Technische Universität München
Fakultät für Informatik, I-14
Prof. Harald Räcke
Robin Münk

SoSe 2025
Übungsblatt 3
9. Mai 2025

Algorithmen und Datenstrukturen

Aufgabe 1 Laufzeiten

Gegeben seien Algorithmen A_1 , A_2 und A_3, welche die Laufzeiten

$$T_1(n) = c_1 n,\ T_2(n) = c_2 n^3 \text{ und } T_3(n) = c_3 2^n$$

haben, wobei die $c_i, i = 1,2,3$, konstant sind. Für jeden Algorithmus bezeichne $m_i, i = 1,2,3$, die maximale Größe der Eingabe, die innerhalb einer fest vorgegebenen Zeit T auf einem Rechner R verarbeitet werden kann.

Wie ändern sich die Zahlen $m_i, i = 1,2,3$, wenn der Rechner R durch einen k-mal schnelleren Rechner ersetzt wird?

Beispiellösung:

Sei jeweils m' die maximale Größe der Eingabe, die in Zeit T auf einem k-mal schnelleren Rechner verarbeitet werden kann. Dann,

a)

$$\begin{aligned} \frac{T_1(m')}{k} &= c_1 m \\ T_1(m') &= c_1 k m \\ m' &= k m \end{aligned}$$

b)

$$\begin{aligned} \frac{T_2(m')}{k} &= c_2 m^3 \\ T_2(m') &= c_2 (m\sqrt[3]{k})^3 \\ m' &= m\sqrt[3]{k} \end{aligned}$$

c)

$$\begin{aligned} \frac{T_3(m')}{k} &= c_3 2^m \\ T_3(m') &= c_3 2^{m+\log_2 k} \\ m' &= m + \log_2 k \end{aligned}$$

Aufgabe 2 Maximale Teilsequenz - Bruteforce

a) **Implementierung.** Schreiben Sie eine C/C++ Funktion, die als Eingabe einen `vector<int> A` bekommt und den Wert der maximalen Teilsequenz $y(A)$ berechnet.

$$y(A) = \max_{i \leq j} \sum_{k=i}^{j} x_k$$

Ihr Algorithmus muss nicht die optimale Laufzeit haben. Nutzen Sie gerne auch das Testing Framework[1].

b) **Analyse.** Geben Sie die Anzahl der Grundoperationen an, die Ihr Algorithmus abhängig von der Länge der Eingabe n benötigt (analog der Schulmethode für die Multiplikation von Zahlen).

Als Grundoperationen betrachten wir Multiplikation und Addition von Integern.

Beispiellösung:

Sie finden die Lösungen zu Programmieraufgaben in `blatt1solution.cpp`.

Dieser Algorithmus ist sehr ineffizient. Wie er optimiert werden kann, wird in einer späteren Vorlesung behandelt.

Die genaue Anzahl der Grundoperationen lässt ich bei diesem Algorithmus schwer feststellen. Wenn man Integeradditionen und Integermultiplikationen zählt, ergibt sich folgende Formel:

$$\sum_{l=1}^{n}(1+\sum_{r=l}^{n}(1+\sum_{k=l}^{r}2))$$

Wir geben eine Abschätzung an und zeigen, dass die Anzahl der Operationen ein Polynom 3ten Grades ist, d.h die Anzahl der Operationen hat die folgende Form: $a_3 * n^3 + a_2 * n^2 + a_1 * n + a_0$, für Konstanten $a_i, 0 \leq i \leq 3$

$$\sum_{l=1}^{n}(1+\sum_{r=l}^{n}(1+\sum_{k=l}^{r}2)) = \sum_{l=1}^{n}1 + \sum_{l=1}^{n}\sum_{r=l}^{n}1 + \sum_{l=1}^{n}\sum_{r=l}^{n}\sum_{k=l}^{r}2 \leq n + n^2 + 2n^3$$

$$\begin{aligned}\sum_{l=1}^{n}(1+\sum_{r=l}^{n}(1+\sum_{k=l}^{r}2)) &\geq \sum_{l=1}^{n}\sum_{r=l}^{n}2(r-l+1) \geq 2\sum_{l=1}^{n}\sum_{r=1}^{n-l+1}r \\ &\geq 2\sum_{l=1}^{n}\frac{(n-l+1)(n-l+2)}{2} \geq \sum_{l=1}^{n}(n-l+1)^2 \\ &= \sum_{l=1}^{n}l^2 = \frac{1}{6}n(n+1)(2n+1) \geq \frac{1}{3}n^3\end{aligned}$$

Aufgabe 3 (P) Exam Room Order

Ein Prüfungsraum besteht aus einer Reihe von n Sitzplätzen mit den Aufschriften von 0 bis $n-1$.

