Vierde college complexiteit

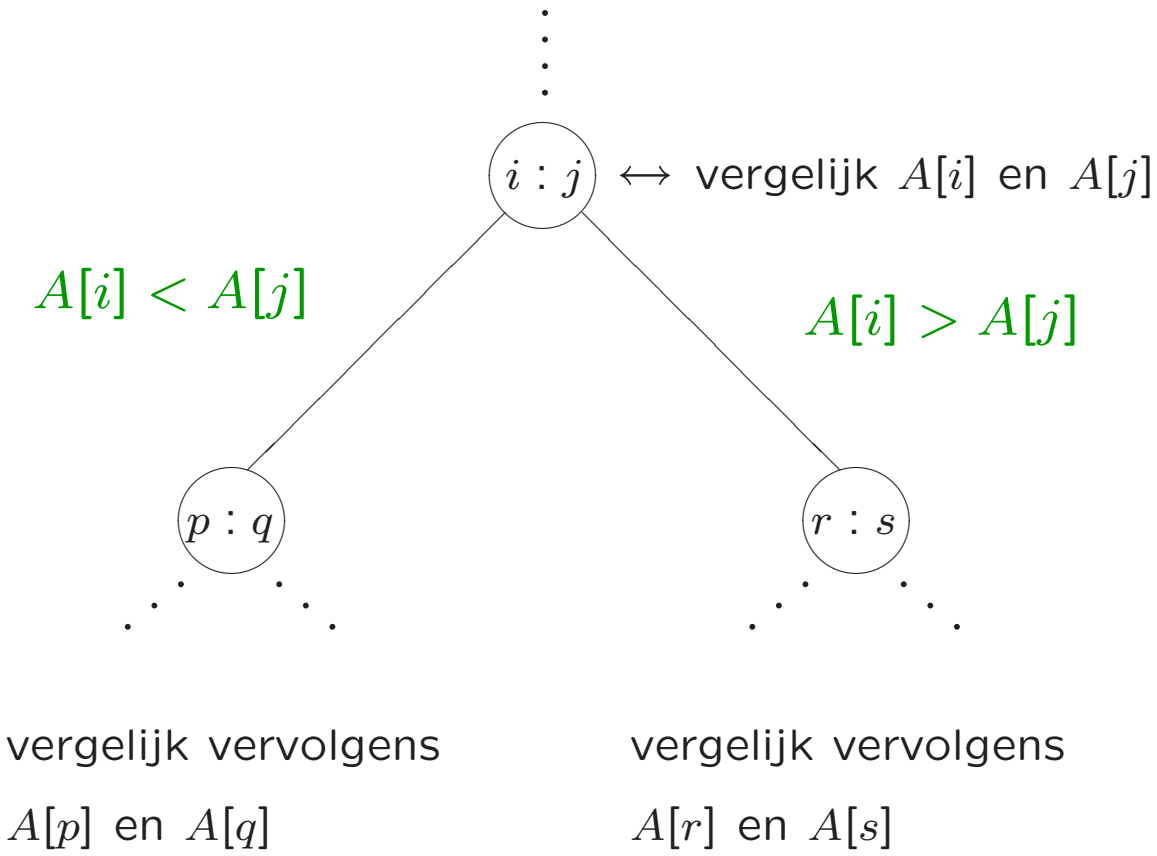
19 februari 2007

Selectie en adversary argument

algoritme gebaseerd op het doen van arrayvergelijkingen $A[i] < A[j]$

beslissingsboom*: binaire boom waarin de interne knopen corresponderen met arrayvergelijkingen en de bladeren/externe knopen met het eindresultaat; een pad vanaf de wortel naar een blad correspondeert met een executie van het algoritme

*alle $A[i]$ zijn verschillend



Beslissingsboom voor algoritmen gebaseerd op arrayvergelijkingen

Stelling

Elk algoritme dat de grootste (resp. kleinste) bepaalt uit een array met n elementen, en dat uitsluitend gebaseerd is op het doen van arrayvergelijkingen, doet ten minste $\lceil \lg n \rceil$ vergelijkingen in de worst case.

