**Prokoll**

*1. Beschreibung des verwendeten Algorithmus:*

Der in diesem Code verwendete Algorithmus ist Dijkstra's Algorithmus, der zur Lösung des kürzesten Pfadproblems in einem Graphen mit nicht negativen Kantenlängen verwendet wird.

Der Algorithmus funktioniert wie folgt:

* Er initialisiert die Distanzen von der Startstadt zu allen anderen Städten als unendlich (`Integer.MAX\_VALUE`) und die Distanz von der Startstadt zu sich selbst als 0.
* Er verwendet eine Prioritätswarteschlange (PriorityQueue), um die Stadt mit der kürzesten Distanz zu bestimmen.
* Bei jeder Iteration entfernt er die Stadt mit der kürzesten Distanz aus der Warteschlange, besucht alle ihre unbesuchten Nachbarstädte und aktualisiert ihre Distanzen, wenn der neue Pfad kürzer ist.
* Der Algorithmus wird beendet, wenn alle Städte besucht wurden oder die Warteschlange leer ist.

*2. O-Notation:*

Die Zeitkomplexität von Dijkstra's Algorithmus ist O((V+E)logV), wobei V die Anzahl der Knoten (Städte) und E die Anzahl der Kanten (Straßen zwischen Städten) ist. Dies liegt daran, dass jede Kante und jeder Knoten höchstens einmal bearbeitet wird und für jede Operation die Prioritätswarteschlange in log(V) Zeit aktualisiert wird.

*3. Experimente und Messungen zur Laufzeit:*

Um die Laufzeit des Algorithmus zu messen, können Sie die Systemzeit vor und nach der Ausführung des Algorithmus erfassen und die Differenz berechnen. Hier ist ein einfacher Weg, dies in Java zu tun:

````

long startTime = System.nanoTime();

// Ausführen des Algorithmus...

long endTime = System.nanoTime();

long duration = (endTime - startTime); // Dauer in Nanosekunden

```

Sie könnten verschiedene Tests mit unterschiedlichen Eingabegrößen durchführen und die Laufzeiten aufzeichnen, um zu sehen, wie sich die Leistung des Algorithmus ändert, wenn die Größe des Problems zunimmt. Beachten Sie jedoch, dass die tatsächliche Laufzeit von vielen Faktoren abhängt, darunter die spezifische Hardware und das Betriebssystem, das Sie verwenden.