# Protokoll:

Adjaceny list : ###

**[Dijkstra's Algorithmus:](https://de.wikipedia.org/wiki/Dijkstra-Algorithmus)** [Findet den kürzesten Pfad zwischen zwei Stationen in einem Graphen. Der implementierte Algorithmus basiert auf einer modifizierten Version von Dijkstra's Algorithmus und verwendet eine Prioritätswarteschlange, um die Station mit der geringsten Entfernung effizient zu extrahieren.](https://de.wikipedia.org/wiki/Dijkstra-Algorithmus)

[Der Algorithmus startet mit dem Startknoten und initialisiert die Distanzen zu allen anderen Knoten im Graphen als unendlich, außer dem Startknoten selbst, für den die Distanz als 0 gesetzt wird. Der Algorithmus verwendet auch Hilfsstrukturen, wie eine Vorgängerliste und eine Liste für die Anzahl der Linienwechsel.](https://de.wikipedia.org/wiki/Dijkstra-Algorithmus)

[Der Algorithmus durchläuft eine Schleife, in der er den Knoten mit der geringsten Distanz auswählt und als besucht markiert. Für jeden Nachbarn dieses Knotens werden die Distanz und die Anzahl der Linienwechsel aktualisiert, falls ein kürzerer Weg gefunden wird. Die Aktualisierung der Distanz und der Linienwechselkosten erfolgt durch Hinzufügen der Kosten des aktuellen Knotens und der Kantenkosten zwischen den Knoten. Wenn die Aktualisierung erfolgt, wird der Nachbarknoten in die Prioritätswarteschlange eingefügt.](https://de.wikipedia.org/wiki/Dijkstra-Algorithmus)

[Der Algorithmus endet, wenn entweder alle Knoten besucht wurden oder der Zielknoten erreicht wurde. Anschließend wird der kürzeste Pfad durch Rückverfolgung der Vorgängerliste von Zielknoten zu Startknoten erstellt.](https://de.wikipedia.org/wiki/Dijkstra-Algorithmus)

[Die Implementierung verwendet auch die Zeitmessungsfunktionen, um die Laufzeit des Algorithmus zu erfassen. Der gComplexity usingrd in den Funktionen find\_path und find\_path1 umgesetzt, wobei find\_path1 die optimierte Version ist, die die Prioritätswarteschlange verwendet (Experimente bezüglich der Zeiteffizienz).](https://de.wikipedia.org/wiki/Dijkstra-Algorithmus)**[Time-Complexity not using heapq:](https://de.wikipedia.org/wiki/Dijkstra-Algorithmus)** [O((V+E) V) (V = Stationen) E = (Edges/Connections); lineare Suche](https://de.wikipedia.org/wiki/Dijkstra-Algorithmus)**[Time-Complexity using heapq:](https://de.wikipedia.org/wiki/Dijkstra-Algorithmus)** [O((V+E) log V) (V = Stationen) E = (Edges/Connections)  
heapq-Operationen(pushing & popping) : O(log V)](https://de.wikipedia.org/wiki/Dijkstra-Algorithmus)