Merk op

We hadden voor het opsporen van het maximum/minimum al een scherpere ondergrens gevonden, namelijk $n - 1$.

Probleem

Gegeven n verschillende getallen, opgeslagen in een array A: $A[1], A[2], \ldots, A[n]$. Laat verder een geheel getal k met $1 \leq k \leq n$ gegeven zijn. Gevraagd de $A[i]$ die kleiner is dan precies $k-1$ andere $A[j]$'s.
M.a.w.: we zoeken de k-de in grootte.

Klasse van algoritmen

We bekijken algoritmen die uitsluitend gebaseerd zijn op het doen van arrayvergelijkingen.

De **complexiteit** van het **probleem**:

Ondergrens

Elk algoritme gebaseerd op arrayvergelijkingen doet voor het selectieprobleem in de worst case altijd ten minste $\lceil \lg n \rceil$ vergelijkingen (beslissingsboomargument).

Bovengrens

Het probleem kan worden opgelost door het array eerst aflopend te sorteren. De k-de in grootte is dan $A[k]$. Sorteren kan met $\Theta(n \lg n)$ vergelijkingen (zie later), dus selectie is $O(n \lg n)$.

Opmerkingen

Zowel de bovengrens als de ondergrens kunnen scherper. We bekijken hierna steeds (het precieze aantal vergelijkingen in) de worst case.

1. $k = 1$: het maximum, of
   $k = n$: het minimum.

   Het kan met $n - 1$ vergelijkingen. Dit is optimaal (al gezien)!

2. (Variant) Het maximum en minimum beide opsporen.

   Voor de hand ligt een algoritme met $2n - 3$ vergelijkingen, maar het kan met $\lceil \frac{3n}{2} \rceil - 2$. Dit is optimaal (adversary-argument, zie verderop).

   Voor deze variant moet natuurlijk $n \geq 2$ gelden.

3. $k = 2$: de op een na grootste (dus $n \geq 2$).

   Voor de hand ligt een algoritme met $2n - 3$ vergelijkingen, maar het kan met $n + \lceil \lg n \rceil - 2$ vergelijkingen (toernooimethode). Dit is optimaal (ook via een adversary-argument).

4. $k = \lceil \frac{n}{2} \rceil$: de mediaan (= de middelste in grootte).

   Er zit een gat tussen de best bekende ondergrens (ongeveer $2n$) en het best bekende algoritme. We kunnen dus niets over de optimaliteit van dat algoritme zeggen.

Een optimaal algoritme:

```
if A[1] > A[2] then // n ≥ 2
    grootste := A[1]; kleinste := A[2];
else
    grootste := A[2]; kleinste := A[1];
i := 3;                                  // voor het gemak: n even
while i < n do                           // anders kleine aanpassing
    if A[i] > A[i + 1] then                                        *
        gr := A[i]; kl := A[i + 1];
    else
        gr := A[i + 1]; kl := A[i];
    fi
    if gr > grootste then                                          *
        grootste := gr;
    fi
    if kl < kleinste then                                          *
        kleinste := kl;
    fi
    i := i + 2;
od
```

De toernooimethode is een optimaal algoritme voor het vinden van de op een na grootste.

Terminologie:

wedstrijd $\longleftrightarrow$ vergelijking;

winnaar $\longleftrightarrow$ grootste van de twee;

speler $\longleftrightarrow$ array-element; etcetera.

Complexiteit:

De toernooimethode doet $n + \lceil \lg n \rceil - 2$ arrayvergelijkingen. Dit is optimaal.

Geschikte implementatie:

met behulp van een soort heap (zie opgave 26).

Algoritme (voor het gemak met $n = 2^\ell$):

- Laat de spelers twee aan twee tegen elkaar spelen ($\frac{n}{2}$ wedstrijden).
- Laat de $\frac{n}{2}$ winnaars weer twee aan twee tegen elkaar spelen; de $\frac{n}{4}$ winnaars daarvan weer, etcetera.
- Herhaal dit totdat je één speler overhoudt: dit is de eindwinnaar, dus de grootste van allemaal.
- Er zijn nu $n - 1$ wedstrijden gespeeld, en er waren ℓ rondes nodig ($\ell = \lg n$).

