Aufgabenblatt 7 (Teamaufgabe)

Einführung in die Programmierung 2
LVA-Nr. 185.A92
2019 S
TU Wien

Thema:
Präfixsuche

Aufwand:
ca. 16 Stunden pro Person

Ausgabe:
15. 3. 2019

Feedback (Deadline):
23. 5. 2019

Abgabe (Deadline):
31. 5. 2019, 13:00 Uhr
Lösungen hochladen

Einführung

In vielen Gebieten der Informatik (Rechtschreibkontrollen, prädiktive Texteingabe in Mobiltelefonen, Routing) ist es wichtig, Datenmengen effektiv nach Präfixen durchsuchen zu können.

Für dieses Aufgabenblatt werden Ihnen unter `www.complang.tuwien.ac.at/franz/teamaufgabe-data.zip` mehrere Dateien für die Simulation eines Sensorennetzwerks (`initial-collectors-N.csv`, `queries-N.txt`) zur Verfügung gestellt.

Ihre Aufgabe besteht darin, die Daten auszulesen und damit die unter "Szenario" beschriebene Simulation durchzuführen.

Szenario

Um die Methanemissionen durch auftauende Permafrostböden zu erheben, werden im Zuge eines internationalen Projekts mehrere Millionen Sensoren über dem Polarkreis abgeworfen.

Damit nicht jeder Sensor eine energieintensive Satellitenverbindung zum zuständigen Forschungszentrum aufrechterhalten muss, haben zu jedem Zeitpunkt nur einige Sensoren – die sogenannten Sammelpunkte – ein aktives Satellitenmodul. Grundsätzlich kann aber jeder Sensor als Sammelpunkt betrieben werden und jeder Sammelpunkt fungiert gleichzeitig auch als Sensor.

Wenn die Daten eines Sensors ausgelesen werden sollen, verbindet sich das Forschungszentrum über Satellit zum entsprechenden Sammelpunkt. Dieser baut eine stromsparende Funkverbindung zum angesprochenen Sensor auf und übermittelt dessen Daten wiederum über Satellit dem Forschungszentrum.

Die Zuteilung von Sensor zu Sammelpunkt basiert auf einer ID aus acht Kleinbuchstaben, die jedem Sensor eindeutig zugeteilt wird. Der Sammelpunkt eines Sensors s ist jener Sammelpunkt, dessen ID sich mit der ID von s das längste Präfix teilt. Falls sich mehrere Sammelpunkte ein Präfix gleicher Länge mit s teilen, wird der Sammelpunkt mit der lexikographisch kleinsten ID ausgewählt.

aaa	**aab**	aac	aad	aba	abb	abc	abd
aca	acb	acc	acd	ada	**adb**	adc	add
baa	bab	bac	bad	bba	bbb	bbc	bbd
bca	bcb	bcc	bcd	bda	bdb	bdc	bdd
caa	cab	cac	cad	cba	cbb	cbc	cbd
cca	ccb	ccc	ccd	cda	cdb	cdc	cdd
daa	dab	dac	dad	**dba**	dbb	dbc	dbd
dca	dcb	dcc	dcd	dda	ddb	ddc	**ddd**

Vereinfachte Darstellung mit kürzeren IDs. Sammelpunkte in schwarz und zugehörige Sensoren farbig hinterlegt. Die mit b beginnenden Sensoren sind aab zugeordnet, weil er der Sammelpunkt mit der lexikographisch kleinsten ID und mit dem längsten geteilten Präfix (dem leeren Präfix) ist.

Um die Topologie der Sensorennetzwerks dynamisch anzupassen und die Batterien der Sammelpunkte zu schonen, werden folgende Regeln eingehalten:

Ein illustriertes Beispiel findet sich im Anhang.

1. Wenn der Sammelpunkt n nach der Beantwortung einer Anfrage an den Sensor s 1000 Anfragen beantwortet hat, wird sein Satellitenmodul abgeschaltet. Das Satelitenmodul von s wird aktiviert.

2. Ein Adressbereich kann unter Umständen auch geteilt werden, indem ein neuer Sammelpunkt hinzugefügt wird. Zu diesem Zweck merkt sich jeder Sammelpunkt, wieviele Anfragen er beantwortet hat, seitdem ein neuer Sammelpunkt im ihm zugeteilten Adressbereich aktiviert wurde.

   Wenn der Sammelpunkt n nach der Beantwortung einer Anfrage an den Sensor s 250 Anfragen seit der letzten Teilung seines Adressbereichs beantwortet hat, wird das Satellitenmodul von s aktiviert. Der Teilungszähler von n wird wieder auf 0 gesetzt.

