Runde 1 Aufgabe 2: Vollgeladen

Bearbeiter/-in dieser Aufgabe:
Joshua Benning

7. September 2021

Inhaltsverzeichnis

Aufgabenstellung

Familie Meier will mit dem Auto zu Verwandten in Portugal reisen. Die Reiseroute steht fest, und ohne Übernachtungen wird es nicht gehen. Die Meiers haben deshalb zahlreiche Hotels entlang der Route im Internet angesehen und für jedes eine Bewertung ermittelt. Die Geschwister Lara und Paul dürfen auswählen, in welchen Hotels übernachtet wird. Im schon etwas betagten Auto der Meiers können die beiden ihre Handys nicht aufladen. Deshalb können sie pro Tag maximal sechs Stunden lang fahren. Insgesamt darf die Reise höchstens fünf Tage dauern. Unter diesen Bedingungen wollen Lara und Paul möglichst schöne Hotels auswählen: Die niedrigste Bewertung eines Hotels ihrer Auswahl soll so hoch sein wie möglich.

Schreibe ein Programm, das die Auswahl der Hotels übernimmt. Dein Programm soll die Angaben zu den Hotels, nämlichihre Position entlang der Route und ihre Bewertungen einlesen und eine Auswahl nach den Wünschen von Lara und Paul ausgeben.

Lösungsidee

Ich baue beginnend mit dem Startpunkt einen gerichteten Graphen auf und finde darin den effizientesten Pfad.

Umsetzung

Die Implementierung erfolgt in Java. Beginnend bei dem Startpunkt (Wegpunkt im weiteren Text auch Node) betrachte ich die möglichen nächsten Haltepunkte. Ob eine Node als Wegpunkt in Frage kommt, entscheide ich dabei nach 2 Kriterien. 1. Ob die Node in der maximalen Fahrzeit von 360 Minuten erreichbar ist und 2. Ob bei Auswahl der Node das Ziel überhaupt rechnerisch noch erreichbar ist. Sprich ob die verbleibenden Nächte multipliziert mit 360 größer sind als die verbleibende Distanz. Hier bieten sich nun zwei mögliche Lösungen an. Man könnte mit einem Deepsearch-Algorithmus alle mögliche Pfade bestimmen oder hier bereits das am besten bewertete Hotel verwenden. Dabei bestünde natürlich die Möglichkeit in eine „Sackgasse“ zu geraten. Dies ist aber durch die Auswahl der möglichen Children ausgeschlossen. Daher verwende ich einen „Greedy“-Algorithmus, sprich gehe nur den besten Pfad entlang. Während eine Tiefensuche der maximalen Kantenlänge 4 immer eine Laufzeit von 4n besitzt, variiert die des Greedy-Algorithmus je nach Bewertungen der Hotels und erreicht nur bei immer gleicher Bewertung eine Laufzeit von 4n . Im Normalfall ist also der Greedy-Algorithmus deutlich effizienter.

Beispiele

Beispiel 1

Wegpunkt bei Minute 1680, Nacht:5, Bewertung:5.0, Typ:TARGET
Wegpunkt bei Minute 1360, Nacht:4, Bewertung:2.8, Typ:HOTEL
Wegpunkt bei Minute 1007, Nacht:3, Bewertung:2.8, Typ:HOTEL
Wegpunkt bei Minute 687, Nacht:2, Bewertung:4.4, Typ:HOTEL
Wegpunkt bei Minute 347, Nacht:1, Bewertung:2.7, Typ:HOTEL
Wegpunkt bei Minute 0, Nacht:0, Bewertung:5.0, Typ:START
Benötigte Zeit: 0.028 s Niedrigste Bewertung: 2.7

Beispiel 2

Wegpunkt bei Minute 1737, Nacht:5, Bewertung:5.0, Typ:TARGET
Wegpunkt bei Minute 1380, Nacht:4, Bewertung:5.0, Typ:HOTEL
Wegpunkt bei Minute 1053, Nacht:3, Bewertung:4.8, Typ:HOTEL
Wegpunkt bei Minute 700, Nacht:2, Bewertung:3.0, Typ:HOTEL
Wegpunkt bei Minute 341, Nacht:1, Bewertung:2.3, Typ:HOTEL
Wegpunkt bei Minute 0, Nacht:0, Bewertung:5.0, Typ:START
Benötigte Zeit: 0.039 s Niedrigste Bewertung: 2.3

Beispiel 3

Wegpunkt bei Minute 1793, Nacht:5, Bewertung:5.0, Typ:TARGET
Wegpunkt bei Minute 1433, Nacht:4, Bewertung:1.7, Typ:HOTEL
Wegpunkt bei Minute 1076, Nacht:3, Bewertung:3.8, Typ:HOTEL
Wegpunkt bei Minute 717, Nacht:2, Bewertung:0.3, Typ:HOTEL
Wegpunkt bei Minute 360, Nacht:1, Bewertung:1.0, Typ:HOTEL
Wegpunkt bei Minute 0, Nacht:0, Bewertung:5.0, Typ:START
Benötigte Zeit:0.036 s Niedrigste Bewertung: 0.3

