KI – Hausaufgabe 1

Erik Bauer | 215204632

Aufgabe 1 – Der Staubsauger Agent

Teilaufgabe 1.

Siehe beigefügte Python-Datei `staubsauger-2d-aufgabe.py` in den Bereichen zwischen `## S Added` und `## E Added`.
Der Roboter bewegt sich, sofern vorhanden, in die Richtung eines benachbarten, nicht besuchten Feldes. Sind alle benachbarten Felder besucht, so bewegt er sich zufällig in eine Richtung. War seine Bewegung ohne Effekt, d.h. seine vorherige und derzeitige beobachtete Position ist identisch, bewegt er sich nicht in die vorher zufällig gewählte Richtung.

Teilaufgabe 2.

Nein, er reinigt seine Umgebung i.d.R. nicht in der minimalen Anzahl Bewegungen es sei denn, seine Startposition ist günstig gesetzt, in meiner Implementierung die untere linke Ecke. Jedoch selbst dann bewegt er sich gegen die Wände, da er sie nicht sehen kann. Dies liegt, unter Anderem, an der „Kurzsichtigkeit“ des Roboters, da er lediglich nur die direkt benachbarten Felder in seine Richtungsentscheidung einbezieht. Des Weiteren fehlt ihm die Funktion aus der Entfernung Verunreinigungen festzustellen. Ebenfalls fehlt eine Funktionalität, welche zwischen mehreren noch nicht besuchten Richtungen Anhand eines Qualitätsmerkmals bzw. einer Kostenfunktion entscheidet.
Ein optimaler Roboter benötigt also von Anfang an ein Verständnis seiner Umgebung, Fläche und Verschmutzungen, um seinen Pfad planen zu können, sowie eine Suchfunktion, um den kostengünstigsten Pfad zu finden. Im Bezug auf den Roboter ist dieser Pfad derjenige, welcher, von der Startposition ausgehend, alle verunreinigten Felder mindestens einmal und so wenig saubere wie möglich besucht.

Aufgabe 2 – Problemlösen durch Suchen

Teilaufgabe 1.

Breitensuche:

```
Schritt 1: Markiere Start => [ S ]
Schritt 2: Expandiere von Start => [ [S,1], [S,2], [S,7] ]
Schritt 3: Expandiere => [ [S,1,4], [S,2,5], [S,2,6], [S,7,Z] ]
Schritt 4: Ziel gefunden mit Pfad [S,7,Z] => Stopp
```

Tiefensuche:
Da die Suche hier abhängig von der Wahl des expandierten Knoten abhängt, wurden alle Knoten expandiert bis das Ziel oder eine Wiederholung gefunden wurde. Selbiges ist auch im Graphen zu sehen. Eine mögliche Entscheidung wäre z.Bsp. erst den Knoten mit der geringsten Raumnummer zu expandieren. Ist nur ein Folgeknoten vorhanden wurde in der Textuellen Beschreibung ein extra aufschreiben der Zwischenschritte ausgespart.

```
Schritt 1: Markiere Start => [ S ]
S a2: Expandiere mit 1,4,8,Z => [ S,1,4,8,Z ]
S a3: Ziel gefunden mit Pfad [ S,1,4,8,Z ] => Stopp

S b2: Expandiere mit 2 => [ S,2 ]
S b3.w: Expandiere mit 5,2 => Wiederholung
S b3: Expandiere mit 6 => [ S,2,6 ]
S b4.w: Expandiere mit 3,6 => Wiederholung
S b4: Expandiere mit Z => [ S,2,6,Z ]
S b5: Ziel gefunden mit Pfad [ S,2,6,Z ] => Stopp

S c2: Expandiere mit 7,Z => [ S,7,Z ]
S c3: Ziel gefunden mit Pfad [ S,7,Z ] => Stopp
```

A*:
Die voraussichtlichen Kosten werden jeden Schritt für neue Knoten berechnet, die Berechnung wird bei dem ersten Auftreten notiert.

```
Schritt 1: Markiere Start => [ S ]
Kosten: (1):85+272=357; (2):217+219=436; (7):173+383=556
Schritt 2: Expandiere mit 1 => [ [S,1] ]
Kosten: (4):85+80+253=418; (2):436; (7):556
Schritt 3: Expandiere mit 4 => [ [S,1,4] ]
Kosten: (8):85+80+250+57=472; (2):436; (7):556
Schritt 4: Expandiere mit 2 => [ [S,1,4], [S,2] ]
Kosten: (8):472; (5):217+186+318=711; (6):217+103+150=470; (7):556
Schritt 5: Expandiere mit 6 => [ [S,1,4], [S,2,6] ]
```

