Karlsruher Institut für Technologie

Stefan Walzer
ITI Prof. Sanders
Wintersemester 2023/2024

Übungsblatt 11
Game Theory & Yao's Principle

Randomisierte Algorithmik

Aufgabe 1 – Untere Schranken für randomisiertes Sortieren

Sei $n \in \mathbb{N}$ und **Inputs** die Menge aller Permutation von $\{1, \dots, n\}$. Sei **Algos** die Menge aller vergleichsbasierten *deterministischen* Sortieralgorithmen.

(i) Nenne ein $A \in$ **Algos** (ohne Beweis) mit $\max_{I \in \textbf{Inputs}} C(A, I) = O(n \log n)$.

(ii) Fingerübung: Für $n = 3$ gibt es einen randomisierten Algorithmus $\mathcal{A}$ der jeden deterministischen Algorithmus in folgendem Sinne schlägt:

$$\min_{A \in \textbf{Algos}} \max_{I \in \textbf{Inputs}} C(A, I) = 3 > 2 + \tfrac{2}{3} = \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)].$$

Unser Plan im Folgenden ist zu zeigen, dass dieser Vorteil in O-Notation verschwindet.

(iii) Zeige, dass es keine „schwierige Eingabe" gibt, dass nämlich gilt:

$$\max_{I \in \textbf{Inputs}} \min_{A \in \textbf{Algos}} C(A, I) = n - 1.$$

Um gleich die bestmöglichen Kosten für eine Eingabe*verteilung* in den Blick nehmen, benötigen wir etwas Vorbeitung:

(iv) Zeige, dass in einem Binärbaum mit k Blättern die durchschnittliche Tiefe eines Blattes mindestens $\lfloor \log_2(k) \rfloor$ beträgt.
Hinweis: Zeige dafür, dass die durchschnittliche Blatttiefe bei balancierten Bäumen minimal ist, genauergesagt, dass sich jeder Baum, in dem sich die Blatttiefen um mindestens 2 unterscheiden, so umbauen lässt, dass die durchschnittliche Blatttiefe sinkt, die Anzahl der Blätter aber gleichgeblieben ist.

(v) Folgere nun mithilfe von Yaos Prinzip, dass gilt:

$$\min_{\mathcal{A} \text{ Vert. auf } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)] = \Omega(n \log n).$$

Lösung 1

(i) Mergesort.

(ii) Wir zeigen zunächst die linke Gleichung. Sei $A \in$ **Algos** beliebig. Bei Eingabe $I = (x, y, z)$ können wir aus Symmetriegründen (zwischen den Elementen) annehmen, dass A zunächst x und y vergleicht und $x < y$ herausfindet. Aus Symmetriegründen (zwischen $<$ und $>$) können wir annehmen, dass er als nächstes x und z vergleicht. Nun könnte es sein, dass $x < y$ und $x < z$ gilt, dann ist die Reihenfolge von y und z noch ungeklärt und ein weiterer Vergleich ist nötig. Also gibt es einen Eingabe I sodass $C(A, I) = 3$. Damit gilt $\max_{I \in \textbf{Inputs}} C(A, I) = 3$. Weil A beliebig war gilt:

$$\min_{A \in \textbf{Algos}} \max_{I \in \textbf{Inputs}} C(A, I) = 3.$$

Ein randomisierter Algorithmus kann "raten" welches Element das mittlere Element ist und die beiden anderen Elementen mit diesem Vergleichen. Wenn der Algorithmus richtig geraten hat ist nach zwei Vergleichen die vollständige Reihenfolge geklärt. Anderfalls ist der dritte Vergleich nötig. Dieses $\mathcal{A}$ erfüllt $\mathbb{E}_{A \sim \mathcal{A}}[C(A, I)] = \frac{1}{3} \cdot 2 + \frac{2}{3} \cdot 3 = 2 + \frac{2}{3}$ für jede Eingabe I wie gewünscht.

(iii) Die Reihenfolge von „min" und „max" erlaubt es uns, den Algorithmus auf die Eingabe I zuzuschneiden. Sei dafür π die Permutation[1], die I sortiert. Wir definieren A in Abhängigkeit von I so:

```
Algorithm A(I'):
    vertausche Elemente von I' gemäß π
    fail ← FALSE
    for i = 1 to n − 1 do // prüfe ob sortiert
        if I'[i] > I'[i + 1] then
            fail ← TRUE
            break
    if fail then
        MergeSort(I')
```

Wir müssen uns drei Dinge überlegen:

Der Algorithmus ist korrekt, also $A \in$ Algos. Das ist klar: Der Algorithmus ordnet seine Eingabe I' zunächst gemäß π um und überprüft, ob I' dadurch sortiert ist. Falls ja, tut er nichts mehr, sonst benutzt er MergeSort.

