Kurztext zu Gallenbacher Kap. 1

Inhaltliche Schritte

- Einführung in die Problemstellung Routenplaner
- Wahl einer Miniwelt
- Ausgangspunkt Landkarte

1. Aufgabe:
   Versuchen Sie eine **Lösung selbst** herauszufinden.

- **Reflexionsphase**: Wie sind Sie vorgegangen?
  - Vermutung brute-force
  - Probleme von **brute-force** → Berechnungsaufwand eventuell zu groß
- Methode der Abstraktion:
  Reduzieren der Informationen auf das für das Problem Wesentliche

2. Aufgabe:
   Suchen Sie nach Informationstypen, notieren Sie, bewerten Sie.

- Neuzeichnen der Karte, allein mit Symbolen für die Namen der Städte, den Linien als Wegen und ihren Längen.
- Methode der Gleichformung:
  ...auf Grundelemente zurückführen.

von der Karte zur Arbeitsvorlage

3. Aufgabe:
   Überlegen Sie, was die Eigenschaften der "normalen" Elemente sind und ob man die speziellen nicht auch wie normale darstellen kann.

- Neuzeichnen der Karte, da auch Kreuzungen ohne Städte bei dieser Problemstellung wie Städte behandelt werden können.
- Einschub:
  Fachausdruck für das Erreichte ist **Graph**; Begriffe **Knoten** und **Kanten**

Dijkstra und die Ameisen

- Bild des **Ameisenstamms** für das Suchen der kürzesten Verbindung zu einem Nachbarort
- Methodisch:
  farbige Markierung der Wege
- **Reflexionsphase**:
  Erreicht ist: kürzester Weg von ***I*** nach ***B*** ist gefunden.

anders bei Gallenbacher!

- Dort weiter, gleichzeitig aber die anderen **Wege halten**, so dass der nächste kürzeste Weg zu ***C*** mit 40 gefunden wird.

4. Aufgabe:
   Finden Sie heraus, was im nächsten Schritt passiert.

- **Reflexionsphase**:
  In dem Moment, wo ***M*** mit der Länge 43 erreicht wird, ist klar, dass der Weg von ***C*** zu ***M*** aus dem Graphen der kürzesten Wege entfernt werden kann.
- Erreichen der Knoten ***X*** und ***P***, Streichen der Kanten ***B-X*** und ***C-X***.

5. Aufgabe:
   Führen Sie das Verfahren zu Ende.

- Ergebnis vergleichen: ***I-P-K-F-O***, Länge 123.
- Reflexionsphase: Interessant für Informatiker:
  - **Reduzierung des Aufwands** gegenüber brute-force
  - Algorithmisierung möglich
- Begriffsdefinition Algorithmus

Erarbeitung der Formulierung des Algorithmus von Dijkstra

- Die ersten Schritte werden noch einmal mit auf die Formalisierung fokussierten Formulierungen vollzogen.
- Angabe der Algorithmus-Beschreibung

6. Aufgabe:

- Führen Sie nun den Algorithmus für das gegebene Beispiel fort.
- Üben Sie am Beispiel des kürzesten Weges von ***L*** nach ***E***.

Formulierung des Ergebnisses

Der Algorithmus von Dijkstra bestimmt immer die kürzesten Wege von einem Startpunkt zu vielen anderen Punkten.

7. Aufgabe:
   Untersuchen Sie, wie das Navigationssystem es nach der erstmaligen Berechnung des Weges schafft, in kurzer Zeit ohne eine Neuberechnung auf ein Verhalten des Fahrers zu reagieren, der sich nicht an den optimalen Weg gehalten hat.

- Reflexionsphase:
  Berechnung des Aufwandes bei brute-force (exponentiell) und Dijkstra (quadratisch) im Vergleich.
  np-vollständige Probleme

Der Algorithmus von Dijkstra

Beschreibung entsprechend der Vorlage von Gallenbacher[1]

Algorithmus – Beschreibung bei Gallenbacher:

0. Markiere die Startstadt rot, weise ihr die Kennzahl 0 zu. Bezeichne diese als aktuelle Stadt.
1. Gehe aus von der aktuellen Stadt zu allen direkt erreichbaren Nachbarstädten.

... und führe das Folgende für jede Nachbarstadt durch:

Errechne die Summe aus der Kennzahl an der aktuellen Stadt und der Länge der Strecke dorthin.

- Ist die Nachbarstadt bereits rot markiert, mache nichts.
- Hat die Nachbarstadt keine Kennzahl, weise ihr die Summe als Kennzahl zu. markiere die Strecke zur aktuellen Stadt.
- Hat die Nachbarstadt eine Kennzahl größer der Summe, streiche die dortige Kennzahl sowie die Markierung. Weise ihr danach die Summe als neue Kennzahl zu. Markiere die Strecke zur aktuellen Stadt.

2. Betrachte alle Städte, die zwar eine Kennzahl haben, aber noch nicht rot markiert sind. Suche die Stadt mit der kleinsten Kennzahl.
3. Bezeichne diese als aktuelle Stadt. Weisen mehrere Städte die kleinste Kennzahl auf, wähle eine beliebige davon als aktuelle Stadt.
4. Markiere die aktuelle Stadt rot, zeichne die dort markierte Strecke in rot ein.
5. Falls es noch Städte gibt, die nicht rot markiert sind, weiter bei (1.)

Diese Formulierung lässt sich nicht ganz einfach in Scheme umsetzen, da sie sich an prozeduraler / objektorientierter Programmierung orientiert und nicht an funktionaler. Insbesondere der Punkt 5 ist ärgerlich, hier sollte besser eine Formulierung wie "...bis es keine Städte mehr gibt, die nicht rot markiert sind" gewählt werden.

1 Jens Gallenbacher: Abenteuer Informatik; Spektrum Akademischer Verlag