Technische Universität Berlin **SS 2013**
Fakultät II, Institut für Mathematik
Sekretariat MA 5–2, Frau Kiefer

Prof. Dr. Martin Skutella
Torsten Gellert
Martin Groß
Dr. Max Klimm

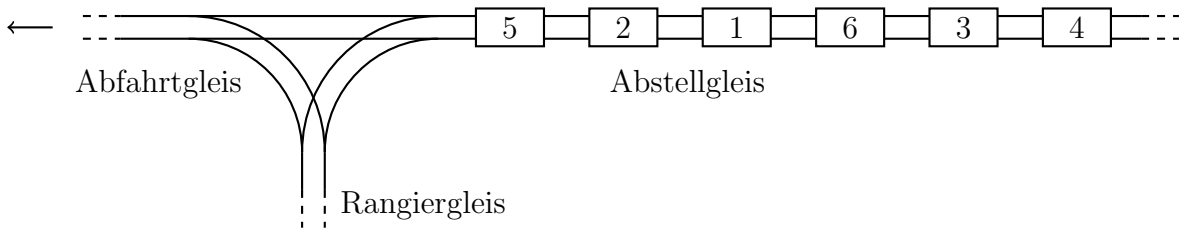
Aylin Acikel, Katharina Bütow, Christian Döblin, Alexander Hopp, Daniel Kuske, Olivia Röhrig, Robert Rudow, Daniel Schmand, Hendrik Schrezenmaier, Mona Setje-Eilers, Judith Simon, Sebastian Spies, Fabian Wegscheider, Jan Zur

2. Übungsblatt Computerorientierte Mathematik II

Abgabe: 23.04.2013 (bis 14:15 Uhr in MA 004)

1. Aufgabe $(5 + 2 + 2 + 1$ Punkte)

Auf einem Rangierbahnhof stehen auf einem Abstellgleis n Güterwaggons, die zu einem Zug der Länge n zusammengestellt werden sollen. Dabei sollen die Waggons in eine bestimmte Reihenfolge gebracht werden. Für die Rangierarbeiten steht ein Rangiergleis zur Verfügung, in das einige Waggons bei Bedarf zwischendurch abgestellt werden können. Der fertige Zug mit den Wagen in der Reihenfolge von 1 bis n soll am Ende auf dem Abfahrtgleis stehen, wo er dann den Bahnhof nach links verlässt. Die Situation sieht wie folgt aus:



Die Waggons sind mit Zahlen $1, \ldots, n$ benannt, die angeben, in welcher Reihenfolge sie am Ende auf dem Abfahrtgleis stehen sollen. Auf dem Abstellgleis können sie in beliebiger Reihenfolge stehen.

(a) Formuliert einen Algorithmus als Pseudocode der für alle Permutationen von $n \in \mathbb{N}$, Waggons unter Benutzung des Rangiergleises die richtige Reihenfolge auf dem Abfahrtgleis herstellt. Der Algorithmus erhält als Eingabe einen Stack, der die Wagen auf dem Abstellgleis enthält und soll als Ausgabe eine Queue mit den Wagen auf dem Abfahrtgleis liefern. Ihr dürft für den Algorithmus außerdem das Rangiergleis wie einen Stack benutzen. Begründet, warum eure gewählte Datenstruktur passend ist.

Hinweis: Für einen Stack sind nur die Funktionalitäten `boolean contains(element)`, `void push(element)`, `element pop()`, verfügbar, die Queue verfügt nur über die Methode `void add(element)`. Dabei sagt `contains(element)`, ob sich `element` in dem jeweiligen Stack befindet oder nicht; `push(element)` fügt am oberen Ende des jeweiligen Stacks `element` hinzu; `pop` entfernt es wieder und gibt `element` als Funktionswert zurück; `add` fügt `element` in die Queue `abfahrtgleis` ein.

In eurem Verfahren wird sicher eine bestimmte Teilaufgabe auftauchen, die wiederholt gelöst werden muss. Schreibt hierfür eine eigene Unterroutine, die ihr mit passenden Input-Parametern formal korrekt definiert und nach Bedarf aufruft.

(b) Welche Laufzeit hat euer Algorithmus? Geht davon aus, dass alle Stack- und Queue-Operationen außer `contains` in $\mathcal{O}(1)$ durchgeführt werden können, und dass `contains` auf einen n-elementigen Stack oder eine n-elementige Queue angewendet jeweils eine Laufzeit in $\mathcal{O}(n)$ benötigt. Begründet eure Antwort.

(c) Rechnet mit Hilfe eures Algorithmus das Rangierbeispiel in der Abbildung oben durch und zeichnet für jede Waggonverschiebung nach, welchen Inhalt die beiden Stacks und die Queue jeweils haben.

(d) Wie müssen die 6 Waggons im Beispiel auf dem Abstellgleis aufgestellt werden, so dass der Algorithmus möglichst lange läuft?

2. Aufgabe (3 + 7 Punkte)

Eine Firma benötigt von euch ein in-place Sortierverfahren, d.h. ein Sortierverfahren, bei dem keine Teile des Arrays kopiert werden. Dabei sollen möglichst wenig Schreibvorgänge auf das Array verursacht, da in den Produkten der Firma oftmals Speicher benutzt wird, der nur begrenzt oft beschrieben werden kann. Als einen Schreibvorgang betrachten wir dabei, wenn einem Array-Element etwas zugewiesen wird.

(a) Schlagt den besten Sortieralgorithmus vor, den ihr aus der Vorlesung kennt und gebt die Anzahl der Schreibvorgänge im Worst-Case genau an (mit Begründung).

(b) Entwickelt für das Problem im Pseudocode einen bezüglich der Anzahl der Schreibvorgänge optimalen Sortieralgorithmus. Beweist außerdem, dass es keinen Algorithmus mit weniger Schreibvorgängen geben kann. Analysiert auch die Best-Case und Worst-Case Laufzeit. Ihr könnt davon ausgehen, dass die Elemente des Arrays paarweise verschieden sind.