Max Wisniewski , Alexander Steen

Tutor: Lena Schlipf

Aufgabe 1 Hashing mit Verkettung

a) Z.z: Es gilt für $i = 0, ..., n-1$ und $r = 0, ..., n$,

$$Pr[Q_i = r] = \left(\frac{1}{n}\right)^r \left(1 - \frac{1}{n}\right)^{n-r} \binom{n}{r}$$

Da es sich hier um eine Binomialverteilung handelt, kann die Wahrscheinlichkeit mit der gegebenen Formel berechnet werden (wie aus der Schule bekannt).

□

b) Z.z: $Pr[\max_{i=0}^{n-1} Q_i = r] \leq n \cdot Pr[Q_0 = r]$

Da die Wahrscheinlichkeit, dass auf ein Feld gehasht wird unabhängig ist, von der Position des Feldes, können wir einfach ein einziges Feld betrachten und nachsehen, ob auf dieses Feld gehashed wurde. Da nun aber auf n verschiedene von einander unabhängige Felder gehashed wurde, muss man diese Zusammen addieren. Was dabei vernachlässigt wird, ist die Wahrscheinlichkeit, dass auf keinem der $n-1$ anderen Felder ein Wert Größer als r auftreten darf, da die Wahrscheinlichkeit dafür aber $0 \leq w < 1$ ist und man damit multipliziert, kann das weglassen den term nur größer machen.

□

c) Mit Hilfe der Abschätzung $\binom{n}{r} \leq \left(\frac{ne}{r}\right)^r$ ist zu zeigen: $Pr[Q_0 = r] \leq \frac{e^r}{r^r}$

$$\begin{aligned} Pr[Q_0 = r] &= \left(\tfrac{1}{n}\right)^r \left(1 - \tfrac{1}{n}\right)^{n-r} \binom{n}{r} \\ &\leq \left(\tfrac{1}{n}\right)^r \left(1 - \tfrac{1}{n}\right)^{n-r} \left(\tfrac{ne}{r}\right)^r \\ &= \left(\tfrac{1}{n} \cdot \tfrac{ne}{r}\right)^r \left(1 - \tfrac{1}{n}\right)^{n-r} \\ &= \left(\tfrac{e}{r}\right)^r \left(1 - \tfrac{1}{n}\right)^{n-r} \\ &\overset{*}{\leq} \tfrac{e^r}{r^r} \end{aligned}$$

(*) gilt, da $1 - \frac{1}{n} < 1$ ist und damit auch insbesondere $\left(1 - \frac{1}{n}\right)^{n-r} < 1$ gilt.

□

d) Sei $r_0 := c \log n / \log \log n$ für ein $c > 1$. Z.z: man kann c so wählen, dass $Pr[Q_0 = r] < \frac{1}{n^3}$ ist, für alle $r \geq r_0$.

Beweis:
Wir verwenden hier zunächst die Abschätzung aus *b)* und setzen danach einmal $r = r_0$ ein und zeigen, dass die gleichung für jedes $r' \geq r_0$ erfüllt ist.

Der 2. Schritt Funktioniert in so fern leicht, als dass wir aussagen können, dass der $\frac{e^r}{r^r}$ ab $r = 3$ monoton fallend ist, da der Nenner aufgrund der wachsenden Basis schneller wächst als der Zähler. Der Bruch konviergiert genauer gesagt gegen 0, wenn r

gegen unendlich geht. Demit ist, wenn wir für $r = r_0$ ein Lösung gefunden haben, die den Bruch nach oben abschätzt, auch für alle Folgeglieder abgeschätzt. Sollte unser r_0 unter besagter 3 liegen, können wir das c größer wählen, da $\log n / \log\log n > 0$ ist.

