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

□

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

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

(*) 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:

$$\exists c > 1 \forall r \geq r_0 : Pr[\max_{i=0}^{n-1} Q_i \geq r_0] \leq 1/n$$

□

e) Z.z:

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

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

□

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?

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

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.