Karlsruher Institut für Technologie

Institut für Theoretische Informatik
T.T.-Prof. Dr. Thomas Bläsius
https://scale.iti.kit.edu

Übungsblatt 09

Algorithmen I – Sommersemester 2023

Abgabe im ILIAS bis 30.06.2023, 18:00 Uhr

Die Abgabe erfolgt als *eine* PDF-Datei über das Übungsmodul in der Gruppe deines Tutoriums im ILIAS.
Beachte bitte die Hinweise zum Bearbeiten auf der Webseite.

Aufgabe 1 - Instazahn (8 Punkte)

Während die NSA das Fernverkehrsnetz umgestaltet hat, ist den Nagern aufgefallen, dass die ursprünglich entsandten Biber gar nicht stumpfsinnig, sondern einfach nur abgelenkt sind. Wie Zombiber streifen sie ziellos am Bahnhof umher, die Augen ausschließlich auf ihre Smartphones gerichtet. Eine weitere Spezialeinheit, das *FBI* (furchtloses Biber Indiziensammlerteam), wurde eingeschaltet um herauszufinden, was die Biber beschäftigt.

Die Untersuchung hat ergeben, dass sich die Biber sozial vernetzt haben und ununterbrochen Bilder und Videos austauschen. Geschickt hat das FBI die zugehörige Online-Plattform gehackt und die erhaltenen Daten analysiert. Dabei fällt dem FBI auf: Biber Alina hat mit ihrem Freund Bob gleich viele gemeinsame Freunde wie mit ihrer Freundin Charlie. So etwas scheint aber auch noch für mehr Biber aufzutreten, das kann kein Zufall sein! Aus den Beobachtungen haben sie eine Hypothese formuliert, die sie die *Gemeinsame-Partner-Theorie (GPT)* nennen: *Jeder Biber, der mindestens zwei Freunde hat, hat zwei Freunde, die gleich viele gemeinsame Freunde mit ihm haben.*

Dr. Meta gratuliert dem FBI zu dieser augenscheinlich unnützen Entdeckung, kommt aber nicht umher sich zu fragen, wie die sonst so unorganisierten Biber das hinbekommen haben...

a) Beschreibe, wie GPT als Aussage über einen Graphen modelliert werden kann. (2 Punkte)

Hinweis: Was sind die Knoten? Zwischen welchen Knoten gibt es eine Kante? Ist der Graph gerichtet oder nicht? Brauchst du Kantengewichte?

Und jetzt wurdest auch du gehackt! Das FBI hat deine Formalisierung gestohlen und Dr. Meta vorgelegt. Nachdem das Problem so auf das Wesentliche reduziert wurde, wird ihm jetzt einiges klarer. Er bedankt sich lobend beim FBI und vermutet, dass das Beobachtete nicht etwa ein Biber-gemachtes Phänomen, sondern eine Eigenschaft eines jeden Graphen ist. Er beauftragt das FBI dies zu beweisen.

Deine Aufgabe ist es, einen Beweis zu finden, bevor sie es tun. Die Aufgabe erscheint zunächst viel zu schwer. Aber dann fällt dir ein: "Was die können, kann ich schon lange!". Du hackst das FBI... Unter `/Users/common/neighbors/` findest du genau eine Datei, aus der du folgenden Pseudocode extrahierst.

```
1:  COUNT(G = (V, E): Graph, v ∈ V) : ℕ
2:      for u ∈ N(v) do
3:          b := 0
4:          for w ∈ N(u) do
5:              if w ∈ N(v) then
6:                  b += 1
7:              end
8:          end
9:          PRINT(b)
10:     end
```

b) Beschreibe in Worten was b zählt. Gib den Wertebereich an Zahlen an, die der Algorithmus ausgeben kann. (2 Punkte)

c) Angenommen, es gibt einen Knoten $u \in N(v)$, bei dem $b = 0$ ausgegeben wird. Was bedeutet das für u und v? Gib den Wertebereich an Zahlen an, die der Algorithmus ausgeben kann, wenn diese Situation eintritt. (1 Punkt)

d) Angenommen, es gibt einen Knoten $u \in N(v)$, bei dem $b = |N(v)| - 1$ ausgeben wird. Was bedeutet das für u und v? Gib den Wertebereich an Zahlen an, die der Algorithmus ausgeben kann, wenn diese Situation eintritt. (1 Punkt)

e) Benutze die Erkenntnisse aus den vorherigen Teilaufgaben, um GPT mithilfe deiner Formalisierung aus Teilaufgabe a) zu beweisen. (2 Punkte)
Hinweis: Was fällt auf, wenn du die Kardinalitäten der Wertebereiche mit der Größe der Nachbarschaft von v vergleichst?

