Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Thomas Seidl
Janina Sontheim, Florian Richter

München, 27.06.2019

Algorithmen und Datenstrukturen

SS 2019

Übungsblatt Global 5: Hashing und Graphen

Aufgabe Global 5-1 *Knobelei: Staatsempfang*

Auf einem Staatsempfang wird ein Spion vermutet. Niemand kennt ihn, doch er soll ausfindig gemacht werden, bevor er Staatsgeheimnisse ausspionieren und entkommen kann. Die anwesenden $2n+1$ Diplomaten sind untereinander gut bekannt. Wählt man n beliebige Diplomaten aus, dann findet man stets einen Diplomaten außerhalb dieser Menge, der mit all diesen n Diplomaten bekannt ist. Bekanntheit ist dabei immer eine wechselseitige Relation. Eine Befragung jedes einzelnen zur Identifizierung wäre aber zu auffällig. „Kein Problem! Wir fragen denjenigen, der alle anderen Diplomaten kennt.“. Gibt es diese Person immer?

Lösungsvorschlag:

Erstmal: Eine Clique C ist eine Menge von Knoten, die vollständig vernetzt sind. Wir betrachten die Menge der Diplomaten als Knoten und eine Kante beschreibt die Bekanntheit zweier Diplomaten miteinander. Falls es nun eine Clique der Größe $n+1$ gäbe, hätten wir die Antwort. Denn die restliche Menge besteht aus n Diplomaten und wir wissen, dass dann einer der Clique mit allen übrigen n bekannt sein muss. Damit wäre er tatsächlich mit allen Knoten bekannt. Wir zeigen daher, dass es auch so eine Clique der Größe $n+1$ gibt. Dazu wählen wir zuerst eine Menge von n Diplomaten aus. Es gibt in der übrigen Menge einen Knoten, der mit allen n gewählten Knoten verbunden ist. Wir wählen eine Paarung aus (egal, welche) und fügen zu dieser 2-Clique $n-2$ weitere beliebige Punkte hinzu. Wieder greift die Eigenschaft, dass in der verbleibenden Menge ein Knoten mit allen n gewählten Knoten verbunden ist. Daher auch mit unserer 2-Clique. Wir fügen ihn hinzu und erhalten eine 3-Clique. Dies wiederholen wir, bis wir eine $n+1$-Clique erhalten. Danach können wir die Clique nicht vergrößern, da die restliche Menge weniger als $n+1$ Knoten enthält und die Eigenschaft nicht mehr gilt. Das ist aber nicht schlimm, denn da wir eine $n+1$-Clique gefunden haben, muss es mit dem Argument zuvor auch einen Knoten geben, der mit allen anderen Knoten verbunden ist.

Aufgabe Global 5-2 *Hashing Funktionsgüte*

Diskutieren Sie, inwiefern sich die folgenden Funktionen zum Hashing eignen. Begründen Sie Ihre Antwort.

- $f : \mathbb{N} \to \{0, 1, 2, \ldots, 10\}, x \mapsto \lfloor x/10 \rfloor$
- $g : \mathbb{N} \to \{0, 1, 2, \ldots, 49\}, x \mapsto 2^x \mod 50$
- $h : \mathbb{N} \to \{0, 1, 2, \ldots, 10\}, x \mapsto x \mod 10$
- $i : \mathbb{N} \to \{0, 1, 2, \ldots, 49\}, x \mapsto x \mod 50$

Lösungsvorschlag:

f: einfach zu berechnen, surjektiv, Verteilung auf Bildbereich gleichverteilt, ähnliche Schlüssel nah beieinander. Achtung: Eigentlich sollte die Funktion von $\{0, \ldots, 100\}$ abbilden, nicht von den natürlichen Zahlen. Es können natürlich nicht alle Werte aus $\mathbb{N}$ abgebildet werden, zum Beispiel ist $f(110) = 11$. Die Funktion ist daher nicht wohldefiniert. Eine andere Reparaturmöglichkeit wäre, ein $\mod 11$ zu benutzen.

g: einfach zu berechnen, nicht surjektiv (keine ungeraden Zahlen größer 1 getroffen), annähernd gleichverteilt (1,2 nur einmal getroffen), ähnliche Schlüssel weit verteilt.

h: einfach zu berechnen, nicht surjektiv (10 nie getroffen), gleichverteilt, Streuung gut (schlechter als g).

i: einfach zu berechnen, surjektiv, gleichverteilt, Streuung gut.

Aufgabe Global 5-3 *Hashing Diskussion*

Diskutieren Sie Unterschiede zwischen offenem und geschlossenem Hashing bzgl. Kollisionen, Überlauf und Laufzeiten. Wie kann man das Konzept von Doppelhashing (mehrere Hashfunktionen) auf offenes Hashing anwenden?

Lösungsvorschlag:

Kollisionen: Kollisionen müssen beim geschlossenen Hashing innerhalb der Tabelle aufgelöst werden. Dann Sondieren oder verwerfen. Offenes Hashing: Kollisionen erzeugen Overhead in angehängten Datenstrukturen.

Überlauf: Kein Problem beim offenen Hashing. Geschlossenes Hashing benötigt Rehashing oder Verwerfen alter Einträge.

Laufzeiten: Geschlossenes Hashing hat stehts konstanten Aufwand, da Tabelle konstant groß. Offenes Hashing hängt von der angehängten Datenstruktur ab. Falls Listen O(n) zum Suchen wegen linearer Suche.

Offenes Hashing kann weitere Hashtabellen angehängt nutzen. So können weitere Hashfunktionen genutzt werden, um baumartige Blockstruktur (ähnlich B-Baum) zu generieren.

Zweite Möglichkeit Cuckoo-Hashing: Hashe mit zwei Funktionen. An einer Stelle muss Objekt zu finden sein. Wenn Kollision an beiden Stellen entstehen, dann hashe ein Objekt an seiner anderen möglichen Position ein. Rekursiv auflösen und irgendwann abbrechen.