Philipps-Universität Marburg
Fachbereich Mathematik und Informatik
Arbeitsgruppe Algorithmik

Wintersemester 2024/2025

Version 2024-10-25 09:59

Prof. Dr. Sebastian Wild
Dr. Nikolaus Glombiewski

Übungen zur Vorlesung
Effiziente Algorithmen

Abgabe: 01.11.2024,
bis **spätestens** 19:00 Uhr
über die ILIAS Plattform

Übungsblatt 1

Aufgabe 1.1: Korrektheisbeweis (1+4) (5 Punkte)

Gegeben ist folgender Algorithmus, der die Potenz x^n für $n \in \mathbb{N}$ berechnet. Dabei ist `div` die ganzzahlige Division und `mod` der Rest der Division.

Algorithm 1: Power

```
Input: x, n ∈ ℕ
p = 1;
if x ≠ 0 then
    while n ≠ 0 do
        if n mod 2 == 1 then
            p = p · x;
        n = n div 2;
        x = x · x;
else
    p = 0;
return p;
```

a) Durchlaufen Sie den Algorithmus für folgende Werte: $x = 2$, $n = 7$. Geben Sie die Werte von x, n, p nach jedem Schleifendurchlauf an.

b) Verifizieren Sie die Korrektheit des Algorithmus mit Hilfe des Hoare-Kalküls. Geben Sie hierbei insbesondere Vor- und Nachbedingungen sowie eine geeignete Schleifeninvariante an.

Aufgabe 1.2: Lemmata über Asymptotiken (2+1) (3 Punkte)

Beweisen Sie die folgenden Lemmata:

1. Sei $P(n) = a_0 + a_1 n + \cdots + a_k n^k$ ein beliebiges Polynom mit $a_k > 0$. Dann gilt $|P(n)| \sim a_k n^k$.
2. Für alle $\alpha, \epsilon \in \mathbb{R}^+$ gilt $n^\alpha \in o(n^{\alpha+\epsilon})$.

Aufgabe 1.3: O-Notation (1+1+2+2) (6 Punkte)

Beweisen oder widerlegen Sie folgende Behauptungen:

a) $2^{2n} \sim 4^n$

b) $\sqrt[n]{n} = o(1)$

c) Für beliebige Funktionen: $f, g : \mathbb{N} \to \mathbb{R}$ gilt $f = O(g)$ oder $g = O(f)$.

d) $(n+m)^4 \in \Theta(n^4)$ für $m \in \mathcal{O}(1)$

Aufgabe 1.4: Algorithmen Entwurf (1+2+2+1) (6 Punkte)

In der Vorlesung wurden mehrere Lösung für das Maximum Subarray Problem vorgestellt. In dieser Aufgabe sollen Sie eine Lösung für die 2-dimensionale Variante des Problems erarbeiten. Die Definition für das 2D Maximum Subarray Problem ist wie folgt:

Gegeben sei eine ganzzahlige $n \times n$ Matrix M, sodass $M[i,j]$ mit $0 \leq i, j \leq n$ den Wert der Matrix in der i-ten Zeile und der j-ten Spalte liefert. Sei $z_k(x_1, x_p) = \sum_{i=x_1}^{x_p} M[i,k]$ die Summe der Zeilen x_1 bis x_p der k-ten Spalte der Matrix. Ferner sei $S(x_1, x_p, y_1, y_q) = \sum_{i=y_1}^{y_q} z_i(x_1, x_p)$ die Summe der Submatrix von M der Spalten y_1 bis y_q mit jeweils den Zeilen x_1 bis x_p. Bestimmen Sie die Koordinaten der maximalen Submatrix $S_{\max}$ in M, d.h.

$S_{\max} = \left\{(x_1, x_p, y_1, y_q) \;\middle|\; \forall(x'_1, x'_p, y'_1, y'_q) : S(x_1, x_p, y_1, y_q) \geq S(x'_1, x'_p, y'_1, y'_q)\right\}$.

Sie finden die Code Basis für diese Aufgabe auf der Kurswebseite. Ferner ist die Code Basis auch eingebettet in diesem PDF (letzteres könnte bei einigen PDF Viewern nicht funktionieren):

SubArrayProblem.java

Anmerkung: $S_{\max}$ gibt eine Menge zurück, im Code und in dieser Aufgabenstellung reicht es eine mögliche Lösung zurückzugegeben.

a) Die `bruteForce` Methode der `SubArrayProblem` Klasse liefer eine (sehr naive) Lösung für 2D Maximum Subarray Problem zurück. Geben Sie eine möglichst kleine obere Schranke in O-Notation für den `bruteForce` Algorithmus an. Begründen Sie Ihre Lösung.

b) Beschreiben Sie textuell einen Algorithmus, welcher das Problem in Zeit $\mathcal{O}(n^3)$ löst.

   Hinweis: Sie können die $\Theta(n)$ Lösung für das 1-dimensionale Problem aus der Vorlesung verwenden. Begründen Sie die Laufzeit Ihrer Lösung.

c) Implementieren Sie die Ihre Lösung aus b) in einer Methode `efficient` in der Klasse `SubArrayProblem`.

d) Erstellen Sie in der Klasse `SubArrayProblem` ein geeignetes Experiment, um die Laufzeiten von `bruteForce` und `efficient` experimentell zu ermitteln. Plotten Sie einen Vergleich der Ergebnisse und beschreiben Sie diese.