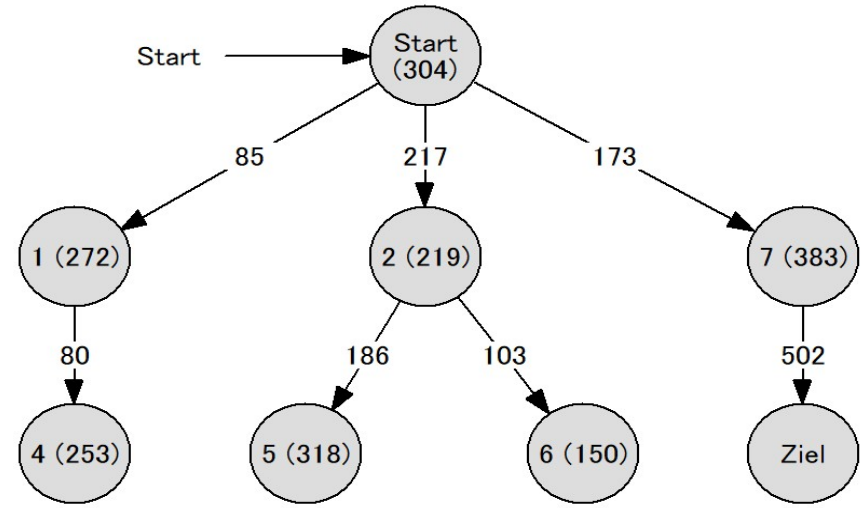
```
Kosten: (8):472; (5):711; (3):217+103+183+189=692; (Z):217+103+167=487; (7):556
Schritt 6: Expandiere mit Z => [ [S,1,4], [S,2,6,Z] ]
Schirtt 7: Ziel gefunden mit Pfad [S,2,6,Z] => Stopp
```

Greedy:

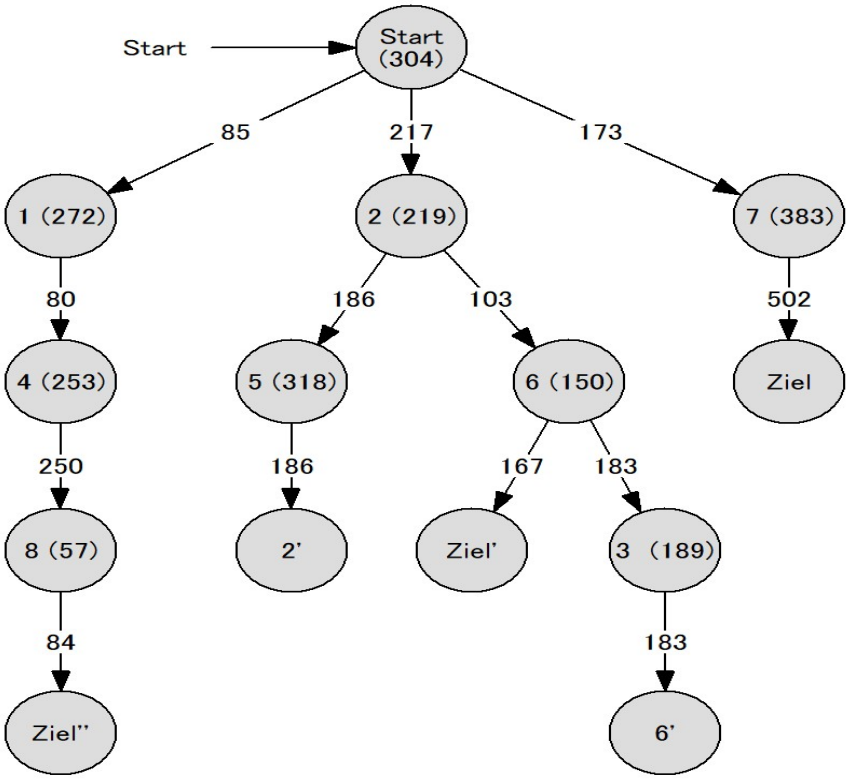
```
Schritt 1: Markiere Start => [ S ]
Kosten: (1):272; (2):219; (7):383
Schritt 2: Expandiere mit 2 => [ S,2 ]
Kosten: (5):318; (6):150
Schritt 3: Expandiere mit 6 => [ S,2,6 ]
Kosten: (3):189; (Z):0
Schritt 4: Expandiere mit Z => [ S,2,6,Z ]
Schritt 5: Ziel gefunden mit Pfad [ S,2,6,Z ] => Stopp
```

