Karlsruher Institut für Technologie

Stefan Walzer
ITI Prof. Sanders
Wintersemester 2023/2024

Übungsblatt 15 – Peeling

Randomisierte Algorithmik

Aufgabe 1 – Deterministisches Schälen in Linearzeit

(a) Der Algorithmus constructByPeeling ist nichtdeterministisch (in der Wahl von i in der While-Schleife). Zeige, dass dennoch für zwei mögliche Ausführungen von constructByPeeling stets die gleiche Menge von Schlüsseln unplatziert bleibt. (Insbesondere beeinflusst der Nichtdeterminismus nicht Erfolg/Misserfolg.)

(b) Verfeinere den Pseudocode so, dass ein deterministischer Algorithmus herauskommt, der erkennbar Laufzeit $O(n)$ hat. Nutze geeignete Hlifsdatenstrukturen deiner Wahl.

Lösung 1

(a) Sei $X = (x_1, \ldots, x_k)$ die Folge von Schlüsseln, die eine vollständige Ausführung von constructByPeeling platziert (in dieser Reihenfolge) bevor die While-Schleife verlassen wird. Sei $Y = (y_1, \ldots, y_\ell)$ eine andere mögliche Folge. Es genügt zu zeigen, dass jedes Element, das in X vorkommt, auch in Y vorkommt: Aus Symmetriegründen folgt dann, dass X und Y die selben Elemente (möglicherweise in anderer Reihenfolge) enthalten.

Wir machen einen Beweis durch Widerspruch und nehmen an, $j \in [k]$ ist der kleinste Index sodass x_j nicht in Y vorkommt. Von der X-Ausführung wird x_j an einem Index i_j platziert, der für keinen der Schlüssel aus $S \setminus \{x_1, \ldots, x_{j-1}\}$ in Frage kommt. Also kommt der Index i_j auch für keinen Schlüssel aus $S \setminus \{y_1, \ldots, y_\ell\}$ in Frage, denn $\{y_1, \ldots, y_\ell\} \supseteq \{x_1, \ldots, x_{j-1}\}$ nach Wahl von j. Nach Annahme ist x_j am Ende der Y-Ausführung noch in S. Dann wäre aber i_j ein Index der nur für x_j und keinen anderen verbleibenden Schlüssel in Frage kommt. Damit wäre ein weiterer Durchlauf der While-Schleife möglich. Damit beschreibt Y keine vollständige Ausführung von constructByPeeling. Widerspruch.

(b) Wir legen dem Algorithmus den bipartiten Cuckoo-Graphen wie in der Vorlesung definiert zugrunde. Wir benutzen eine Warteschlange Q. Invariante ist, dass Q alle Tabellenknoten von Grad 1 enthält (und eventuell weitere Tabellenknoten). Es spielt keine Rolle ob Q eine FIFO/LIFO oder andere Art von Warteschlange ist.

Algorithm constructByPeeling-Linear(S, h_1, h_2, h_3)**:**

```
T ← [⊥, ..., ⊥] // leere Tabelle
G ← (S, [m], {(x, h_i(x)) | x ∈ S, i ∈ [3]}) // cuckoo graph
Q ← ∅ // leere Warteschlange
for i ∈ [m] do
    if deg_G(i) = 1 then
        Q.push(i)
while not Q.isEmpty do
    i ← Q.pop
    if deg_G(i) = 1 then
        x ← eindeutiger Nachbar von i
        T[i] ← x
        S ← S \ {x}
        for j ∈ {h_1(x), h_2(x), h_3(x)} do
            lösche die Kante (x, j) aus G
            if deg_G(j) = 1 then
                Q.push(j)
if S = ∅ then
    return T
else
    return NOT-PEELABLE
```

Nach Annahme lässt sich G am Anfang in Zeit $O(m)$ konstruieren und jede weitere Einzeloperation lässt sich in $O(1)$ ausführen. Weil jeder Knoten nur einmal zu einem Grad 1 Knoten werden kann (weil wir nur Kanten löschen), wird für jedes $i \in [m]$ nur einmal Q.push(i) ausgeführt. Damit ist die Anzahl der While-Schleifendurchläufe auch höchstens m (weil da jeweils ein Element aus Q entfernt wird). Insgesamt ergibt sich $O(m)$.

