Übungsaufgabe

Graphalgorithmen Grundlegende Traversierung: DFS und BFS und deren Anwendungen

Universität:	Technische Universität Berlin
Kurs/Modul:	Algorithmen und Datenstrukturen
Erstellungsdatum:	September 6, 2025

Aufgabe 1: Grundlagen der DFS- und BFS-Traversierung

Betrachten Sie den ungerichteten Graphen $G_1 = (V_1, E_1)$ mit

$V_1 = \{A, B, C, D, E, F, G\}, \qquad E_1 = \{(A, B), (A, C), (B, D), (B, E), (C, F), (D, E), (E, F), (F, G)\}.$

Es handelt sich um einen zusammenhängenden Graphen. Startknoten ist A.

a) Tiefensuche (DFS) ab A
Führen Sie DFS ab A durch. Geben Sie die Besuchsreihenfolge der Knoten in einer Liste an und notieren Sie die Parent-Beziehungen aller Knoten im DFS-Baum (z. B. `parent[A] = -`, `parent[B] = A`, ...). Skizzieren Sie ggf. den DFS-Baum.

b) Breitensuche (BFS) ab A
Führen Sie BFS ab A durch. Geben Sie die Besuchsreihenfolge der Knoten im BFS-Baum an, notieren Sie die Abstände dist(v) von A zu jedem Knoten und schildern Sie die BFS-Baum-Struktur (Eltern-Beziehungen).

c) Vergleich von DFS- und BFS-Traversierung
Diskutieren Sie grob, welche Eigenschaften DFS- bzw. BFS-Bäume in Bezug auf Abdeckung, Pfadlängen und Eindeutigkeit der Traversierung haben. Erwähnen Sie insbesondere, dass BFS in diesem ungewichteten Graphen kurze Pfade liefert und DFS andere Strukturen erzeugt.

Aufgabe 2: Anwendungen von DFS und BFS

Betrachten Sie einen weiteren Graphen $G_2 = (V_2, E_2)$ mit

$$V_2 = \{u, v, w, x, y, z, t\}, \qquad E_2 = \{(u, v), (u, w), (v, x), (w, y), (x, z), (y, z)\}.$$

Der Graph besitzt zwei Komponenten: eine aus $\{u, v, w, x, y, z\}$ und eine isolierte Komponente $\{t\}$.

a) BFS-gestützte kürzeste Pfade
Verwenden Sie BFS beginnend bei u, um die Abstände $\text{dist}(v)$ von u zu allen Knoten zu bestimmen. Geben Sie die Abstände an und nennen Sie den kürzesten Pfad von u nach z.

b) Bestimmung der Zusammenhangskomponenten per DFS
Nutzen Sie DFS, um die Anzahl der Zusammenhangskomponenten von G_2 zu bestimmen. Geben Sie für jeden Knoten die Komponentennummer $\text{comp}(v)$ an (z. B. Komponente 1, 2, ...).

c) Reflexion zu Anwendungen
Nennen Sie zwei konkrete Anwendungsbeispiele, in denen DFS bzw. BFS in der Praxis zur Traversierung von Graphen genutzt werden (z. B. Bestimmung von Komponenten, kürzeste Pfade in ungewichteten Graphen, oder Aufbau von Traversierungsbäumen). Erläutern Sie kurz den Nutzen der jeweiligen Traversierung für das jeweilige Beispiel.

Lösungen

Lösung zu Aufgabe 1

a) Tiefensuche (DFS) ab A
Besuchsreihenfolge: $[A, B, D, E, F, C, G]$
Parent-Beziehungen im DFS-Baum: `parent[A] = -`, `parent[B] = A`, `parent[D] = B`, `parent[E] = D`, `parent[F] = E`, `parent[C] = F`, `parent[G] = F`

DFS-Baum (Kanten des DFS-Baums): $\{(A, B), (B, D), (D, E), (E, F), (F, C), (F, G)\}$

b) Breitensuche (BFS) ab A
Besuchsreihenfolge: $[A, B, C, D, E, F, G]$
Abstände $\text{dist}(v)$ von A zu jedem Knoten:

$$\text{dist}(A) = 0, \quad \text{dist}(B) = 1, \quad \text{dist}(C) = 1, \quad \text{dist}(D) = 2, \quad \text{dist}(E) = 2, \quad \text{dist}(F) = 2, \quad \text{dist}(G) = 3$$