Beispiel 4

Wegpunkt bei Minute 1510, Nacht:5, Bewertung:5.0, Typ:TARGET
Wegpunkt bei Minute 1301, Nacht:4, Bewertung:5.0, Typ:HOTEL
Wegpunkt bei Minute 979, Nacht:3, Bewertung:4.7, Typ:HOTEL
Wegpunkt bei Minute 658, Nacht:2, Bewertung:4.6, Typ:HOTEL
Wegpunkt bei Minute 340, Nacht:1, Bewertung:4.6, Typ:HOTEL
Wegpunkt bei Minute 0, Nacht:0, Bewertung:5.0, Typ:START
Benötigte Zeit: 0.094 s Niedrigste Bewertung: 4.6

Beispiel 5

Wegpunkt bei Minute 1616, Nacht:5, Bewertung:5.0, Typ:TARGET
Wegpunkt bei Minute 1271, Nacht:4, Bewertung:5.0, Typ:HOTEL
Wegpunkt bei Minute 987, Nacht:3, Bewertung:5.0, Typ:HOTEL
Wegpunkt bei Minute 636, Nacht:2, Bewertung:5.0, Typ:HOTEL
Wegpunkt bei Minute 317, Nacht:1, Bewertung:5.0, Typ:HOTEL
Wegpunkt bei Minute 0, Nacht:0, Bewertung:5.0, Typ:START
Benötigte Zeit: 0.063 s Niedrigste Bewertung: 5.0

Quellcode

```java
/**
 * Gibt alle Nodes in Range einer Node an
 * @param node Ausgangsnode
 * @param targetPosition Position des Zielpunkts
 * @return Gibt eine Liste von Nodes an die im Zeitlimit liegen und
 * mathemathisch zielführend sind
 */
private List<Node> getNodesInRange(Node node, int targetPosition) {
    int startPosition = node.getPosition();
    List<Node> nodesInRange = new ArrayList<>();
    for (Node nodeToCheck : nodes) {
        if (nodeToCheck.getPosition() > startPosition &&
                nodeToCheck.getPosition() - startPosition <= maxDistancePerLine
                && canBeUsedToReachTarget(nodeToCheck, targetPosition, node.getNight())) {
            nodesInRange.add(nodeToCheck);
        }
    }
    return nodesInRange;
}

/**
 * Methode zur Berechnung des optimalen Pfads
 * @param targetPosition Position des Ziels
 * @return Gibt die Zielnode zurück wobei über die Parents der Pfad hergeleitet werden kann
 */
public Node calculateOptimalPath(int targetPosition) {
    // Liste aller Optionen für das Weiterfahren
    List<Node> openNodes = new ArrayList<>();

    Node start = new Node(0, 0, 5f, Type.START);
    this.nodes.add(new Node(targetPosition, 0, 5f, Type.TARGET));
    openNodes.add(start);

    while (!openNodes.isEmpty()) {
        Node current = openNodes.get(0);
        if (current.getNight() > this.maxLines - 1) {
            openNodes.remove(current);
            continue;
        }
        for (Node node : getNodesInRange(current, targetPosition)) {
            node.setParent(current);
            node.setNight(current.getNight() + 1);
            if (node.getParentedRating() >= current.getParentedRating()) {
                current.setParentedRating(node.getParentedRating());
            }
            if (node.getType() == Type.TARGET) {
                return node;
            }
            openNodes.add(node);
            openNodes.sort(Comparator.comparing(Node::getParentedRating).reversed());
        }
        openNodes.remove(current);
    }
    return null;
}
```

```java
/**
 * Methode um zu überprüfen ob bei Auswahl einer Node das Ziel immer noch erreicht werden kann
 * @param node Zu überprüfende Node
 * @param targetPosition Zielposition
 * @param night Die momentane Nacht
 * @return Wahr wenn die Node verwendet werden kann
 */
private boolean canBeUsedToReachTarget(Node node, int targetPosition, int night) {
    // Check if the node at its current night can be used to reach the target
    int remainingDistance = targetPosition - node.getPosition();
    int possiblyTraveledDistance = (5 - night - 1) * 360;
    return remainingDistance - possiblyTraveledDistance <= 0;
}


/**
 * Modellklasse zur Darstellung des Graphens
 */
public class Node implements Cloneable {
    private final int position;
    private int night;
    private final float rating;
    private final Type type;
    private float parentedRating;
    private Node parent;

    /**
     * @param position Position des Wegpunkts
     * @param depth Nacht in der an dem Wegpunkt gestoppt wurde
     * @param rating Bewertung des Hotels, bei Start und Ende 5.0
     * @param type Type des Wegpunkts (HOTEL; START; TARGET)
     */
    public Node(int position, int depth, float rating, Type type) {
        this.position = position;
        this.night = depth;
        this.rating = rating;
        this.parentedRating = this.rating;
        this.type = type;
    }

    public int getPosition() {
        return position;
    }

    public Type getType() {
        return type;
    }

    public int getNight() {
        return night;
    }

    public void setNight(int night) {
        this.night = night;
    }

    public void setParent(Node parent) {
        this.parent = parent;
    }
```

```
    public float getParentedRating() {
        return parentedRating;
    }

    public void setParentedRating(float parentedRating) {
        this.parentedRating = parentedRating;
    }

    @Override
    public String toString() {
        return "Wegpunkt bei Minute " + position + ", Nacht:" + night + ", Bewertung:" + rating + ", Typ:" + type +
                "\n" + parent;
    }
}
```

Quellen / Material

Aufgabestellung: https://bwinf.de/bundeswettbewerb/40/1/ [03.09.2022 ~ 20:15 Uhr]

Beispieldaten: https://bwinf.de/bundeswettbewerb/40/1/ [03.09.2022 ~ 20:15 Uhr]