Jedem neu ankommenden Prüfling wird ein fixer Platz zugewiesen. Dieser Platz ist immer so zu wählen, dass der Abstand zu dem nächstgelegenen besetzten Platz maximal ist. Sollte es mehrere

[1] Weitere Informationen in *Setup Programmieraufgaben* in Moodle

solche Plätze geben, wird der mit der niedrigsten Nummer gewählt. Der erste zu besetzende Platz ist die Nummer 0.

In dieser Aufgabe implementieren Sie eine Funktion, die diese Reihenfolge für ein gegebenes n berechnet. Nutzen Sie gerne auch das Testing Framework[1].

a) Überlegen Sie eine geeignete Datenstruktur, um die besetzten Plätze zu speichern. Beachten Sie dabei insbesondere diese Fragen:

   - Was ist über Wertebereich/maximale Anzahl der Datenpunkte bekannt?
   - Welche Operationen soll die Datenstruktur zum Einfügen/Auslesen anbieten? Welche Informationen über die Daten sind wichtig?
   - Hilft es, die Daten sortiert abzuspeichern?
   - Gibt es bereits eine Implementierung in der C++ Standard Library?

b) Schreiben Sie eine Funktion `nextSeat`, die aus Ihrer gewählten Datenstruktur die Nummer des nächsten zu besetzenden Platzes berechnet.

c) Implementieren Sie eine Funktion `examRoomOrder`, welche die gesamte Reihenfolge der vergebenen Sitzplätze für ein gegebenes n berechnet. Benutzen Sie dafür ihre gewählte Datenstruktur und die Funktion `nextSeat`.

Beispiellösung:

a) Wir müssen abspeichern welche Plätze bereits besetzt sind. Die Werte sind die Zahlen 0 bis $n-1$, wovon höchstens n abgespeichert werden müssen. Die Datenstruktur soll das Einfügen von Zahlen unterstützen. Beim Auslesen interessiert uns nur welche Plätze besetzt sind. Insbesondere ist es nicht wichtig, in welcher Reihenfolge die Plätze bisher besetzt wurden. Wir benötigen keinen expliziten Zugriff auf den i-ten besetzten Platz.

   Wir wählen daher als Datenstruktur eine *Menge*. Die Sortierung hilft an dieser Stelle sehr viel: wenn die Plätze nach ihrer Nummer sortiert sind, sind die freien Plätze genau der *Abstand von aufeinanderfolgenden Zahlen* in unserer Menge! In der C++ Standard Library gibt es eine Implementierung von genau diesem Konzept, die sortierte Menge `set`.

b) Wie beobachtet, iterieren wir durch unser `set` und berechnen den Abstand von je zwei aufeinanderfolgenden Elementen. Für ein solches Paar a und b an besetzten Plätzen, sind also $b-a-1$ Plätze zwischen ihnen frei. Setzen wir nun jemanden genau in die Mitte, bleibt nach links und rechts ein Abstand von mindestens $\lfloor (b-a)/2 \rfloor$.

   Wichtig ist hier ein Sonderfall: gibt es ein Paar mit Abstand 4 und ein anderes Paar mit Abstand 5, garantieren beide einen Abstand von 2. Nach Angabe gilt hier die kleine Nummer, als kommt es darauf an, welches Paar die niedrigeren Zahlen hat. Das kann auch das Paar sein, das den geringeren Abstand von 4 hat.

c) Wir initialisieren unser `set` und eine Liste, in der wir die Reihenfolge speichern. Die ersten beiden Elemente sind (bei $n \geq 2$) nach Angabe immer 0 und $n-1$. Danach rufen wir wiederholt unsere Funktion aus b) auf und fügen den Index sowol in `set` als auch in die Ausgabe hinzu.

Sie finden die Lösungen zu Programmieraufgaben in `blatt1solution.cpp`.