CSVGraphUtilJoin – Pre-Layout

ZoomIn:

* Server:
  + Variante 1 (Redundanz vermeiden):
    - Der Client schickt alle Wrapper-IDs aus der vorherigen Ansicht an den Server.
    - Dazu muss über alle Cytoscape-Kanten iteriert werden.
    - Der Server filtert diese Wrapper heraus.
    - Der Server filtert alle Wrapper bei denen nicht mindesten 1 Knoten im neuen Zielbereich liegt heraus.
    - Damit werden keine Wrapper gesendet, welche schon visualisiert sind.
  + Variante 2 (Mit Redudanz klarkommen):
    - Sendet alle Wrapper, bei denen mindestens 1 Knoten im Zielbereich liegt.
    - Damit sind alle Wrapper aus der vorherigen höheren Ebene redundant, bei denen mindestens 1 Knoten immer noch im Zielbereich liegt. Möglicherweise alle, möglicherweise sind auch alle Knoten redundant.
* Client:
  + Variante 1: Knoten-Collection der Knoten des alten Zielbereichs, Knoten-Collection der neuen Knoten im neuen Zielbereich
    - Der Client muss ermittelt haben, um wieviele Knoten der Zielbereich leerer geworden ist. Dazu einmal durch alle Knoten des alten Zielbereichs iterieren und alle Knoten aus der Collection (NUR aus der Collection) entfernen, die nicht im neuen Zielbereich liegen. Eine Knoten-Collection der Knoten im Zielbereich ist erforderlich.
    - Einmal durch cy.edges() iterieren und kannten entfernen, wenn beide Knoten außerhalb des Zielbereichs
    - Neue Wrapper werden hinzugefügt, solange die Kapazität des Zielbereichs noch nicht erreicht ist. Von den Knoten dieser Wrapper, wird die laufende Variable des kleinsten Grades aktualisiert. *Liegt nur einer der beiden Knoten des Wrappers im Zielbereich, wird dieser zur „Neu, nicht-definitiv Knoten im Zielbereich“-Collection (NNDKZ)hinzugefügt. Liegen beide Knoten im Zielbereich wird der größere Knoten zur Collection „Knoten-Collection im Zielbereich“ (KZ) hinzugefügt und der Kleinere zur NNDKZ. Bei Gleichheit werden beide zur KZ hinzugefügt. (Ungültige Überlegung weil, identity stream und nicht-identity stream nicht sortiert sind!!!!).*
    - Ist die Kapazität des Zielbereichs erreicht, werden Knoten mit größerem Grad im Zielbereich als mindestens einer der Knoten in NNDKZ stattdessen hinzugefügt. Dies kann über einen Vergleich mit einer laufenden Variablen des kleinsten Grades ermittelt werden. Ist dies der Fall, wird durch die ganze Collection iteriert und die neue Variable des kleinsten Grades ermittelt und der Austauschknoten gefunden.
    - Folgende Fälle:
      * Knoten 1 und 2 sind identisch (identityEdge): Füge hinzu, wenn größer, und lösche kleinsten
      * Knoten 1 und 2 sind nicht identisch
        + Knoten 1 und 2 sind im Zielbereich

Ein Knoten ist größer: Füge keinen hinzu, weil Identitätswrapper noch kommt

Beide Knoten sind größer als die kleinsten beiden: Der ganze Wrapper wird hinzugefügt und zwei kleinste bisherige Knoten im Zielbereich gelöscht

* + - * + Nur ein Knoten ist im Zielbereich: Knoten wird hinzugefügt, wenn größer, und ein bisheriger Knoten im Zielbereich gelöscht. Dann wird auch der Rest des Wrappers hinzugefügt.
    - Nach Time Out wird die Operation abgebrochen
    - Alle Knoten außerhalb des Zielbereiches, welche nun keinen Nachbarn haben, werden gelöscht. Dazu einmal durch Cytoscape.nodes() iterieren.
    - Die beiden Collections fusionieren.
  + Variante 2:
    - Bei Verwendung einer Wrapper-Collection, können zuerst alle Wrapperkanten gelöscht werden (aus der Collection und aus Cytoscape), bei denen beide Knoten nicht mehr im Zielbereich liegen (die Knoten noch nicht).
    - Durch Iterieren durch die Knoten-Collection ermittelt, um wieviele Knoten der Zielbereich leerer geworden ist.
    - Neue Wrapper auf Redundanz testen: Alle Wrapper mit Wrapper-Collection vergleichen
    - Ab hier sehr ähnlich zu Variante 1

ZoomOut:

* Server:
  + Variante 1 (Redundanz vermeiden):
    - Der Client schickt alle Wrapper-IDs aus der vorherigen Ansicht an den Server.
    - Dazu muss über alle Cytoscape-Kanten iteriert werden.
    - Der Server filtert diese Wrapper heraus.
    - Der Server filtert alle Wrapper bei denen nicht mindesten 1 Knoten im neuen Zielbereich liegt heraus.
    - Damit werden keine Wrapper gesendet, welche schon visualisiert sind.
  + Variante 2 (Mit Redundanz klarkommen):
    - Sendet alle Wrapper, bei denen mindestens 1 Knoten im Zielbereich liegt, angefangen mit den größten Wrappern.
    - Damit können alle Wrapper aus der vorherigen tieferen Ebene redundant sein.
* Client:
  + Variante 1: Knoten-Collection des Zielbereichs
    - Eine laufende Variable des Knoten mit kleinstem Grad wird ermittelt
    - Sind beide Knoten im Zielbereich und BEIDE (Nicht wenn nur einer, denn der identityWrapper regelt das) größer als die beiden kleinsten Knoten in der momentanen Visualisierung, werden die zwei kleinste Knoten ausgetauscht. Dabei werden jeweils die Variablen des Knoten mit kleinstem/zweitkleinstem Grad aktualisiert.
    - Liegt nur ein Knoten im Zielbereich und ist dieser größer als der kleinste in der momentanen Visualisierung, wird der Wrapper hinzugefügt und der Knoten kommt in die NNDKZ. Der kleinste Knoten wird gelöscht und die Variable des kleinsten Knoten aktualisiert.
    - Abbruch nach Timeout
    - Alle Nachbarn der entfernten Knoten aus der KZ, die vor dem ZoomOut außerhalb der Visualisierung waren müssen jetzt berücksichtigt werden, weil sie nicht automatisch durch größere Knoten ausgetauscht werden. Daher werden AM ENDE DER OPERATION alle diese Nachbarn gelöscht, falls sie ohne weitere Nachbarn sind und Inzidenz 0 haben. Eine höhere Inzidenz würde bedeuten, dass ein anderer, neuer, valider Wrapper diesen Knoten benötigt, dann würde er nicht gelöscht. Hierfür muss die Inzidenz aller Knoten vor Beginn der Operation auf die Anzahl der Nachbarn gesetzt werden (optimalerweise ist das sowieso für alle Knoten, bis auf solche welche durch identityWrapper entstanden sind, der Fall). Außerdem muss beim Löschen eines Knoten die Inzidenz aller Nachbarn um 1 verringert werden. Damit wird gewährleistet, dass alle Knoten, welche nicht durch identityWrapper entstanden sind und trotzdem isoliert sind, gelöscht werden.
    - Alle Knoten außerhalb des Zielbereiches, welche nun keinen Nachbarn haben, werden gelöscht. Dazu einmal durch Cytoscape.nodes() iterieren.
  + Variante 2: Knoten-Collection des Zielbereichs, Wrapper-Collection
    - Wrapper mit Wrapper-Collection vergleichen
    - Ab hier ähnlich wie Variante 1

Panning:

* Überlegung: Wenn man von einem dünn-besiedelten Bereich in einen stärker besiedelten Bereich pannt, kann es sein, dass viele große Knoten nicht visualisiert werden, weil die capacity nicht besonders groß ist 🡪 Client: alle Knoten aus der KZ in die NNDKZ überführen.
* Server:
  + Filtert alle Wrapper raus, die nicht mindestens 1 Knoten im neuen Teilbereich haben
  + Filtert DAVON alle Wrapper raus, die den zweiten Knoten im alten Zielbereich haben.
* Client: Knoten-Collection, „Neue Knote”-Collection
  + Ermitteln, um wieviele Knoten der Zielbereich leerer geworden ist (Besser: wie hoch die verbleibende Kapazität ist). Dazu einmal durch die Knoten-Collection iterieren.
  + Einmal durch cy.edges() iterieren und kannten entfernen, wenn beide Knoten außerhalb des Zielbereichs
  + Neue Wrapper hinzufügen bis Kapazität erreicht ist.
  + Wenn Kapazität erreicht, neue Knoten gegen die Variable des Knoten mit kleinstem Grad testen (aus „neue Knoten“-Collection) und eventuell kleinsten oder die beiden kleinsten Knoten austauschen.
  + Wenn timeout, dann Operation abbrechen.
  + Knoten außerhalb des Zielbereichs, welche keine Nachbarn haben, löschen.
  + Knoten-Collections fusionieren.