Bemerkung. Anstatt eine Adjazenzlistendarstellung von G zu wählen, genügt im vorliegenden Fall auch eine sparsamere Darstellung. Für jede Tabellenposition speichern wir zwei Dinge: Ihren Grad (den sie in G hätte) sowie die *Summe* der mit ihr verbundenen Elemente aus S (dies nimmt an, dass das Universum D eine Gruppe ist, zum Beispiel $D = \mathbb{Z}_{2^{64}}$). Man kann an dieser Darstellung erkennen ob der Grad eines Tabellenknotens 1 ist und in diesem Fall den eindeutigen Nachbarn extrahieren. Wenn der Grad größer 1 ist kann man durch Subtrahieren einen gegebenen Nachbarn herauslöschen. Diese Datenstruktur kommt ohne Zeiger aus.

Aufgabe 2 – Peeling mit 2 Hashfunktionen

Angenommen wir verwenden den Schälalgorithmus constructByPeeling in einem Setting mit nur zwei Hashfunktionen h_1 und h_2. Wir gehen vereinfachend davon aus, dass $h_1(x) \neq h_2(x)$

für alle $x \in D$ gilt und unter dieser Einschränkung die Paare $(h_1(x), h_2(x))$ uniform zufällig und für verschiedene $x \in D$ unabhängig sind.[1] Zeige:

(a) Es werden alle Schlüssel platziert genau dann wenn folgender Graph kreisfrei ist:

$$G = ([m], \{\{h_0(x), h_1(x)\} \mid x \in S\})$$

Beachte: G ist als Multigraph zu verstehen.

(b) Sei $\alpha > 0$ eine Konstante und $n = \lfloor \alpha m \rfloor$. Zeige, dass es (für $m \to \infty$) mit Wahrscheinlichkeit $\Omega(1)$ einen Kreis gibt.
Hinweis: Es genügt nach Kreisen der Länge 2 (also Doppelkanten) zu fanden.
Siehe `https://en.wikipedia.org/wiki/Birthdayproblem#Arbitrarynumberofdays`.

Es gibt damit keinen Schälbarkeitsschwellwert, wie es ihn für $k \geq 3$ gibt, denn für kein $\alpha = \Theta(1)$ hat constructByPeeling mit hoher Wahrscheinlichkeit Erfolg.

Lösung 2

(a) Wir können constructByPeeling als Algorithmus auf G auffassen, der wiederholt Knoten von Grad 1 sucht. Ist v ein solcher Knoten, dessen inzidente Kante von einem Schlüssel x herkommt, dann wird x in v platziert und die Kante zu x wird gelöscht. Es werden also alle Schlüssel platziert, wenn das sukzessive Löschen von Kanten mit einem Endpunkt von Grad 1 mit dem leeren Graphen endet.

Starten wir mit einem Wald G, dann ist G in jedem Schritt ein Wald (löschen von Kanten aus einem Wald liefert wieder einen Wald). Solange der Wald nicht leer ist, also mindestens ein Baum mit mindestens einer Kante übrig ist, hat dieser ein Blatt von Grad 1 an dem weitergearbeitet werden kann. Ergo: Wenn G ein Wald ist, werden alle Schlüssel platziert.

Hat G dagegen mindestens einen Kreis, der von Schlüsseln $x_1, \ldots, x_k$ herkommt, dann kann keiner dieser Schlüssel der erste sein, der von constructByPeeling platziert wird, denn beide möglichen Positionen des Schlüsseln kommen auch für einen zyklisch benachbarten Schlüssel in Frage. Also werden nicht alle Schlüssel platziert.

(b) Wir können uns vorstellen, dass wir beim Bestimmen der Hashwerte aller Schlüssel n mal mit Zurücklegen aus einer Urne mit $\binom{m}{2}$ Bällen ziehen, die den möglichen Kanten

[1] Um das sicherzustellen können wir zum Beispiel $h_2(x) := (h_1(x) + h_{\text{diff}}(x)) \bmod m$ definieren für ein $h_{\text{diff}} : D \to [m-1]$.

