Single Source Shortest Path Suche auf OpenStreetMap

By Richard Wiedenhöft & Matthias Schilling

Überblick

- Problemformulierung
- Konzept
- Umsetzung
- Probleme
- Ergebnis

Problemformulierung

- OpenStreetMap Kartendaten als Grundlage eines Graphen
- Single Source Shortest Path: Von einem Quellenknoten alle kürzesten Abstände berechnen
- Ergebnis als Heatmap visualisieren

Konzept

Die Kartendaten einlesen und daraus einen Graph erstellen

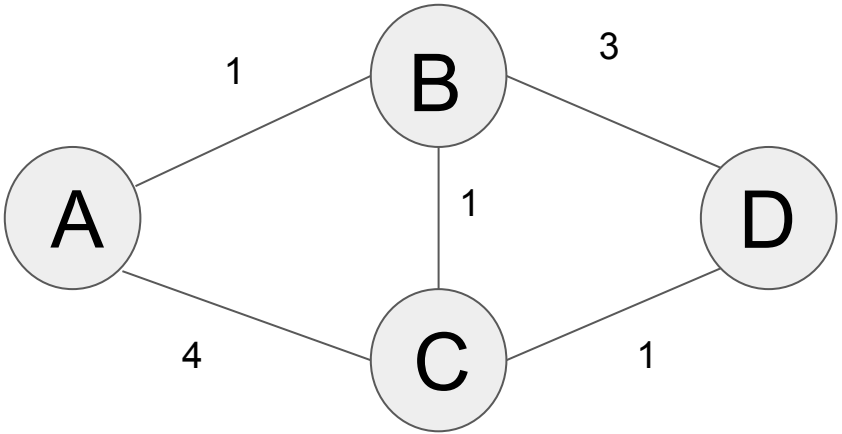
- Karte liegt als Protobuf (.pbf) vor
- Viele Unnütze Daten(Alte oder ungebaute Straßen, Gebäude/nicht relevante Ding)
- Viele redundante Knoten(eine Eingangs und eine Ausgangskante)

Konzept

Die kürzesten Abstände Berechnen

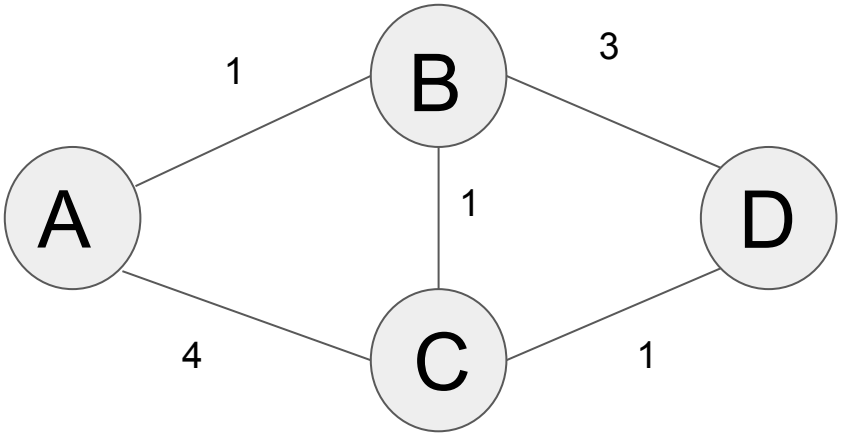
- Den geparsten Graphen als Adjazenzmatrix
- Die Kürzesten Wege mit Naive Dijkstra
- Dijkstra:
  - Kleiner Synchronisierungsaufwand
  - Einfacher Algorithmus
  - Intuitive Parallelisierbarkeit

