Algorithmen & Datenstrukturen

Übungsstunde 13

Michelle Honegger - CAB G59 - 15.12.2025

Miniquiz 12

- Passwort: **johnson8**
- 9 Fragen
- 11 Minuten
- 0.111 Punkte pro richtige Antwort

Programm

- Miniquiz
- Theory Recap
- Semester Recap Part II
- Kahoot!

Besprechung Miniquiz

Kruskal's algorithm on a connected graph $G = (V, E)$ has a runtime of $O(|E| + |V| \log(|V|))$.

- Wahr
- Falsch

Suppose we run Kruskal's algorithm on a connected graph $G = (V, E)$. Then for a vertex $u \in V$, consider the quantity N_u defined as the number of times we change $\texttt{repr}[u]$. What is the tightest upper bound for N_u that holds for any connected graph?

- a. $N_u \leq O(1)$
- b. $N_u \leq O(\log_2(|V|))$
- c. $N_u \leq O(|V|)$

Asymptotically, Floyd-Warshall's runtime is always at least as fast as Johnson's algorithm's runtime.

- Wahr
- Falsch

After the k-th iteration of the outer loop of the Floyd-Warshall algorithm, $d^k_{u,v}$ contains

- a. the length of the shortest $u - v$ path using at most k edges
- b. the length of the shortest $u - v$ path that uses only intermediate vertices from $\{1, \ldots, k\}$

Suppose you run Floyd-Warshall on a directed, weighted graph $G = (V, E)$ with vertices $V = \{1, 2, 3\}$.

Assume the DP table of d^3 is as follows:

0	-2	-3
2	0	-1
3	1	0

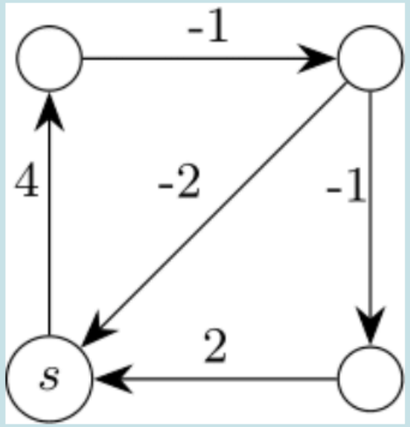
Is it true that any such G will have no negative cycle?

- Wahr
- Falsch

Suppose we have a graph $G = (V, E)$ with weights $c : E \to \mathbb{R}$. Recall in Johnson's algorithm we use Bellman-Ford as a subroutine on a modified version of G to find a function $h : V \to \mathbb{R}$ such that we get a new set of weights $\hat{c} : E \to \mathbb{R}$ defined as $\hat{c}((u, v)) = c((u, v)) + h(u) - h(v)$ for $(u, v) \in E$ which guarantees $\hat{c}((u, v)) \geq 0$.

Then we have $h(v) \leq 0$ for all $v \in V$.

- Wahr
- Falsch



Suppose you run Johnson's algorithm on the above graph and compute the $h : V \to \mathbb{R}$ function as in lecture. What is the value of $h(s)$?

Antwort:

Let $G = (V, E)$ be a directed graph with vertices $V = \{1, 2, 3, 4\}$.

Let A_G be the *adjacency matrix* of G. Suppose that $(A_G)^3$ (i.e., *the adjacency matrix to the power 3*) is equal to

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

How many directed triangles (a collection of three edges which form a closed walk) does G have?

Antwort:

Let $G = (V, E)$ be a directed graph, which has self loops on every vertex, and let A_G be its *adjacency matrix*.

Suppose there exists some $i \in \mathbb{N}$ such that A_G^i has all non-zero entries, then all pairs of vertices are reachable from one another.

