Complexiteit 2023 — college 4

28 februari 2023

Selectie is $O(n)$

Insertion sort

Recurrente betrekkingen

Vorige keer

- beslissingsboomargument:
  - hoogte boom h, worst case $h + 1$
  - n knopen: $h \geq \lceil \lg(n+1) \rceil - 1 = \lfloor \lg n \rfloor$
  - b bladeren: $h \geq \lceil \lg b \rceil$
  - "makkelijk", niet altijd even scherp
- adversary-argument
  - zie het probleem als een twee-spelerspel
  - adversary (tegenstander) houdt jou zo lang mogelijk bezig
  - adversary-strategie geeft ondergrens op worst case
- toernooimethode: optimaal algoritme voor het vinden van de op één na grootste

Vandaag

- Een lineair algoritme voor het selectieprobleem
- Insertion sort
- Recurrente betrekkingen

Selectieprobleem is $O(n)$

Probleem

Gegeven een array $A = A[1], \ldots, A[n]$ met n verschillende getallen, en een geheel getal k met $1 \leq k \leq n$. Gevraagd het k-de element in grootte (in volgorde van klein naar groot). Formeler: gevraagd het element $A[i]$ dat groter is dan precies $k - 1$ andere $A[j]$'s.

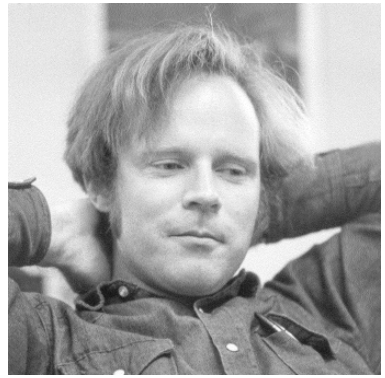
Complexiteit

Het selectieprobleem is $O(n)$. We bewijzen dit door een algoritme te geven dat de k-de in grootte vindt in $O(n)$ vergelijkingen in de worst case*.

*Blum, Floyd, Pratt, Rivest, Tarjan, 1973

Even tussendoor

Manuel Blum*
CAPTCHA†



Robert Floyd*
Floyd-Warshall†

Vaughan Pratt
Knuth-Morris-Pratt†

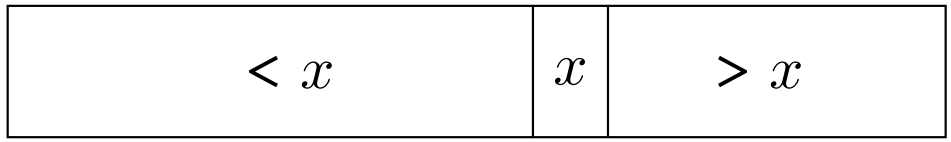
Ron Rivest*
Rivest-Shamir-Adleman†

Robert Tarjan*
Tarjan's SCC†

*Turing Award
†en heel veel meer

Het idee

- Kies (bijv. willekeurig) een element x uit het array (de spil*)
- Reorganiseer het array als volgt:



- Dit kost ongeveer n arrayvergelijkingen
- De k-de in grootte zit in het linker- óf het rechtergedeelte
- Ga dus met één van beide gedeeltes verder en herhaal
- Met geluk: telkens twee gelijke stukken, in totaal ongeveer $n + \frac{n}{2} + \frac{n}{2^2} + \frac{n}{2^3} + \ldots + \frac{n}{2^m} \leq 2n$ vergelijkingen
- Average case is $O(n)$, maar worst case is $\Theta(n^2)$
- Oplossing: kies goede spillen

*Engels: pivot

$O(n)$-algoritme

Algoritme:

