Projekt Fortgeschrittene Datenstrukturen im SS 2023

In diesem Dokument wird das Programmierprojekt für das Sommersemester 2023 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.

Predecessor-Datenstrukturen und Range-Minimum-Queries

In diesem Projekt müssen folgende Datenstrukturen implementiert werden, die in der Vorlesung vorgestellt worden sind:

1. Eine Predecessor-Datenstrukture (entweder y-Fast-Tries *oder* Elias-Fano) und
2. drei Datenstrukturen (naive, $O(n \log n)$ Platz und $O(n)$ Platz), die Range-Minimum-Queries besantworten.

Ein- und Ausgabe

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

```
ads_programm [pd|rmq] eingabe_datei ausgabe_datei
```

Der Parameter *pd* sorgt dafür, dass eine Predecessor-Datenstruktur für der Eingabedatei erstellt wird und der Parameter *rmq* sorgt dafür, dass eine Range-Minimum-Query-Datenstruktur entsprechend der Eingabedatei erstellt wird.

pd. Hier enthält die erste Zeile eine Zahl $n \in \mathbb{N}$ *in sortierter Reihenfolge* und wird gefolgt von n Zeilen bestehend aus Zahlen, die mit maximal 64 Bits repräsentiert werden können. Anschließend folgt eine beliebige Anzahl an Zeilen der Form:

- $i \in \mathbb{N}_0$, welche eine Predecessor-Anfrage darstellt oder

Beispiel:

```
4
0
1
4
7
5
4
3
```

Führt zu einer Ausgabedatei, welche die folgenden Zahlen zeilenweise enthält: 4, 4, 1.

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

rmq. Hier enthält die erste Zeile eine Zahl $n \in \mathbb{N}$ und wird gefolgt von n Zeilen bestehend aus Zahlen, die mit maximal 64 Bits repräsentiert werden können. Anschließend folgt eine beliebige Anzahl an Zeilen der Form:

- s, e mit $s, e \in \mathbb{N}_{\not\epsilon}$ und $s \leq$, welches einem Intervall für eine Range-Minimume-Query darstellt.

Beispiel:

```
4
1
0
3
7
```

```
0,3
1,2
2,3
```
Führt zu einer Ausgabedatei, welche die folgenden Zahlen zeilenweise enthält: 1, 1, 2.

Neben der Ausgabedatei soll noch eine Ausgabe in der Konsole der Form
`RESULT algo=rmq name<first_last_name> time=<running_time_without_output_in_ms> space=<required_space_in_bits>`
erzeugt werden. Beispiel:
`RESULT algo=rmq nameflorian_kurpicz time=512 space=1234`

Minimalanforderungen

Das Projekt darf in einer der folgenden Programmiersprache umgesetzt werden: C, C++, Rust oder Python. 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 **genau** zweiseitigen 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 17.07.2023 um 23:59 Uhr deutscher Zeit per Mail an `kurpicz@kit.edu` gesendet werden (gerne als Link zu einem Repository). Spätere Abgaben können leider nicht berücksichtigt werden.