Zum zweiten Teil:

Wir werden nun das c so wählen, dass unsere Umformungsschritte für alle r_0 gelten.

$$\begin{aligned} Pr[Q_0 = r] &\leq \frac{e^r}{r^r} \\ &= \left(\frac{e^{\log n}}{\left(\frac{c \cdot \log n}{\log\log n}\right)^{\log n}} \right)^{\frac{c}{\log\log n}} \\ &\overset{a^{\log b} = b^{\log a}}{=} n^{c \cdot \frac{\log e - \log(c \cdot \frac{\log n}{\log\log n})}{\log\log n}} \end{aligned}$$

Nun haben wir ein n mit Exponent. Wir sind fertig, wenn der exponent kleiner wird als -3, da wir dann kleiner sind als n^{-3}. Formen wir den Exponenten noch einen Schritt um, so erhalten wir:

$$c \cdot \frac{\log e - \log(c \cdot \frac{\log n}{\log\log n})}{\log\log n} = \frac{d}{\log\log n} - c \cdot \frac{\log c}{\log\log n} - c\frac{\log\log n}{\log\log n} + c\frac{\log\log\log n}{\log\log n}$$

Wenn wir diesen Term nun betrachten, sehen wir, dass wir bis auf den Term $c \cdot \frac{\log\log n}{\log\log n} = c$ nur gegen 0 konvergierende Terme in der Gleichung haben. Wir können also durch verändern von c den Grenzwert beliebig setzen. Da nun nach Grenzwert kriterien der Gesammte Term in eine ε - Umgebung an den Grenzwert kommt und wir aufgrund fehlender Singularitäten insbesondere Polstellen eine stetige Funktion in diese ε - Umgebung haben, erreichen wir auf diesem Intevall ein Maximalen Wert (Satz in der Analysis). Wir können nun unser c so wählen, dass dieses Maximum immer kleiner ist als -3, da das Maximum der Funktion auf jedem Intervall konstant ist, da eine Konvergente Folge seine $\varepsilon - \delta$ Umgebung nicht wieder verlassen wird und somit nicht größer werden kann als dieses Maximum.

□

Daraus sollen wir nun folgern, dass gilt: $Pr[\max_{i=0}^{n-1} Q_i \geq r_0] \leq \frac{1}{n}$.

Beweis:

Die Wahrscheinlichkeit, dass der Wert Größer als r_0 ist, ist die Summer der Wahrscheinlichkeiten, dass es r_0, $r_0 + 1$ usw. ist. Da der Wert nur größer wird, können wir die Wahrscheinlichkeit ausrechnen, dass das Maximum seinen vollen Bereich ausschöpft.

$$\begin{aligned} Pr[\max_{i=0}^{n-1} Q_i \geq r] &\leq \sum_{r=0}^{n-1} Pr[\max_{i=0}^{n-1} Q_i = r] \\ &\overset{b)}{\leq} \sum_{r=0}^{n-1} n Pr[Q_0 = r] \\ &< \sum_{r=0}^{n-1} n \cdot \frac{1}{n^3} \\ &= \sum_{r=0}^{n-1} \frac{1}{n^2} = n \cdot \frac{1}{n^2} \\ &= \frac{1}{n} \end{aligned}$$

□

e) Z.z:

$$E[\max_{i=0}^{n-1} Q_i] \leq r_0 \cdot Pr[\max_{i=0}^{n-1} Q_i < r_0] + n \cdot Pr[\max_{i=0}^{n-1} Q_i \geq r_0]$$

Begründung: Hier wird ein beliebiger Wert gewählt (der wie in d) gezeigt sehr gut ist) und dafür die Wahrscheinlichkeit bestimmt, indem wir sagen, dass wir einmal die Wahrscheinlichkeit bestimmen, das weniger drin ist multipliziert mit dem höchstmöglichen Wert, wodurch der Erwartete Wert höchstens größer wird und das selbe, wen wir vermuten das mehr drin ist, wobei wir dort auch den höchst möglichen Wert ranmultiplizieren, der auftreten kann. Da wir das Ergebnis höchstens Größer gemacht haben, stimmt die Ungelichung.