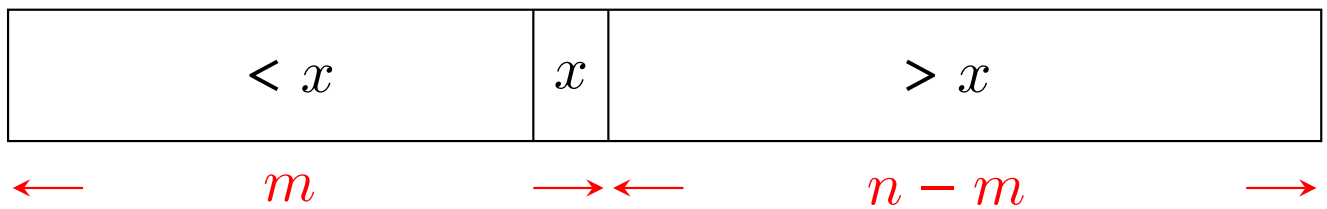
1. verdeel de getallen in $\lfloor \frac{n}{5} \rfloor$ groepjes van 5 elementen, en één groepje met de resterende $n \bmod 5$*
2. vind de mediaan van elk van de $\lceil \frac{n}{5} \rceil$ groepjes, bijvoorbeeld met Bubblesort of opgave 24
3. vind de mediaan x van de in stap 2 gevonden $\lceil \frac{n}{5} \rceil$ medianen: recursie†

*deze 5 is niet willekeurig gekozen, maar is optimaal

†de mediaan van ℓ elementen is de $\lceil \frac{\ell}{2} \rceil$-de in grootte

Vervolg algoritme

4. Partitioneer (ongeveer zoals bij Quicksort) alle elementen rond x. Stel dat m getallen $\leq x$ zijn en $n - m$ getallen $> x$.



5. Vind de k-de in grootte uit m stuks als $k \leq m$, of de $(k - m)$-de uit $n - m$ als $k > m$: recursie

Afschatting

Na stap 3 van het algoritme geldt:

- ten minste $3 \times (\lceil \frac{1}{2} \lceil \frac{n}{5} \rceil \rceil - 2 \geq \frac{3n}{10} - 6$ elementen zijn groter (resp. kleiner) dan x.
- er zijn dus hooguit $\lceil \frac{7n}{10} \rceil + 6$ elementen $\leq x$ (resp. $\geq x$)

Gevolg: in stap 5 wordt het algoritme recursief aangeroepen op hooguit $\lceil \frac{7n}{10} \rceil + 6$ elementen.

Voorbeeld

12	15	11	2	9	5	0	7	3	21	44	40	1	18	20	32	19	35	37	39
13	16	14	8	10	16	6	33	4	27	49	46	52	25	51	34	43	56	72	79
17	23	24	28	29	30	31	36	42	47	50	55	58	60	63	65	66	67	81	83
22	45	38	53	61	41	62	82	54	48	59	57	71	78	64	80	70	76	85	87
96	95	94	86	89	69	68	97	73	92	74	88	99	84	75	90	77	93	98	91

Blauw is de mediaan der medianen (x in stap 3); groen is gegarandeerd lager, evenals het linker oranje gedeelte; rood en het rechter oranje gedeelte zijn gegarandeerd hoger. In het algemeen zijn gegarandeerd ongeveer $\frac{3n}{10}$ elementen uit het array kleiner resp. groter dan x. In het ergste geval ga je de recursie dus in op de resterende $\approx 70\%$ van het array.

Recurrente betrekking

Laat $T(n)$ het aantal vergelijkingen zijn dat dit (recursieve) algoritme doet in de worst case. Onder de aanname dat $T(n)$ stijgend is, geldt:

$$T(n) = \begin{cases} \text{constant} & n \leq \ldots \\ T(\lceil \frac{n}{5} \rceil) + T(\lceil \frac{7n}{10} \rceil + 6) + O(n) & n > \ldots \text{(bv. } 80) \end{cases}$$

- $T(\lceil \frac{7n}{10} \rceil + 6) < n$ voor $n > 20$
- $O(n)$ komt van stappen 1, 2 en 4 samen

Bewering:

$T(n) \leq cn$ voor geschikte c en $n \geq \ldots$
ofwel: $T(n) \in O(n)$

Selectie is $\Omega(n)$

Opgave 26

Elk algoritme voor het selectieprobleem dat is gebaseerd op arrayvergelijkingen doet in de worst case ten minste $\lceil \frac{n}{2} \rceil$ van zulke vergelijkingen.

Gevolg

Het selectieprobleem is $\Omega(n)$ (en dus $\Theta(n)$).

