Datenstrukturen: Was ist wichtig?
(Fragestellungen der Klausur)

Was ist wichtig:
Asymptotische Notation und Laufzeit-Analyse

- Asymptotische Notation
  - Wie ist O, o, Ω und Θ definiert?
  - Beherrsche ich die asymptotische Notation?
    - Ist $\sum_{i=1}^{n} i = O(n)$?

Was ist wichtig: Asymptotische Notation und Laufzeit-Analyse

- Asymptotische Notation
  - Wie ist O, o, Ω und Θ definiert?
  - Beherrsche ich die asymptotische Notation?
    - Ist $\sum_{i=1}^{n} i = O(n)$?
    - Ist $\sum_{i=1}^{n} i^k = \Theta(n^{k+1})$?

Was ist wichtig:
Asymptotische Notation und Laufzeit-Analyse

- Asymptotische Notation
  - Wie ist O, o, Ω und Θ definiert?
  - Beherrsche ich die asymptotische Notation?
    - Ist $\sum_{i=1}^{n} i = O(n)$?
    - Ist $\sum_{i=1}^{n} i^k = \Theta(n^{k+1})$?
    - Gilt $\sum_{i=0}^{n} a^i = \Theta(1)$?

Was ist wichtig:
Asymptotische Notation und Laufzeit-Analyse

- Asymptotische Notation
  - Wie ist O, o, Ω und Θ definiert?
  - Beherrsche ich die asymptotische Notation?
    - Ist $\sum_{i=1}^{n} i = O(n)$?
    - Ist $\sum_{i=1}^{n} i^k = \Theta(n^{k+1})$?
    - Gilt $\sum_{i=0}^{n} a^i = \Theta(1)$?
    - Und was bedeutet $\Theta(1)$ überhaupt?

Was ist wichtig: Asymptotische Notation und Laufzeit-Analyse

- Asymptotische Notation
  - Wie ist O, o, Ω und Θ definiert?
  - Beherrsche ich die asymptotische Notation?
    - Ist $\sum_{i=1}^{n} i = O(n)$?
    - Ist $\sum_{i=1}^{n} i^k = \Theta(n^{k+1})$?
    - Gilt $\sum_{i=0}^{n} a^i = \Theta(1)$?
    - Und was bedeutet $\Theta(1)$ überhaupt?
  - Habe ich den Logarithmus verstanden?
    - Wie ist denn überhaupt $\log_a n$ definiert?
    - Für welches a gilt $\log_a n = 5$?

Was ist wichtig: Asymptotische Notation und Laufzeit-Analyse

- Asymptotische Notation
  - Wie ist O, o, Ω und Θ definiert?
  - Beherrsche ich die asymptotische Notation?
    - Ist $\sum_{i=1}^{n} i = O(n)$?
    - Ist $\sum_{i=1}^{n} i^k = \Theta(n^{k+1})$?
    - Gilt $\sum_{i=0}^{n} a^i = \Theta(1)$?
    - Und was bedeutet $\Theta(1)$ überhaupt?
  - Habe ich den Logarithmus verstanden?
    - Wie ist denn überhaupt $\log_a n$ definiert?
    - Für welches a gilt $\log_a n = 5$?
- Laufzeit-Analyse:
  - Wie analysiere ich eine **for-Schleife**, wie eine **while-Schleife** und wie ein **rekursives Programm**?

Was ist wichtig: Asymptotische Notation und Laufzeit-Analyse

- Asymptotische Notation
  - Wie ist O, o, Ω und Θ definiert?
  - Beherrsche ich die asymptotische Notation?
    - Ist $\sum_{i=1}^{n} i = O(n)$?
    - Ist $\sum_{i=1}^{n} i^k = \Theta(n^{k+1})$?
    - Gilt $\sum_{i=0}^{n} a^i = \Theta(1)$?
    - Und was bedeutet $\Theta(1)$ überhaupt?
  - Habe ich den Logarithmus verstanden?
    - Wie ist denn überhaupt $\log_a n$ definiert?
    - Für welches a gilt $\log_a n = 5$?