   Diese Regel wird nur angewandt, wenn nicht zuvor schon die Regel 1 angewandt wurde.

3. Damit sich nicht zu viele aktive Sensoren im Netzwerk befinden, werden nach jeweils 500000 Anfragen an das gesamte Netzwerk die Satellitenmodule all jener Sammelpunkte deaktiviert, die seit der letzten Anwendung dieser Regel keine Anfragen mehr beantwortet haben.

Implementieren Sie eine Datenstruktur, die die Sammelpunkte effizient verwalten kann, um die obigen Regeln umzusetzen. Lesen Sie die Dateien `initial-collectors-N.csv` ein, um die Sammelpunkte des Sensorennetzwerks zu initialisieren. Speisen sie danach die Anfragen aus `queries-N.txt` in das Netzwerk und simulieren Sie die obigen Regeln.

Geben Sie am Ende der Simulation folgende Eckdaten aus:

1. maximale Größe des Netzwerks während der Simulation
2. minimale Größe des Netzwerks während der Simulation
3. letzer Sammelpunkt der aufgrund von Regel 1 deaktiviert wurde
4. letzer Sammelpunkt der aufgrund von Regel 2 aktiviert wurde

Datenformat

Verwenden Sie die Klasse `Scanner` um die Daten aus der mitgelieferten Dateien einzulesen. Initialisieren Sie das `Scanner`-Objekt auf folgende Weise, um die Datei auf die gleiche Weise einlesen zu können wie Daten aus der Standardeingabe:

```
try(Scanner s = new Scanner(
        new File(System.getProperty("user.dir") +
            "/data/initial-collectors-0.csv"), "UTF-8")) {
    // Benutzen Sie das Scanner-Objekt s hier
} catch(FileNotFoundException e) {
    // initial-collectors-0.csv wurde nicht gefunden
    System.exit(1);
}
```

Tipp: Machen Sie sich mit der Methode `useDelimiter` in `Scanner` vertraut um das Einlesen zu vereinfachen.

In den Dateien `initial-collectors-N.csv` steht jeder Datensatz in einer eigenen Zeile. Innerhalb einer Zeile werden die verschiedenen Dateneinträge mit einem Semikolon (`;`) getrennt. Die Einträge sind in folgender Reihenfolge:

```
ID;Anfragen (gesamt);Anfragen seit letzter Teilung
```

Zum Beispiel:

```
obblujkm;468;218
xbwodgtq;687;187
```

Die Dateien `queries.txt` bestehen aus einer Liste von IDs, wobei eine ID in jeder Zeile steht. Das Einlesen mit einem `Scanner`-Objekt funktioniert ähnlich.

Datenstrukturen

Es gibt in der Literatur verschiedene Datenstrukturen, mit denen die Simulation implementiert werden kann. Zwei unserer Empfehlungen finden Sie im Folgenden kurz beschrieben. Lesen Sie die angegebene Literatur und erarbeiten Sie danach eine Lösung. Arbeiten Sie außerdem die allgemeinen und spezifischen Fragen aus, sodass Sie sie in der Übungseinheit 7 gegebenenfalls beantworten können.

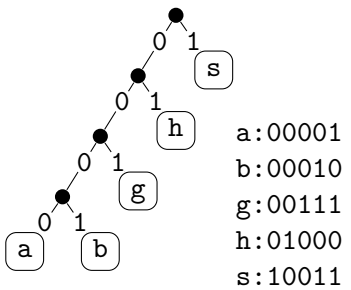
Wenn Sie eine andere Datenstruktur implementieren, werden Ihnen natürlich ebenfalls Fragen dazu gestellt.

Allgemeine Fragen

- Wieso haben Sie sich für die verwendete Datenstruktur entschieden?
- Wie verhält sich diese Datenstruktur bezüglich verschiedener Eingabedaten? (gleichmäßig verteilt, einem gewissen Muster folgend)
- Wie verhält sich der Implementierungsaufwand der Operationen auf der Datenstruktur?
- Wie verhält sich der Laufzeitaufwand der Operationen auf der Datenstruktur?

Binary Trie

Ein Binary Trie ist eine binäre Baumstruktur. Um ein Element in einem Binary Trie zu speichern, wird es zuerst in einen binären String fixer Länge konvertiert. Dieser binäre String wird als "Pfadangabe" innerhalb des Baumes benutzt: `0` entspricht links, `1` entspricht rechts.



