Probeklausur

Algorithmen und Datenstrukturen

Universität:	Technische Universität Berlin
Kurs/Modul:	Algorithmen und Datenstrukturen
Bearbeitungszeit:	120 Minuten
Erstellungsdatum:	September 6, 2025

Bearbeitungszeit: 120 Minuten.

Aufgabe 1.

(a) Gegeben sei der ungerichtete, ungewichtete Graph $G = (V, E)$ mit $V = \{s, a, b, c, d, t\}$ und $E = \{(s, a), (s, b), (a, c), (b, c), (c, d), (d, t)\}$. Bestimmen Sie die BFS-Reihenfolge der Knoten, falls mit Startknoten s gestartet wird. Geben Sie die Reihenfolge als Liste an.

(b) Derselbe Graph. Bestimmen Sie eine DFS-Reihenfolge, beginnend bei s. Geben Sie die Reihenfolge der Knoten an.

(c) Geben Sie die Zeitkomplexität von BFS in Abhängigkeit von $n = |V|$ und $m = |E|$ an. Begründen Sie kurz.

(d) Entscheiden Sie, ob der Graph Zyklen enthält, und begründen Sie Ihre Antwort.

Aufgabe 2.

(a) Gegeben sei die Folge von Zahlen $A = [8, 3, 5, 2, 9, 1, 6, 4]$. Sortieren Sie diese Folge mithilfe des Heapsort-Verfahrens. Beschreiben Sie in kurzen Schritten die wesentlichen Phasen und geben Sie die endgültige sortierte Folge an.

(b) Geben Sie die worst-case Zeitkomplexität von Mergesort an und begründen Sie diese.

(c) Welche der folgenden Sortieralgorithmen sind stabil: Quicksort, Mergesort, Heapsort? Begründen Sie Ihre Aussagen.

(d) Diskutieren Sie den Speicherbedarf der drei Algorithmen in Bezug auf zusätzlichen Platz (in-place vs. extra Speicher).

Aufgabe 3.

(a) Gegeben sei ein gewichteter Graph $G = (V, E)$ mit $V = \{s, u, v, w, t\}$ und Kanten: $s-u$ mit Gewicht 3, $s-v$ mit Gewicht 2, $u-w$ mit Gewicht 4, $v-w$ mit Gewicht 1, $w-t$ mit Gewicht 5, $u-t$ mit Gewicht 10. Berechnen Sie den kürzesten Weg von s nach t mittels Dijkstra und geben Sie die Distanz sowie die Knotenfolge an.

(b) Erläutern Sie kurz, warum Dijkstra nicht mit negativ behafteten Kanten funktioniert.

(c) Nennen Sie zwei Unterschiede zwischen Dijkstra und Bellman-Ford in Bezug auf Anwendungsbereich und Komplexität.

(d) Welche Informationen benötigt man zur Bestimmung der Komplexität der Implementierung einer Dijkstra-Variante mit einem Min-Heap?

Aufgabe 4.

(a) Gegeben sei das Flussnetzwerk $\mathcal{N} = (V, E, c)$ mit Knoten $\{S, A, B, C, T\}$ und Kapazitäten: $S \to A : 100$, $S \to B : 50$, $A \to C : 75$, $B \to C : 125$, $C \to T : 100$, $A \to T : 50$. Bestimmen Sie den Max-Flow von S nach T.

(b) Definieren Sie eine S-T-Schranke und erklären Sie, wie deren Kapazität den Max-Flow begrenzt. Geben Sie exemplarisch eine minimale Schnittmenge in diesem Graphen an.

(c) Geben Sie die zeitliche Komplexität bekannter Max-Flow-Algorithmen in Abhängigkeit von $|V|$ und $|E|$ an.

(d) Diskutieren Sie Auswirkungen einer zusätzlichen Kante $B \to A$ mit Kapazität 20 auf das Max-Flow-Verhalten.