Wiederholung: Dijkstra



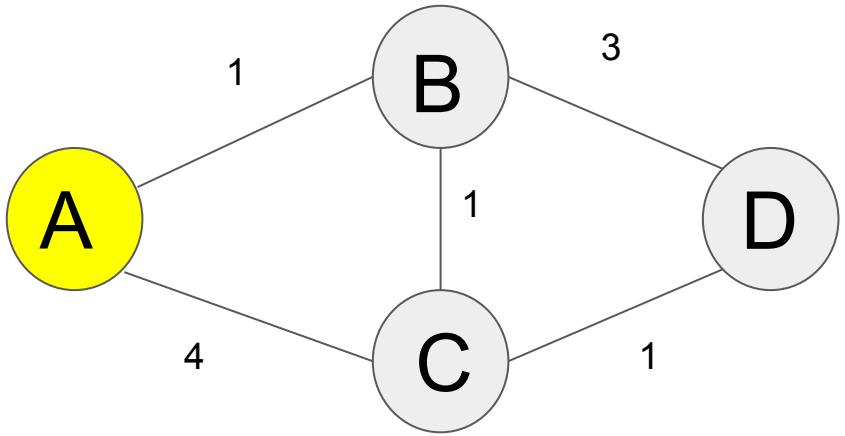
1. Suche den unbesuchten Knoten mit dem Geringsten Abstand
2. Aktualisiere Abstände der benachbarten Knoten
3. Markiere den Knoten als Besucht