Sorteren

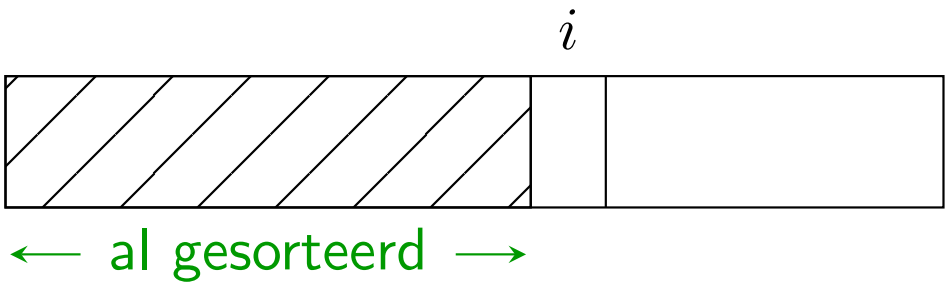
Probleem

Gegeven een rij (array) $A = A[1], \ldots, A[n]$. Sorteer A oplopend, dus $A[i] \leq A[i+1]$ voor alle i $(1 \leq i < n)$; $A[i] < A[i+1]$ als alle elementen verschillend zijn.

We bekijken eerst sorteeralgoritmen die zijn gebaseerd op het doen van arrayvergelijkingen.

Insertion sort

Insertion sort



Conceptueel idee: itereer over i en voeg $A[i]$ op de juiste plek in het gesorteerde stuk $A[1], \ldots, A[i-1]$ in door herhaald te vergelijken met de linkerbuur en indien nodig te verwisselen.

Insertion sort

Het algoritme is gebaseerd op het doen van arrayvergelijkingen ($A[i] < A[j]$).

```
1 for i := 2 to n do
  |   // nu A[i] invoegen op de juiste plek in
  |       A[1], ..., A[i − 1]
2 |   x := A[i];
3 |   j := i − 1;
4 |   while j > 0 and A[j] > x do
5 |   |   A[j + 1] := A[j];
6 |   |   j := j − 1;
7 |   od
8 |   A[j + 1] := x;
9 od
```

Observaties

- het aantal arrayvergelijkingen is een goede maat voor de complexiteit
- insertion sort doet eigenlijk steeds compare-exchange-operaties: vergelijk en verwissel (indien nodig). Deze zijn hier vermomd als verschuivingen, waarna pas in de laatste stap $A[i]$ daadwerkelijk pas wordt neergezet.
- de "verwisselingen" zijn steeds buurverwisselingen

Basisoperatie

Merk op:

- de for-loop (regel 1) doet precies $n-1$ iteraties: $\Theta(n)$
- de vergelijking $A[j] > x$ gebeurt minstens één keer per iteratie v/d for-loop, dus totaal *minstens* $n-1$ keer: $\Omega(n)$
- het vergelijken van array-elementen (want $x = A[i]$) is een zinnige operatie voor sorteren

Dan:

- regels 1, 2, 3 en 8 gebeuren elk precies $n-1 \in \Omega(n)$ keer*
- eerste deel van regel 4 ($j > 0$) gebeurt per iteratie v/d for-loop hooguit één keer vaker dan het tweede deel. Dit is dus hooguit een constante factor 2.
- regels 5 en 6 gebeuren alleen als regel 4 waar is, dus hooguit even vaak als de test op regel 4 (ook orde van grootte)

*je kan ook vinden dat regel 1 maar één keer gebeurt; dan geldt $1 \in \Omega(n)$

Complexiteit Insertion sort

We tellen het aantal vergelijkingen $A[j] > x$ (waar $x = A[i]$).

1. Best case: $B(n) = \sum_{i=2}^{n} 1 = n - 1$
2. Worst case: $W(n) = \sum_{i=2}^{n} (i - 1) = \frac{1}{2} n(n-1)$
3. Average case*: $A(n) = \frac{1}{4} n(n-1) + n - \sum_{i=1}^{n} \frac{1}{i} \in \Theta(n^2)$

*onder de aanname dat alle $A[i]$'s verschillend zijn en dat alle $n!$ permutaties (ordeningen) van $A[1], \ldots, A[n]$ even waarschijnlijk zijn. We middelen dan over alle mogelijke permutaties en dat zijn in essentie alle mogelijke invoerrijtjes.