- Wahr
- Falsch

Floyd-Warshall

- DP über Zwischenknoten $= O(n^3)$
- dp[i,j] = min(dp[i,j], dp[i,k]+dp[k,j]
- For k
  - For i
    - For j

Johnson

- Erstelle super source
- Berechne Potentiale h(v) von Knoten mittels Bellman Ford
- Reweighting $w'(u,v)=w(u,v)+h(u)-h(v)$
  - Neue Gewichte nicht negativ
- Dijkstra von jedem Knoten

Semester Recap Part II

Problem	**Zeit**	**Speicher**	**DP-Zustand**
Maximum Subarray Sum (MSS)	$O(n)$	$O(1)$	Bestes Subarray bis Position i
Subset Sum	$O(nW)$	$O(nW)$	$dp[i][w]$: erreichbar mit ersten i Elementen
Knapsack (0/1)	$O(nW)$	$O(nW)$	Max. Wert mit Gewicht $\leq w$
Longest Common Subsequence (LCS)	$O(nm)$	$O(nm)$	LCS von Präfixen $[1..i], [1..j]$
Edit Distance	$O(nm)$	$O(nm)$	Kosten zum Transformieren von Präfixen
Jump Game	$O(n)$	$O(n)$	Erreichbarkeit bis Index i
Longest Ascending Subsequence (LAS)	$O(n \log n)$	$O(n)$	Minimales Endelement je Länge
Floyd–Warshall	$O(n^3)$	$O(n^2)$	Kürzeste Wege mit $\leq k$ Zwischenknoten

DP-Ideen – Kurzüberblick

- **MSS (Kadane):** Lokale Entscheidung: erweitern oder neu starten.
- **Subset Sum / Knapsack:** Binäre Entscheidung pro Element (nehmen / nicht nehmen).
- **LCS / Edit Distance:** DP über zwei Sequenzen, Vergleich von Präfixen.
- **Jump Game:** Erreichbarkeit / greedy-artige DP über Positionen.
- **LAS:** DP über Längen, Optimierung mittels Binary Search.
- **Floyd–Warshall:** DP über erlaubte Zwischenknoten.

Begriff	**Beschreibung**
Graph	$G = (V, E)$ mit Knotenmenge V und Kantenmenge E
Gerichtet	Kanten sind geordnete Paare (u, v)
Ungerichtet	Kanten sind ungeordnete Paare $\{u, v\}$
Gewichtet	Gewichtsfunktion $w : E \to \mathbb{R}$
Grad	Anzahl inzidenter Kanten $\deg(v)$
Handschlaglemma	$\sum_{v \in V} \deg(v) = 2\|E\|$ (ungerichtete Graphen)

Konzept	**Kernaussage**
Eulerweg	Benutzt jede Kante genau einmal
Eulerzyklus	Eulerweg mit gleichem Start- und Endknoten
Existenz Eulerzyklus	Alle Knoten haben geraden Grad
Hamiltonpfad	Besucht jeden Knoten genau einmal
Hamiltonproblem	NP-vollständig, keine einfache Charakterisierung

Kantentyp	**Bedingung**	**Bedeutung**
Tree Edge	Intervall geschachtelt	Kante im DFS-Baum
Back Edge	Rückkante zum Vorfahren	Zyklus im Graphen
Forward Edge	Nachfahr, keine Baumkante	Nur in gerichteten Graphen
Cross Edge	Intervalle disjunkt	Zwischen Subbäumen

Wichtige Implikationen:

- Back Edge $\Rightarrow$ gerichteter Zyklus
- Back Edge $\Rightarrow$ keine topologische Sortierung

Algorithmus	**Problem**	**Gewichte**	**Laufzeit**	**Bemerkung**
BFS	SSSP	unge- wichtet	$O(\|V\| + \|E\|)$	Kürzeste Wege nach Kantenanzahl
Dijkstra	SSSP	≥ 0	$O((\|V\| + \|E\|) \log \|V\|)$	Greedy, PQ
Bellman–Ford	SSSP	negativ erlaubt	$O(\|V\|\|E\|)$	Erkennt negative Zyklen
Floyd–Warshall	APSP	negativ erlaubt	$O(\|V\|^3)$	DP über Zwischen-knoten
Johnson	APSP	keine neg. Zyklen	$O(\|V\|(\|E\| + \|V\|) \log \|V\|)$	BF + Dijkstra

Algorithmus	**Strategie**	**Laufzeit**	**Datenstruktur**
Prim	Knoten-basiert	$O((\|V\| + \|E\|) \log \|V\|)$	Priority Queue
Borůvka	Komponenten	$O(\|E\| \log \|V\|)$	—
Kruskal	Kanten-basiert	$O(\|E\| \log \|E\|)$	Union–Find