STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM

MR 110

Iets quicker dan quicker.

(Informatie, 11(1969), p 30-32)

door

M.H. van Emden

1969

IETS QUICKER DAN QUICKER

door ir. M. H. van Emden

Lezing gehouden voor het Nederlands Rekenmachine Genootschap op 18 oktober 1968

0 *Samenvatting*

Het sorteerprincipe van Hoare („quicksort") laat wiskundige beschouwingen toe over de efficiëntie ervan. Deze wordt vergeleken met de theoretisch haalbare efficiëntie. Een nieuw sorteerprogramma volgens het principe van Hoare wordt besproken, waarin een gedeelte van de theoretisch mogelijke verbetering gerealiseerd wordt.

1 *Quicksort en quickersort*

Het onderwerp van deze lezing is een algoritme voor het sorteren (dat wil zeggen, in monotoon niet-dalende volgorde plaatsen) van een rij getallen. Het betreffende onderzoek werd gedaan op het Mathematisch Centrum door prof. dr. F. E. J. Kruseman Aretz en mijzelf. Het algoritme werd door Hoare [1961] gepubliceerd in de vorm van een in ALGOL 60 geschreven procedure.

We zullen in het volgende de te sorteren getallen aanduiden als a [1], a [2], . . ., a [n], of kortweg als a [1 : n]. Het principe van quicksort wordt veel gebruikt voor het sorteren van hoeveelheden getallen, die geheel in een random access-geheugen kunnen staan. Quicksort heeft het voordeel snel te werken en bovendien weinig meer geheugenruimte (plaats voor ongeveer $\log_2 n$ gehele getallen) te gebruiken dan nodig is voor het opbergen van de getallen zelf. Scowen [1965] heeft een versie gepubliceerd („quickersort"), dat op dit laatstgenoemde punt een kleine verbetering geeft. Of deze versie inderdaad sneller is, hangt af van de specifieke implementatie van ALGOL 60.

Het principe van quicksort is eenvoudig en laat zich als volgt omschrijven. De waarde van één van de te sorteren getallen, het zogenaamde *referentie-element*, wordt bepaald, stel het is $y = a[j]$. Stel nu dat er in een linker (rechter) uiteinde van a geen getal staat dat groter (kleiner) is dan y. Exakter gezegd, stel dat er gehele getallen p en q bestaan ($1 \leqq p < q \leqq n$) zodanig dat altijd geldt:

$$a[i] \leqq y \text{ voor } 1 \leqq i \leqq p$$
$$a[i] \geqq y \text{ voor } q \leqq i \leqq n \qquad \ldots (1)$$

Stel bovendien dat p (q) het grootste (kleinste) getal is dat hieraan voldoet, dat wil zeggen $a[p+1] > y$ en $a[q-1] < y$. Wanneer we nu deze twee verwisselen, kunnen we p (q) met tenminste 1 verhogen (verlagen) terwijl de betrekkingen (1) hun geldigheid behouden.
Dit proces kan voortgezet worden totdat $p + 1 = q$. Wanneer j (de index van het referentie-element $y = a[j]$) kleiner is dan q stellen we $r = p$, anders $r = q$. Vervolgens verwisselen we a [j] en a [r], zodat we de situatie hebben gekregen, dat er links (rechts) van a [r] = y geen enkel element staat dat groter (kleiner) is dan y.

Door nu de stukken a [1 : r—1] en a [r+1 : n] afzonderlijk te sorteren is de hele rij a [1 : n] gesorteerd.