Der Algorithmus ist schnell für I. Wenn die Eingabe I' mit I übereinstimmt (der Eingabe, auf die A zugeschnitten ist), dann ist I' nach den Vertauschungen gemäß π sortiert. Dann sind nur die $n - 1$ Vergleiche der Schleife nötig, um dies zu verifizieren.

[1] Mit „Permutation" meint man manchmal Anordnungen einer Menge (wie am Anfang der Aufgabe), manchmal einen Operator, der eine gegebene Folge umordnet (so ist es jetzt gemeint).

Es geht nicht noch schneller. Angenommen es wurden weniger als $n-1$ Vergleiche angestellt. Wir betrachten den Graphen $G = ([n], E)$ der eine Kanten zwischen i und j enthält falls das $i$te Elemente der Eingabe mit dem $j$ten Element der Eingabe verglichen wurde. Wegen $|E| < n-1$ ist G unzusammenhängend. Damit ist die relative Reihenfolge dieser Zusammenhangskomponenten nicht geklärt. (Formal: Wenn eine Eingabe mit der Bauminformation konsistent ist, dann auch eine weitere Eingabe. Also "weiß" der Algorithmus noch nicht was die Eingabe war, was aber nötig ist um die Ausgabe zu konstruieren.)

(iv) Sei T ein Binärbaum mit k Blättern und minimaler durchschnittlicher Blatttiefe. Dann ist T ein voller Binärbaum, das heißt jeder Knoten hat 0 oder 2 Kinder (Grund: Innere Knoten mit nur einem Kind kann man herausbasteln und die durchschnittliche Blatttiefe reduzieren). Seien L und L' die größte bzw. kleinste Tiefe eines Blattes in T. Wir zeigen nun $L' \geq L-1$. Angenommen das ist nicht so, dann gilt $L' \leq L-2$. Sei ℓ_1 ein Blatt auf Tiefe L und ℓ_2 sein Geschwisterknoten (existiert weil T voll ist), der ebenfalls ein Blatt sein muss (nach Wahl von L als maximale Tiefe). Sei ℓ' ein Blatt auf Tiefe L'. Wir hängen nun ℓ_1 und ℓ_2 um und machen sie zu Kindern von ℓ'. Dadurch entsteht wieder ein Binärbaum. Die Summe der Blatttiefen verändert sich aus folgenden Gründen:

- Die Blätter ℓ_1 und ℓ_2 haben nun Tiefe $L'+1$ anstatt L.
- Der Knoten p, der Vater von ℓ_1 und ℓ_2 war, ist nun ein Blatt der Tiefe $L-1$.
- Der Knoten ℓ' auf Tiefe L' ist jetzt kein Blatt mehr.

Die Summe der Blatttiefen verändert sich damit um

$$2((L'+1)-L) + (L-1) - L' = L' - L + 1 \leq L - 2 - L + 1 = -1$$

ist also kleiner geworden. Damit ist die durchschnittliche Blatttiefe kleiner geworden, was ein Widerspruch zur Wahl von T ist.

Also gilt $L' \geq L-1$, d.h. die Tiefen der Blätter unterscheiden sich um höchstens 1. *Wäre* die durchschnittliche Blatttiefe von T weniger als $\lfloor \log_2(k) \rfloor$, dann hätte mindestens ein Blatt eine Tiefe von weniger als $\lfloor \log_2(k) \rfloor$ und alle Blätter hätten Tiefe höchstens $\lfloor \log_2(k) \rfloor$. Damit gibt es weniger als $2^{\lfloor \log_2(k) \rfloor} \leq k$ Knoten – ein weiterer Widerspruch.

Also hat T durchschnittliche Blatttiefe mindestens $\lfloor \log_2(k) \rfloor$. Weil T nach Wahl minimale durchschnittliche Blatttiefe hat, gilt dies auch für alle anderen Binärbäume.