Fragen zum Binary Trie

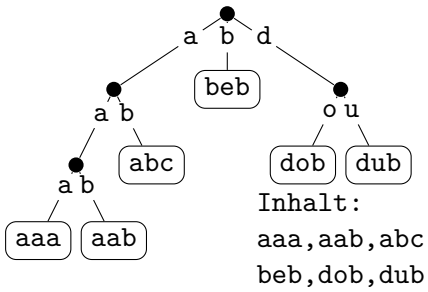
- Wie haben Sie die Konvertierung in den binären String implementiert? Worauf mussten Sie achten?
- Haben Sie eine Form von Pfadkomprimierung (siehe Literatur) implementiert?

Literatur zum Binary Trie

- Folien:
  `http://cw.fel.cvut.cz/old/_media/courses/a4m33pal/paska13trie.pdf`
- siehe außerdem Quellen für Indexed-Trie (Binary Tries können als Spezialfall betrachtet werden)

Indexed-Trie

Ein Indexed-Trie (oft auch einfach nur "Trie") hat Ähnlichkeiten zu einem Binary-Trie, mit dem Unterschied, dass nicht auf binärer Ebene gearbeitet wird, sondern auf direkt auf dem Alphabet. Ein Knoten innerhalb eines Indexed-Tries hat bis zu N Kinder, wobei N die Größe des Alphabets ist.



Fragen zum Indexed-Trie

- Wie speichern Sie die Kindknoten in Ihrer Datenstruktur? Was sind die vor und Nachteile Ihrers Ansatzes?
- Haben Sie eine Form von Pfadkomprimierung (siehe Literatur) implementiert?

Literatur zum Indexed-Trie

- Folien (ab 5.2):
  `https://algs4.cs.princeton.edu/lectures/52Tries.pdf`
- Skriptum (ab 9.4):
  `https://www.cs.cmu.edu/~avrim/451f11/recitations/rec0921.pdf`
- Video:
  `https://www.youtube.com/watch?v=TRg9DQFu0kU`

Vorgehensweise

Die wichtigsten Kenntnisse, auf denen die Teamaufgabe aufbaut, sollten Sie bereits erworben haben oder in den nächsten Vorlesungseinheiten erwerben (baumförmige Datenstrukturen). Es wird erwartet, dass Sie sich darauf aufbauend selbständig weitergehendes Wissen über den effizienten Umgang mit Präfixen erarbeiten. Dieses Thema wird in künftigen Vorlesungen nicht behandelt. **Fangen Sie so bald wie möglich an, um eventuell auftretende Fragen rechtzeitig klären zu können.**

nicht zuwarten

Manche in künftigen Vorlesungen eingeführte Techniken könnten Details der Implementierung verbessern. Diese dürfen natürlich zum Einsatz kommen, es gibt aber keine Verpflichtung dazu. Essenzielle Schwierigkeiten in der Aufgabenstellung werden davon wahrscheinlich nicht betroffen sein. Die Details zum Einlesen von Dateien werden erst nach der Abgabe behandelt.

siehe Punkt: Datenformat

Dies ist eine Teamaufgabe: Arbeiten sie an jedem Aspekt der Aufgabe

Teams wie bei Ad-hoc-Aufgaben; im Team lösen

gemeinsam; optimalerweise am gleichen Ort und zur gleichen Zeit. Arbeiten Sie in den selben Teams wie bei den Ad-Hoc-Aufgaben (sollte das nicht möglich sein, suchen Sie sich eine Partnerin oder einen Partner aus der selben Übungsgruppe).

Um die volle Punktzahl (33) zu erreichen, müssen Sie spätestens bis 23. 5. 2019 Feedback bei der Tutorin oder beim Tutor Ihrer Übungsgruppe einholen und bis zur Abgabe einarbeiten – je früher, desto besser. Wenn Sie kein Feedback einholen, können Sie höchstens die halbe Punktzahl erreichen.

Feedback einholen, Mail an Tutorin/Tutor

Falls es im Team Probleme gibt (wenn zum Beispiel ein Teil des Teams die Übung abbricht oder nicht mitarbeitet), wenden Sie sich bitte ebenfalls möglichst frühzeitig an die Tutorin oder den Tutor Ihrer Übungsgruppe.

Mail an Tutorin/Tutor bei Problemen jeder Art

Für inhaltliche Fragen steht Ihnen das Diskussionsforum in TUWEL und das Programmiercafé zur Verfügung.