2 *De snelheid van quicksort*

Wanneer we de snelheid van sorteer-algoritmen willen vergelijken, kunnen we dat het beste doen met hun *gemiddelde* snelheid. Dit impliceert dat er gemiddeld wordt over een zekere verzameling van getallenrijen, die het algoritme eventueel zou moeten sorteren. Nu zijn de getallen zelf niet in de eerste plaats van belang; we kunnen ze vervangen denken door hun rangnummers in de gesorteerde rij. Een te sorteren getallenrij kunnen we dus abstraheren tot een bepaalde permutatie van de rangnummers 1, . . ., n.
Voor de onderstaande beschouwingen maken we de volgende veronderstelling over de verzameling getallenrijen waarvoor we de gemiddelde snelheid van een sorteer-algoritme bepalen:
Bij een aselekte trekking uit deze verzameling heeft elke permutatie van de rangnummers 1, . . .,n een even grote kans (= 1/n!) van optreden.
Uit het hiervoor beschreven principe van quicksort volgt, dat wanneer de keuze van de oorspronkelijke rij hieraan voldoet, het ontstaan van alle nog te sorteren deelrijen hieraan voldoet.
Verder moeten we een maat aangeven voor de hoeveelheid „werk" die verricht wordt bij het sorteren. Uit theoretische overwegingen verdient het de voorkeur als eenheid te nemen: het vergelijken van twee getallen uit de rij. Deze eenheid zullen we aanduiden als „*komparitie*".
We zullen in het volgende met behulp van informatietheoretische overwegingen komen tot een verwachting van het aantal komparities dat nodig is om een getallenrij, die aselekt getrokken is uit de boven beschreven verzameling, te sorteren. Hiertoe kan men het beste het sorteerproces beschouwen als het *winnen van informatie*. Wanneer we namelijk verslag zouden leggen van de successievelijk verrichte verwisselingen, zouden we, terugwerkend vanuit de gesorteerde rij, de oorspronkelijke rij kunnen rekonstrueren. Tijdens het sorteren moeten we daarom een hoeveelheid *informatie* winnen gelijk aan de *onzekerheid* die inherent is in het aselekt trekken van een getallenrij uit de verzameling.
We veronderstelden, dat alle permutaties een even grote kans van optreden zouden hebben. We hebben daarom te doen met een diskrete kansverdeling op n! mogelijke uitkomsten, ieder met kans 1/n!. Vervolgens Shannon is de entropie H van deze kansverdeling gelijk aan de onzekerheid inherent in deze situatie:

$$H = -\sum_{i=1}^{N} p_i \log p_i$$

Dit geeft voor $N = n!$ en $p_i = 1/n!$:

$$H = \log n!$$

Wanneer het grondtal van de logaritme 2 is, vinden we de entropie in *bits*, de gebruikelijke eenheden van *selektieve informatie*.

Het sorteerproces moet aan informatie $\log_2 n!$ bits opbrengen en dat gaat met behulp van de komparities. De vraag is nu: hoeveel komparities hebben we nodig om een hoeveelheid informatie van $\log_2 n!$ te winnen? Laten we daartoe kijken naar de informatie-opbrengst van één komparitie. Deze geeft antwoord op de vraag: Is a[i] $<$ y? Er wordt één van twee mogelijke uitkomsten gegeven; de ene met kans r/n, de andere met kans 1—r/n. Hiermee wordt een informatie gewonnen van:

$$H = -(r/n)\ \log_2\ (r/n)\ -(1-r/n)\ \log_2\ (1-r/n) \text{ bits} \qquad \ldots (2)$$

waarbij r weer het rangnummer van y is. Wanneer $r = n/2$, dan is H maximaal en wel gelijk aan 1 bit. Bij de best denkbare strategie krijgen we steeds $r = n/2$; de gemiddelde informatie-opbrengst per komparitie is dan 1 bit en we moeten gemiddeld

$$t_n = \log_2 n! = 1.44\ \ln n! \qquad \ldots (3)$$

komparities doen. Dit is de *theoretische ondergrens* voor de verwachting van het aantal komparities. Een praktisch uitvoerbare strategie zal er in het algemeen gemiddeld meer nodig hebben, nooit minder.
Hoeveel meer hangt ervan af hoeveel de gemiddelde informatie-opbrengst per komparitie minder is dan de maximaal mogelijke van 1 bit. Wanneer we aannemen dat f_r de kans is dat r die bepaalde waarde aanneemt, vinden we, door (2) over f te middelen, als gemiddelde informatie-opbrengst per komparitie:

$$H = -\sum_{r=1}^{n} f_r\ ((r/n)\ \log_2\ (r/n) + (1-r/n)\ \log_2\ (1-r/n)) \text{ bits.}$$

We zijn vooral geïnteresseerd in grote waarden n, waarvoor deze som nadert tot:

$$H = -\int_0^1 x\ g(x)\ \log_2\ (x)\ dx - \int_0^1 (1-x)\ g(x)\ \log_2\ (1-x)\ dx \text{ bits, waarin } f_r = g(r/n).$$

We veronderstellen g symmetrisch ten opzichte van $x = .5$, dat wil zeggen $g(x) = g(1-x)$, zodat:

$$H = -2\int_0^1 x\ g(x)\ \log_2\ (x)\ dx \text{ bits,}$$

de gemiddelde informatie-opbrengst per komparitie. De benodigde hoeveelheid informatie is $\log_2 n!$ bits. De verwachting voor het benodigde aantal komparities is dan:

$$t_n = \frac{\log_2 n!}{-2\int_0^1 x\ g(x)\ \log_2\ (x)\ dx}$$

Uit deze formule kunnen we al enige interessante konklusies trekken. Bij quicksort geldt: $g(x) = 1$ voor $0 \leqq x \leqq 1$, zodat we vinden:

$$t_n = 2\ln n!$$

Wanneer we dit vergelijken met (3) dan zien we dat er theoretisch maximaal een verbetering van 28 % mogelijk is in gemiddeld aantal komparities ten opzichte van quicksort.

