Algorithmen und Komplexität

Dr. Bruno Becker

Übungsblatt 5

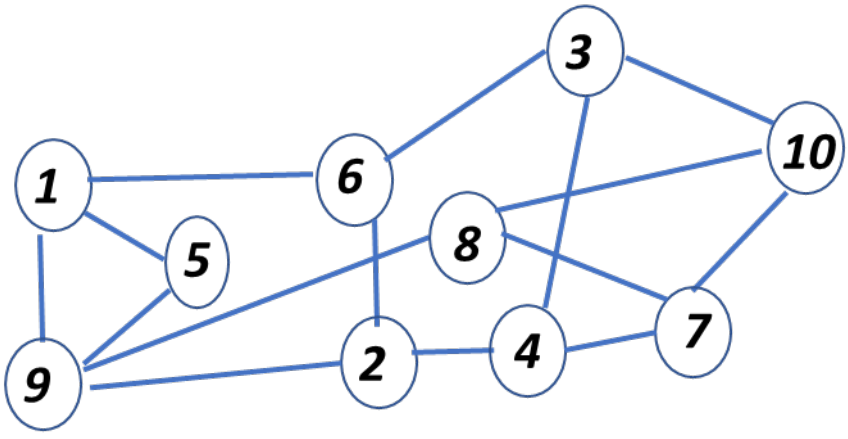
Aufgabe 1

Gegeben sei eine Adjazenz-Liste *A*, die Graphen mit maximal *n* Knoten abspeichern kann. Die verkettete Liste mit den Nachbarn eines Knoten sei einfach verkettet und unsortiert, d.h A[i] zeigt auf den ersten Nachbarn von *i* (oder ist *null*).

a) Erstellen Sie eine Methode *InsertNode(int i)*, die einen Knoten mit dem Schlüssel *i* (*i*<*n*) in die Adjazenzliste *A* einfügt.
b) Erstellen Sie eine Methode *InsertEdge(int i, j),* die eine Kante <*i,j*> in den Graphen einfügt. Wie oft kommt die Kante der Adjazenzliste vor?
c) Erstellen Sie eine Methode *DeleteNode(int i),* die einen Knoten und alle angrenzenden Kanten aus dem Graphen löscht.
d) Welchen Aufwand haben die Operationen im mittleren und schlechtesten Fall?
e) Was ändert sich am Aufwand, wenn die Nachbarlisten der Knoten sortiert gehalten werden?

Aufgabe 2

Gegeben sei folgender Graph:



a) Erstellen Sie für diesen Graph eine Adjazenzmatrix und eine Adjazenzliste.
b) Führen Sie ausgehend vom Startknoten 1 eine Tiefensuche durch. Dokumentieren Sie alle Rekursionsaufrufe und geben Sie am Schluss einen zugehörigen Pfad-Baum (DFS-Baum)

Aufgabe 3

Gegeben sei der Graph aus Aufgabe 2 und seine Implementierung in der Adjazenzliste wie in Aufgabe 2 a).

a) Führen Sie ausgehend vom Startknoten 1 eine Breitensuche durch. Dokumentieren Sie alle Rekursionsaufrufe und geben Sie am Schluss einen zugehörigen Pfad-Baum (BFS-Baum) an.
b) Für welchen Zielknoten ergibt die Pfadlänge durch Breitensuche den größten Unterschied zur Tiefensuche?

Aufgabe 4

Eine Kante in einem zusammenhängenden, ungewichteten Graphen heißt ***Brücke***, wenn das Entfernen dieser Kante den Graphen in zwei Teile zerfallen lässt.

Entwerfen Sie einen möglichst effizienten Algorithmus, der zu einem gegebenen Graphen G und einer Kante <i,j> ermittelt, ob die Kante <i,j> eine Brücke ist.