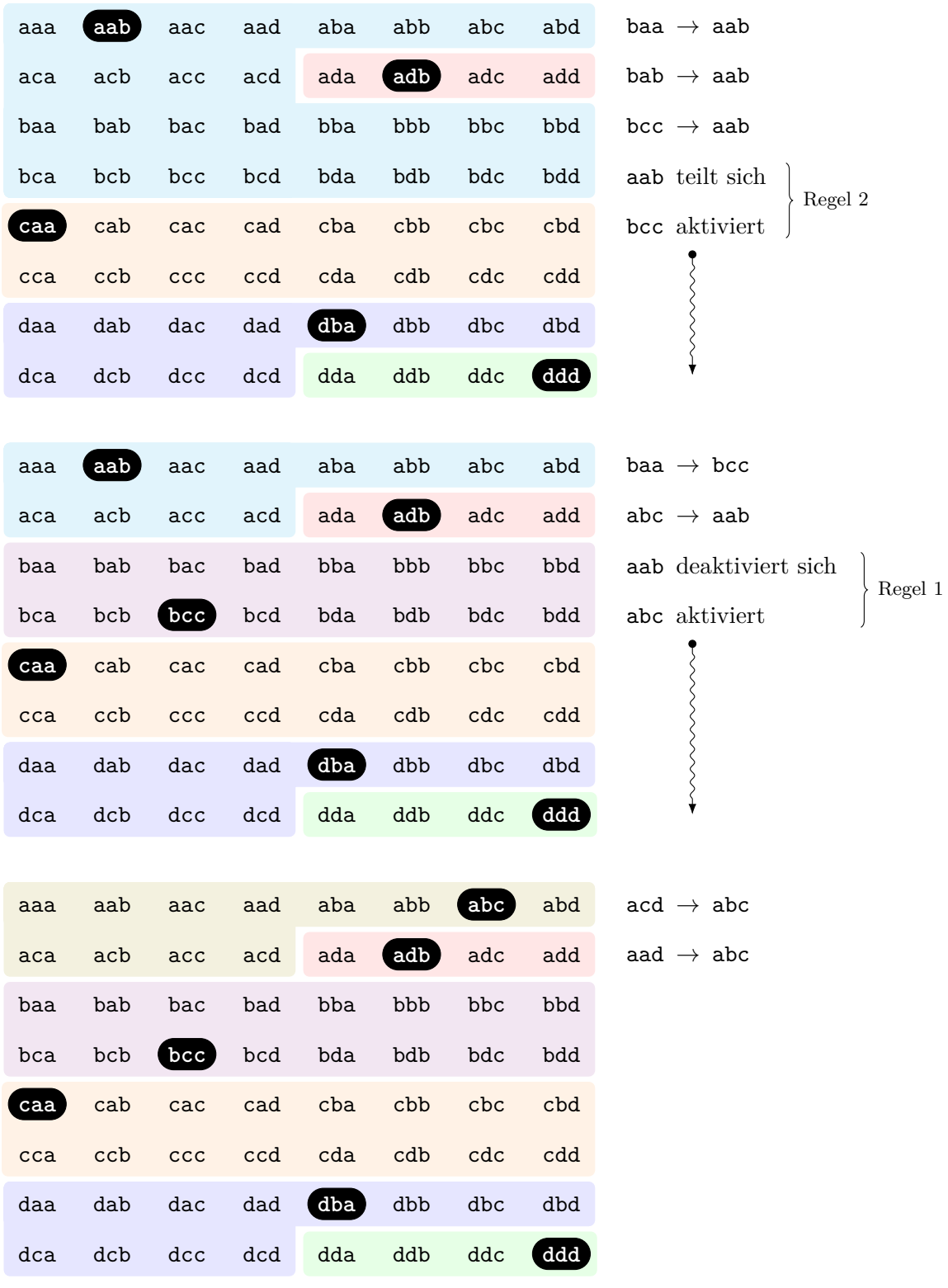
Beispielanfragen

```
> Simulation of 'initial-collectors-0.csv' and 'queries-0.txt'
minimal size of network: 5
maximal size of network: 6
last deactivated collector: aabxxxxx
last activated collector: bccxxxxx

> Simulation of 'initial-collectors-3.csv' and 'queries-3.txt'
minimal size of network: 10
maximal size of network: 4622
last deactivated collector: ushahkah
last activated collector: anahpamo

> Simulation of 'initial-collectors-5.csv' and 'queries-5.txt'
minimal size of network: 387303
maximal size of network: 401559
last deactivated collector: ifllvaag
last activated collector: iscolupd
```

Illustration



Netzwerk aus `initial-collectors-0.csv` mit Anfragen aus `queries-0.txt`