Aufgabe 2: Rechenrätsel

Teilnahme-ID: 63175

Bearbeiter/-in dieser Aufgabe:
Lars Noack

2. März 2022

Inhaltsverzeichnis

1 Lösungsidee

Unter den Bedingungen, die das Rätsel erfüllen muss, gibt es nur eine, die die Aufgabe schwer macht und mich daran hindert Rätsel mit 1000 Operatoren zu generieren. Die anderen führen nur dazu, dass die Aufgabe schwerer zu implementieren ist. Dies ist:

a) das Rätsel eindeutig lösbar ist

Auch wenn ich danach gesucht habe, habe ich keinen Mathe-Trick oder Kniff gefunden, der dies ermöglicht ohne das Rätsel langweilig werden zu lassen. Daher hat allein die Validierung ob ein Rätsel eine Laufzeit von $O(4^n)$, was sehr schlecht ist. Beispielsweise wenn $n = 15$, muss man allein zum validieren $4^15 = 1.073.741.824$ Rechnungen berechnen. Da man auf jeden fall $n = 15$ schaffen muss, nehmen wir einfach mal $1.073.741.824 = 10^9$.

1.1 naive Herangehensweise

Eine naive Lösungsidee wäre jetzt ein schon gelöstes Rätsel (mit Operatoren und Operanden) zu generieren und danach zu validieren, ob dieses Rätsel eindeutig ist. Dies müsste man machen bis das Ergebniss eindeutig ist. Man muss aber bedenken, dass im Durchschnitt die Anzahl der benötigten Versuche sich erhöht, wenn sich auch n erhöht. Dies liegt daran, dass es mit jedem Operator mehr, eine Möglichkeit mehr gibt, die Eindeutigkeit zu verlieren. Dies resultiert in einer Laufzeit von $O(cn \cdot 4^n)$ (c ist eine Konstante, die man entweder experimentell oder unter Berücksichtigung zu vieler Edge-Cases errechnen kann) Entspricht.

Aber ist das wirklich so dramatisch? **Ja.** Ich habe es ausprobiert. Man muss bei z.B. $n = 15$ $10^9 \cdot nc$ Rätsel lößen.

1.2 erfolgreiche Herangehensweise

Aber wie lößt man es dann?
Wenn man sich jetzt noch einmal die Aufgabenstellung genau anschaut kann man sehen, dass lediglich die Anzahl der Operatoren gegeben ist. Somit hat man Freiheiten in:

- Dem Wert der Operanden
- Der Art der Operatoren
- Dem Ergebniss

Was man auf jeden Fall machen muss ist zu validieren, ob das Rätsel eindeutig ist. Ich validiere hier, indem ich zuerst den Therm ausrechne, der resultiert wenn ich die gegebenen Operanden mit den gegebenen Operatoren "vermische". Dannach generiere ich alle möglichen Operatorenkombinationen. Diese schreibe ich dann zwischen die zu validieren Operanden und berechne das Ergebnis des Resultierenden Therms. Dann schaue ich ob sich das zuerst berechnete Ergebniss mehrere male in meinen restlichen berechneten Ergebnissen befindet. Wenn dies der Fall ist war das Rätsel nicht Eindeutig, sonst ist dies Eindeutig.

Wenn man jetzt anstatt Operanden und Operatoren zufällig zu generieren lediglich Operanden zufällig generiert, und dann die Validierung ohne Operanden durchführt, kann man am Ende des Algorythmusses alle Rätsel, deren Ergebniss einzigartig ist herausschreiben. Somit bekommt man eine ganze Liste mit möglichen Rätseln. Dann ist der Letzte Schritt nur noch das ïnterresanteste"Rätsel herauszufinden, da einige nicht so interesant sind. (z.B. nur multiplikation als Operator)

2 Laufzeitanalyse

2.1 Theoretische Laufzeitanalyse

Das erste, dass einem bei der Laufzeitanalyse auffällt, ist dass ich in Zeile 212[1] eine while schleife habe, die läuft, biss es Ergebnise gibt. Dies ist, da es mit einer bestimmten warscheinlichkeit sein kann dass kein Ergebniss kommt, und ich somit die funktion um Ergebnisse aufrufe, bis welche kommen. Dies bedeuted der worst case ist eine Laufzeit von $O(\infty)$, was aber nicht realistisch ist und nie passieren wird.

Der rest des Programmes, also nicht, die eindeutigen Therme zu finden, sondern nur den diversesten auszuwählen. Ist vernachlässigbar.