entsprechen. Dass wir paarweise verschiedene Bälle ziehen hat Wahrscheinlichkeit:

$$\begin{aligned}
&1 \cdot \left(1 - \frac{1}{\binom{m}{2}}\right) \cdot \left(1 - \frac{2}{\binom{m}{2}}\right) \cdot \ldots \cdot \left(1 - \frac{n-1}{\binom{m}{2}}\right) \\
&\leq 1 \cdot \left(1 - \frac{1}{m^2/2}\right) \cdot \left(1 - \frac{2}{m^2/2}\right) \cdot \ldots \cdot \left(1 - \frac{n-1}{m^2/2}\right) \\
&\leq \exp\left(-\frac{1}{m^2/2}\right) \cdot \exp\left(-\frac{2}{m^2/2}\right) \cdot \ldots \cdot \exp\left(-\frac{n-1}{m^2/2}\right) \\
&= \exp\left(-\frac{n(n-1)/2}{m^2/2}\right) = \exp\left(-\frac{n(n-1)}{m^2}\right) \overset{m\to\infty}{\longrightarrow} \exp(-\alpha^2).
\end{aligned}$$

Im ersten Schritt nutzen wir dass $1 + x \leq e^x$ für $x \in \mathbb{R}$ gilt. Später nutzen wir $1 + 2 + \cdots + n - 1 = n(n-1)/2$. Die Wahrscheinlichkeit, dass wir nicht verschiedene Bälle ziehen und entsprechend eine Doppelkante in G vorkommt, ist damit im Grenzwert mindestens $1 - e^{-\alpha^2}$ und damit $\Omega(1)$. Insbesondere gibt es mit Wahrscheinlichkeit $\Omega(1)$ einen Kreis.

Aufgabe 3 – Der Schälalgorithmus bleibt nicht erst spät stecken[2]

Für eine Schlüsselmenge $S \subseteq D$ der Größe $n = |S|$ und Hashfunktionen $h_1, h_2, h_3 \sim \mathcal{U}([m]^D)$ betrachten wir den Cuckoo Graphen wie in der Vorlesung:

$$G = G_{S,h_1,h_2,h_3} = (S, [m], \{(x, h_i(x)) \mid x \in S, i \in [3]\})$$

Wir nehmen lediglich $\alpha = \frac{n}{m} \leq 1$ an. Sei $S' \subseteq S$ die Menge der Schlüssel, die vom Schälalgorithmus nicht entfernt werden können (es gilt $|S'| \in \{0, \ldots, n\}$). Wir wollen zeigen, dass $\Pr[|S'| \in \{1, \ldots, \delta m\}] = O(1/m)$ ist für eine Konstante $\delta > 0$ (die später gewählt wird).

Intution: Entweder gilt $S' = \emptyset$ oder $|S'|$ ist $\Omega(m)$; alles dazwischen ist unwahrscheinlich.

(a) Beobachte: $|S'| = 1$ ist nicht möglich.

(b) Zeige: $|N(S')| \leq \frac{3}{2}|S'|$. Hierbei ist $N(S')$ die Menge aller Nachbarn von S' in G.

(c) Zeige, dass es eine Konstante C gibt, sodass für $s \in \{2, \ldots, n\}$ folgendes gilt:

$$p_s := \Pr[\exists X \subseteq S, |X| = s : \exists Y \subseteq [m], |Y| = \lfloor \tfrac{3}{2}s \rfloor : N(X) \subseteq Y] \leq \left(C \cdot \frac{s}{m}\right)^{s/2}.$$

Hinweis: Einfach brutal Union-Bound über alle Möglichkeiten von X und Y verwenden. Nützlich ist außerdem die Abschätzung $\binom{n}{k} \leq (\frac{ne}{k})^k$ für Binomialkoeffizienten. Ignoriere die Gaussklammern, das machen alle so.

(d) Zeige (i) $p_2 + p_3 + p_4 + p_5 = O(1/m)$, (ii) $\sum_{s=6}^{\sqrt{m}} p_s = O(1/m)$, (iii) $\sum_{s=\sqrt{m}}^{m/(2C)} p_s = O(1/m)$.

(e) Wähle $\delta = 1/2C$ und zeige $\Pr[|S'| \in \{1, \ldots, \delta m\}] \leq \sum_{s=2}^{\delta m} p_s = O(1/m)$.

[2]Diese Aufgabe ist etwas aufwändig. Sie ist vor allem auf dem Blatt weil sonst der Beweis der Vorlesung unvollständig wäre.

