Algorithmen und Datenstrukturen

Sommersemester 2022

Prof. Dr. Philipp Jenke

HAW HAMBURG | Department Informatik

Aufgabe 3

Dieses Aufgabenblatt baut auf den Vorlesungen auf:

- Graphen,
- Graphen-Suche,
- Bäume.

Deliverables:

- PDF mit Lösungen zu den Theorie-Aufgaben
- Code für Praxisaufgabe (Code-Konventionen, Tests)

1 Theorie

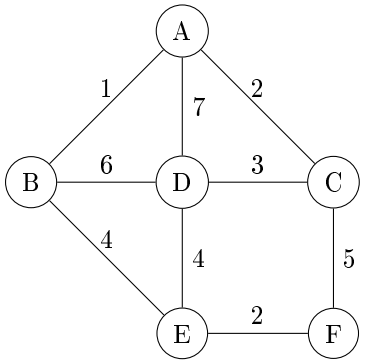
1.1 Minimaler Spannbaum

Bestimmen Sie den minimalen Spannbaum für den folgenden Graphen mit dem Algorithmus von Prim:

Geben Sie die Zwischenschritte und den jeweiligen Zustand der Datenstrukturen bei der Suche an.

1.2 AVL-Baum

Bauen Sie einen AVL-Baum aus einem leeren Baum auf, indem Sie nacheinander die folgenden Elemente einfügen: $15, 17, 20, 10, 12$. Geben Sie die Zwischenschritte und die jeweils (falls notwendig) verwendeten Balancier-Operationen an.



2 Praxis

In dieser Aufgabe verwenden Sie den A*-Algorithmus, um auf einer Hex-Feld-Karte einen Pfad zu planen. Sie implementieren konkret eine Smelloscope[1]-Suche. Auf der Karte befindet sich dazu eine Figur, die ein Ziel erreichen muss. Es ist bekannt, dass das Ziel einen Duft absondert, der umso stärker wird, je näher man dem Ziel kommt. Hat die Figur das Ziel erreicht, wird direkt ein neues Ziel definiert.

Im Vorgabeframework finden Sie eine Visualisierung einer Karte. Der Geruch, der als Heuristik verwendet werden soll, ist in den Zellen farbcodiert. Ihr Einstieg in den Code findet sich in der Anwendungsklasse `Smelloscope`. Konkret müssen Sie dort den berechnten Pfad über die Methode

```
List<Cell> getPath(Cell start, Cell target)
```

setzen.

2.1 Graph

Implementieren Sie eine Datenstruktur für einen Graphen basierend auf einer Adjazenzliste in einer Klasse `Graph`. Die in den Knoten referenzierten Elemente sollen variabel bleiben, verwenden Sie also Generics. Testen Sie die Klasse mit eigenen kleinen Test-Graphen. Der Graph soll nicht für die Smelloscope-Anwendung und auch nicht für den A* angepasst sein. Beispielsweise müssen im Graphen auch andere Elemente als `Cell` möglich sein und die Heuristik soll nicht Teil des Graphen sein.

Anschließend müssen Sie für die Karte einen passenden Graphen aufbauen. Legen Sie für jede Zelle einen Knoten im Graphen an. Die Zelle ist jeweils als Element im Knoten abgelegt. Auf die Zellen können Sie über den entsprechenden Iterator (`getCellIterator()`) in der Klasse `Map` zugreifen. Jede Zelle kann bis zu sechs Nachbarzellen haben, die Sie mit der Methode `getNeighborCell()` erreichen. Eine Zelle kann auch belegt sein (Baum in der Visualisierung). Solche Zellen sollen nicht im Graphen erscheinen. Ob eine Zelle belegt ist, prüfen Sie mit `isOccupied()`.

[1] *https://futurama.fandom.com/wiki/Smell-O-Scope*, abgerufen am 9.7.2020

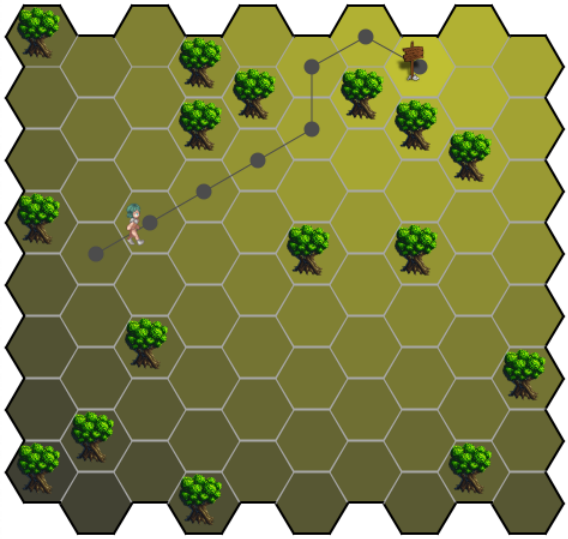


Abbildung 1: Pfadsuche in einer Hex-Karte mit A*. Die Heuristik ist durch die Farbe der Zellen codiert. Der berechnete Pfad ist als Linienzug visualisiert.

2.2 A*-Suche

Implementieren Sie nun für den Graphen die A*-Suche in einer Klasse `AStar`. Testen Sie die Klasse mit den Beispielen für A* aus der Vorlesung. Die `AStar`-Klasse soll nicht für das Smelloscope maßgeschneidert sein. Sie soll also für beliebige Graphen anwendbar sein. Entwickeln Sie eine Abstraktionsebene (z.B. ein Interface) für die Heuristik. Für die Smelloscope-Anwendung brauchen Sie denn eine passende Implementierung, um die Geruchswerte für den A* zur Verfügung zu stellen.

Integrieren Sie eine Instanz der Klasse dann in die Smelloscope-Vorgabe. Als Heuristik verwenden Sie den Duft-Wert der Zellen. Diesen können Sie für jede Zelle mit `getSmell()` abfragen. Der Duft-Wert wird kleiner, je näher Sie dem Ziel kommen (vergleiche: der Scheinriese Tur Tur bei Jim Knopf). Der Duft-Wert wird aktualisiert, wenn ein neues Ziel gewählt wurde. Bei jedem Aufruf der oben genannten Methode `getPath(Cell start, Cell target)` müssen Sie eine Liste von Zellen auf dem (kürzesten) Pfad von der Zelle `start` bis zur Zelle `ziel` erzeugen. `start` und `ziel` müssen beide in der Liste enthalten sein.

Die Heuristik ist ziemlich gut. Es ist also zu erwarten, dass der A*-Algorithmus nur wenige Zellen prüfen muss. Lassen Sie sich die Anzahl der geprüften Zellen ausgeben.

Hinweis: Die Karten werden bei jedem Start zufällig generiert. Daher kann es vorkommen, dass es in der Karte Sackgassen gibt und man ein Ziel nicht erreichen kann. Dieser Fall wird nicht abgefangen. Starten Sie in dem Fall die Anwendung einfach neu.