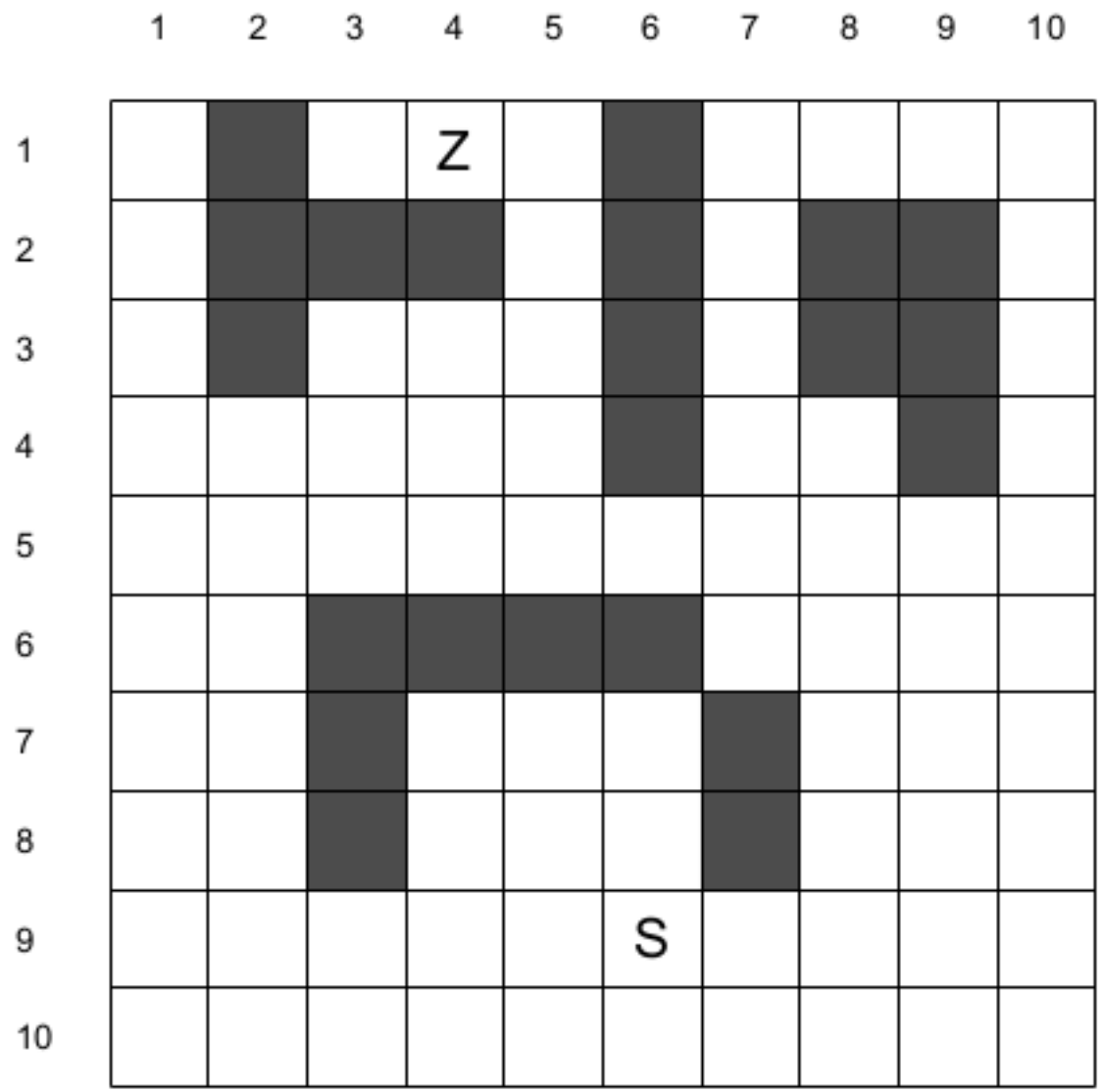
A-Stern Programm

Analyse eines

Programms zu A*

für das einfache Labyrinth

A-Stern Programm



A-Stern Programm

Module bzw Abschnitte

- Für Prioritätswarteschlange **ordneAlleEin**
  - Prädikat **vor** (konkret **kuerzer**)
- Feld
  - Dictionary der Hindernisse *(***gesperrt***)*
  - Prädikat **zulaessig** (verwendet **gesperrt**)
- Kosten
  - **kosten** und **restkosten** (aktuelle → Länge Weg)
- Expansion der Ebenen
  - **expandiere** und **nachfolger**
- Aufrufhülle

A-Stern Programm

Prädikat vor (kuerzer)

```
def kuerzer(weg1, weg2):
    return gesamtKosten(weg1)<gesamtKosten(weg2)
```

*zur Funktion **gesamtKosten** → weiter hinten*

A-Stern Programm

Prädikat zulaessig

```
def zulaessig(feld):
    for koordinate in feld:
        if koordinate<1: return False
        if koordinate>10: return False
    if feld[1] in gesperrt[feld[0]]: return False
    return True
```

A-Stern Programm

Dictionary der Hindernisse

```
gesperrt={
   1:[2,6],
   2:[2,3,4,6,8,9],
   3:[2,6,8,9],
   4:[6,9],
   5:[],
   6:[3,4,5,6],
   7:[3,7],
   8:[3,7],
   9:[],
   10:[]
}
```

A-Stern Programm

Funktion nachfolger

```
def nachfolger(feld):
    ergebnis=[]
    for fall in [[-1,0],[1,0],[0,-1],[0,1]]:
        if zulaessig([feld[0]+fall[0],feld[1]+fall[1]]):
            ergebnis+=[[feld[0]+fall[0],feld[1]+fall[1]]]
    return ergebnis
```

A-Stern Programm

Funktion kosten

```
def kosten(weg):
    return len(weg)-1
```

Funktion restkosten (*Manhattandistanz*)

```
def restkosten(feld):
    return abs(feld[0]-ziel[0])+abs(feld[1]-ziel[1])
```

Funktion gesamtKosten

```
def gesamtKosten(weg):
    return kosten(weg)+restkosten(weg[len(weg)-1])
```

A-Stern Programm

Funktion expandiere

```
def expandiere(weg):
    fortsetzungen=nachfolger(weg[len(weg)-1])
    gefiltert=[]
    for knoten in fortsetzungen:
        if not knoten in weg:
            gefiltert.append(knoten)
    fortsetzungen=[]
    for i in range(len(gefiltert)):
        fortsetzungen.append(weg+[gefiltert[i]])
    return fortsetzungen
```

A-Stern Programm

Funktion expansionsSchritt

```
def expansionsSchritt(prioWS, ziel):
    wegZumZiel=None
    while wegZumZiel==None:
        prioWS=ordneAlleEin(expandiere(prioWS[0]), prioWS[1:])
        wegZumZiel=amZiel(prioWS, ziel)
    return wegZumZiel
```

A-Stern Programm

Funktion amZiel

```
def amZiel(prioWS, ziel):
    for weg in prioWS:
        if ziel in weg:
            return weg
    return None
```

A-Stern Programm

Aufrufhülle aStern

```
def aStern(start, ziel):
    return expansionsSchritt([[start]], ziel)
```