public List<Vertex> berechneKuerzestenWeg(Vertex von, Vertex nach)

setze alle Markierungen im Graphen auf false

bereiteAbstandstabelleVor(von)

Der Knoten, den wir uns gerade ansehen, z.B. A.
Hilfsvariable: aktuellerKnoten

Falls wir uns gerade Knoten A ansehen, speichert diese Variable den Nachbarn B.
Hilfsvariable: aktuellerNachbarknoten

Alle Nachbarn von A.
Hilfsvariable: List der Nachbarn

Die Kante zwischen A und B.
Hilfsvariable: aktuelleKante

(Im Array) der Wegabschnitt des aktuellen Nachbarknotens, wenn wir neue Wege berechnen.
Hilfsvariable: abschnittNachbarknoten

Brauchen wir, wenn wir den Wegabschnitt des Nachbarknotens im Array suchen.
Hilfsvariable: tempIndex

für i von 0 bis zum letzten Index des Arrays

speichere den aktuellen Knoten (A) aus dem Array

speichere die Nachbarn des aktuellen Knotens [B, C, D, ...]

while-Schleife -> toFirst() und next() nicht vergessen
iteriere durch die Liste der Nachbarn [B, C, D, ...]

speichere den aktuellen Nachbarknoten (B)

Nachbarknoten (B) bereits markiert?

T

F

Hier aufpassen, sonst Endlosschleife!
gehe zum nächsten Nachbarknoten (C)

∅

Die Klasse Graph bietet einen entsprechenden Getter an.
speichere die aktuelle Kante (A - B)

tempIndex = ...
suche und speichere den aktuellen
Index des Nachbarknotens (B) im Array

speichere mithilfe von tempIndex den Wegabschnitt
des aktuellen Nachbarknotens (B)

berechne die Dauer der Strecke zum Nachbarknoten (B)
ausgehend vom aktuellen Knoten (A)

Ist der neue Weg besser?

T

F

abschnittNachbarknoten.vorgaenger = ...
...
überschreibe die Daten im Wegabschnitt des Nachbarknotens (B)

∅

markiere den aktuellen Knoten (A)

sortiere die Abstandsstabelle ab Index i + 1

erstelleErgebnisAusTabelle(nach)

return ergebnis