Lernzettel: Graphen & Algorithmen (30min Test)

1. Grundlagen & Darstellung

Wichtig für: Ankreuzaufgaben und “Wahr/Falsch”-Fragen zur Speicherkomplexität.

Graph-Typen (Seite 371)

- **Gerichtet:** Pfeile (Einbahnstraßen). Kanten sind Paare (u,v).
- **Ungerichtet:** Linien (Beide Richtungen). Kanten sind Mengen {u,v}.
- **Gewichtet:** An den Kanten stehen Zahlen (Kosten, Distanz).

Speicherformen (Vergleich)

Hier wird oft nach dem Speicherbedarf oder der Geschwindigkeit gefragt.

Typ	Beschreibung	Speicherbed	Kante finden (Laufzeit)	Gut für…	Folie
Adjazenzma	2D-Tabelle. 1 = Kante da, 0 = keine.	O(n^2) (Hoch!)	O(1) (Sehr schnell)	Dichte Graphen (viele Kanten)	S. 374-377
Adjazenzlis	Liste pro Knoten mit Nachbarn.	O(n + m) (Gering)	O(Grad des Knotens)	Dünne Graphen (wenige Kanten)	S. 378

(n = Anzahl Knoten, m = Anzahl Kanten)

---

2. Graphtraversierung (Durchlauf)

Wichtig für: “Nummeriere die Knoten in der Reihenfolge des Besuchs”-Aufgaben.

Tiefensuche (DFS - Depth First Search)

- **Prinzip:** Gehe so tief wie möglich in den Pfad, bevor du umkehrst (Backtracking).
- **Ablauf:** Startknoten -> Erster Nachbar -> Dessen Nachbar … -> Sackgasse -> Zurück.
- **Folien:** S. 383 - 399 (Beispielsimulation anschauen!)

Topologisches Sortieren

- **Was ist das?** Eine Reihenfolge, bei der alle Abhängigkeiten erfüllt sind (z.B. Unterhose vor Hose anziehen).
- **Wann geht das?** NUR bei **DAGs** (Gerichtete Azyklische Graphen). **Keine Kreise erlaubt!**
- **Algorithmus (Kahn):**
  1. Berechne “Eingangsgrad” (In-Degree) für alle Knoten (wie viele Pfeile zeigen auf mich?).
  2. Nimm alle Knoten mit In-Degree 0 in eine Liste.
  3. Entferne Knoten aus Liste -> verringere In-Degree der Nachbarn.
  4. Wiederholen.
- **Folien:** S. 400 - 412

Luca.Schmidt@grenzebach.com

---

3. Kürzeste Pfade (SSSP & APSP)

Wichtig für: Den Kern des Tests. Du musst wissen, welcher Algorithmus wann benutzt wird.

Die Übersicht (Auswendig lernen!) - Folie 446

Algorithmus	Typ	Kanten	Laufzeit (grob)
Dijkstra	Einer zu allen (SSSP)	Nur positiv	O(E * log V)
Bellman-Ford	Einer zu allen (SSSP)	Auch negativ	O(V * E)
Floyd-Warshall	Alle zu allen (APSP)	Auch negativ	O(V^3)

(V = Anzahl Knoten, E = Anzahl Kanten)

Floyd-Warshall (Alle zu Alle)

- **Funktionsweise:** Drei verschachtelte Schleifen (k, i, j).
- **Prinzip:** Prüfe, ob der Weg von i nach j kürzer ist, wenn man einen Umweg über k macht.
- **Formel:** `D[i][j] = min(D[i][j], D[i][k] + D[k][j])`
- **Folien:** S. 447 (Formel), S. 467 (Code/Laufzeit O(n^3))

Dijkstra (Der Klassiker)

- **Voraussetzung:** Keine negativen Kanten!
- **Prinzip (Greedy):**
  1. Alle Distanzen auf unendlich, Startknoten auf 0.
  2. Nimm den unbesuchten Knoten mit der *kleinsten* Distanz.
  3. "Relaxieren": Prüfe alle Nachbarn. Ist `Weg zum aktuellen Knoten` + `Kante` < `alter Weg beim Nachbarn`? Wenn ja: Update.
- **Laufzeit:** O(E * log V) (bei Nutzung eines Heaps/Priority Queue).
- **Folien:** S. 473 (Text), S. 475-498 (Ausführliches Beispiel - geh das einmal im Kopf durch!).

A* (A-Star) Suche

- **Ziel:** Schneller als Dijkstra durch Zielgerichtetheit.
- **Formel:** `f(n) = g(n) + h(n)`
  - `g(n)`: Tatsächliche Kosten vom Start bis n.
  - `h(n)`: Geschätzte Kosten von n bis zum Ziel (Heuristik).
- **Bedingung:** Die Heuristik `h(n)` muss **zulässig** sein (darf die wahren Kosten nie überschätzen, z.B. Luftlinie).
- **Folien:** S. 510 (Formel), S. 513 (Eigenschaften).

---

4. Minimale Spannbäume (MST)

Wichtig für: Aufgaben wie “Zeichnen Sie den MST ein”. **Ziel:** Alle Knoten verbinden, Kosten minimieren, **keine Kreise**.

Algorithmus von Prim

- **Strategie:** Starte bei einem Knoten und lass den Baum wachsen.
- **Vorgehen:** Nimm immer die billigste Kante, die von einem *bereits besuchten* Knoten zu einem *unbesuchten* Knoten führt.
- **Gut für:** Dichte Graphen.
- **Folien:** S. 416 - 433 (Beispiel).

Algorithmus von Kruskal

- **Strategie:** Betrachte nur die Kanten.
- **Vorgehen:**
  1. Sortiere **alle** Kanten nach Gewicht (aufsteigend, billigste zuerst).
  2. Nimm die billigste Kante, sofern sie keinen Kreis schließt.
  3. Wiederhole, bis alle Knoten verbunden sind.
- **Erkennung von Kreisen:** Nutzt “Union-Find” Struktur (oder “Chef”-Logik in den Folien).
- **Folien:** S. 438 - 442.

Checkliste

1. **Unterschied Matrix vs. Liste** (S. 374 vs 378).
2. **Topologisches Sortieren** geht nur ohne Kreise (DAG) (S. 411).
3. **Dijkstra** geht nicht bei negativen Kanten (S. 446).
4. **Floyd-Warshall** hat Laufzeit O(n^3) (S. 467).
5. **A-Star Formel:** f = g + h (S. 513).
6. **Prim** wächst vom Startknoten, **Kruskal** nimmt global billigste Kanten (Vergleich S. 416 vs 439).

Luca.Schmidt@grenzebach.com