- Laufzeit-Analyse:
  - Wie analysiere ich eine **for-Schleife**, wie eine **while-Schleife** und wie ein **rekursives Programm**?
  - Habe ich verstanden, wie man **Rekursionsgleichungen** aufstellt?

Was ist wichtig:
Listen, Stacks, Queues und Heaps

- **Listen**:
  - Wann benutze ich eine Liste?
  - Was ist der große Nachteil von Listen und warum tritt dieser Nachteil zum Beispiel für die Adjazenzlisten-Darstellung von Graphen nicht auf?

Was ist wichtig:
Listen, Stacks, Queues und Heaps

- **Listen**:
  - Wann benutze ich eine Liste?
  - Was ist der große Nachteil von Listen und warum tritt dieser Nachteil zum Beispiel für die Adjazenzlisten-Darstellung von Graphen nicht auf?
- **Stacks**:
  - Wie implementiere ich ein rekursives Programm?
  - Wie schnell werden die Pop- und Push-Operationen ausgeführt und wie schnell ist ein Lookup?

Was ist wichtig:
Listen, Stacks, Queues und Heaps

- **Listen**:
  - Wann benutze ich eine Liste?
  - Was ist der große Nachteil von Listen und warum tritt dieser Nachteil zum Beispiel für die Adjazenzlisten-Darstellung von Graphen nicht auf?
- **Stacks**:
  - Wie implementiere ich ein rekursives Programm?
  - Wie schnell werden die Pop- und Push-Operationen ausgeführt und wie schnell ist ein Lookup?
- **Queues**:
  - Wie sind denn überhaupt die Operationen Enqueue und Dequeue definiert? Und wie schnell werden sie implementiert?
  - Und wo haben wir Queues mit großem Vorteil eingesetzt?

Was ist wichtig:
Listen, Stacks, Queues und Heaps

- **Listen**:
  - Wann benutze ich eine Liste?
  - Was ist der große Nachteil von Listen und warum tritt dieser Nachteil zum Beispiel für die Adjazenzlisten-Darstellung von Graphen nicht auf?
- **Stacks**:
  - Wie implementiere ich ein rekursives Programm?
  - Wie schnell werden die Pop- und Push-Operationen ausgeführt und wie schnell ist ein Lookup?
- **Queues**:
  - Wie sind denn überhaupt die Operationen Enqueue und Dequeue definiert? Und wie schnell werden sie implementiert?
  - Und wo haben wir Queues mit großem Vorteil eingesetzt?
- **Heaps**:
  - Was ist eine Priority Queue?
  - Wie funktionieren Repair_up und Repair_down?
  - Wie navigiert man im Heap?

Was ist wichtig: Bäume und Graphen

- **Bäume**:
  - Welche Datenstrukturen gibt es und was sind Vor- und Nachteile?

Was ist wichtig: Bäume und Graphen

- **Bäume**:
  - Welche Datenstrukturen gibt es und was sind Vor- und Nachteile?
  - Wie besuche ich alle Knoten eines Baums?
    - ★ Habe ich Inorder, Postorder und Präorder verstanden?
    - ★ Wie sieht der Pseudocode aus?

Was ist wichtig: Bäume und Graphen

- **Bäume**:
  - Welche Datenstrukturen gibt es und was sind Vor- und Nachteile?
  - Wie besuche ich alle Knoten eines Baums?
    - Habe ich Inorder, Postorder und Präorder verstanden?
    - Wie sieht der Pseudocode aus?
  - Habe ich die **rekursive Programmierung** für Bäume verstanden?
    - Wie zähle ich die Anzahl der Blätter, wie bestimme ich Tiefe und Höhe, wie bestimme ich die Anzahl der Knoten?
    - Was ist ein Rekursionsbaum?