Lösungen

Bearbeitungszeit: 120 Minuten.

Aufgabe 1.

(a) BFS-Reihenfolge (Startknoten s): $[s, a, b, c, d, t]$.
Begründung: Von s aus werden in der Reihenfolge a dann b besucht; danach wird c über a bzw. b erreichbar; anschließend d und schließlich t.

(b) DFS-Reihenfolge (Startknoten s): $[s, a, c, b, d, t]$.
Begründung: Mit der Nachbarsortierung {a, b, d} an c ergibt sich der Pfad $s \to a \to c \to b \to d \to t$.

(c) Zeitkomplexität von BFS: $\mathcal{O}(n + m)$.
Begründung: Jede Vertex wird genau einmal enqueued/dequeued und jede Kante genau einmal betrachtet.

(d) Zyklen: *Ja.* Beispiel: $s - a - c - b - s$ bildet einen Zyklus (auch über s–a–c–b–s sichtbar).

Aufgabe 2.

(a) Heapsort auf $A = [8, 3, 5, 2, 9, 1, 6, 4]$.
Phasen (kurz): - Phase 1: Build-Max-Heap aus $A \rightarrow$ Max-Heap: $[9, 8, 6, 4, 3, 1, 5, 2]$.
- Phase 2: Wiederholtes Extrahieren des Maximums und Heapify im Rest: Am Ende erhält man die aufsteigend sortierte Folge.

(b) Worst-Case-Zeitkomplexität von Mergesort: $\mathcal{O}(n \log n)$ (genau $\Theta(n \log n)$).
Begründung: Bei jeder Ebene der Teilung fallen insgesamt n Elemente an; es gibt $\log_2 n$ Ebenen, und das Zusammenführen kostet $\Theta(n)$ pro Ebene.

(c) Stabilität der Algorithmen:
- Quicksort: Nicht stabil in der typischen Implementierung.
- Mergesort: Stabil, wenn der Merge-Schritt so implementiert ist, dass bei gleichen Schlüsseln das Element aus dem linken Teil vor dem aus dem rechten Teil kommt.
- Heapsort: Nicht stabil.

(d) Speicherbedarf (zusätzlicher Platz):
- Quicksort: In-place-Variante möglich; jedoch benötigt man für Rekursionen ($\mathcal{O}(\log n)$ im Durchschnitt; worst-case $\mathcal{O}(n)$).
- Mergesort: Zusätzlich $\mathcal{O}(n)$ Speicher für das Zusammenführen (außer bei speziellen in-place-Varianten).
- Heapsort: In-place; $\mathcal{O}(1)$ zusätzlichen Speicher.

Aufgabe 3.

(a) Dijkstra auf $G = (V = \{s, u, v, w, t\}, E)$ mit Gewichten:
$s-u : 3,\ s-v : 2,\ u-w : 4,\ v-w : 1,\ w-t : 5,\ u-t : 10.$
Ergebnis: kürzester Weg $s \to v \to w \to t$ mit Distanz 8. Die Distanz nach s ist $\text{dist}(t) = 8$ und der Pfad ist [s, v, w, t] (Kosten 2 + 1 + 5).

(b) Dijkstra funktioniert nicht bei negativ behafteten Kanten, weil die Annahme, dass eine beim Herausnehmen aus der Prioritätsstruktur gefundene Distanz final ist, verletzt werden kann, wenn es negative Kanten gibt.
Beispiel: $\{s, a, b\}$ mit $s-a = 2,\ s-b = 2,\ a-b = -1$. Falls b zuerst verarbeitet wird, könnte eine günstigere Distanz zu b erst später durch die Kante $a \to b$ entdeckt werden; dies verletzt die Korrektheit von Dijkstra.