Best case

Het minimale aantal vergelijkingen van Insertion sort is $n - 1$.

- de eerste vergelijking op regel 4 is altijd aanvankelijk waar (want $i > 1$), dus de tweede vergelijking ($A[j] > x$) wordt elke iteratie minstens één keer uitgevoerd. Totaal $n - 1$ iteraties, dus minstens $n - 1$ keer.
- $n - 1$ kan worden gehaald: dan moet elke keer óf direct $A[i - 1] \leq A[i]$ zijn (dus oplopend gesorteerd), of $j \leq 0$ vóór de tweede vergelijking $A[j] > x$
- twéé goede rijtjes: oplopend gesorteerd, of alleen de eerste twee elementen fout.

Worst case

Het maximale aantal vergelijkingen van Insertion sort is $\frac{1}{2}n(n-1)$.

- de tweede vergelijking op regel 4 kan hooguit gebeuren zolang $j > 0$: hooguit $i - 1$ keer. Totaal dus hooguit $\sum_{i=2}^{n}(i-1) = \frac{1}{2}n(n-1)$ keer.
- dit kan worden gehaald: dan moet elke keer voor alle $j > 1$ ook gelden $A[j] > A[i]$. Voor $j = 1$ maakt het niet meer uit.
- slechte rijtjes: alle rijtjes waarvoor voor elke $A[i]$ geldt dat $A[i]$ het kleinste of het op één na kleinste element is van $A[1], \ldots, A[i]$

Inversies

Definitie: een inversie van de permutatie $A[1], \ldots, A[n]$ is een paar $(A[i], A[j])$ waarvoor $i < j$ en $A[i] > A[j]$. M.a.w.: een inversie is een paar $(A[i], A[j])$ dat verkeerd om staat.

Merk op: *elk* sorteeralgoritme moet *alle* aanwezige inversies opheffen.

Verder: als een sorteeralgoritme altijd hooguit één inversie opheft per arrayvergelijking, dan is het aantal vergelijkingen dat wordt gedaan om $A[1], \ldots, A[n]$ te sorteren *ten minste* het aantal inversies van A.

Ten slotte: een buurverwisseling (zoals bij Insertion sort) heft altijd precies één inversie op (aangenomen dat de buren verkeed om staan).

Worst case

Stelling

Het maximale aantal inversies dat kan voorkomen in een rijtje van n verschillende waarden is $\binom{n}{2} = \frac{1}{2}n(n-1)$. Dit treedt (o.a.) op bij een aflopend gesorteerd rijtje.

Gevolg

Elk algoritme dat sorteert met behulp van arrayvergelijkingen en dat per vergelijking hooguit één inversie opheft, doet ten minste $\frac{1}{2}n(n-1)$ vergelijkingen in de worst case.

Conclusie: Insertion sort is optimaal wat betreft de worst case, binnen de beschreven klasse van algoritmen.

Average case

Stelling

Het *gemiddeld* aantal inversies in een permutatie van n verschillende waarden (bijvoorbeeld de getallen 1 t/m n) is $\frac{1}{4}n(n-1)$. Dit onder de aanname dat alle $n!$ permutaties even waarschijnlijk zijn.

Gevolg

Elk algoritme dat sorteert met behulp van arrayvergelijkingen en dat per vergelijking hooguit één inversie opheft, doet ten minste $\frac{1}{4}n(n-1)$ vergelijkingen in de average case.

Conclusie: Insertion sort doet gemiddeld $\frac{1}{4}n(n-1) + n - 1 - \sum_{i=2}^{n} \frac{1}{i}$ vergelijkingen, dus Insertion sort is optimaal in orde van grootte wat betreft de average case, binnen de beschreven klasse van algoritmen.

Inversies

Voor sorteeralgoritmen gebaseerd op arrayvergelijkingen, waarbij per arrayvergelijking hooguit één inversie wordt opgeheven*, geldt:

- # arrayvergelijkingen ≥ # inversies invoerarray
- # arrayvergelijkingen in de worst case ≥ $\frac{1}{2}n(n-1)$

Als je een beter sorteeralgoritme wilt (gebaseerd op arrayvergelijkingen), moet je dus elementen verwisselen die verder van elkaar liggen, zoals gebeurt bij Mergesort, Quicksort en Shellsort (en veel meer).

