Technische Universität München
Institut für Informatik
Lehrstuhl für Effiziente Algorithmen
Prof. Dr. Ernst W. Mayr
Dmytro Chibisov
Stefan Eckhardt
Hanjo Täubig

WS 2007/08
Aufgabenblatt 5
19. November 2007

Praktikum Algorithmen-Entwurf

(Abgabetermin: Montag, den 26.11.2007, 14.00Uhr)

Matchings in bipartiten Graphen

Aufgabe 1

Gegeben ist ein ungerichteter, bipartiter Graph $G = (V, E)$ mit $V = V_1 \cup V_2$, wobei $V_1 \cap V_2 = \emptyset$ und alle Kanten einen Knoten aus V_1 mit einem aus V_2 verbinden. Implementieren Sie den Algorithmus von Hopcroft und Karp, der in Zeit $O(\sqrt{|V|} \cdot |E|)$ ein Matching maximaler Kardinalität berechnet.

Verwenden Sie die Darstellungsmöglichkeiten von GraphWin, um die Arbeitsweise des Algorithmus, d.h. jede einzelne simultane Breitensuche und anschließende Tiefensuche, anschaulich darzustellen. Zu jedem Zeitpunkt soll das vorläufige Matching am Bildschirm klar erkennbar sein. Auch die Invertierung augmentierender Pfade soll gut mitverfolgt werden können.

Zusätzlich soll vor jeder simultanen Breitensuche die obere Schranke

$$2 \cdot \left\lfloor \frac{|M|}{|M'| - |M|} \right\rfloor + 1$$

für die Länge eines kürzesten augmentierenden Pfades ausgegeben werden, wobei M das aktuelle Matching und M' ein beliebiges Matching maximaler Kardinalität ist. Da M' unbekannt ist, soll hier $|M'| = |V|/2$ eingesetzt werden.

Hinweise

Als Eingabe für Ihren Algorithmus können Sie die vier ungerichteten, bipartiten Graphen `bipartite1.gw` bis `bipartite4.gw` verwenden, die in den üblichen Verzeichnissen zu finden sind. Bei diesen Graphen hat das User-Label der Knoten in V_1 den Wert “1”, das der Knoten in V_2 den Wert “2”. Alle diese Graphen enthalten ein perfektes Matching, d.h. ein Matching mit $|V|/2$ Kanten.

Aufgabe 2

Gegeben ist ein ungerichteter, bipartiter Graph $G = (V, E)$ mit $V = V_1 \cup V_2$. Jeder Kante $e \in E$ ist eine ganze Zahl $w(e) \in \mathbb{Z}$ als Gewicht zugeordnet. Beachten Sie, dass auch negative Kantengewichte erlaubt sind! Implementieren Sie einen effizienten Algorithmus, der in Zeit $O(|V|^2 \log |V| + |V| \cdot |E|)$ ein Matching maximaler Kardinalität berechnet, das unter allen Matchings maximaler Kardinalität in G minimales Gewicht hat. (Das Gewicht eines Matchings ist gleich der Summe der Gewichte seiner Kanten.) Erweitern Sie den Algorithmus so, dass der Benutzer wählen kann, ob ein Matching mit minimalem oder mit maximalem Gewicht (unter allen Matchings maximaler Kardinalität) berechnet werden soll. Am Ende soll Ihr Programm das Gewicht des berechneten Matchings ausgeben.

Als Unterroutine zur Berechnung kürzester Pfade sollten Sie Ihre Implementierung des Algorithmus von Dijkstra von Blatt 4 verwenden.

Verwenden Sie die Darstellungsmöglichkeiten von GraphWin, um die Arbeitsweise des Algorithmus anschaulich darzustellen. Es bietet sich an, zusätzlich zu dem Fenster mit dem Eingabegraphen ein zweites GraphWin-Fenster zu öffnen, in dem der modifizierte Graph G' mit gerichteten Kanten und zusätzlichen Knoten s und t angezeigt wird.

Hinweise

Als Eingabe für Ihren Algorithmus können Sie die vier ungerichteten, bipartiten Graphen `wbipartite1.gw` bis `wbipartite4.gw` verwenden, die in den üblichen Verzeichnissen zu finden sind. Bei diesen Graphen hat das User-Label der Knoten in V_1 den Wert "1", das der Knoten in V_2 den Wert "2". Die Gewichte der Kanten sind als Strings im zugehörigen User-Label gespeichert. Alle Kantengewichte sind ganze Zahlen zwischen -2 und 17, und Ihr Programm kann die Kantengewichte durch Setzen der Breite einer Kante gleich dem Betrag ihres Gewichts optisch gut darstellen (Kanten mit negativem Gewicht können gestrichelt gezeichnet werden).

Die folgende Tabelle gibt für die vier Eingabegraphen das Gewicht eines Matchings maximaler Kardinalität und minimalen bzw. maximalen Gewichts an.

	wbipartite1.gw	wbipartite2.gw	wbipartite3.gw	wbipartite4.gw
minimal	29	-2	10	46
maximal	33	-1	22	81