Aufgabe 2 - Baumschule (5 Punkte)

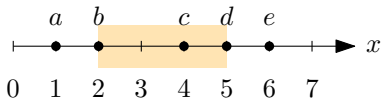
In dieser Aufgabe wollen wir uns mit der Konstruktion von $(2, 3)$-Bäumen beschäftigen.

a) Beschreibe, wie in Linearzeit aus einer gegebenen sortierten Folge ein $(2, 3)$-Baum konstruiert werden kann. Begründe, warum dein Algorithmus das geforderte Laufzeitverhalten hat. (3 Punkte)

b) Beschreibe, wie in Linearzeit zwei gegebene $(2,3)$-Bäume zu einem $(2,3)$-Baum zusammengefügt werden können. Begründe, warum dein Algorithmus das geforderte Laufzeitverhalten hat. (2 Punkte)

Aufgabe 3 - Bereichsanfragen (8 Punkte)

Im folgenden beschäftigen wir uns mit Punkten und Intervallen. Wir starten in Dimension 1. Hier ist ein Punkt durch eine ganzzahlige x-Koordinate definiert. Nun brauchen wir eine Datenstruktur, die Punkte so verwaltet, dass *Range Queries* unterstützt werden. Eine Range Query wird durch ein Intervall $[q_1, q_2] \subseteq \mathbb{Z}$ repräsentiert, wobei alle Punkte gesucht sind, für die $x \in [q_1, q_2]$ gilt.



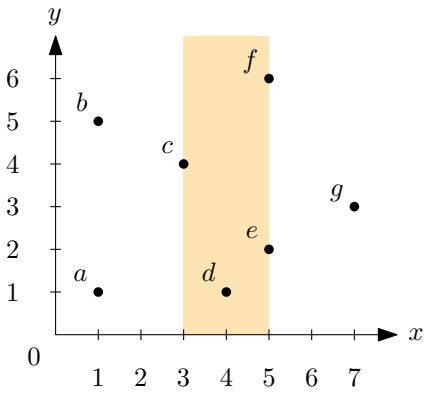
Eine Punktmenge $\{a, b, c, d, e\}$ mit Range Query $Q = [2, 5]$. Die gesuchten Punkte sind $\{b, c, d\}$.

Wir wollen nun eine Datenstruktur entwerfen, die die folgenden Operationen unterstützt:

- `insert(`p`)`: Fügt einen Punkt p in Zeit $O(\log(n))$ ein, falls er noch nicht in der Datenstruktur enthalten ist.
- `remove(`p`)`: Entfernt einen Punkt p in Zeit $O(\log(n))$, falls er in der Datenstruktur enthalten ist.
- `get(`q_1`, `q_2`)`: Gibt in $O(\log(n) + k)$ alle Punkte aus, nach der die Range Query $[q_1, q_2]$ fragt, wobei k die Anzahl dieser Punkte ist.

a) Beschreibe, wie mithilfe von (a, b)-Bäumen die gewünschten Operationen umgesetzt werden können. Begründe, warum die Operationen das geforderte Laufzeitverhalten haben. (3 Punkte)

Nun erweitern wir unseren Horizont um eine zusätzliche Dimension! Woah! Ein Punkt $p = (x, y) \in \mathbb{Z}^2$ wird nun durch eine x- und eine y-Koordinate definiert. Unsere Range Queries interessieren sich jedoch immer noch für nur eine Dimension. Inbesondere fragt eine Range Query nach allen Punkten, deren x-Koordinate in $[q_1, q_2]$ enthalten ist.



Eine Punktmenge mit Range Query $Q = [3, 5]$. Die gesuchten Punkte sind $\{c, d, e, f\}$.

Achtung: Anders als im 1D-Fall kann es jetzt, wie im obigen Beispiel, mehrere Punkte mit der gleichen x-Koordinate geben.
Wir möchten nun die Datenstruktur aus Teilaufgabe a) so anpassen, dass sie trotzdem noch funktioniert. Dazu modifizieren wir zunächst den (a, b)-Baum aus Teilaufgabe a), indem in Blättern jetzt Listen von Punkten gespeichert werden können.

b) Beschreibe, wie `insert`, `remove` und `get` mit der neuen Datenstruktur im 2D-Fall umgesetzt werden können und begründe, dass `get` die gewünschte Laufzeit hat. (2 Punkte)
Hinweis: Über die Laufzeit von `insert` und `remove` musst du dir in dieser Teilaufgabe keine Gedanken machen.

c) Jetzt musst du dir Gedanken über die Laufzeit von `insert` und `remove` machen. Gebe für beliebig große n jeweils einen Zustand der Datenstruktur aus Teilaufgabe b) (als 2D-Punktemenge mit n Punkten) und eine `insert`- oder `remove`-Operation an, sodass diese Operation die gewünschte Laufzeit überschreitet. (1 Punkt)

d) Du darfst die Datenstruktur nun beliebig anpassen. Beschreibe eine Datenstruktur, die neben `get` auch `insert` und `remove` in 2D in gewünschter Laufzeit umsetzen kann. (2 Punkte)