Beating the Worst Case
Wintersemester 2023/2024
Jean-Pierre von der Heydt & Marcus Wilhelm

Karlsruher Institut für Technologie

Übungsblatt 0: Aufnahmeprüfung

Ausgegeben am 25.10.2020. Geschätzter Zeitaufwand: 3 Wochen

Auf der Webseite[1] befindet sich eine ZIP-Datei mit 800 verschiedenen Graphen, welche wir auf geheime Art und Weise gesammelt haben. Eure Aufgabe besteht darin, zwei verschiedene Algorithmen zu implementieren und auf diesen Graphen auszuwerten. Euer Ziel ist es, die Graphen zu finden auf denen der jeweilige Algorithmus besonders gut funktioniert.

Als Abgabeformat erwarten wir eine Datei in der die Dateinamen der identifizierten Graphen (per Zeilenumbruch separiert) aufgelistet sind.

```
0.gr
3.gr
8.gr
11.gr
14.gr
...
```

Abbildung 1: Ausschnitt aus einer möglichen Abgabedatei `solution1.txt`

Aufgabe 1: Aufwärmübung

Das Framework[2] für dieses Praktikum beinhaltet grundlegende Funktionalität, um in C++ mit Graphen zu arbeiten. Um es zu nutzen, erstelle deinen eigenen Fork dieses Repositories. Lies dann die Readme-Dateien des Frameworks und bringe das Beispielprogramm (run) wie beschrieben zum laufen.

Aufgabe 2: Kürzeste *st*-Wege (200 Punkte)

Um kürzeste Wege auf ungewichteten Netzwerken zu ermitteln, kann man bekannterweise die Breitensuche verwenden. Eine beliebte Beschleunigungstechnik besteht darin, die Breitensuche *bidirektional* auszuführen. Bei der Suche nach einem st-Pfad werden hierbei nicht nur ausgehend von s, sondern (abwechselnd) auch von t aus neue adjazente Knoten exploriert. Dazu wird zusätzlich zum Zustand der Vorwärtssuche („besucht"-Markierung und Queue der gefundenen Knoten) auch der Zustand der Rückwärtssuche verwaltet. Das Wechseln zwischen den beiden Richtungen erfolgt

[1] `https://scale.iti.kit.edu/teaching/2023ws/beating_wc`

[2] `https://gitlab.kit.edu/kit/iti/scale/templates/praktikum_beating_the_worst_case_framework`

jeweils nachdem ein kompletter Layer exploriert wurde, wobei ein Layer alle Knoten sind die den selben Abstand von s (bzw. t) haben. Die Wechselstrategie für die wir uns hier interessieren fährt dabei jeweils mit der Richtung fort, bei der die Kosten für die Exploration des nächsten Layers kleiner sind. Diese Kosten werden mithilfe Summe der Knotengrade des vorhergehenden Layers abgeschätzt.

Die Worst-Case Laufzeit der bidirektionalen Breitensuche ist, wie bei der klassischen Breitensuche, in $O(n + m)$. Allerdings wurde festgestellt, dass es Graphklassen gibt, auf denen die bidirektionale Suche einen asymptotischen speedup gegenüber der unidirectionalen Variante hat. Das bedeutet, dass der Faktor, um den die bidirektionale Breitensuche schneller ist wachsend in n ist. Unter den gegebenen Netzwerken gibt es eine geheime Teilmenge von 200 Graphen, auf denen die bidirektionale Breitensuche *keinen* asymptotischen speedup hat. Implementiert den Algorithmus und evaluiert ihn auf den gegebenen Netzwerken, um die geheime Teilmenge zu ermitteln.

Aufgabe 3: Vertex Cover (200 Punkte)

Für einen Graphen $G = (V, E)$ nennen wir eine Teilmenge $X \subseteq V$ ein *Vertex Cover*, falls für jede Kante $\{u,v\} = e \in E$ gilt $u \in X$ oder $v \in X$. Das Problem zu entscheiden, ob es ein Vertex Cover einer gegebenen Größe gibt ist NP-schwer. Es existieren aber polynomielle Reduktionsregeln, mit denen eine äquvalente kleinere Instanz erzeugt werden kann. Eine solche Reduktionsregel ist die Dominanzregel.

Für zwei benachbarte Knoten $u, v \in V$ sagen wir, dass u v dominiert, wenn $N[v] \subseteq N[u]$ gilt[3]. Die Dominanzregel besagt, dass es eine minimale Knotenüberdeckung gibt, die u einschließt. Somit kann man die Instanz reduzieren, indem man u in die Knotenüberdeckung aufnimmt und es aus dem Graphen entfernt. Um die Wirksamkeit der Dominanzregel zu bewerten, wenden wir sie erschöpfend an, d. h., bis keine dominanten Knoten mehr übrig sind. Darüber hinaus entfernen wir isolierte Knoten. Wir beziehen uns auf die Anzahl c der Knoten in der größten zusammenhängenden Komponente der verbleibenden Instanzen als die Kerngröße.

Unter den gegebenen Netzwerken gibt es eine geheime Teilmenge von 200 Graphen, auf denen die Regel zu *besonders kleinen* Kernen führt. Implementiert die Dominanzregel und evaluiert sie auf den gegebenen Netzwerken, um die geheime Teilmenge zu ermitteln.

[3] hier bezeichnet $N[v]$ die geschlossene Nachbarschaft von v, d.h. $N[v] = N(v) \cup \{v\}$