)  'nicht-loesbar)
  ((member ziel (first wege))  (first wege))
  (else
   (a*
    start
    ziel
    (fuege-alle-ein
     (expandiere (first wege) ziel) vor? (rest wege))))))
```