(c) Zwei Unterschiede zwischen Dijkstra und Bellman-Ford:
- Anwendungsbereich: Dijkstra setzt Nicht-Negativität der Kantengewichte voraus; Bellman-Ford funktioniert auch mit negativen Gewichten (und erkennt negative Zyklen).
- Komplexität: Dijkstra mit Binär-Heap $\mathcal{O}((n + m) \log n)$; Bellman-Ford $\mathcal{O}(nm)$.

(d) Informationen zur Komplexität einer Dijkstra-Variante mit Min-Heap:
- $|V| = n$ und $|E| = m$; Repräsentation als Adjazenzliste (typisch) vs. Adjazenzmatrix (Auswirkungen auf Nachbarsuche).
- Zugriffs- und Update-Kosten der Heap-Operationen: Extract-Min $\mathcal{O}(\log n)$ pro Vertex; Decrease-Key pro Kantenrelaxation $\mathcal{O}(\log n)$.
- Anzahl der Operationen: mindestens n Extract-Min und bis zu m Decrease-Key-Operationen.
- Ergebnis: Gesamtkosten $\mathcal{O}((n + m) \log n)$ (bei Binär-Heap).

Aufgabe 4.

(a) Max-Flow von $\mathcal{N} = (V, E, c)$ mit $V = \{S, A, B, C, T\}$ und Kapazitäten $S \to A : 100,\ S \to B : 50,\ A \to C : 75,\ B \to C : 125,\ C \to T : 100,\ A \to T : 50$.
Eine zulässige Max-Flow-Zuweisung ist
- $f(S, A) = 100, \quad f(S, B) = 50$
- $f(A, T) = 50, \quad f(A, C) = 50$
- $f(B, C) = 50$
- $f(C, T) = 100$.

Die Knotenbilanz stimmt, und alle Kanten liegen innerhalb ihrer Kapazitäten. Daraus folgt- Flow(S,T) = 150.

(b) Eine S-T-Schranke (S-T-Cut) ist eine Kantenmenge, die durch Trennung der Knotenmenge in zwei Teile entsteht, so dass S im linken Teil und T im rechten Teil liegt; die Kapazität ist die Summe der Kapazitäten der Kanten vom linken zum rechten Teil. Eine minimale Schnittmenge ist z.B. der Schnitt $\{S\} \mid \{A, B, C, T\}$. Kapazität: $c(S \to A) + c(S \to B) = 100 + 50 = 150$. Nach dem Max-Flow-Min-Cut-Satz entspricht dies dem Max-Flow-Wert.

(c) Zeitkomplexitäten bekannter Max-Flow-Algorithmen (in Abhängigkeit von $|V|, |E|$): - Ford-Fulkerson (allgemein, abhängig von der augmentierenden Pfadlänge; informell abhängig von der Summe der Flüsse)
- Edmonds-Karp (BFS-basierte Implementierung) $\mathcal{O}(|V| \cdot |E|^2)$.
- Dinic (mit Schichten und Fluss-Skalierung) $\mathcal{O}(|V|^2|E|)$ im Worst-Case; $\mathcal{O}(|E|\sqrt{|V|})$ bzw. bessere Grenzwerte in speziellen Modellen (z. B. unit capacities).
- Push-Relabel (Goldberg-Tarjan) $\mathcal{O}(|V|^3)$ im Worst-Case.

(d) Auswirkungen einer zusätzlichen Kante $B \to A$ mit Kapazität 20.
Die maximale Flussobergrenze bleibt unverändert bei 150 (sie ist durch die Summe der Ausflusskapazitäten von S nach $\{A, B\}$ gegeben). Die zusätzliche Kante $B \to A$ kann jedoch alternative Flussrichtungen ermöglichen und gegebenenfalls eine andere Ausnutzung der Kapazitäten ermöglichen (z. B. Umverteilung von Flüssen zwischen A und B), ändert aber nichts an der maximalen Flussgrenze. Eine feasible Max-Flow-Zuweisung mit $f(B \to A) = 20$ lässt sich z. B. durch Umverteilung von Anteilen realisieren, bleibt aber insgesamt bei 150.