Aufgabe 4: Urlaubsfahrt

Team-ID: 00587

Team-Name: Doge.NET

Bearbeiter dieser Aufgabe:
Johannes von Stoephasius

19. November 2019

Inhaltsverzeichnis

1 Lösungsidee

1.1 Definitionen

Im Folgenden wird ”Weg” als Alias für eine Menge von zu besuchenden Tankstellen verwendet.

Weiterhin wird mit ”bestem Weg” oder ”optimalem Weg”, der Weg mit den wenigsten Stopps, und geringsten Preis um vom Ausganspunkt zu einer gegebenen Tankstelle zu kommen bezeichnet. Hierbei wird wie in der Aufgabenstellung eine minimal Stoppzahl priorisiert.

1.2 Kernidee

Der Algorithmus ist in 2 Teile geteilt; Der Nebenalgorithmus errechnet den günstigsten Preis, der benötigt ist um einen bestimmten Weg abzufahren.

Der Hauptalgorithmus errechnet, unter Verwendung des Nebenalgorithmus, den optimalen Weg von einer Tankstelle zu einer anderen.

1.3 Hauptalgorithmus

Der Hauptalgorithmus baut nach und nach eine Zuordnung auf, die für jede Tankstelle den besten Weg zu ihr enthält.

Diese Zuordnung wird für eine beliebige Tankstelle ermittelt, indem zuerst für alle vorherigen Tankstellen eine Zuordnung ermittelt wird. Sind die optimalen Wege zu allen Tankstellen vor einer gegebenen Tankstelle errechnet, kann die Ermittlung des optimalen Wegs zu dieser beginnen. Zuerst werden alle

optimale Wege zu den vorherigen Tankstelle durchiteriert. Dabei wird zuerst überprüft von welchen dieser Tankstellen die aktuell betrachtete Tankstelle überhaupt erreicht werden kann.
Wenn nicht wird zur nächsten Tankstelle derr Zuordnung übergegangen.
Wenn doch, so wird ein neuer Weg gebildet, der aus dem vorherigen Weg und der betrachtete Tankstelle besteht.
Von allen dieser neuen Wege werden nun die kürzesten ausgewählt, also alle Wege mit einer Stoppzahl gleich der minimalen Stoppzahl. Danach wird der Preis aller übrigen Wege ermittelt, und von denen wird der billigste ausgewählt. Dieser Weg ist dann der optimale Weg vom Start zur aktuell betrachteten Tankstelle. Dieser wird dann in die Zuordnung eingetragen.

Dieses Verfahren wird für alle Tankstellen iterativ durchgeführt, bis der optimale Weg zum Streckenende Teil der Zuordnung ist. Dieser ist dann der optimale Weg vom Start zum Streckenende.

1.4 Nebenalgorithmus

Der Nebenalgorithmus berechnet den besten Preis für einen gegebenen Weg.

Dafür werden vorerst die Tankstellen dem Preis nach aufsteigend geordnet. Zuerst wird von der preiswertesten Tankstelle aus eine Strecke definiert, die von der Tankstelle aus über die maximalen Tanklänge reicht, oder wenn das Streckenende in dieser enthalten ist, bis zum Ziel geht.

Danach wird für die nächst-preiswerteste Tankstelle auch eine solche Strecke definiert, es sei denn, es gibt Überschneidungen mit einer bereits eingetragenen Strecke. In diesem Fall wird die bereits eingetragene Strecke durch maximales Volltanken des halb-leeren Tanks an dieser Tankstelle erweitert.

Dieses Verfahren wird so lange wiederholt, bis nur noch eine Strecke existiert, die den gesamten Weg abdeckt. Dieses Verfahren liefert das ideale Ergebnis, da preislich aufsteigend immer die beste Teillösung gefunden wird.

2 Umsetzung

2.1 Codestruktur

- Urlaubsfahrt.cs
  - enthält die statische Klasse `Urlaubsfahrt`, die die statische Methode `GetTrack` enthält, die den Hauptalgorithmus ausführt
  - `GetTrack` erhält als Parameter:
    - die Streckenlänge
    - das Start-Benzin
    - die mit einer maximalen Tankfüllung zurückgelegt werden kann
    - eine Liste aller Tankstellen
- GasStation.cs
  - enthält die Position und den Benzinpreis einer Tankstelle
- Track.cs
  - berechnet für einen Weg den besten Preis, unabhängig von den benötigten Stopps
  - enthält eine Liste von allen Tankstellen des Weges
  - ein Track kann folgendermaßen gebildet werden:
    - aus einer Tankstelle
    - aus einem Track und einer Tankstelle
    - aus nichts; das statische Feld `EmptyTrack` repräsentiert einen komplett leeren Weg
  - die Methode `GetPriceTo` führt den Nebenalgorithmus aus, und berechnet entweder bis zu einem arbiträren Punkt, oder zu einer Tankstelle den Preis aus
- Extensions.cs
  - enthält Methoden, die generische Klassen oder Interfaces erweitern
    - `AllMins<TSource>` gibt von einem IEnumerable von TSource die Elemente zurück, die bei einer Funktion von TSource zu `IComparable<T>` gleichwertig zum Minimum sind

2.1.1 Implementierung

3 Beispiele

Genügend Beispiele einbinden! Die Beispiele von der BwInf-Webseite sollten hier diskutiert werden, aber auch eigene Beispiele sind sehr gut – besonders wenn sie Spezialfälle abdecken. Aber bitte nicht 30 Seiten Programmausgabe hier einfügen!

4 Quellcode

Unwichtige Teile des Programms sollen hier nicht abgedruckt werden. Dieser Teil sollte nicht mehr als 2–3 Seiten umfassen, maximal 10.