*zoals bij algoritmen die gebruikmaken van buurverwisselingen, zoals Insertion sort en Bubblesort

Inversies opheffen

Stel dat $A[i]$ en $A[i+k]$ $(k > 0)$ verkeerd om staan en dat we die verwisselen. Hoeveel inversies worden dan ten minste respectievelijk ten hoogste opgeheven?

Situatie:

met $A[i] > A[i+k]$. Verwissel nu $A[i]$ en $A[i+k]$.

Recurrente betrekkingen

- rij van Fibonacci: $0, 1, 1, 2, 3, 5, 8, 13, 21$
- vanaf het derde element: som van de voorgaande twee
- een recurrente betrekking is een voorschrift om een waarde $T(n)$ te berekenen door middel van zijn voorganger(s), dus bijvoorbeeld $T(n-1)$ of $T(\frac{n}{2})$, of ...

Voor de Fibonacci-getallen geldt:

$$T(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ T(n-1) + T(n-2) & n > 1 \end{cases}$$

Recurrente betrekkingen

Een ander voorbeld van een recurrente betrekking:

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T(\frac{n}{2}) + n & n = 2^k > 1 \end{cases}$$

- vorm een vermoeden voor een formule (of sommatie) d.m.v.:
  - herhaalde substitutie en afleiden algemene vorm; of
  - probeer wat termen door te rekenen: zie je een patroon?
- bewijs* je vermoeden met volledige inductie

Oplossing: $T(n) = n + n \lg n \in \Theta(n \lg n)$

*of niet

Recurrente betrekkingen

De vorige recurrente betrekking, maar nu voor algemenere n, dus niet alleen voor tweemachten:

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T(\lfloor \frac{n}{2} \rfloor) + n & n > 1 \end{cases}$$

Dan geldt: $T(n) \in O(n \lg n)$ (en overigens ook $T(n) \in \Theta(n \lg n)$)

Dit kan worden bewezen door met behulp van volledige inductie bijvoorbeeld aan te tonen dat $T(n) \leq 2n \lg n$ voor alle $n > 1$.

Recurrente betrekkingen

Ten slotte nog twee voorbeelden om op te lossen:

1. $T(n) = \begin{cases} 3 & n = 1 \\ T(n-1) + n - 1 & n > 1 \end{cases}$

   Oplossing: $T(n) = 3 + \frac{1}{2}n(n-1)$

2. $T(n) = \begin{cases} 0 & n = 1 \\ 2T(\frac{n}{4}) + \sqrt{n} & n = 4^k > 1 \end{cases}$

   Oplossing: $T(n) = \sqrt{n}\log_4 n = \frac{1}{2}\sqrt{n}\lg n$

Om te onthouden van vandaag

- selectie kan in $O(n)$
- Insertion sort
  - best case: $n - 1$
  - worst case: $\frac{1}{2}n(n-1)$, optimaal voor buurvergelijkingen
  - average case: $\Theta(n^2)$, optimaal in orde van grootte voor buurvergelijkingen
- inversie: paar $(A[i], A[j])$ zodat $i < j$ maar $A[i] > A[j]$, sorteeralgoritmen moeten alle inversies opheffen
- recurrente betrekkingen
  - vorm een vermoeden (herhaalde substitutie of doorrekenen eerste x termen)
  - bewijs met volledige inductie
  - oefenen

(Werk)college

Volgende college: dinsdag 7 maart, 9u00–10u45, zaal C1 (Gorlaeus)

Werkcollege: zodadelijk van 11u00 tot 12u45, computerzalen 302–304, 306–308 en 303 (Snellius)
Opgaven uit het dictaat: 32, 14, 15, 31, 18

Huiswerk

Huiswerk 1

- uitwerking komt binnenkort op de website
- cijfers volgen binnen twee werkweken op Brightspace

Huiswerk 2

- volgende week op de website, deadline eind maart