Hinweise zur Bearbeitung von Hausaufgabe Nr. 1

- Die Bearbeitung muss in Gruppen von 3-4 Personen erfolgen
- Abgabe bis 18.05.2017, 23:59 Uhr, per Email an stefan.luedtke2@uni-rostock.de.
- Die Lösungen müssen als **PDF-Datei** abgegeben werden. Zusätzlich muss der Quellcode als **Python-Datei** abgegeben werden.
- Sowohl auf den Lösungsblättern als auch im Quellcode (als Kommentar) müssen die **Namen und Matrikelnummern aller Gruppenmitglieder** angegeben werden.
- Die Zahlen im Rand geben die erreichbaren Punkte pro Aufgabe an.

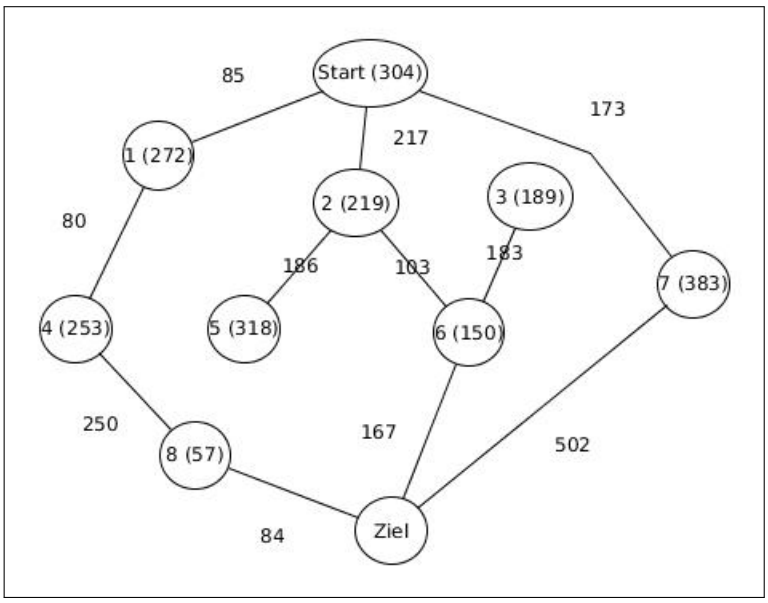
Der Staubsauger-Agent

In dieser Aufgabe soll der aus der Übung bekannte Staubsauger-Agent erweitert werden, sodass er mit einer 2D-Umgegung umgehen kann. Laden Sie dazu die Datei "Staubsauger2d.zip" aus Stud.IP und entpacken sie diese. Die entsprechende Umgebung, inklusive einer GUI, ist darin bereits implementiert. Das Fortschreiten der Zeit kann durch die Leertaste ausgelöst werden. 20

1. Implementieren Sie die die `program`-Methode des `ModelBasedVacuumAgent`. Dieser soll wie folgt funktionieren: Wenn an der aktuellen Position Staub festgestellt wurde, soll gesaugt werden. Wenn nicht, soll sich der Roboter in eine der 4 Richtungen bewegen. Dabei soll eine Richtung gewählt werden, die der Roboter vorher noch nicht gesehen hat. Wenn alle Richtungen schon gesehen wurden, soll zufällig eine gewählt werden. Achten Sie bei der Implementierung darauf, dass der Roboter sich nicht über den "Rand" der Umgebung hinausbewegen kann – wenn doch eine entsprechende Aktion ausgeführt wird, hat dies keine Effekt. Berücksichtigen Sie dies – beispielsweise soll der Roboter nicht wiederholt versuchen, sich in Richtung des Randes zu bewegen, weil das dahinterliegende Feld noch nicht erkundet wurde. 10
2. Ist der so implementierte Agent *optimal*, d.h. reinigt er die Umgebung in der minimalen Anzahl Bewegungen? Warum / warum nicht? Wenn er nicht optimal ist, was müsste ein optimaler Agent tun (hier reicht eine abstrakte, textuelle Beschreibung)? 10

Problemlösen durch Suchen

1. Ein mobiler Roboter soll auf einem Fabrikgelände navigieren. Eine Herausforderung besteht darin, einen Weg zu einem bestimmten Raum (dem Ziel) zu finden. Die Räume sind so verbunden, wie in der folgenden Abbildung dargestellt. Die Heuristik zum Ziel ist an jedem Knoten in Klammern angegeben, die Kosten jeder Kante stehen an den Kanten: 15



a) Benutzen Sie die folgenden Suchalgorithmen um einen Weg zwischen dem Start und dem Ziel zu finden. 10

   i. Breitensuche
   ii. Tiefensuche
   iii. A^*. Notieren Sie die Berechnung für jeden Schritt.
   iv. Greedy. Notieren Sie die Berechnung für jeden Schritt.

b) Zeichnen Sie den entstehenden Suchbaum (analog zu der Darstellung in der Vorlesung) für jeden Suchalgoritmus. 5

2. Gegeben sei die aus der Übung bekannte Formalisierung von Suchproblemen als Graph. 40

a) Laden Sie die Zip-Datei "Hausaufgabe1.zip" aus dem Stud.IP und entpacken sie diese. Öffnen Sie die Datei "Hausaufgabe1_aufgabe.py" und definieren Sie einen Graphen, der das in Aufgabe 1 angegebene Suchproblem modelliert. Definieren Sie die Distanzen so wie in der Abbildung oben zu sehen. 10

b) Im Gegensatz zu den vorherigen Aufgaben wird in dieser Aufgabe die Heuristik nicht aus der Position der Städte *berechnet*, sondern die Heuristik (für einen konkreten Aufangs-Zustand) ist schon angegeben. Ändern Sie die Implementierung von `heuristic(a,b)` entsprechend (die Funktion soll die Heuristik nicht berechnen, sondern in einem Dictionary nachschlagen). 10

c) Implementieren Sie die Greedy-Suche. Diese soll in jedem Schritt den Knoten mit der geringsten Heuristik wählen. 10

d) Implementieren Sie die notwendige `main`-Funktion, so dass der Aufruf

```
1 python YOURFILE.py
```

nach dem Suchalgoritmus fragt (z.B. “Give the search algorithm: (greedy / a*)”) und, nach Angabe, alle notwendigen Schritte mit dem gewählten Algorithmus auf der Konsole ausgibt. Außerdem soll der gefundene Pfad ausgegeben werden. Dafür müssen sie evtl. eine weitere Funktion `reconstruct_path (came_from,start,goal)` implementieren, die aus der Ausgabe des Algorithmus (dem Vorgänger jedes Knotens) den gefundenen Pfad von Start zum Ziel erzeugt. 10