(v) Betrachten wir den Entscheidungsbaum T_A der ein beliebiges $A \in \mathbf{Algos}$ beschreibt. Ohne Einschränkung schauen wir nur solche Algorithmen an, die einen Vergleich „$x < y$" nur dann anstellen, wenn sowohl „true" als auch „false" als Ergebnis noch möglich sind, also jeder Berechnungspfad auch von mindestens einer Eingabe beschritten wird. Das bedeutet, dass T_A ein voller Binärbaum ist. Für jede der $n!$ möglichen Eingaben ist eine andere Umordnung der Elemente nötig, also muss jede Eingabe zu einem anderen Blatt in A führen. Also hat A exakt $n!$ Blätter. Nach (iii) ist die durchschnittliche Blatttiefe von A mindestens $\lfloor \log_2(n!) \rfloor$. Wenn wir nun die Gleichverteilung $\mathcal{I}$ auf den Eingaben anschauen, dann ist die erwarte Anzahl von Vergleichen, die A für $I \sim \mathcal{I}$ anstellt exakt

die durchschnittliche Blatttiefe von T_A, also ebenfalls mindestens $\lfloor \log_2(n!) \rfloor$. Aus $n! \geq (n/2)^{n/2}$ folgt $\lfloor \log_2(n!) \rfloor \geq \lfloor (n/2) \log_2(n/2) \rfloor = \Omega(n \log n)$. Damit ist nach dem Satz von Yao

$$C \overset{\text{Yao}}{\geq} \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}}[C(A, I)] \geq \lfloor \log_2(n!) \rfloor = \Omega(n \log n).$$

Aufgabe 2 – Yaos Prinzip ohne Schnick-Schnack(-Schnuck)

Beweise Yaos Prinzip ohne auf spieltheoretische Sätze zurückzugreifen (kein Satz von Nash, Loomis, etc). Beweise also, dass im Setting der Vorlesung für eine beliebige Verteilung $\mathcal{A}_0$ auf **Algos** und eine beliebige Verteilung $\mathcal{I}_0$ auf **Inputs** gilt:

$$\max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}_0}[C(A, I)] \geq \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_0}[C(A, I)].$$

Hinweis: Der spieltheoretische Vorbau der Vorlesung ist hier nicht erforderlich, weil wir nicht zeigen möchten, dass „=“ möglich ist.

Lösung 2

Es gilt:

$$\max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}_0}[C(A, I)] \geq \mathbb{E}_{A \sim \mathcal{A}_0, I \sim \mathcal{I}_0}[C(A, I)] \geq \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_0}[C(A, I)].$$

In Worten: In der Mitte werden sowohl A als auch I *zufällig* gewählt. Links wird I nicht zufällig, sondern mit dem Ziel einer Maximierung gewählt, weshalb das Ergebnis größer wird. Rechts wird hingegen A nicht zufällig sondern mit dem Ziel einer Minimierung gewählt, weshalb das Ergebnis kleiner wird. Das löst bereits die Aufgabe.

Bemerkung. Formal könnte man den ersten Schritt (und den zweiten analog) noch kleinteiliger aufschreiben, indem man folgende zwei abstrakte Einsichten verwendet:

1. $\max_{x \in X} f(x) \geq \mathbb{E}_{x \sim \mathcal{X}}[f(x)]$.
2. $\mathbb{E}_{x \sim \mathcal{X}}[\mathbb{E}_{y \sim \mathcal{Y}}[g(x, y)]] = \mathbb{E}_{x \sim \mathcal{X}, y \sim \mathcal{Y}}[g(x, y)]$.

wobei f und g Funktionen sind, $\mathcal{X}$ eine Verteilung auf einer Menge X und $\mathcal{Y}$ eine Verteilung auf einer Menge Y.

Die erste Gleichung erlaubt ein Maximum durch einen Erwartungswert abzuschätzen, die zweite Gleichung erlaubt ein zweistufiges Zufallsexperiment und einen „erwarteten Erwartungswert“ in ein einstufiges Zufallsexperiment zu überführen. Nutzt man diese Gleichungen für $X = \textbf{Inputs}$, $Y = \textbf{Algos}$, $\mathcal{X} = \mathcal{I}_0$, $\mathcal{Y} = \mathcal{A}_0$, $f(I) = \mathbb{E}_{A \sim \mathcal{A}_0}[C(A, I)]$ und $g(A, I) = C(A, I)$ so ergibt sich die erste Ungleichung von oben.

Aufgabe 3 – Empfehlung: Simulating the Evolution of Teamwork

Der Youtube-Kanal *Primer* beschäftigt sich mit evolutionärer Spieletheorie. In folgendem Video geht es unter anderem darum, alle möglichen 2-Spieler-Spiele mit zwei reinen Strategien danach zu klassifizieren, wieviele und welche Arten von Nash-Equilibria für diese existieren. Das Video ist unterhaltsam und lädt zum mitdenken ein, ist aber nur bedingt vorlesungsrelevant.

https://www.youtube.com/watch?v=TZfh8hpJIxo