Algoritme (vervolg):

- Nu moet de op een na grootste nog gevonden worden.
- Dit moet een van de ℓ spelers zijn die in het toernooi van de grootste verloren heeft, en wel de grootste van die ℓ.
- Het kost nog $\ell - 1$ vergelijkingen om deze te bepalen.

Als $n = 2^\ell$ doet de toernooimethode dus $n + \lg n - 2$ array-vergelijkingen in totaal. Dit is optimaal.

Opmerking: als n geen tweemacht is, is een kleine aanpassing nodig. Het aantal vergelijkingen wordt daarmee: $n + \lceil \lg n \rceil - 2$.

Een algoritme speelt een vraag-en-antwoord-spel tegen een adversary (tegenstander).

- algoritme: wil zo veel mogelijk informatie krijgen om met zo weing mogelijk vragen het probleem op te lossen
- adversary: wil zo weinig mogelijk informatie prijsgeven om ervoor te zorgen dat het algoritme zo veel mogelijk vragen moet stellen om de/een oplossing te vinden
- belangrijkste spelregel: consistentie. De adversary geeft alleen antwoorden die consistent zijn met eerder gegeven informatie.

Voorbeelden:

- datum raden (maand + dag)
- galgje: woord raden binnen 10 stappen
- X zoeken in een rij met n verschillende elementen

- de adversary beantwoordt de vragen van het algoritme volgens een of andere adversary-strategie
- deze strategie wil het algoritme dwingen zo veel mogelijk vragen te stellen
- de adversary bouwt zo tijdens de uitvoering van een algoritme een bad-case invoer op
- de adversary(-strategie) zorgt ervoor dat hij op elk moment een invoer kan geven die consistent is met de op de gestelde vragen gegeven antwoorden
- het aantal stappen (=vragen) dat elk *willekeurig* algoritme *ten minste* tegen de adversary(-strategie) moet uitvoeren om het juiste antwoord te krijgen geeft een ondergrens op de worst case complexiteit van het probleem.

Stelling

Elk algoritme gebaseerd op arrayvergelijkingen dat het minimum en het maximum van n (verschillende) waarden vindt, doet in het slechtste geval ten minste $\lceil \frac{3n}{2} \rceil - 2$ vergelijkingen.

Bewijs

De status van een array-element geven we aan met:

1. W: ≥ 1 keer gewonnen, nooit verloren
2. V: ≥ 1 keer verloren, nooit gewonnen
3. WV: ≥ 1 keer gewonnen en ≥ 1 keer verloren
4. N: nog nooit "gespeeld"

Adversary-strategie

wedstrijd x-y	uitslag	N	W	V	WV	type
N-N	x > y	-2	$+1$	$+1$	$--$	1
V-N	x < y	-1	$+1$	$--$	$--$	1
W-N	x > y	-1	$--$	$+1$	$--$	1
W-W	consistent	$--$	-1	$--$	$+1$	2
V-V	consistent	$--$	$--$	-1	$+1$	2
W-V	x > y	$--$	$--$	$--$	$--$	
WV-WV	consistent	$--$	$--$	$--$	$--$	
WV-N	x > y	-1	$--$	$+1$	$--$	1
WV-W	x < y	$--$	$--$	$--$	$--$	
WV-V	x > y	$--$	$--$	$--$	$--$	
	begin	n	0	0	0	
	eind	0	1	1	$n-2$	

Merk op dat de uitslagen altijd kunnen, want de waarde van een V (resp. W) mag altijd omlaag (resp. omhoog).

vergelijking x-y	actie van de adversary (constructie bad case)
N-N	kies "frisse" waarden zodat x>y
W-N	kies "frisse" waarde voor y (<x) je mag de waarde van x ook ongestraft verhogen; dat beïnvloedt eerdere uitslagen namelijk niet
W-V	x mag ongestraft verhoogd worden (of y verlaagd) zodat x > y wordt
W-W	laat winnen welk van de twee groter is
WV-V	y mag ongestraft verlaagd worden