Lösung 3

(a) Ein einzelner Schlüssel kann sich nicht selbst im Weg stehen.

(b) Je nachdem, ob man Doppelkanten zählt oder nicht, gehen von S' genau $3|S'|$ oder höchstens $3|S'|$ Kanten aus. Weil jeder Behälter in $N(S')$ von zwei verschiedenen Schlüsseln aus S' getroffen werden muss (sonst könnte man weiterschälen) hat jeder Knoten in $N(S')$ mindestens zwei Nachbarn in S'. Für die Anzahl e von Kanten zwischen S' und $N(S')$ gilt also $3|S'| \geq e \geq 2|N(S')|$. Umstellen liefert die Behauptung.

(c) Es gibt $\binom{n}{s}$ Möglichkeiten um X zu wählen und $\binom{m}{(3/2)s}$ Möglichkeiten um Y zu wählen. Für beliebiges $x \in X$ gilt $\Pr[N(\{x\}) \subseteq Y] = (|Y|/m)^3$, denn alle drei Hashwerte müssen (unabhängig voneinander) in der Menge Y landen. Wir erhalten somit folgendes (auf dem Weg fassen wir einige Konstanten durch ein $C = \Theta(1)$ zusammen):

$$\begin{aligned} p_s &= \Pr[\exists X \subseteq S, |X| = s : \exists Y \subseteq [m], |Y| = \lfloor \tfrac{3}{2}s \rfloor : N(X) \subseteq Y] \\ &\leq \binom{n}{s}\binom{m}{(3/2)s}\left(\frac{(3/2)\cdot s}{m}\right)^{3s} \leq \left(\frac{ne}{s}\right)^{s}\left(\frac{me}{(3/2)s}\right)^{(3/2)s}\left(\frac{(3/2)\cdot s}{m}\right)^{3s} \\ &\leq \left(\frac{n^2e^2}{s^2}\right)^{s/2}\left(\frac{2^3m^3e^3}{3^3s^3}\right)^{s/2}\left(\frac{3^6s^6}{2^6m^6}\right)^{s/2} \leq \left(\frac{Cm^2m^3s^6}{s^2s^3m^6}\right)^{s/2} = \left(\frac{Cs}{m}\right)^{s/2}. \end{aligned}$$

(d) (i) Es gilt $p_2 = O(m^{-1})$, $p_3 = O(m^{-3/2})$, $p_4 = O(m^{-2})$, $p_4 = O(m^{-5/2})$. Passt also.

(ii) Jeder der Summanden in $\sum_{s=6}^{\sqrt{m}} \left(\frac{Cs}{m}\right)^{s/2}$ ist höchstens $\left(\frac{C\sqrt{m}}{m}\right)^{6/2} = O(m^{-3/2})$. Da es $O(m^{1/2})$ Summanden gibt, summieren diese sich zu höchstens $O(m^{-1})$.

(iii) Jeder der Summanden in $\sum_{s=\sqrt{m}}^{m/(2C)} \left(\frac{Cs}{m}\right)^{s/2}$ ist höchstens $\left(\frac{1}{2}\right)^{\sqrt{m}/2} = O(m^{-2})$ (*sehr* grob abgeschätzt). Da es $O(m)$ Summanden gibt, summieren diese sich zu höchstens $O(m^{-1})$.

(e) Wähle $\delta = \frac{1}{2C}$. Angenommen $|S'| = s$ für ein $s \in \{1, \ldots, \delta m\}$. Sei $X = S'$ und $Y' := N(S')$. Nach (b) gilt $|Y'| \leq \frac{3}{2}s$ also gibt es eine Menge $Y \supseteq Y'$ mit $|Y| = \lfloor \frac{3}{2}s \rfloor$ und $N(S') \subseteq Y$. Die Mengen X und Y erfüllen das Ereignis dessen Wahrscheinlichkeit wir mit p_s beschränkt haben. Zusammenfassend können wir also schlussfolgern:

$$\Pr[|S'| \in \{1, \ldots, \delta m\}] \leq \underbrace{\Pr[|S'| = 1]}_{0} + \sum_{s=2}^{\delta m} \Pr[|S'| = s] \leq \sum_{s=2}^{m/(2C)} p_s = O(1/m).$$