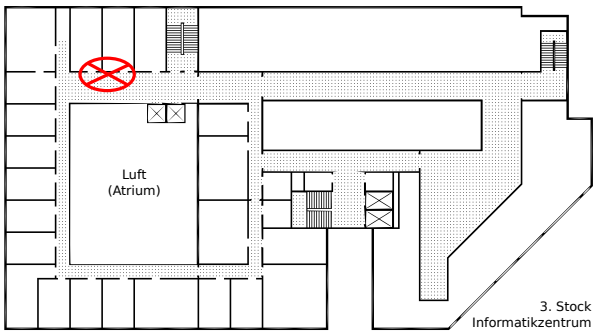
Algorithmen und Datenstrukturen 2 **SoSe 2022**
Prof. Dr. Sándor Fekete **Abgabe:** 20.07.22
Matthias Konitzny **Rückgabe:** ab 27.07.22

Hausaufgabenblatt 5

Abgabe der Lösungen bis zum 20.07.22 um 12:00 Uhr im Hausaufgabenschrank bei Raum IZ 337 (siehe Skizze rechts). Es werden nur mit einem dokumentenechten Stift (kein Rot!) geschriebene Lösungen gewertet. **Bitte die Blätter zusammenheften und vorne deutlich mit beiden Namen, Matrikel- Übungs- und Gruppennummer versehen!**



Hausaufgabe 1 (Wahlaufgabe): (8 Punkte)

Im Folgenden kann zwischen einer theoretischen und einer praktischen Aufgabe gewählt werden. Es muss lediglich **eine** der Aufgaben bearbeitet werden, um die vollen Punkte zu erreichen. (Hinweis: Möchtest du dennoch beide Aufgaben bearbeiten, werten wir die Aufgabe mit der höheren Punktzahl.)

Theoretische Aufgabe (Graphenprobleme) (4+1+3 Punkte)

In dieser Aufgabe betrachten wir die folgenden zwei Entscheidungsprobleme:

1. Vertex Cover:

   Gegeben: Ein Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}$

   Frage: Existiert eine Menge $VC \subseteq V$ mit $|VC| \leq k$, sodass für jede Kante $\{u, v\} \in E$ gilt: $|\{u, v\} \cap VC| \geq 1$ (Mindestens ein Endknoten ist in VC)?

2. Independent Set:

   Gegeben: Ein Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}$

   Frage: Existiert eine Menge $IS \subseteq V$ mit $|IS| \geq k$, sodass für jede Kante $\{u, v\} \in E$ gilt: $|\{u, v\} \cap IS| \leq 1$ (Höchstens ein Endknoten ist in IS)?

a) Betrachte den Graphen G aus Abbildung 1. Gib ein Vertex Cover der Größe $k = 4$ an. Gibt es ein kleineres Vertex Cover? Begründe deine Antwort.

b) Betrachte weiterhin den Graphen G aus Abbildung 1. Gib ein Independent Set der Größe $k = 2$ an.

c) Sei $H = (V, E)$ ein Graph.

   Zeige: VC ist genau dann ein Vertex Cover in H, wenn $IS = V \setminus VC$ ein Independent Set in H ist.

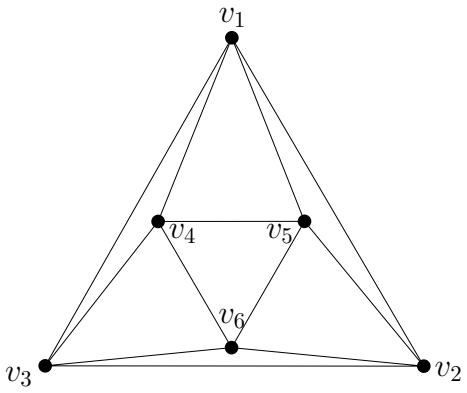


Abbildung 1: Der Oktaeder-Graph G.

