# ABC-hits opsommen

**Willem Jan Palenstijn**
(aantekeningen: Arjen Stolk)

Een *ABC-hit* is een drietal positieve gehele getallen $a$, $b$, $c$ waarvan elk tweetal onderling ondeelbaar is, $c$ de som is van $a$ en $b$ en waarvoor $\mathrm{rad}(abc)$ kleiner dan $c$ is.

We gaan een algoritme bekijken dat gegeven een positief geheel getal $N$ alle ABC-hits met $c$ kleiner dan $N$ opsomt in tijd die essentiëel lineair in $N$ is. Het idee van het algoritme is als volgt:

- Laat $c$ lopen van 2 tot $N$.
- Bepaal voor elke $c$ een verzameling mogelijke waarden voor $b$ die zeker alle ABC-hits bevat. Sluit dus alleen waarden uit waarvan zeker is dat ze geen ABC-hit zijn.
- Voor elk paar $(b, c)$ bepalen we $a = c - b$ en controleren of $(a, b, c)$ een ABC-hit is.

Om de looptijd van het algoritme binnen de gewenste limiet te houden, moeten we bepalen welke $b$'s er bij gegeven $c$ mogelijk een hit opleveren en het aantal van die $b$'s moet niet te groot zijn. Aangezien $a$, $b$ en $c$ onderling ondeelbaar zijn, geldt $\mathrm{rad}(abc) = \mathrm{rad}(a)\mathrm{rad}(b)\mathrm{rad}(c)$ en dus weten we dat

$$N > c > \mathrm{rad}(abc) \geq \mathrm{rad}(b)\mathrm{rad}(c).$$

Dit is een tamelijk grove afschatting, maar hij voldoet voor de analyse van dit algoritme. Door logaritmen te nemen vinden we de relatie:

$$\log \mathrm{rad}(b) + \log \mathrm{rad}(c) < \log N.$$

Het algoritme krijgt als invoer de bovengrens $N \in \mathbf{Z}_{>2}$ en doorloopt dan de volgende stappen:

1. Bepaal $\log \mathrm{rad}(n)$ voor alle $n < N$.
2. Sorteer een kopie van deze lijst op radicaal.
3. Laat $c$ lopen van 2 tot $N$.
4. Laat voor elke $c$ met behulp van de gesorteerde lijst $b$ lopen over alle waarden waarvoor $\log \mathrm{rad}(b) + \log \mathrm{rad}(c) < \log N$.
5. Bepaal voor elk paar $(b, c)$ of ze onderling ondeelbaar zijn, laat vervolgens $a$ gelijk zijn aan $c - b$ en bepaal de kwaliteit van het drietal $(a, b, c)$.

We gaan nu laten zien dat elke stap van dit algoritme voldoende snel kan worden uitgevoerd om essentiëel lineaire looptijd te krijgen.

Allereerst het maken van de tabel met radicalen. We doen dit met een variant op de zeef van Eratosthenes.

1. Laat $r(n) = 0$ voor alle $1 < n < N$.
2. Laat $p$ lopen van 2 tot $N$.
3. Als $r(p) = 0$, tel dan $\log p$ bij $r(k)$ op voor elk veelvoud $k$ van $p$ dat tussen 1 en $N$ ligt.

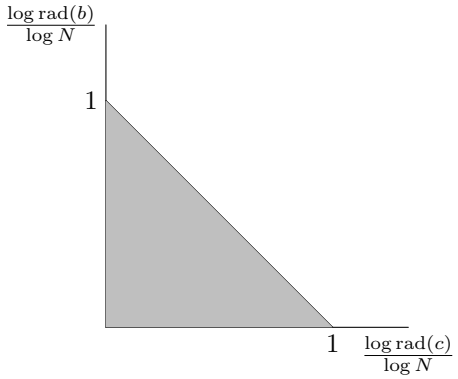
Elk priemgetal dat we tegenkomen heeft $r(p) = 0$, deze wordt dus $\log p = \log \mathrm{rad}(p)$. Een samengesteld getal krijgt een term $\log p$ voor elke priemdeler $p$ van het getal en dus geldt na afloop $r(n) = \log \mathrm{rad}(n)$ voor alle $1 < n < N$. Nu moeten we nog bepalen hoe lang deze stap duurt. Merk op dat het aantal keer dan $r(n)$ veranderd wordt gelijk is aan het aantal priemdelers van $n$. Dit aantal is van boven begrensd door ${}^2\log N$. Het totale aantal stappen is dus hoogstens $N \log N$ en elke stap is polynomiaal in $\log N$. Deze zeefstap loopt dus in $O(N^{1+\epsilon})$ tijd.

Nu moet er een kopie van deze lijst gesorteerd worden. Hiervoor gebruiken we het Heapsort algoritme, dat loopt in $N \log N$ polynomiale stappen in $\log N$, dus ook deze stap kost $O(N^{1+\epsilon})$ tijd.

