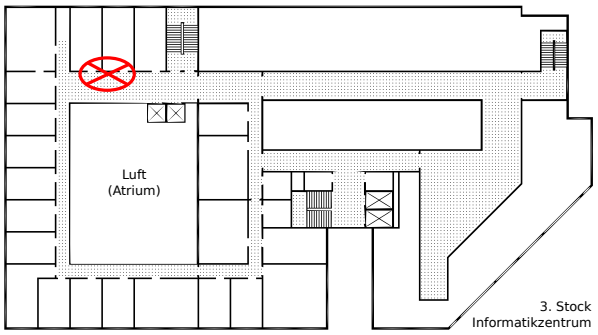
Abteilung Algorithmik **Sommer 2013**
Institut für Betriebssysteme und Rechnerverbund
TU Braunschweig

Prof. Dr. Sándor P. Fekete
Stephan Friedrichs

Algorithmen und Datenstrukturen II
Übung 4 vom 19. 06. 2013

Abgabe der Lösungen bis zum Mittwoch, den 03. 07. 2013 um 14:00 im Hausaufgabenrückgabeschrank.

Bitte die Blätter vorne deutlich mit eigenem Namen sowie Matrikel- und Gruppennummer versehen!



Aufgabe 1 (NP-Reduktionen): Betrachte die 3SAT Instanz $E = (x_1 \vee \bar{x}_3 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_5) \wedge (x_2 \vee \bar{x}_4 \vee \bar{x}_5) \wedge (\bar{x}_1 \vee x_4 \vee x_5)$.

a) Reduziere E auf eine Instanz von Subset Sum (Problem 1.8 aus dem Skript). (Hinweis: Beispiel 1.34)

b) Gib einen Algorithmus an, der eine beliebige 3SAT-Instanz in polynomieller Zeit in eine Instanz von Subset Sum umwandelt, die genau dann eine Lösung besitzt, wenn die 3SAT-Instanz lösbar ist.

c) Mittels des Algorithmus aus der vorigen Teilaufgabe können wir eine Instanz von 3SAT erst in eine von Subset Sum umrechnen und diese dann mit Algorithmus 1.11 (Dynamic Programming für Subset Sum) in Zeit $O(nZ)$ lösen. Folgt daraus, dass wir 3SAT in polynomieller Zeit lösen können? Begründe deine Aussage.

(4+4+2 Punkte)

Aufgabe 2 (Approximation von Set Cover): Gegeben sind eine endliche Menge U und eine Familie $\mathcal{F}$ von Teilmengen von U. $C \subseteq U$ *überdeckt* alle Elemente in C. Ein *Set Cover (SC)* von $(U, \mathcal{F})$ ist eine Auswahl der Mengen in $\mathcal{F}$, $F = \{F_1, \ldots, F_m\} \subseteq \mathcal{F}$, die U überdeckt: $U = \bigcup_{i=1}^{m} F_i$. Gesucht ist ein Set Cover kleinstmöglicher Kardinalität. Es darf angenommen werden, dass $\mathcal{F}$ ein Set Cover von $(U, \mathcal{F})$ ist.

Da Vertex Cover $\mathcal{NP}$-vollständig ist und SC eine Verallgemeinerung von Vertex Cover darstellt, ist auch SC $\mathcal{NP}$-vollständig. Deswegen macht es Sinn, Approximationsalgorithmen für SC zu betrachten. Algorithmus 1 ist ein Greedy-Approximationsalgorithmus für SC, der im Folgenden genauer untersucht werden soll. Dabei dienen Zeilen 7 und 8 lediglich der Analyse und spielen für die Korrektheit von Algorithmus 1 keine Rolle.

```
1: function GreedySC(U, F)
2:     C ← ∅
3:     C̄ ← U
4:     SC ← ∅

5:     while C ≠ U do
6:         S ← max_{S∈F} |S ∩ C̄|

7:         α ← 1/|S ∩ C̄|
8:         ∀s ∈ S ∩ C̄ :   price(s) ← α

9:         C ← C ∪ S
10:        C̄ ← C̄ \ S
11:        SC ← SC ∪ {S}
12:    end while

13:    return SC
14: end function
```

Algorithmus 1: Greedy Approximation von Set Cover

Sei OPT die Größe eines kleinsten Set Covers von $(U, \mathcal{F})$.

a) Konstruiere ein Beispiel $(U, \mathcal{F})$, dessen optimale Lösung mindestens zwei Teilmengen $F_1, F_2 \in \mathcal{F}$ benötigt und bei dem Algorithmus 1 nicht die optimale Lösung ausgibt.

b) Wende Algorithmus 1 auf Dein Beispiel an. Gib dabei für jede Iteration S, $S \cap \bar{C}$, α, C sowie $\bar{C}$ an. Führe nach Ablauf des Algorithmus für alle $e \in S$ price(e) auf.

c) Zeige: Algorithmus 1 ist korrekt, gibt also immer ein Set Cover aus und terminiert.

d) Zeige: In jeder Iteration von Algorithmus 1 gilt $\alpha \leq \mathrm{OPT}/|\bar{C}|$. (Hinweis: In jeder Iteration kann $\bar{C}$ mit $\leq$ OPT Elementen überdeckt werden.)

e) Sei $U = \{e_1, \ldots, e_n\}$, wobei die Elemente in der Reihenfolge aufgelistet seien, in der Algorithmus 1 sie in C einfügt. Folgere aus der letzten Teilaufgabe: Für alle k gilt $\mathrm{price}(e_k) \leq \frac{\mathrm{OPT}}{n-k+1}$.

f) Folgere: Algorithmus 1 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 price(e_i) auf U.)

(1+3+2+2+1+1 Punkte)