Übungsblatt 6

Abgabe am 26.11.2018

Aufgabe 1: Implementation von A* (5+2+2 Punkte)

Implementieren Sie den A^* Algorithmus aus der Vorlesung. Dazu stehen Ihnen im Moodle die Dateien `AStarGUI.java`, die lediglich für die Visualisierung sorgt, und `AStar.java`, in der Sie Ihre Implementierung schreiben, zur Verfügung. Wenn Sie `AStarGUI.java` laufen lassen, werden Sie eine Fläche mit Hindernissen und einen Start- und Zielpunkt sehen.

a) Füllen Sie die Funktion `solveAStar()` mit dem A* Algorithmus. Wenn Sie den Knopf "Start" in der Benutzeroberfläche der GUI drücken, wird diese Funktion ausgeführt und die Suche und der gefundene Pfad animiert. Mehr dazu in der Datei `AStar.java`. Verwenden Sie als Heuristik zunächst die Manhattan-Metrik. Benutzen Sie außerdem die Klasse `QueueItem`, die bereits in der Datei `AStar.java` steht.

b) Füllen Sie die Funktion `solveDijkstra()` in der Datei `Dijkstra.java` mit dem Dijkstra Algorithmus.

c) Vergleichen Sie Dijkstra und den A* Algorithmus. Was fällt besonders auf? Lassen Sie A* auch einmal mit der euklidischen Distanz als Heuristik laufen und beziehen Sie das Ergebnis ebenfalls in den Vergleich mit ein (schriftlich auf ihrer Abgabe).

Hinweis zur Abgabe: Bitte laden Sie dieses Mal die fertigen Dateien `AStar.java` und `Dijkstra.java` im Moodle hoch. Das können Sie unter 'Abgabe - AStar' in der Woche 6 erledigen, aber höchstens bis zum **26.11.2018, 15:45 Uhr**.

Aufgabe 2: Approximation stückweise linearer Funktionen (3 + 3 Punkte)

Gegeben seien n Paare von Punkten $(x_i, f(x_i)) \in \mathbb{R}^2$, $i = 1, \ldots, n$, mit $x_1 < x_2 < \ldots < x_n$ für eine Funktion $f : \mathbb{R} \to \mathbb{R}$. Wir wollen f durch eine möglichst einfache Funktion g approximieren, für die $g(x_i) = f(x_i)$ für manche (nicht notwendigerweise alle) x_i gilt und die zwischen diesen x_i linear ist.

Es gilt also eine Balance zu finden zwischen der Anzahl linearer Segmente von g und dem Approximationsfehler. Formal wollen wir für gegebene Parameter $\alpha, \beta > 0$ den Ausdruck

$$\alpha \cdot (\text{Anzahl linearer Segmente von } g) + \beta \cdot \sum_{i=2}^{n-1} |f(x_i) - g(x_i)|^2 \tag{1}$$

über alle Funktionen g minimieren, die folgendes erfüllen:

- $g(x_1) = f(x_1)$, $g(x_n) = f(x_n)$,
- g ist stückweise linear und Ecken von g sitzen nur auf Punkten $(x_i, f(x_i))$.

a) Zeigen Sie, dass sich dieses Problem als graphentheoretisches Problem formulieren lässt und beschreiben Sie, wie es sich lösen lässt.

b) Lösen Sie das Problem für $\alpha = \beta = 1$ und

$$(x_1, f(x_1)) = (0, 0), \quad (x_2, f(x_2)) = (5, 5), \quad (x_3, f(x_3)) = (10, 2), \quad (x_4, f(x_4)) = (15, 0),$$

d.h. berechnen Sie die Funktion g, die (1) für diese Daten minimiert.

Aufgabe 3: Floyd-Warshall-Algorithmus (5 Punkte)

Die in der Vorlesung präsentierte Variante des Floyd-Warshall-Algorithmus berechnet nur die

Kürzeste-Pfad-Distanz zwischen allen Knoten, nicht aber kürzeste Pfade selbst (also Pfade, die die Kürzeste-Pfad-Distanz realisieren). Adaptieren Sie den Algorithmus so, dass er kürzeste Pfade repräsentiert durch eine Vorgängermatrix mitberechnet. Geben Sie den adaptierten Algorithmus in Pseudo-Code an und begründen Sie seine Korrektheit.

Freiwillige Zusatzaufgabe 1 (1+1+2 Punkte) Stellen Sie sich vor, Sie gründen ein Unternehmen, dass mit Drohnen kleine Pakete oder andere Güter innerhalb einer Gruppe von Städten V transportiert. In jeder Stadt machen Sie Gewinn: Ein Besuch in Stadt i erzeugt einen Profit von p_i Euro. Allerdings kostet der Transport von Stadt i zu Stadt j auch $c_{ij} > 0$ Euro. Sie möchten nun eine zyklische Route finden, sodass das Verhältnis von Profit zu Kosten maximiert wird. Betrachten Sie dafür den Graphen $G = (V, E)$, dessen Knoten die Städte sind und die Kanten die Verbindungen zwischen je zwei Städten. Für einen beliebigen (einfachen) Kreis C in diesem Graphen, sei das Profit-zu-Kosten Verhältnis gegeben als

$$r(C) = \frac{\sum_{(i,j)\in C} p_j}{\sum_{(i,j)\in C} c_{ij}}.$$

Sei r^* das maximal mögliche Verhältnis. Eine Möglichkeit, r^* zu bestimmen, ist die binäre Suche: zunächst raten wir ein Verhältnis r und testen daraufhin, ob es zu groß oder zu klein ist. Betrachten Sie nun ein beliebiges $r > 0$. Geben Sie jeder Kante (i, j) das Gewicht $w_{ij} = rc_{ij} - p_j$.

a) Zeigen Sie, dass $r < r^*$ gilt, wenn es einen Kreis mit negativem Gewicht gibt.

b) Zeigen Sie, dass $r > r^*$ gilt, wenn alle Kreise im Graphen strikt positive Gewichte haben.

c) Geben Sie einen effizienten Algorithmus an, der als Input eine gewünschte Genauigkeit $\varepsilon > 0$ verlangt und einen einfachen Kreis C zurückgibt, für den gilt $r(C) \geq r^* - \varepsilon$. Begründen Sie die Korrektheit Ihres Algorithmus und untersuchen Sie die Laufzeit in Abhängigkeit von $|V|, \varepsilon$ und $R = \max_{(i,j)\in E}(p_j/c_{ij})$.