Gezeichnete Suchbäume:

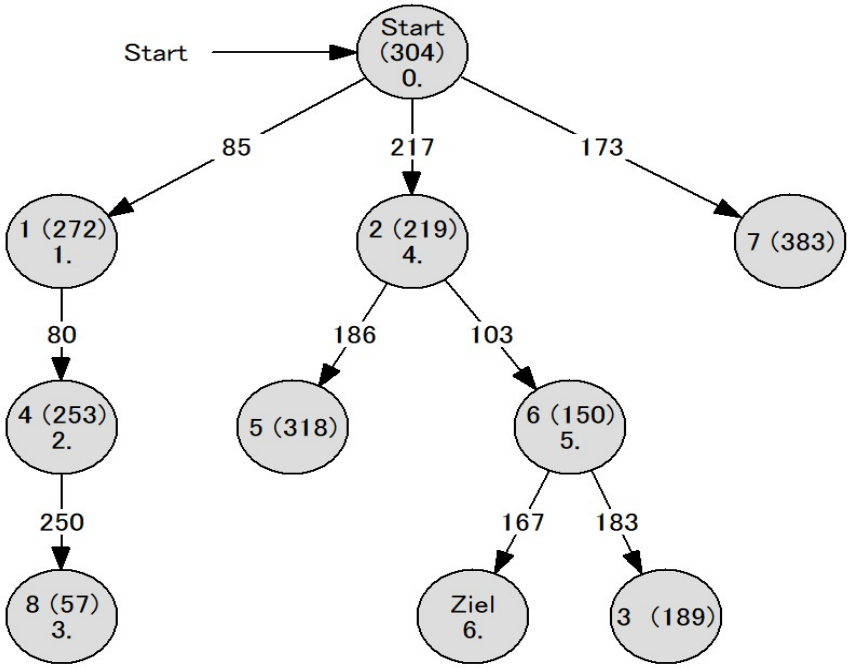
Breitensuche:



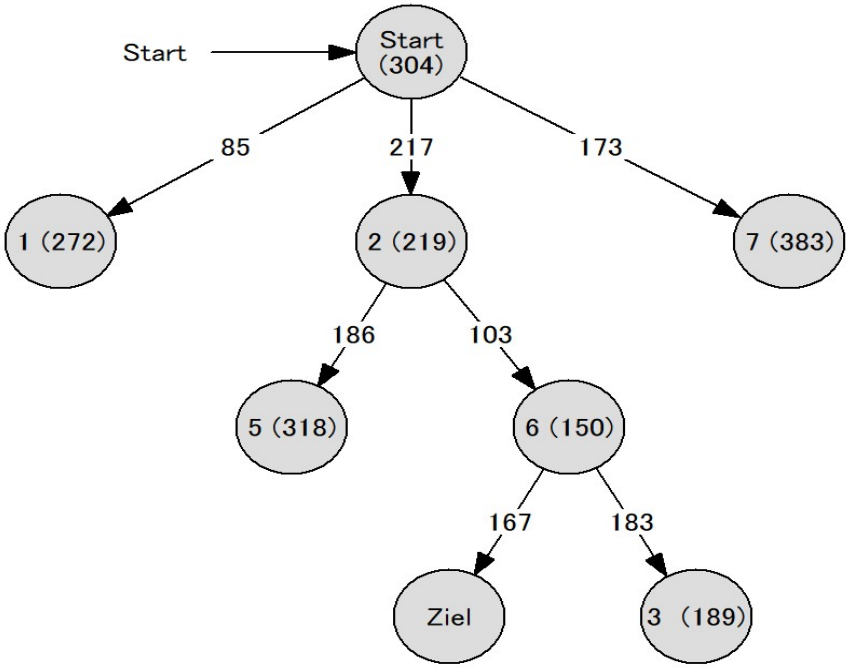
Tiefensuche:



A*:


Start
Start
(304)
0.
85
217
173
1 (272)
1.
2 (219)
4.
7 (383)
80
186
103
4 (253)
2.
5 (318)
6 (150)
5.
250
167
183
8 (57)
3.
Ziel
6.
3 (189)

Greedy:


Start
Start
(304)
85
217
173
1 (272)
2 (219)
7 (383)
186
103
5 (318)
6 (150)
167
183
Ziel
3 (189)

Teilaufgabe 2.

Siehe beigefügte Python-Datei `Hausaufgabe1_aufgabe.py` in den Bereichen zwischen `## S Added` und `## E Added`.