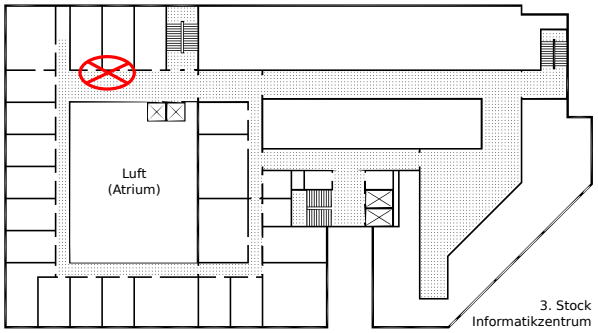
Algorithmen und Datenstrukturen II **SoSe 2019**
Dr. Linda Kleist **Abgabe:** 11.07.19
Arne Schmidt **Rückgabe:** 17./19.07.19

Übungsblatt 5

Abgabe der Lösungen bis zum 11.07.19 um 13:15 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 eigenem Namen, Matrikel- und Gruppennummer, sowie Studiengang versehen!**



Dieses Blatt besteht aus einer Präsenzaufgabe, die in der kleinen Übung besprochen wird, sowie aus zwei Hausaufgaben, die abgegeben werden müssen und bewertet werden.

Präsenzaufgabe:

a) Wir betrachten die Hashfunktion $t(i, x) \equiv (2x^2 - 4 + (4x + 1)i) \mod 13$. Wie lässt sich $t(i, x)$ als rekursive Funktion realisieren?

b) Wir betrachten ein anfangs leeres Array A der Größe 13 mit den Speicherzellen $A[0], \ldots, A[12]$. In diesem Array führen wir offenes Hashing mit der Hashfunktion $t(x, i)$ aus Teil a) durch.

Dabei ist x ein einzusetzender Schlüssel und i die Nummer des Versuches, x in eine unbesetzte Speicherzelle des Arrays zu schreiben, beginnend bei $i = 0$. Berechne zu jedem der folgenden Schlüssel die Position, die er in A bekommt:

$$4, \quad 2, \quad 10, \quad 8, \quad 5, \quad 15, \quad 1$$

Dabei sollen die Schlüssel in der gegebenen Reihenfolge eingefügt werden und der Rechenweg soll klar erkennbar sein.

c) Wie ist der Belegungsfaktor β definiert und welchen Belegungsfaktor besitzt A aus Teil b) nach Einfügen der Elemente?

d) Sei $h(x)$ eine Hashfunktion und $\text{Prob}[h(x) = j] = \frac{1}{m}$, wobei m die Größe der Hashtabelle bezeichne. Wie hoch ist die Wahrscheinlichkeit, dass drei Datensätze kollisionsfrei in eine Hashtabelle der Größe $m = 20$ eingefügt werden können? Wie hoch ist die Wahrscheinlichkeit (in Prozent) bei vier, sechs, acht, zehn, 15 oder 20 Datensätzen?

Hausaufgabe 1 (Bin Packing - First Fit): **(4+4 Punkte)**

Ein möglicher Algorithmus für Bin Packing ist *First Fit*: Verwende die Objekte in der gegebenen Reihenfolge und packe das Objekt in den ersten Bin, in das es noch reinpasst.

a) Zeige: First Fit ist eine 2-Approximation für Bin Packing.

b) Konstruiere für beliebiges $N \in \mathbb{N}$ eine Sequenz von Objekten, sodass die optimale Verteilung $N^* > N$ Behälter benötigt, First Fit aber mindestens $\frac{5}{3}N^*$ Behälter verwendet.

Hausaufgabe 2 (Set Cover): **(4+3+2+2+1 Punkte)**

Gegeben sind eine endliche Menge U und eine Familie $\mathcal{F}$ von Teilmengen von U. Wir nehmen an, dass jedes Element aus U in einer Menge in $\mathcal{F}$ vorkommt. Ein *Set Cover (SC)* von $(U, \mathcal{F})$ ist eine Menge $F \subseteq \mathcal{F}$, die U überdeckt, d.h. für jedes Element $u \in U$ gibt es eine Menge $M \in F$ mit $u \in M$. Gesucht ist ein Set Cover kleinstmöglicher Kardinalität.

Als Beispiel betrachte $U := \{1,2,3,4,5,6\}$ und $\mathcal{F} := \{\{1,2\},\{1,4\},\{3,6\},\{2,3,4\},\{1,2,5\},\{2,3\}\}$. $F := \{\{1,4\},\{3,6\},\{1,2,5\}\}$ ist ein Set Cover von $(U,\mathcal{F})$. Es kann schnell überprüft werden, dass es kein Set Cover mit zwei Elementen aus $\mathcal{F}$ gibt; daher ist F ein kleinstes Set Cover.

a) Zeige, dass Set Cover NP-schwer ist. (Hinweis: Nutze Vertex Cover.)

Da Set Cover also NP-schwer ist, bietet es sich an, Approximationsalgorithmen zu betrachten. Der folgende Algorithmus (GreedySC) bestimmt ein Set Cover.

```
1:  function GreedySC(U, F)
2:      C := ∅                                  ▷ Menge der bereits überdeckten Elemente
3:      C̄ := U                                  ▷ Menge der noch zu überdeckenen Elemente
4:      SC := ∅                                 ▷ Set Cover
5:      while C ≠ U do
6:          S := argmax_{M∈F} |M ∩ C̄|           ▷ Menge mit den meisten nicht-überdeckten Elementen
7:          α := 1/|S ∩ C̄|
8:          For each s ∈ S ∩ C̄ do kosten(s) := α
9:          C := C ∪ S
10:         C̄ := C̄ \ S
11:         SC := SC ∪ {S}
12:     return SC
```

b) Wende GreedySC auf folgende Instanz an: $U := \{1,\dots,10\}$, $\mathcal{F} := \{F_1,\dots,F_5\}$ mit $F_1 = \{1,2,3,7,9\}$, $F_2 = \{4,5,6,8,10\}$, $F_3 = \{1,2,3,4,5,6\}$, $F_4 = \{7,8\}$ und $F_5 = \{9,10\}$.
Gib dabei für jede Iteration S, $S \cap \bar{C}$, α, C sowie $\bar{C}$ an. Führe nach Ablauf des Algorithmus für alle $e \in S$ die kosten(e) auf.

c) Zeige: In jeder Iteration von GreedySC gilt $\alpha \leq \mathrm{OPT}/|\bar{C}|$.
(Hinweis: In jeder Iteration kann $\bar{C}$ mit $\leq$ OPT Elementen überdeckt werden.)

d) Sei $U = \{e_1,\dots,e_n\}$, wobei die Elemente in der Reihenfolge aufgelistet seien, in der sie GreedySC in C einfügt. Folgere mit Hilfe von c), dass Für alle k gilt kosten$(e_k) \leq \frac{\mathrm{OPT}}{n-k+1}$.

e) Folgere, dass GreedySC liefert eine H_n-Approximation von Set Cover, wobei $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$.
(Hinweis: Verteile die Kosten des gefundenen Set Covers mit Hilfe von kosten(e_i) auf U.)