UNIVERSITÄT BONN
RHEINISCHE FRIEDRICH-WILHELMS UNIVERSITÄT BONN
INSTITUT FÜR INFORMATIK DER UNIVERSITÄT BONN

Institut für Informatik 1
Computational Analytics
Prof. Dr. Petra Mutzel
M. Sc. Jonas Charfreitag
M. Sc. Philip Mayer

Übungen zur Vorlesung
Graphenalgorithmen
Sommersemester 2024

Übungsblatt 2
Besprechung:
30.04. und 02.05.

Bitte beachten Sie die Hinweise zu den Übungen und der Übungsgruppenverteilung im eCampus. Stellen Sie Fragen zu Vorlesung und Übung vorzugsweise im dort angelegten Forum.

Sie sind unbedingt dazu eingeladen, die Aufgaben in Gruppen zu bearbeiten. Um die Punkte für einen Aufgabenteil bekommen zu können müssen Sie natürlich in der Lage sein, die Aufgabe(n) allein und vollständig vorstellen zu können.

Aufgabe 2.1 – Adjazenzarray für ungerichtete Graphen (2 Punkte)

In der Vorlesung wurde das Adjazenzarray für gerichtete Graphen vorgestellt. Die Datenstruktur kann zum Speichern ungerichteter Graphen angepasst werden, indem jede $\{u, v\}$ Kante zwei mal in das Edgearray abgespeichert wird. Einmal in dem zum Knoten u und einmal in dem zum Knoten v gehörenden Bereich.

In einigen Algorithmen ist es wichtig zu jeder Kante (u, v) schnell Zugriff auf die sog. Gegenkante (v, u) zu haben. Geben Sie eine *Realisierung* einer Erweiterung des Adjazenzarrays für ungerichtete Graphen an, die diese Anforderung erfüllt. Analysieren Sie die Laufzeit für das Löschen von Knoten und den Speicherbedarf Ihrer Lösung.

Eine *Realisierung* beschreibt genauer als klassischer Pseudocode, wie die relevanten Datenstrukturen im Hauptspeicher repräsentiert werden und wie einzelne Schritte des Algorithmus konkret ausgeführt werden.

Aufgabe 2.2 – Sortieren in Linearzeit (2 Punkte)

Gegeben sei ein ungerichteter Graph G in Adjazanzlisten Repräsenation. Beschreiben Sie einen Algorithmus, der in linearer Zeit eine Kopie von G erzeugt, in der alle Adjazenlisten entsprechend der Konten IDs sortiert sind. Der Algorithmus soll dabei über jede Adjazenliste von G nur genau einmal iterieren und ohne die explizite Verwendung von Sortieralgorithmen (wie Radix Sort o.ä.) auskommen.

Aufgabe 2.3 – [Programmieraufgabe] Implementierung Adjazenzarray (5 Punkte)

Schreiben Sie ein Programm, das einen ungerichteten Graphen in Form einer ungeordneten Kantenlist einliest und als Adjazenzarray repräsentiert. Laden Sie Ihr Programm zur Evaluation als Lösung für die '2_UAA' Aufgabe des Contests 'GA_SS24' auf unserer Competitive Programming Plattform hoch. Es empfiehlt sich unbedingt, die auf der Webseite einsehbaren Beispieleingaben lokal zu testen. Sollten Sie hierbei Probleme haben, hilft Ihnen möglicherweise dieses Tutorial. Machen Sie sich außerdem Gedanken über die theoretische Effizienz Ihrer Implementierung. Das Hochladen von ineffizientem Code wird mit hoher Wahrscheinlichkeit zu einem Überschreiten des auf der Plattform eingestellten 'Timelimits' führen.

Sie können davon ausgehen, dass jeder Knoten der Eingabegraphen mindestens Grad 1 hat.

Eingabe (Extended Edgelist Format): Die erste Zeile der Eingabe enthält die Anzahl der Knoten $|V|$ und die Anzahl der Kanten $|E|$, durch ein Leerzeichen getrennt. Alle weiteren Zeilen enthalten jeweils eine Kante, repräsentiert durch die Knoten IDs der inzidenten Knoten. Die Knoten IDs sind aus $\{0, \ldots, |V| - 1\}$.

Ausgabe: Die geforderte Ausgabe für einen Eingabe-Graphen $G = (V, E)$ besteht aus zwei Zeilen. Die erste Zeile repräsentiert das Indexarray, die zweite das Kantenarray Ihrer Adjazenzarray Implementierung. Jeder Eintrag des Arrays ist durch ein Leerzeichen vom nächsten getrennt.