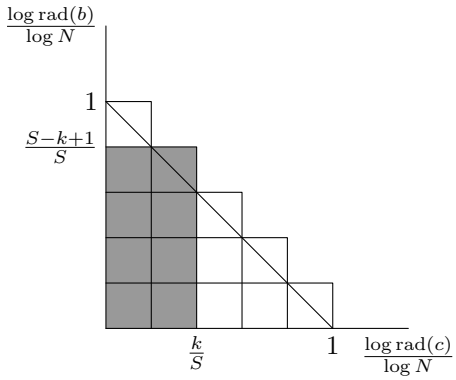
Nu komen we aan bij de lus over alle potentiële paren $(b, c)$. We gaan allereerst afschatten hoeveel van deze drietallen er zijn. Eerder hebben we gezien dat we gegeven $c$ alleen die $b$'s hoeven te bekijken die voldoen aan

$$\log \mathrm{rad}(b) + \log \mathrm{rad}(c) < \log N.$$

We zoeken dus paren $(b, c)$ die in het onderstaande plaatje een punt binnen het grijze gebied opleveren.



We gaan deze punten als volgt tellen. Laat $S$ enig positief geheel getal zijn. We verdelen het interval tussen 0 en 1 op zowel de horizontale als de verticale as in $S$ stukken. Vervolgens vragen we ons af hoeveel paren $(b, c)$ een punt opleveren binnen een rechthoek van de vorm zoals die in het onderstaande plaatje te zien is.




Voor zo'n paar moet gelden dat

$$\frac{\log \mathrm{rad}(b)}{\log N} < \frac{S-k+1}{S} \quad \text{en} \quad \frac{\log \mathrm{rad}(c)}{\log N} < \frac{k}{S},$$

oftewel,

$$\mathrm{rad}(b) < N^{\frac{S-k+1}{S}} \quad \text{en} \quad \mathrm{rad}(c) < N^{\frac{k}{S}}.$$

De stelling van Granville zegt het volgende:
*Laat $\alpha$ een reëel getal groter of gelijk aan 1 zijn. Dan geldt*

$$\lim_{N\to\infty} \frac{\log \#\Big\{x < N : \mathrm{rad}(x) < N^{1/\alpha}\Big\}}{\log N} = \frac{1}{\alpha}.$$

Iets minder formeel zegt de stelling dat als $N \to \infty$ dan is $\#\Big\{x < N : \mathrm{rad}(x) < N^{1/\alpha}\Big\}$ ongeveer gelijk aan $N^{1/\alpha+\epsilon}$.

Dit is precies wat we nodig hebben om het aantal paren $(b, c)$ af te schatten, want de voorwaarden op $b$ en $c$ zijn precies van het soort waar deze stelling iets over zegt. We zien dus dat het aantal paren $(b, c)$ binnen een rechthoek gaat naar

$$N^{\frac{S-k+1}{S}+\epsilon} N^{\frac{k}{S}+\epsilon} = N^{1+\frac{1}{S}+\epsilon}.$$

Deze afschatting is onafhankelijk van $k$.

We schatten het totale aantal punten nu af met de som over alle rechthoeken van het aantal punten in de rechthoek. Hierbij worden somige punten veel te vaak geteld, maar dat geeft

niet, want we zijn op zoek naar een bovengrens. Omdat we het aantal per rechthoek hebben afgeschat met een uitdrukking die niet afhangt van welke rechthoek we nemen, is het aantal gewoon $S$, een constante, keer het aantal in één rechthoek. We zien dus dat het aantal paren $O(N^{1+\frac{1}{S}+\epsilon})$ is. Omdat dit geldt voor elke $S$, kunnen we de $\frac{1}{S}$ en de $\epsilon$ samen nemen en zien we dat het aantal paren zelfs $O(N^{1+\epsilon})$ is, essentiëel lineair in $N$.

Rest nog te laten zien dat de hoeveelheid werk per paar polynomiaal in $\log N$ is. We moeten van het paar $(b, c)$ bepalen of ze onderling ondeelbaar zijn. Dit kan met de Euclidisch algoritme in tijd polynomiaal in $\log N$. Als het paar geen gemeenschappelijke delers heeft bepalen we nu $a = c - b$ en berekenen de kwaliteit:

$$q(a, b, c) = \frac{\log c}{\log \operatorname{rad}(a) + \log \operatorname{rad}(b) + \log \operatorname{rad}(c)}.$$

We zien dus dat het werk per paar $(b, c)$ polynomiaal in $\log N$ is en dus dat de gehele lus kan worden uitgevoerd in $O(N^{1+\epsilon})$ tijd.

Daarmee hebben we van alle stappen laten zien dat ze in essentiëel lineaire tijd uitgevoerd kunnen worden en dus kost het hele algoritme essentiëel lineaire tijd. Een belangrijk nadeel van dit algoritme is dat de hoeveelheid ruimte die het kost om de tabellen op te slaan lineair in $N$ is.