Wiederholung: Dijkstra



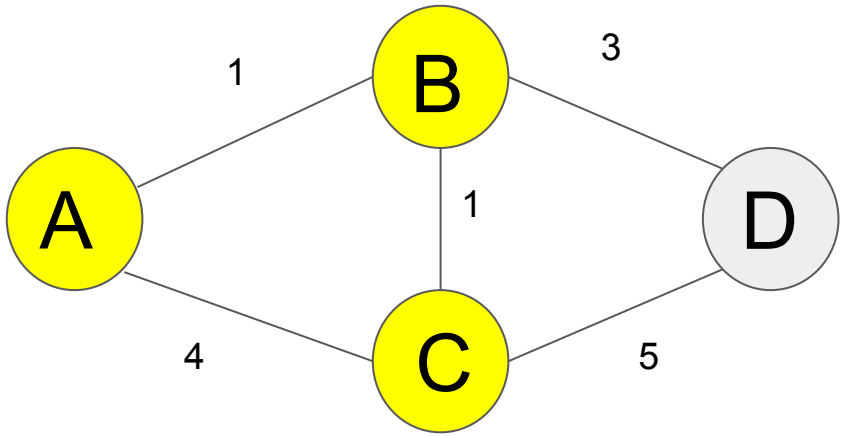
step	A	B	C	D
0	0	INF	INF	INF

Wiederholung: Dijkstra



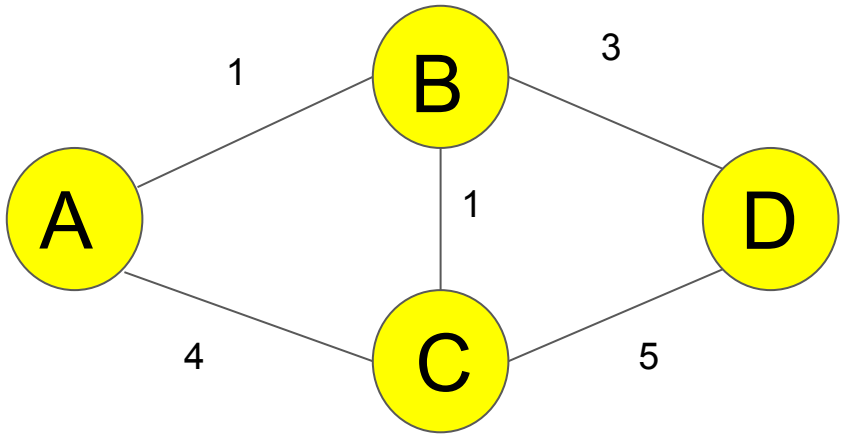
step	A	B	C	D
0	0	INF	INF	INF
1		1	4	INF

Wiederholung: Dijkstra



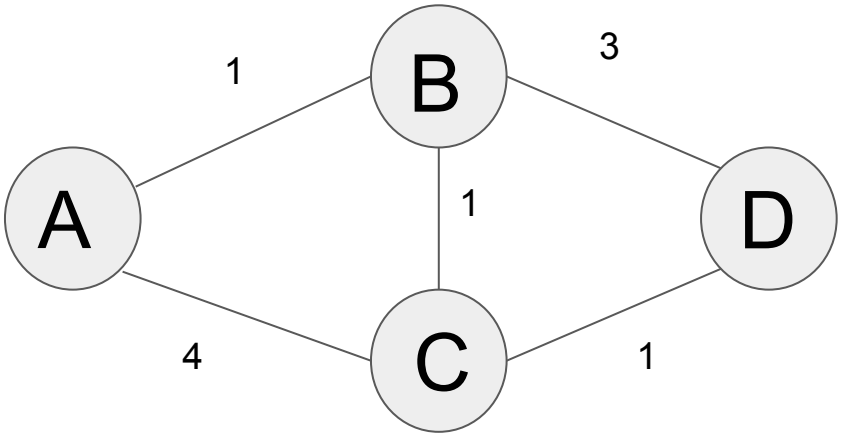
step	A	B	C	D
0	0	INF	INF	INF
1		1	4	INF
2			2	4
3				

Wiederholung: Dijkstra



step	A	B	C	D
0	0	INF	INF	INF
1		1	4	INF
2			2	4
3				

Dijkstra Parallel (P = 2)



Parallel: Kleinsten Abstand eines Unbesuchten Knoten

- Jeder Prozess bekommt einen Zuständigkeitsbereich
- Dort sucht er lokales minimum

Synchronisierung: Minimum der lokalen Minima

Danach: Serielles Update auf jedem Worker

Dijkstra Parallel (P = 2)

Parallel: Kleinsten Abstand eines Unbesuchten Knoten

- Jeder Prozess bekommt einen Zuständigkeitsbereich
- Dort sucht er lokales minimum

Synchronisierung: Minimum der lokalen Minima

Danach: Serielles Update auf jedem Worker

step	A	B	C	D
0	0	INF	INF	INF

Dijkstra Parallel (P = 2)

Parallel: Kleinsten Abstand eines Unbesuchten Knoten

- Jeder Prozess bekommt einen Zuständigkeitsbereich
- Dort sucht er lokales minimum

Synchronisierung: Minimum der lokalen Minima

Danach: Serielles Update auf jedem Worker

Graph der zu bearbeiten ist:

- Wenige Kanten pro Knoten
- Viele Knoten

Umsetzung

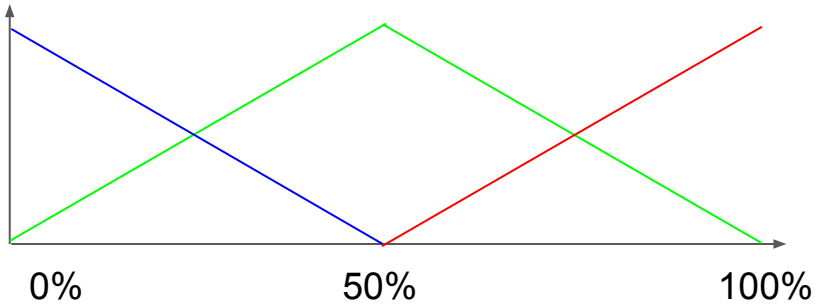
- Protobuf im Grunde einfach. OSMPBF nicht.
- Nachrichtenteile mit Zlib komprimiert
- 2 verschiedene Darstellungsformen von Nodes
- Delta-Codierung von IDs
- Festkommazahlen Marke Eigenbau
- Viele für uns überflüssige Daten
- Erkennung sinnvoller Daten nicht trivial

Umsetzung

- Dijkstra mit Adjazenzmatrix
- Verteile Komplette Adjazenzmatrix
- Parallele Suche des Minimum mit OMP und MPI
- Aktualisierung der Ergebnisse
- Vektorisierung Schwierig
- Grundlage: Eigen::SparseMatrix

Umsetzung

- Visualisierung mit OpenCV
- x-Achse ist Longitude y-Achse ist Latitude
- Farbcodierung von blau über grün bis rot



Probleme

- Vektorisierung Schwierig bei Graphensuche
- Datenstruktur
- Graph sehr Groß/Outputfile sehr groß
- Planetfile erst recht
- Nichttriviale Workload Distribution(*visited* Knoten brauchen weniger Rechenkapazität)

Ergebnis