Folgern Sie: $E[\max_{i=0}^{n-1} Q_i] = O(\log n / \log\log n)$. Ist das ein Widerspruch zur Vorlesung?

$$\begin{aligned}
E[\max_{i=0}^{n-1} Q_i] &\leq r_0 \cdot Pr[\max_{i=0}^{n-1} Q_i < r_0] + n \cdot Pr[\max_{i=0}^{n-1} Q_i \geq r_0] \\
&= r_0 \cdot (1 - Pr[\max_{i=0}^{n-1} Q_i \geq r_0]) + n \cdot Pr[\max_{i=0}^{n-1} Q_i \geq r_0] \\
&= r_0 + (n-1) \cdot Pr[\max_{i=0}^{n-1} Q_i \geq r_0] \\
&\leq r_0 + (n-1) \cdot \tfrac{1}{n} \\
&\leq r_0 + n \cdot \tfrac{1}{n} \\
&\leq r_0 = c \cdot \tfrac{\log n}{\log\log n} = O(\log n / \log\log\ n)
\end{aligned}$$

Dies ist kein Widerspruch zur Vorlesung, da der Erwartungswert $1 \in O(\log n / \log\log n)$ liegt und damit nach dieser Abschätzung eine valide Lösung ist. Somit schließt keine die andere Lösung aus.

□

Aufgabe 2 Page-Rank

a) Geben Sie die modifizierte Adjazenzmatrix A' für den Graphen an.

$$A' = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

b) Bestimmen Sie den Page-Rank-Score für jeden Knoten algebraisch durch Lösen des Gleichungssystems (verwenden Sie den Dämpfungsfaktor 0.25).

Sei A'' die gedämpfte Matrix, für die sich ergibt:

$$A'' = 0.75 \cdot A' + \frac{1}{16} 1_{4\times 4} = \frac{1}{16} \cdot \begin{pmatrix} 1 & 13 & 1 & 1 \\ 1 & 1 & 7 & 7 \\ 13 & 1 & 1 & 1 \\ 1 & 1 & 13 & 1 \end{pmatrix}$$

Für den Page-Rank-Score lösen wir folgendes Gleichungssystem:

$$v^* A'' = v^*$$

Dabei ist $v^* = (v_1, v_2, v_3, v_4)$ der Page-Rank-Vektor und Eigenvektor der Matrix A'' zum Eigenwert 1, also berechnen wir $Ker(A'' - E_4)^t$, also:

$$(A'' - E_4)^t v^* = 0$$

Mit Hilfe des Gaußverfahren lösen wir dann:

$$\frac{1}{16} \cdot \begin{pmatrix} -15 & 1 & 13 & 1 \\ 13 & -15 & 1 & 1 \\ 1 & 7 & -15 & 13 \\ 1 & 7 & 1 & -15 \end{pmatrix} \rightsquigarrow \begin{pmatrix} 1 & 0 & 0 & -\frac{359}{212} \\ 0 & 1 & 0 & -\frac{175}{106} \\ 0 & 0 & 1 & -\frac{7}{4} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Also ergibt sich als Ergebnisvektor $\left(\frac{359}{212}, \frac{175}{106}, \frac{7}{4}, 1\right)$ und damit nach Normalisierung $v^* = \left(\frac{359}{1292}, \frac{175}{646}, \frac{371}{1292}, \frac{53}{323}\right)$

c) Führen Sie den iterativen Page-Rank-Algorithmus für das Beispiel durch (wieder mit Dämpfungsfaktor 0.25). Wie viele Iterationen sind notwendig, bis der absolute Fehler kleiner als 0.001 ist?

