Lernplan: Test 2 (Datenstrukturen & Hashing)

Ziel: Vorbereitung auf den Test am Dienstag. Die Zeit ist knapp (Do-Di), daher sind die Lerneinheiten extrem fokussiert (ca. 15-20 Min pro Tag).

Relevante Folien: PDF 5 bis PDF 8.

Tag 1 (Donnerstag): Stack, Queue & Listen

Fokus: Die grundlegenden linearen Datenstrukturen verstehen und Operationen auf Papier durchführen können.

1. **Stack (Stapel - LIFO):**
   - Operationen: `push(x)` (drauflegen), `pop()` (wegnehmen).
   - **Laufzeit:** `push()`, `pop()`, `top()` → **O(1)** (Best & Worst Case).
   - **Wichtig:** Wie sieht der Stack nach einer Folge von Befehlen aus?
   - *Übung:* Zeichne einen Stack. Führe aus: `push(5)`, `push(3)`, `pop()`, `push(7)`. Was ist `top()`? (Lösung: 7).
   - *Quelle:* PDF 5 (Seiten 169-192).
2. **Queue (Warteschlange - FIFO):**
   - Operationen: `enq(x)` (hinten anstellen), `deq()` (vorne wegnehmen).
   - **Laufzeit:** `enq()`, `deq()` → **O(1)** (Best & Worst Case).
   - *Ringbuffer:* Verstehe, wie `head` und `tail` (oder `count`) im Array wandern. Was passiert bei Array-Ende? (Modulo-Operator!).
   - *Quelle:* PDF 5 (Seiten 193-213).
3. **Listen (Verkettet):**
   - Unterschied Array vs. Verkettete Liste.
   - Einfach verkettet (`next`) vs. Doppelt verkettet (`next`, `prev`).
   - **Laufzeit:**
     - Einfügen/Löschen am Anfang: **O(1)** (Best & Worst).
     - Suchen/Zugriff auf Element: **Best O(1)** (erstes Element), **Worst O(n)** (letztes Element).
     - Einfügen/Löschen in der Mitte: **O(1)** (wenn Position bekannt), sonst **O(n)** (erst suchen).
   - **Exam-Style Frage:** Gegeben eine Liste `A -> B -> C`. Was passiert, wenn `B` gelöscht wird? (`A.next = C`).
   - *Quelle:* PDF 7.

Tag 2 (Freitag): Hashing Grundlagen

Fokus: Wie finde ich Daten schnell wieder? Hashfunktionen und Kollisionen.

1. **Prinzip:** Schlüssel $k \rightarrow$ Hashfunktion $h(k) \rightarrow$ Index im Array.
2. **Kollisionen:** Zwei Schlüssel landen auf demselben Index.
   - **Offenes Hashing (Open Hashing):** Array von Listen (Buckets). Bei Kollision einfach anhängen.
     - **Laufzeit:** Best O(1), Worst O(n) (alle Elemente in einer Liste).

Luca.schmidt@grenzebach.com

- **Geschlossenes Hashing (Closed Hashing / Open Addressing):** Alle Daten direkt im Array. Wenn Platz voll -> such den nächsten freien.
  - **Laufzeit:** Best O(1), Worst O(n) (viele Kollisionen → langes Sondieren).

3. **Übung:**
   - Berechne $h(x) = x \mod 10$.
   - Hashwerte für 12, 22, 32 sind alle 2. Was passiert bei offenem Hashing? (Liste: 12 -> 22 -> 32).
   - *Quelle:* PDF 6 (Seiten 228-243).

Tag 3 (Samstag): Geschlossenes Hashing & Sondieren

Fokus: Das Sondieren (Suchen nach freiem Platz) ist ein klassischer Kandidat für "Schritt-für-Schritt"-Aufgaben.

1. **Lineares Sondieren:**
   - Bei Kollision an Index i: Prüfe $i + 1$, dann $i + 2$, etc. (Modulo nicht vergessen!).
   - **Laufzeit:** Suchen/Einfügen/Löschen → Best O(1) (keine Kollision), Worst O(n) (Clustering).
   - **Löschen:** Man darf nicht einfach "leer" machen, sonst bricht die Suchkette für nachfolgende Elemente. Man braucht einen Marker (z.B. "G" für Gelöscht / Grave).
2. **Übung (Wichtig!):**
   - Table Size $N = 7$. $h(x) = x \mod 7$. Lineares Sondieren.
   - Füge ein: 7, 14, 21.
   - Ablauf:
     - 7 -> Index 0. (Frei -> OK).
     - 14 -> Index 0 (Besetzt) -> Index 1 (Frei -> OK).
     - 21 -> Index 0 (Besetzt) -> Index 1 (Besetzt) -> Index 2 (Frei -> OK).
   - Endzustand Array: `[7, 14, 21, _, _, _, _]`.
   - *Quelle:* PDF 6 (Seiten 245-260).

Tag 4 (Sonntag): Bäume & Traversierung

Fokus: Begriffe und die drei Wege, einen Baum zu durchlaufen.

1. **Begriffe:** Wurzel, Blatt, Tiefe, Höhe.
   - *Binärbaum:* Max. 2 Kinder pro Knoten.
2. **Traversierung (Muss sitzen!):**
   - **Preorder (Hauptreihenfolge):** V-L-R (Vater, Links, Rechts).
   - **Inorder (Symmetrische Reihenfolge):** L-V-R. (Bei Suchbäumen ergibt das die sortierte Folge!).
   - **Postorder (Nebenreihenfolge):** L-R-V.
   - **Laufzeit:** Alle Traversierungen → **O(n)** (jeder Knoten wird genau einmal besucht).
3. **Übung:**
   - Nutze die `data-structures-visualization.html` (Tab "Bäume"), um die Reihenfolgen zu üben.
   - Schreibe die Preorder für den Baum auf Folie 272 (PDF 8) auf.
   - *Quelle:* PDF 8 (Seiten 271-276).

Luca.schmidt@grenzebach.com

Tag 5 (Montag): Binäre Suchbäume (BST)

Fokus: Suchen, Einfügen und Löschen im geordneten Baum.

1. **Eigenschaft:** Links kleiner, Rechts größer.
2. **Suchen & Einfügen:** Simpel. Immer vergleichen und links/rechts abbiegen.
    - **Laufzeit:** Best O(log n) (balanciert), Worst O(n) (entarteter Baum = Liste).
3. **Löschen (Komplex!):**
    - Fall 1: Blatt löschen -> einfach weg.
    - Fall 2: Ein Kind -> Kind rückt nach.
    - Fall 3: Zwei Kinder -> **Inorder Successor** (Nachfolger) suchen. Das ist der kleinste Wert im rechten Teilbaum (einmal rechts, dann ganz links). Dieser Wert ersetzt den gelöschten Knoten.
    - **Laufzeit:** Best O(log n) (balanciert), Worst O(n) (entarteter Baum).
    - *Quelle:* PDF 8 (Seiten 283-290).

Tag 6 (Dienstag - Testtag): Finale Wiederholung

1. **Checkliste vor dem Test:**
    - ☐ Kannst du Stack (LIFO) und Queue (FIFO) unterscheiden?
    - ☐ Kannst du Hashing mit Linearem Sondieren auf Papier durchrechnen? (Achte auf "Gelöscht"-Marker beim Suchen!).
    - ☐ Kannst du für einen gegebenen Baum die Pre-, In- und Postorder hinschreiben?
    - ☐ Weißt du, wie man im BST mit 2 Kindern löscht? (Stichwort: Inorder Successor).

Luca.schmidt@grenzebach.com