Was ist wichtig: Bäume und Graphen

- **Bäume**:
  - Welche Datenstrukturen gibt es und was sind Vor- und Nachteile?
  - Wie besuche ich alle Knoten eines Baums?
    - Habe ich Inorder, Postorder und Präorder verstanden?
    - Wie sieht der Pseudocode aus?
  - Habe ich die **rekursive Programmierung** für Bäume verstanden?
    - Wie zähle ich die Anzahl der Blätter, wie bestimme ich Tiefe und Höhe, wie bestimme ich die Anzahl der Knoten?
    - Was ist ein Rekursionsbaum?
- **Graphen**:
  - Welche Datenstrukturen gibt es und was sind Vor- und Nachteile?

Was ist wichtig: Bäume und Graphen

- **Bäume**:
  - Welche Datenstrukturen gibt es und was sind Vor- und Nachteile?
  - Wie besuche ich alle Knoten eines Baums?
    - Habe ich Inorder, Postorder und Präorder verstanden?
    - Wie sieht der Pseudocode aus?
  - Habe ich die **rekursive Programmierung** für Bäume verstanden?
    - Wie zähle ich die Anzahl der Blätter, wie bestimme ich Tiefe und Höhe, wie bestimme ich die Anzahl der Knoten?
    - Was ist ein Rekursionsbaum?
- **Graphen**:
  - Welche Datenstrukturen gibt es und was sind Vor- und Nachteile?
  - Wie besuche ich alle Knoten eines Graphen?
    - Habe ich Tiefen- und Breitensuche verstanden? Wie sieht ihr Pseudocode aus?
    - Welche Kantentypen gibt es für ungerichtete und gerichtete Graphen? Und warum ist das überhaupt interessant?

Was ist wichtig: Bäume und Graphen

- **Bäume**:
  - Welche Datenstrukturen gibt es und was sind Vor- und Nachteile?
  - Wie besuche ich alle Knoten eines Baums?
    - Habe ich Inorder, Postorder und Präorder verstanden?
    - Wie sieht der Pseudocode aus?
  - Habe ich die **rekursive Programmierung** für Bäume verstanden?
    - Wie zähle ich die Anzahl der Blätter, wie bestimme ich Tiefe und Höhe, wie bestimme ich die Anzahl der Knoten?
    - Was ist ein Rekursionsbaum?
- **Graphen**:
  - Welche Datenstrukturen gibt es und was sind Vor- und Nachteile?
  - Wie besuche ich alle Knoten eines Graphen?
    - Habe ich Tiefen- und Breitensuche verstanden? Wie sieht ihr Pseudocode aus?
    - Welche Kantentypen gibt es für ungerichtete und gerichtete Graphen? Und warum ist das überhaupt interessant?
  - Wie
    - bestimme ich alle Zusammenhangskomponenten,
    - wie stelle ich fest, ob ein Graph bipartit oder kreisfrei ist,
    - und wie führe ich eine topologische Sortierung durch?

Was ist wichtig: Wörterbücher (1/3)

Ein Wörterbuch ist ein abstrakter Datentyp mit den Operationen

- **Lookup**(x): Ist der Schlüssel x gespeichert?
- **Insert**(x): Speichere den Schlüssel x.
- **Remove**(x): Lösche den Schlüssel x.

Was ist wichtig: Wörterbücher (1/3)

Ein Wörterbuch ist ein abstrakter Datentyp mit den Operationen

- **Lookup**(x): Ist der Schlüssel x gespeichert?
- **Insert**(x): Speichere den Schlüssel x.
- **Remove**(x): Lösche den Schlüssel x.

- **Listen** passen sich ideal der Größe der Datenmenge an wie etwa im Fall der „Darstellung dünnbesetzter Matrizen“ oder in der Adjazenzlisten-Darstellung.

Was ist wichtig: Wörterbücher (1/3)

Ein Wörterbuch ist ein abstrakter Datentyp mit den Operationen

- **Lookup**(x): Ist der Schlüssel x gespeichert?
- **Insert**(x): Speichere den Schlüssel x.
- **Remove**(x): Lösche den Schlüssel x.