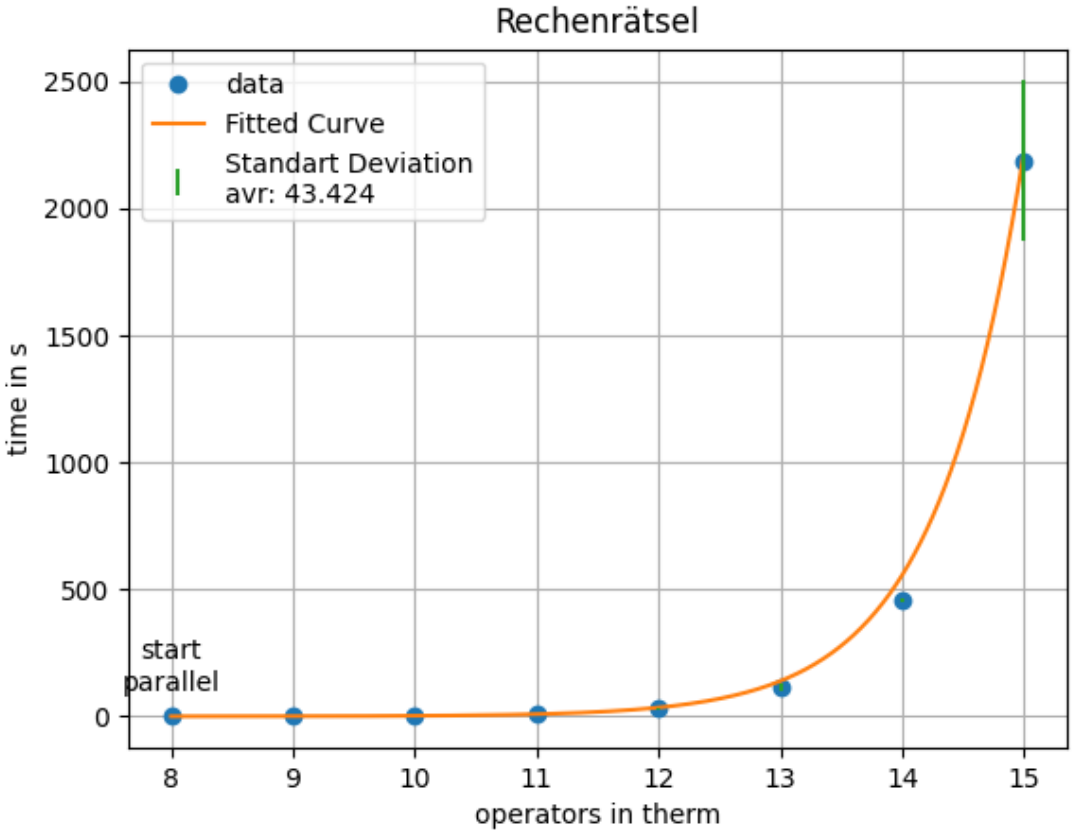
Bei $n > 9$ wird zwar multithreading verwendet, dies sollte aber nicht die Laufzeit verändern.

Das generieren der Operanden ist auch zu vernachlässigen, dann ist dass einzig wichtige, dass generrieren der Operatoren. Ich rechne jede Kombination aus, und überprüfe die eindeutugkeit. Da es 4 Operatoren bzw. Möglichkeiten gibt und es n "Plätze"gibt, macht dass eine LAufzeit von $O(n) = 4^n$ was eine ziemlich schlechte Exponentielle Laufzeit ist.

2.2 experimentelle Laufzeitanalyse

Bestätigen tut dies auch eine experimentelle laufzeitanalyse. Dafür habe ich ein paar mal das programm mit $8 \leq n \leq 15$. Dann habe ich die ergebnisse mit mathplotlib geplottet, die Standartabweichung berechnet, diese geplottet, und eine angeglichene Funktion der Maske $f(n) = f \cdot 4^n$ geplottet.

[1] Die Zeilen sind ziemlich final könnten sich aber noch ändern.



Wie man sieht passt diese angeglichene Funktion ziemlich gut. Der Funktionstherm wäre mit meinen Specs ca. $f(n) = 2.08e-06 \cdot 4^n$

3 Umsetzung

Hier wird kurz erläutert, wie die Lösungsidee im Programm tatsächlich umgesetzt wurde. Hier können auch Implementierungsdetails erwähnt werden.

4 Beispiele

Genügend Beispiele einbinden! Die Beispiele von der BwInf-Webseite sollten hier diskutiert werden, aber auch eigene Beispiele sind sehr gut – besonders wenn sie Spezialfälle abdecken. Aber bitte nicht 30 Seiten Programmausgabe hier einfügen!

5 Quellcode

Unwichtige Teile des Programms sollen hier nicht abgedruckt werden. Dieser Teil sollte nicht mehr als 2–3 Seiten umfassen, maximal 10.

6 Erweiterung