Aufgabe 2: Vollgeladen

Team-ID: 00564

Team-Name: SRZ info3 Gruppe 1

Bearbeiter dieser Aufgabe:
Bruno Hoffmann

16. November 2021

Inhaltsverzeichnis

1. Lösungsidee

Um den besten Pfad zu finden, geht der Algorithmus rekursiv über alle möglichen Fahrtmöglichkeiten in dem er einen Baum generiert und evaluiert die minimale Bewertung der Hotels des Pfades. Meine Lösung ist dabei in C++ 20 implementiert.

Für alle Hotels, die man innerhalb eines Tag vom Startpunkt erreichen kann, wird die Bewertung bestimmt, und wenn diese geringer als die bereits festgestellte Minimalpfadbewertung ist, nicht weiter verfolgt, da bereits ein besserer Pfad gefunden wurde. Ist dies nicht der Fall, wird dieser Prozess für alle Folgehotels innerhalb der Eintagesreichweite, und die Folgehotels in deren Eintagesreichweite wiederholt, bis man entweder nah genug an dem Zielort ist, oder fünf Tage unterwegs war. Wenn man das Ziel erreicht hat, wird die Minimalbewertung des Pfades bestimmt, und wenn diese größer ist, als die des schon registrierten Pfades, wird der Pfad als bester gespeichert und die minimale Bewertung aktualisiert.

2. Umsetzung

Um den Algorithmus umzusetzen verwende ich eine Funktion „eval“ (siehe Quellcode), die als Attribute das aktuelle Hotel, die davor besuchten Hotels als Array und das aktuelle Level (Anzahl der besuchten Hotels) beinhaltet. Ist die Bewertung des aktuellen Hotels kleiner als die bereits festgestellte minimale Bewertung eines Pfades, wird die Funktion beendet, da bereits ein besserer Pfad gefunden wurde und eine Weiterführung des Pfades kein besseres Ergebnis ergeben könnte. Ist das Level größer/gleich 5 oder ist das aktuelle Hotel in Reichweite des Ziels, wird der Evaluierungsprozess eingeleitet, wobei die Minimalbewertung ermittelt wird, und wenn diese höher als die davor festgestellte ist, wird der genommene Pfad und die Minimalbewertung global gespeichert.

Die Hotels werden einfach aus der Datei eingelesen, deren Pfad man in die Konsole eingegeben hat. Kann diese keine Hotels aus der angegebenen Datei auslesen wird ein Fehler ausgegeben.

3. Beispiele

Gesetzt dem Fall, dass es keine Möglichkeit gibt, das Ziel zu erreichen, sei es durch eine zu große benötigte Zeit zwischen zwei Hotels, oder eine allgemein zu große Distanz zu dem Ziel zu groß ist, um es in 5 Tagen zu erreichen, gibt das Programm aus, dass

Wenn es zwei Pfade mit gleicher Bewertung aber unterschiedlichen benötigten Zeiten gibt, wir der Pfad ausgewählt, der weniger Zeit benötigt, da dies auch im Interesse der Fahrer liegen würde, nicht unnötig Zeit zu verschwenden.

4. Quellcode

```
void eval(int hotel, int prevs[], int level = 0) {
	// better path was already found
	if (hotels[hotel].rating <= bestPathRating) return;

	// adding the current hotel to the "list" of previous hotels
	prevs[level] = hotel;

	//if target is in range
	if (endPointTime - hotels[hotel].time <= maxDrivingTime) {
		// get worst rating of path
		double minRating = 10.0;
		for (int i = 0; i < level + 1; i++) {
			if (hotels[prevs[i]].rating < minRating)
				minRating = hotels[prevs[i]].rating;
		}

		// if path is better
		if (bestPathRating < minRating) {
			bestPathRating = minRating;
			bestPathTime = level;
			for (int i = 0; i <= level; i++) {
				bestPath[i] = prevs[i];
			}
			return;
		}
	}

	// needed more than 5 days
	if (level >= maxHotels - 1) return;


	// execute the eval function for all hotels in range
	for (int i = hotel + 1; i < hotelCount; i++) {
		if (hotels[i].time - hotels[hotel].time > maxDrivingTime) break;

		eval(i, prevs, level + 1);
	}
}
```