- **Listen** passen sich ideal der Größe der Datenmenge an wie etwa im Fall der „Darstellung dünnbesetzter Matrizen“ oder in der Adjazenzlisten-Darstellung.
- **Binäre Suchbäume**:
  - \+ Gute erwartete Laufzeit.
  - \+ Ermöglicht die Binärsuche und ist „Ausgangspunkt“ für AVL-Bäume.
  - \- Schlechte worst-case Laufzeit und verlangt relativ viel Speicherplatz.
  - ? Wie implementiert man Lookup, Insert und Remove?

Was ist wichtig: Wörterbücher (1/3)

Ein Wörterbuch ist ein abstrakter Datentyp mit den Operationen

- **Lookup**(x): Ist der Schlüssel x gespeichert?
- **Insert**(x): Speichere den Schlüssel x.
- **Remove**(x): Lösche den Schlüssel x.

- **Listen** passen sich ideal der Größe der Datenmenge an wie etwa im Fall der „Darstellung dünnbesetzter Matrizen“ oder in der Adjazenzlisten-Darstellung.
- **Binäre Suchbäume**:
  - \+ Gute erwartete Laufzeit.
  - \+ Ermöglicht die Binärsuche und ist „Ausgangspunkt“ für AVL-Bäume.
  - \- Schlechte worst-case Laufzeit und verlangt relativ viel Speicherplatz.
  - ? Wie implementiert man Lookup, Insert und Remove?

Was ist wichtig: Wörterbücher (2/3)

- **AVL-Bäume**:
  - \+ Die worst-case Laufzeit ist logarithmisch.
  - \- Relativ viel Speicherplatz notwendig für Zeiger und Balance-Information.
  - ? Wie ist die Balance-Bedingung definiert?
  - ? Wie implementiert man Lookup, Insert und Remove?

Was ist wichtig: Wörterbücher (2/3)

- **AVL-Bäume**:
  - \+ Die worst-case Laufzeit ist logarithmisch.
  - \- Relativ viel Speicherplatz notwendig für Zeiger und Balance-Information.
  - ? Wie ist die Balance-Bedingung definiert?
  - ? Wie implementiert man Lookup, Insert und Remove?
- **(a,b)-Bäume**:
  - \+ Unschlagbar in Anwendungen für langsame Speicher,
  - \- werden aber von Hashing und AVL-Bäumen „geschlagen“, wenn die Daten in einen schnellen Speicher passen.
  - ? Wie ist die (a, b)-Eigenschaft und wie ist die Suchstruktur von (a, b)-Bäumen definiert?
  - ? Wie führt man einen Lookup durch, wie ein Insert oder Remove?
  - ? Wie groß ist die Tiefe von (a, b)-Bäumen mit n Schlüsseln höchstens und wie zeigt man das?

Was ist wichtig: Wörterbücher (3/3)

- **Hashing mit Verkettung**:
  - \+ hat die sehr schnelle erwartete Laufzeit $O(1) + \lambda$,
  - \- aber verlangt relativ viel Speicher.
  - +/- Die worst-case Laufzeit ist schlecht, aber gutes Verhalten in praktischen Anwendungen.

Was ist wichtig: Wörterbücher (3/3)

- **Hashing mit Verkettung**:
  - \+ hat die sehr schnelle erwartete Laufzeit $O(1) + \lambda$,
  - \- aber verlangt relativ viel Speicher.
  - +/- Die worst-case Laufzeit ist schlecht, aber gutes Verhalten in praktischen Anwendungen.
- **Hashing mit offener Adressierung**:
  - \+ ist mit erwarteter Laufzeit $O(1/(1-\lambda))$ etwas langsamer als Hashing mit Verkettung.
  - ? Wie funktioniert das Argument? Welche Annahmen macht man?
  - \- Der Auslastungsfaktor λ darf nicht zu groß werden!
  - +/- Sehr „speicherplatz-freundlich“, mit schlechter worst-case Laufzeit, aber guter Leistung für kleine λ.
  - ? Welche Hashfunktionen wählt man? Wie ist das lineare Austesten, wie ist doppeltes Hashing definiert? Was sind Vor- und Nachteile?
  - ? Wann wähle ich Hashing mit Verkettung, wann Hashing mit offener Adressierung?