Aufgabe 2.4 – Kurzfragen Matching (2 Punkte)

a) Beschreiben Sie den Unterschied zwischen den Definitionen aus der Vorlesung für ein maximum, maximales und perfekte Matching. Geben Sie konkrete Beispiele an, die den Unterschied verdeutlichen.

b) Welche Schranken für maximum Matchings wurden in der Vorlesung eingeführt und wann gelten Sie?

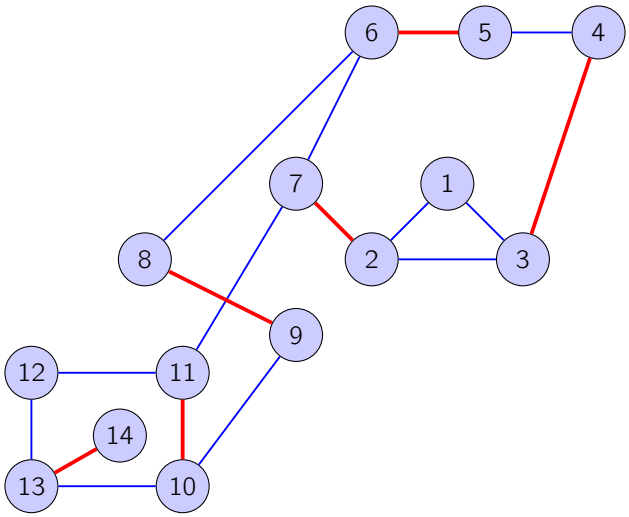
Aufgabe 2.5 – Perfekte Matchings in Bäumen (3 Punkte)

Sei $T = (V, E)$ ein Baum, d.h. T ist ein zusammenhängender, kreisfreier und ungerichteter Graph.
Zeigen Sie: Für T existiert höchstens ein perfektes Matching.

Aufgabe 2.6 – Augmentierende Pfade und Matchings (6 Punkte)

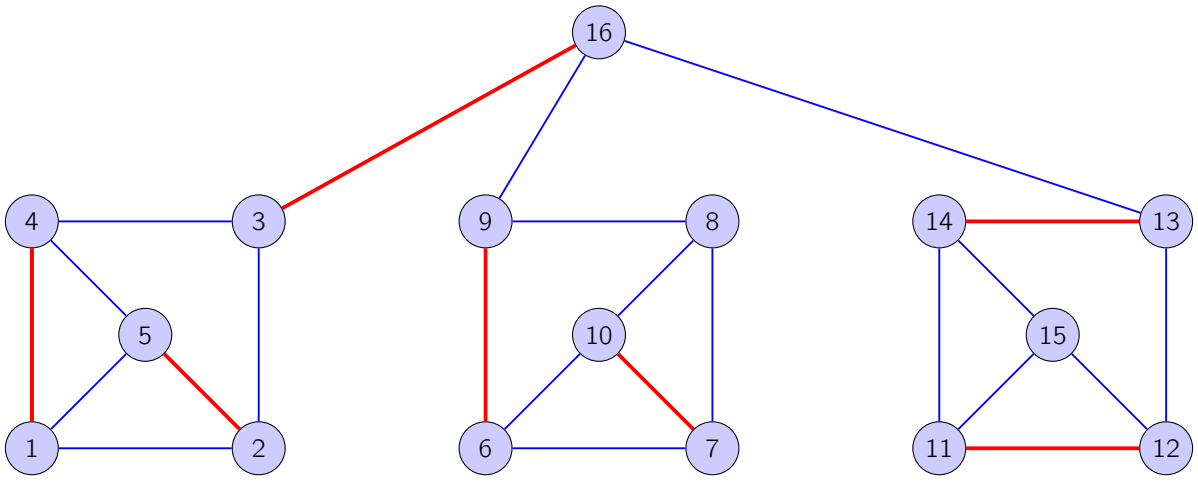
Betrachten Sie die folgenden Graphen G für die ein maximum Matching gesucht ist. Die rot markierten Kanten sind Teil des aktuellen Matchings M.

a)



Geben Sie einen M-augmentierenden Pfad für G an und verwenden Sie ihn um das Matching zu verbessern oder beweisen Sie, dass kein M-verbessernder Pfad existiert.

b)



Begründen / beweisen Sie, warum M für diesen Graphen maximum und/oder maximal ist.