3 *Mogelijkheden tot verhoging van de snelheid*

In het volgende worden twee mogelijkheden besproken om een gedeelte van de theoretisch mogelijke verbetering van het quicksort-principe te realiseren.

a *Referentie-element als mediaan van een steekproef*

We zagen dat een komparitie de maximale informatie-opbrengst heeft als $r = n/2$, dat wil zeggen, als het referentie-element de *mediaan* van de te sorteren rij is. De moeilijkheid is, dat we pas kunnen weten wat de mediaan is wanneer de getallen, althans gedeeltelijk, gesorteerd zijn. Er is echter een verbetering mogelijk wanneer we niet (zoals bij quicksort) $x = r/n$ op zodanige wijze kiezen dat $g(x) = 1$ voor $0 \leqq x \leqq 1$, maar wanneer we proberen g(x) zo groot mogelijk te maken voor x in de buurt van .5. Het gaat dus om een strategie voor het bepalen van het referentie-element op zodanige wijze, dat de kansdichtheid g zoveel mogelijk in het midden gekoncentreerd is.
Een mogelijkheid is het trekken van een steekproef ter grootte van $2k+1$ elementen uit de getallenrij a[1:n]. Van deze steekproef bepalen we de mediaan en deze zal als referentie-element y fungeren. Nu hebben we niet meer $g(x) = 1$ voor $0 \leqq x \leqq 1$ maar in het algemeen:

$$g(x) = Cx^k(1-x)^k \quad \text{met} \quad 1/C = \int_0^1 x^k(1-x)^k dx.$$

Nu is inderdaad g(x) steeds meer in het midden gekoncentreerd naarmate k groter is. Stel dat we nu k als vast percentage van n nemen en vervolgens n onbeperkt laten toenemen, dan zal de kansdichtheid g(x) zich steeds meer in een willekeurige kleine omgeving van $x = .5$ koncentreren. We kunnen op deze manier het theoretisch minimale aantal komparities, $t_n = \log_2 n!$, voor voldoend grote n willekeurig dicht benaderen. Helaas is het zo, dat voor kleine n deze methode minder gunstig is dan de oorspronkelijke, omdat de investering van het mediaan-bepalen er dan niet uit komt. Dit bezwaar geldt niet voor het volgende alternatief.

b *Drijvend referentie-element*

Het zwakke punt van quicksort is, dat het eerste het beste getal van de rij als referentie-element aanvaard wordt, terwijl zijn geschiktheid daarvoor afhangt van zijn relatie tot andere elementen. We zouden eigenlijk de beslissing over wat het referentie-element zal zijn, zo lang mogelijk willen uitstellen. De strategie van het „drijvende referentie-element" onderscheidt zich door niet één bepaald referentie-element aan te houden, maar steeds een zo groot mogelijk interval waarvan elk element nog referentie-element zou kunnen worden. Deze vrijheid kunnen we gebruiken om de kansdichtheid g voor het rangnummer van het uiteindelijke referentie-element zoveel mogelijk in het midden te koncentreren.
Stel dat voor gehele getallen p en q met $1 \leqq p < q \leqq n$ betrekkingen (1) weer gelden. In deze situatie hoeven we nog niet y als referentie-element te aanvaarden, maar wanneer bij voorbeeld:

$x = \max a[i],\ 1 \leqq i \leqq p$ en
$z = \min a[i],\ q \leqq i \leqq n,$

dan kan het referentie-element nog gekozen worden uit alle elementen waarvan het rangnummer niet kleiner is dan dat van x of niet groter dan dat van z. Wanneer we p groter of q kleiner willen maken, kan het voorkomen

dat x of z door een van deze tussengelegen getallen moet moet worden vervangen. Bij de manier waarop dit gedaan wordt, kan rekening gehouden worden met de wenselijkheid het uiteindelijke referentie-element zo dicht mogelijk bij het midden te plaatsen. Wanneer tenslotte p + 1 = q, blijkt dat zowel x als z uiteindelijk referentie-element zou kunnen fungeren. Het rangnummer van x wordt p, dat van z wordt q. De taktiek van het drijvend referentie-element heeft slechts weinig meer bewerkingen nodig dan de oorspronkelijke quicksort.
Voor *grote n* geeft het tijdsbesparing doordat de kansdichtheidsfunktie voor het rangnummer van het uiteindelijke referentie-element meer in het midden is gekoncentreerd dan de uniforme verdelingsfunktie die optreedt bij quicksort.
Voor *kleine n* geeft het tijdsbesparing doordat bij elke slag het aantal nog te sorteren elementen 2 minder wordt in plaats van 1 minder zoals bij quicksort.