Parent-Beziehungen im BFS-Baum: `parent[A] = -`, `parent[B] = A`, `parent[C] = A`, `parent[D] = B`, `parent[E] = B`, `parent[F] = C`, `parent[G] = F`
BFS-Baum-Kanten: $\{(A, B), (A, C), (B, D), (B, E), (C, F), (F, G)\}$

c) Vergleich von DFS- und BFS-Traversierung
- Abdeckung: Beide Algorithmen liefern Spanning Trees für den betrachteten Teilgraphen (und decken damit den jeweiligen Graph vollständig ab, sofern er zusammenhängend ist). In G1 ist der Graph zusammenhängend, daher decken DFS- und BFS-Bäume alle Knoten ab. - Pfadlängen: BFS liefert in ungewichteten Graphen per Definition kürzeste Pfade (Gefangene durch Abstände $\text{dist}(\cdot)$); DFS erzeugt tiefe Pfade, die oft länger sind. Im Beispiel führt der DFS-Baum z. B. den Pfad A–B–D–E–F–G (Länge 5) zum Knoten G, während der kürzeste Pfad A–C–F–G (Länge 3) existiert. - Eindeutigkeit: Beide Traversierungen sind nicht eindeutig – die erzeugten Bäume hängen stark von der Nachbarschaftsreihenfolge ab. Unterschiedliche Reihenfolgen der Nachbarn liefern unterschiedliche DFS-/BFS-Bäume und Distanz- bzw. Elterninformationen.

Lösung zu Aufgabe 2

a) BFS-gestützte kürzeste Pfade
Graph G2: $V_2 = \{u, v, w, x, y, z, t\}$, $E_2 = \{(u, v), (u, w), (v, x), (w, y), (x, z), (y, z)\}$. Eine Komponente ist $\{u, v, w, x, y, z\}$ und eine isolierte Komponente $\{t\}$. Startknoten: u.

- Distanzwerte:

  $$\text{dist}(u) = 0, \quad \text{dist}(v) = 1, \quad \text{dist}(w) = 1, \quad \text{dist}(x) = 2, \quad \text{dist}(y) = 2, \quad \text{dist}(z) = 3, \quad \text{dist}(t) = \infty$$

- Kürzester Pfad von u nach z: zum Beispiel $u \to v \to x \to z$ (Länge 3); alternativ $u \to w \to y \to z$ (ebenfalls Länge 3).
- BFS-Baum (Eltern-Beziehungen): `parent[u] = -`, `parent[v] = u`, `parent[w] = u`, `parent[x] = v`, `parent[y] = w`, `parent[z] = x`, `parent[t] = -`
  BFS-Baum-Kanten: $\{(u, v), (u, w), (v, x), (w, y), (x, z)\}$

b) Bestimmung der Zusammenhangskomponenten per DFS
- Komponentenzahl: 2 (eine Komponente $\{u, v, w, x, y, z\}$ und eine isolierte Komponente $\{t\}$). - Komponentennummern $\text{comp}(v)$ für alle Knoten:

$$\text{comp}(u) = 1,\ \text{comp}(v) = 1,\ \text{comp}(w) = 1,\ \text{comp}(x) = 1,\ \text{comp}(y) = 1,\ \text{comp}(z) = 1,\ \text{comp}(t) = 2.$$

c) Reflexion zu Anwendungen
- DFS-Anwendungen: 1) Bestimmung von Zusammenhangskomponenten (durch wiederholtes Starten einer DFS an unbesuchten Knoten); dabei lassen sich auch Zyklen erkennen. 2) Aufbau von Tiefensuch-Bäumen und Einsatz in Topologischer Sortierung von DAGs sowie Zyklus- und Strukturuntersuchungen in Graphen. - BFS-Anwendungen: 1) Kürzeste Pfade in ungewichteten Graphen (z. B. Ermittlung der minimalen Anzahl von Schritten zwischen Akteuren in einem sozialen Netzwerk oder in Netzwerk-Routing-Szenarien). 2) Aufbau eines Traversierungsbaums (BFS-Baum) mit layer-by-layer-Struktur, die sich gut für Broadcast-/Suchaufgaben eignen (z. B. Verbreitung von Informationen in Netzwerken).