Projekt Fortgeschrittene Datenstrukturen im SS 2024

In diesem Dokument wird das Programmierprojekt für das Sommersemester 2024 beschreiben. Ziel des Projektes ist es, mehrere Datenstrukturen, die in der **Vorlesung Fortgeschrittene Datenstrukturen vorgestellt** wurden, zu implementieren. Neben der Implementierung ist eine Dokumentation, eine Evaluation und eine Abschlusspräsentation Bestandteil des Projektes.

Bitvektor mit Rank- und Select-Support

In diesem Projekt muss ein Bitvektor implementiert werden, der die folgenden Operationen unterstützt.

1. access,
2. rank und
3. select.

Einschränkungen an die Verarbeitung

Um einen fairen Rahmen für die Vergleichende Evaluation zu schaffen, gibt es einige Einschränkungen an die Implementierung:

- Die Anfragen müssen in der Reihenfolgen, in der Sie in der Eingabedatei vorkommen, beantwortet werden. Es ist nicht zulässig erst alle *access*-, dann alle *rank*- und anschließend alle *select*-Anfragen zu beantworten.
- Einzelne Anfragen können parallelisiert werden. Es ist jedoch *nicht* zulässig mehrere Anfragen parallel zu beantworten, d.h., die $i + 1$-te Anfrage darf erst beantwortet werden, wenn das Ergebnis der i-ten Anfrage vorliegt.

Ein- und Ausgabe

Das Programm muss per Kommandozeile steuerbar sein. Die Eingabe hierfür hat das folgende Format.

```
ads_programm eingabe_datei ausgabe_datei
```

Eingabe. Hier enthält die erste Zeile eine Zahl $n \in \mathbb{N}$ gefolgt von einer Zeile, die dem Bitvektor entspricht. Der Bitvektor wird als String über dem Alphabet $\{0, 1\}$ repräsentiert. Nach dem Bitvektor folgen n Zeilen mit Anfragen der Form `access` i, `rank b` i und `select b` i. Hierbei stellt i immer eine gültige Anfrageposition dar und es gilt `b` $\in \{0, 1\}$.

Beispiel:
6
001110110101010111111111
`access` 4
`rank 0` 10
`select 1` 14
`rank 1` 10
`select 0` 3
`access` 5

Ausgabe. Die Ausgabedatei soll für jede Eingabe n Zeilen haben, wobei in jeder Zeile die Antwort für die entsprechende Anfrage steht. Für die Beispieleingabe sieht die Ausgabe wie folgt aus:
1
4
20
6
5

0

Neben der Ausgabedatei soll noch eine Ausgabe in der Konsole der Form
`RESULT name=<first_last_name> time=<running_time_without_output_in_ms> space=<required_space_in_bits>`
erzeugt werden. Beispiel:
`RESULT name=florian_kurpicz time=512 space=1234` Die Ausgaben in der Konsole dürfen hinter dem Gleichheitszeichen keine Leerzeichen enthalten. Die einzelnen Werte sind per Leerzeichen getrennt.

Minimalanforderungen

Das Projekt darf in einer der folgenden Programmiersprache umgesetzt werden: C, C++, Rust, Python, GO oder Java. Wichtig ist aber, dass das Projekt auf einem Linux-System (Ubuntu 20.04.3 LTS) lauffähig ist! Dem Projekt sollte eine detaillierte Anleitung beiliegen, die beschreibt, was zum kompilieren/ausführen benötigt wird und wie genau das Programm ausgeführt werden kann. Das Programm muss das oben beschrieben Ein- und Ausgabeformat unterstützen.

Wichtig: **Es dürfen keine externen Bibliotheken verwendet werden**, die nicht von Florian Kurpicz genehmigt wurden. Bitte per Mail (`kurpicz@kit.edu`) nachfragen, bevor externe Bibliotheken eingebunden werden. Die Standardbibliothek der jeweiligen Programmiersprache kann allerdings ohne Fragen verwendet werden.

Dokumentation, Evaluation und Präsentation

Der Code muss so dokumentiert sein, dass dem Leser klar wird, was an welcher Stelle was genau passiert. Hierfür sollte darauf geachtet werden, dass die Dokumentation auch erklärt *warum* etwas gemacht wird und nicht nur *was* gemacht wird.

Die Evaluation ist Teil der Präsentation. Hier können unter anderem die Laufzeiten des eigenen Ansatzes für unterschiedliche Eingabe gezeigt werden. Ein Vergleich mit anderen Implementierungen ist auch möglich. (Hinweis: Das Ausgabeformat kann direkt zum Erstellen von Diagrammen genutzt werden, siehe `https://github.com/bingmann/sqlplot-tools/`.)

Die Ergebnisse sollen dann in einer **maximal** dreiseitigen Ausarbeitung beschrieben werden (Titel, Name, Matrikelnummer, etc. sowie Bilder zählen nicht zum Seitenlimit). In der Ausarbeitung soll neben der Evaluation der Implementierung darauf eingegangen werden, was implementiert wurde, wie es implementiert wurde und was eventuell besonders an der Implementierung ist. Für die Ausarbeitung ist das LiPICs-Template zu verwenden (`https://www.dagstuhl.de/en/publishing/series/details/LIPIcs`).

Wettbewerb

Des Weiteren gibt es noch einen kleinen Wettbewerb, an dem jedes eingereichte Projekt automatisch teilnimmt. Das Abschneiden im Wettbewerb hat keinen Einfluss auf die Note des Projektes! Bei dem Wettbewerb werden alle Laufzeiten der von mir durchgeführten Tests (gewichtet) addiert. Die geringste Laufzeit gewinnt. Die Gewichtung ist: 50 % Laufzeit und 50 % Platzbedarf.

Deadline

Das Projekt und die Ausarbeitung muss bis zum 12.07.2024 um 23:59 Uhr deutscher Zeit im Ilias hochgeladen werden. Der Upload wird noch freigeschaltet. Dort kann auch gerne ein Link zu einem Repository angegeben werden. Spätere Abgaben oder Abgaben per Mail können leider nicht berücksichtigt werden.