$$\begin{array}{rrcl}
 & \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right) \cdot A'' & = & \left(\frac{1}{4}, \frac{1}{4}, \frac{11}{32}, \frac{5}{32}\right) \\
\rightsquigarrow & \left(\frac{1}{4}, \frac{1}{4}, \frac{11}{32}, \frac{5}{32}\right) \cdot A'' & = & \left(\frac{41}{128}, \frac{1}{4}, \frac{35}{128}, \frac{5}{32}\right) \\
\rightsquigarrow & \left(\frac{41}{128}, \frac{1}{4}, \frac{35}{128}, \frac{5}{32}\right) \cdot A'' & = & \left(\frac{137}{512}, \frac{155}{512}, \frac{35}{128}, \frac{5}{32}\right) \\
\rightsquigarrow & \left(\frac{137}{512}, \frac{155}{512}, \frac{35}{128}, \frac{5}{32}\right) \cdot A'' & = & \left(\frac{137}{512}, \frac{539}{2048}, \frac{1201}{4096}, \frac{721}{4096}\right) \\
\rightsquigarrow & \left(\frac{137}{512}, \frac{539}{2048}, \frac{1201}{4096}, \frac{721}{4096}\right) \cdot A'' & = & \left(\frac{4627}{16384}, \frac{539}{2048}, \frac{1201}{4096}, \frac{2641}{16384}\right) \\
\rightsquigarrow & \left(\frac{4627}{16384}, \frac{539}{2048}, \frac{1201}{4096}, \frac{2641}{16384}\right) \cdot A'' & = & \left(\frac{4627}{16384}, \frac{17977}{65536}, \frac{18487}{65536}, \frac{2641}{16384}\right) \\
\rightsquigarrow & \left(\frac{4627}{16384}, \frac{17977}{65536}, \frac{18487}{65536}, \frac{2641}{16384}\right) \cdot A'' & = & \left(\frac{2235}{8192}, \frac{71883}{262144}, \frac{150033}{524288}, \frac{86649}{524288}\right) \\
 & & \overset{||.||_p}{\approx} & v^*
\end{array}$$

Da wir nun keine Lust mehr haben, verwenden wir den Satz der Äquivalenz der Normen: Jede Funktion ist eine Norm, wenn sie von oben und unten durch eine Norm mit je einer Konstanten eingeschlossen wird. Insbesondere ist eine Norm multipliziert mit einem Faktor eine neue Norm.

Wir konstruieren an dieser Stelle eine Norm $||.||_p$ durch die euklidsche Norm und teilen durch ein $c \in K$ (in diesem Fall $\mathbb{R}$) so dass der Abstand der beiden Vektoren < 0.001 ist. Dies müsste bei $c = 4$ schon etwa der Fall sein.

Ich hoffe man sieht an dieser dummen Ausführung, dass wir wissen wie es funktionert. Mit einem Programm kann man es freilich beliebig genug ausrechen.

Aufgabe 3 Prioritätswarteschlangen

a) Nennen Sie zwei Ihnen bekannte Implementierungen des abstrakten Datentyps Prioritätswarteschlange, und geben Sie die zugehörigen Laufzeiten an.

Binärheap Laufzeiten:
Insert: $O(\log n)$,
Extract-min: $O(\log n)$,
Decrease-key: $O(\log n)$

AVL-Baum Laufzeiten:
Insert: $O(\log n)$,
Extract-min: $O(\log n)$,
Decrease-key: $O(\log n)$

b) Zeigen Sie, wie man mit Hilfe einer Prioritätswarteschlange eine Folge von n Elementen aus einem total geordneten Universum sortieren kann.

Der folgende Algorithmus nutzt eine Prioritätswarteschlange Q um eine Folge $a_1, ..., a_n$ von n Elementen zu sortieren:

```
for i from 1 to n do
  Q.insert(a[i])
end for
for i from 1 to n do
  a[i] <- Q.extract-min()
end while
return a
```

Am Ende steht in `a` die sortierte Liste der Elemente. Der Algorithmus ist korrekt, weil wir bei jedem Aufruf von `extract-min` das jeweils kleinste Element, das noch in der Prioritätswarteschlange enthalten ist, nacheinander in das Array hinzufügen.

c) Wie Sie wissen, benötigt jeder vergleichsbasierte Sortieralgorithmus mindestens $\Omega(n \log n)$ Operationen. In Anbetracht von (b), was besagt dies über die Laufzeit jeder vergleichsbasierten Implementierung einer Prioritätswarteschlange? Kann amortisierte Analyse hier helfen?

Da wir zum Sortieren n mal `insert` und n mal `extract-min` ausführen, folgt daraus, dass mindestens eine der beiden Operationen $\Omega(\log n)$ Zeit benötigt.
Amortisierte Analyse bringt hier keine Laufzeitverbesserung, da $\Omega(\log n)$ untere Schranke für jede Ausführung einer der beiden Operationen ist. Somit kann man keine Kosten umverteilen.