Praktische Aufgabe (Hashing Implementierung) **(1+3+4 Punkte)**

In dieser Aufgabe sollen verschiedene Formen von Hashtabellen implementiert werden. Mache dich vor dem Lösen der Aufgaben mit dem Template unter `https://ibr.cs.tu-bs.de/courses/ss22/aud2/aufgaben/blatt5_template.py` vertraut. Das Template enthält auch eine Reihe von Beispielinstanzen mit denen die Implementierung überprüft werden kann. Für die Überprüfung der Zeilenbegrenzungen werden Kommentare und Leerzeilen ignoriert.

Abgabe: Die Lösung dieser Aufgabe muss in Form einer Python-Datei (.py) an die/den jeweilge*n Übungsleiter*in gesendet werden. In der Mail muss der Name, die Matrikel-Übungs- und Gruppennummer angegeben werden. Die jeweiligen Mailadressen gibt es unter `https://aud2.ibr.cs.tu-bs.de/index.php/organisation/`.

a) Zunächst soll eine primitive Form einer Hashtabelle implementiert werden, welche noch keine Strategie zur Auflösung von möglichen Adresskollisionen bereitstellt. Die Hashtabelle muss allerdings bei einer Anfrage überprüfen, ob ein gesuchter Schlüssel wirklich zu dem gespeicherten Objekt gehört. Die Implementierung darf maximal 30 Zeilen lang sein.

   (Hinweis: Die speziellen „magischen“ Python-Funktionen beginnen und enden mit einem doppelten Unterstrich. So kennst du wahrscheinlich bereits die Methode __INIT__ zum initialisieren einer Klasse. Die hier verwendeten Funktionen __GETITEM__ und __SETITEM__ werden mit dem Operator für eckige Klammern verknüpft und ermöglichen Zugriffe der Form $abc = obj[2]$ bzw. $obj[0] = 1$. Die Methode __DELITEM__ ermöglicht das Entfernen von Einträgen mit Hilfe des Schlüsselwortes DEL (del $obj[2]$).)

b) Eine einfache Methode Adresskollisionen aufzulösen ist die Verwendung verketteter Listen in den eigentlichen Speicherzellen. Implementiere nun eine Hashtabelle welche Kollisionen mit Hilfe dieses Verfahrens auflöst. Es muss *kein* Verfahren für Rehashing implementiert werden. Die Implementierung darf insgesamt maximal 45 Zeilen lang sein.

c) Eine weitere Methode zur Behandlung von Adresskollisionen stellt die Methode der offenen Adressierung dar. Dabei wird Objekten nach einer festen Sondierungsreihenfolge ein neuer Platz zugewiesen, falls der ursprüngliche Platz bereits belegt ist. Implementiere eine Hashtabelle, welche Kollisionen mit Hilfe offener Adressierung behandelt. Die Implementierung darf maximal 60 Zeilen lang sein.

Hausaufgabe 2 (Reduktion): **(3+1+6+2 Punkte)**

In dieser Aufgabe betrachten wir PARTITION und wollen analysieren, wie schwer das Problem ist. Aus der Vorlesung ist bekannt, dass SUBSET SUM NP-vollständig ist.

a) Reduziere PARTITION auf SUBSET SUM.

b) Was sagt die Reduktion aus a) über die Komplexitätsklasse von PARTITION aus?

c) Zeige: PARTITION ist NP-vollständig. (Hinweis: Führe dazu eine Reduktion von SUBSET SUM auf PARTITION durch.)

d) Welche Folge(n) hat es, falls es einen Polynomialzeit-Algorithmus für PARTITION gibt? – Dann gibt es einen Polynomialzeit-Algorithmus für...

- ...jedes Problem in NP. ☐
- ...jedes NP-vollständige Problem. ☐
- ...jedes NP-schwere Problem. ☐

Kreuze die richtige(n) Antwort(en) an und begründe deine Wahl.