4 *Resultaat van een proefneming*

Voor een aselekt getrokken rij van 1001 getallen werden de prestaties van twee ALGOL 60 programma's vergeleken. Het eerste programma was geschreven als een zodanige kruising tussen Hoare's quicksort en Scowen's quickersort, dat het resultaat optimaal is voor het ALGOL-systeem van het Mathematisch Centrum.
Het tweede programma wijkt van het eerste af door de boven beschreven taktiek van het drijvend referentie-element. De vergelijking van deze twee programma's werd 10 maal herhaald en het tweede programma bleek gemiddeld 25 % sneller.
Een nader onderzoek zal moeten uitwijzen hoe de bijdragen van het „kleine n-effekt" en het „grote n-effekt" tot deze 25 % zich verhouden.

5 *Literatuurverwijzingen*

Hoare C. A. R., Algorithm 63; Comm. ACM *4* (1961), p. 321
Hoare C. A. R., Algorithm 64; Comm. ACM *4* (1961), p. 321
Scowen R. S., Algorithm 271; Comm. ACM *8* (1965), p. 669

SERVICEBUREAUS PRESENTEREN ZICH

C-E-I-R N.V.

C-E-I-R n.v., Plaats 24a, 's-Gravenhage, tel.: 070-112187 werd in 1965 opgericht als dochtermaatschappij van C-E-I-R Inc. (Washington, D.C.) en C-E-I-R Ltd. (Londen). De naam C-E-I-R is afgeleid van de oorspronkelijke naam „Corporation for Economic and Industrial Research".
Deze naam wijst nog op het oorspronkelijk doel om econometrische research en industriële planning toe te passen in het bedrijfsleven.
C-E-I-R Inc. werd in 1954 opgericht en heeft op dit moment kantoren in Washington, D.C., New York City, Boston, Los Angeles, San Francisco en Mexico City. Haar staf bestaat uit wiskundigen, economen, statistici, ingenieurs, operations research- en systeem analisten en specialisten op het gebied van computer hardware en software.
Naast de gewone activiteiten zoals management consultancy, het oplossen van problemen en het ontwikkelen van software is C-E-I-R Inc. ook nog actief op een heel ander gespecialiseerd terrein.
Het American Research Bureau houdt zich bezig met het meten van luister- en kijkdichtheid voor Radio en T.V. Het Institute for Advanced Technology geniet een welverdiende reputatie voor de hoge standaard van haar opleidingen die zich bezig houden met ieder mogelijk aspect van computer toepassingen.
In de loop van 1967 werden de aandelen van C-E-I-R Inc. overgenomen door Control Data Corporation. Binnen de Control Data organisatie was men tot de conclusie gekomen dat het niet langer de taak was van een computerfabrikant om zich bezig te houden met het ontwikkelen van applicatie software. De vervaardiging hiervan is kostbaar en in de meeste gevallen voldoen confectieprogramma's maar ten dele.
Om haar cliënten de nodige steun te verlenen bij het ontwikkelen van programma's die gericht zijn op de speciale wensen van de gebruikers werd C-E-I-R overgenomen. Ondanks deze financiële en spirituele band met de computerindustrie blijft de C-E-I-R organisatie onafhankelijk haar taak vervullen als „problem solvers" voor het bedrijfsleven.
Het contact met Washington is zeer intensief. Een overeenkomst tussen beide bureaus zorgt voor de uitwisseling van ervaring en technische gegevens. Hierdoor komen de laatste methoden op het gebied van operationele research en computer toepassingen ter beschikking van alle cliënten van C-E-I-R.
Wanneer als gevolg van omstandigheden de staf in Nederland niet toereikend is, zijn medewerkers van C-E-I-R Inc. voor lange perioden beschikbaar om het tekort aan te vullen.
Het toepassen van besliskundige methode op bestuursproblemen in het bedrijf bevindt zich in Nederland nog in een ontwikkelingsstadium. In haar contacten met cliënten ondervond C-E-I-R n.v. naast enig enthousiasme nog een zekere weerstand en vooroordeel tegen het gebruik van moderne besliskundige methoden. In een aantal gevallen was de reden van deze weerstand dat de bedrijfsleiding zich niet had gerealiseerd dat het aanschaffen van een computer niet voldoende is om alle problemen op te lossen.
Zonder goede software en applicatie programma's plus de organisatie om hiervan gebruik te maken is een computer een kostbaar instrument met een zeer beperkt nut. Men zal moeten leren er rekening mee te houden dat de investeringen die moeten worden gedaan voor software