Complexiteit

Uitwerkingen opgaven

opgave 27a
Voor het maximum zijn n mogelijke uitkomsten (kandidaten), en dan voor het minimum nog $n-1$. Dat zijn dus $n(n-1)$ mogelijke uitkomsten (dus tweetallen (max, min)). Merk op dat je wilt weten welke het maximum is en welke het minimum; het paar $(1,2)$ is dus echt iets anders dan het paar $(2,1)$, immers $(1,2)$ betekent dat $A[1]$ het maximum is en $A[2]$ het minimum en $(2,1)$ betekent juist dat $A[2]$ het maximum is en $A[1]$ het minimum. De uitkomsten zijn dus *geordende* paren indices. Omdat er $n(n-1)$ mogelijke uitkomsten zijn moet een beslissingsboom corresponderend met een algoritme voor dit probleem minstens $n(n-1)$ bladeren hebben, dus $b \geq n(n-1)$. Aangezien voor alle binaire bomen geldt dat $h \geq \lceil \lg b \rceil$, geldt voor dit probleem: $h \geq \lceil \lg n(n-1) \rceil = \lceil \lg n + \lg(n-1) \rceil \geq \lceil 2 \lg n \rceil = \Theta(\lg n)$. Ten slotte, omdat de hoogte van een beslissingsboom het aantal vergelijkingen van het corresponderende algoritme voorstelt hebben we: # vergelijkingen in de worst case is ten minste $\lceil \lg n(n-1) \rceil$, wat overigens niet zo'n scherpe ondergrens is.

opgave 31
In de adversary-argumenten hierna gebruiken we V om de vrager (het algoritme) en T om de tegenstander (adversary) aan te duiden. V vraagt steeds om de waarde van het i-de bit van de string L, en T antwoordt dan met $j \in \{0,1\}$.

Het geval $n = 2$.
In dit geval antwoordt T op elke vraag met 0. V heeft dan minstens twee vragen nodig.

Het geval $n = 3$.
Indien V eerst vraagt naar $L[1]$ neemt T de string 100 in gedachten. V moet dan de andere twee bits ook nog vragen. In feite is het probleem na de eerste vraag teruggebracht tot het geval $n = 2$, waar 2 vragen nodig waren in de worst case. Merk op dat als T hier 0 had geantwoord het algoritme nog maar 1 vraag nodig had gehad, namelijk vragen naar $L[2]$.
Indien V eerst naar $L[2]$ vraagt moet T geen 1 zeggen, want dan zou het algoritme meteen klaar zijn. T antwoordt dus 0, m.a.w. de string ziet er zo uit: $-0-$. Als T op elke volgende vraag 1 antwoordt (dus T neemt eigenlijk 101 in gedachten) wordt V gedwongen nog 2 vragen te stellen, dus in totaal 3.
Indien V eerst naar $L[3]$ vraagt neemt T 001 in gedachten (dit is eigenlijk hetzelfde geval als bij $L[1]$). In alle gevallen moet V dus minstens drie vragen stellen.

Het geval $n = 4$.
In dit geval kan V het antwoord vinden met slechts drie vragen met behulp van het volgende programma (ja/nee betekent hier dat er een/geen substring 00 is). Je kunt inzien dat het niet verstandig is om als eerste $L[1]$ (of $L[4]$) te vragen, want een adversary zou daarop 1 antwoorden, en dan had je het geval $n = 3$ terug. En daar heb je in de worst case 3 vergelijkingen voor nodig, dus samen 4. Ergo, vanuit het standpunt van een algoritme gezien is het verstandig om als eerste vraag naar $L[2]$ of $L[3]$ te vragen.

if $L[2] = 0$ **then**
  if $L[3] = 0$ **then** ja
  else if $L[1] = 0$ **then** ja
    else nee $(*)$ **fi fi**
else if $L[3] = 0$ **and** $L[4] = 0$ **then** ja
  else nee **fi fi**

Merk op dat het bij $(*)$ niet meer nodig is om $L[4]$ te testen omdat je toch al weet dat $L[3] = 1$.

Het geval $n = 5$.
Voor de eerste vraag heeft V vijf mogelijkheden. We zullen voor ieder van deze mogelijkheden laten zien dat T zodanig kan antwoorden dat V vijf vragen nodig heeft om met zekerheid een antwoord te geven.

- Als de eerste vraag $L[1]$ is antwoordt T 0, immers als T 1 zou zeggen is het probleem teruggebracht tot het geval $n = 4$, en dan is er een algorime dat maar 1+3=4 doet. Dus T zegt 0 en neemt daarmee de string $0----$ in gedachten; hiermee wordt bedoeld dat de keuze 0 of 1 voor de laatste vier plekken wordt uitgesteld tot latere vragen. Als nu naar $L[2]$ wordt gevraagd, denkt T aan $01---$, want als hij 0 zou antwoorden was het algoritme al klaar. We hebben nu het geval $n = 3$ terug, dus er zijn nu nog minstens 3 vragen nodig; in totaal wordt V voor dit geval dus gedwongen om ten minste vijf vragen te stellen.
Als (als tweede vraag) naar $L[3]$ wordt gevraagd antwoordt T met 1, dus hij denkt aan $0-1---$. In dat geval kan V worden gedwongen nog 3 vragen te stellen, dus vijf in totaal. (Als V $L[2]$ vraagt als derde vraag zegt T 1; als $L[4]$ als derde gevraagd wordt zegt T 0, want dan moet zeker $L[5]$ ook nog gevraagd worden, als $L[5]$ als derde wordt gevraagd antwoordt T uiteraard ook 0 (anders hoeft $L[4]$ niet meer gevraagd te worden.))
Als (als tweede vraag) naar $L[4]$ wordt gevraagd antwoordt T natuurlijk 0, anders hoeft $L[5]$ niet meer gevraagd te worden. Op elke volgende vraag wordt 1 geantwoord, ofwel, T neemt 01101 in gedachten. V heeft in dit geval eveneens in totaal minstens vijf vragen nodig.
Als (als tweede vraag) naar $L[4]$ wordt gevraagd antwoordt T met 1, dus neemt $0---1$ in gedachten. Het is eenvoudig in te zien dat ook in dit geval minstens vijf vragen nodig zijn. (Merk op dat als hier 0 was geantwoord, dus de string zou er zo uitzien: $0---0$, dan zou V alleen nog maar $L[2]$ en $L[4]$ hoeven te vragen, en dan het antwoord al weten.)

- Geheel analoog als eerst naar $L[5]$ gevraagd wordt.

- Als de eerste vraag $L[2]$ is, denkt T aan $-0---$. Immers als hij hier 1 zou antwoorden hoeft V niet meer naar $L[1]$ te vragen. Ook hier kunnen nu in totaal vijf vragen afgedwongen worden. Ga dit na.

- Geheel analoog als eerst naar $L[4]$ gevraagd wordt.

- Als de eerste vraag $L[3]$ is, denkt T aan $--1---$, en op de tweede vraag antwoordt T altijd met 0. Ook nu moeten er weer minstens vijf vragen gesteld worden. Ga dit na.

Nu het algemene geval. Dit wordt overigens niet gevraagd in de opgave. Als n een veelvoud van 3 plus 1 is (bijvoorbeeld 4 of 13) is er een slim algoritme: vraag de derde bit, als dit een 0 is vraag dan de tweede bit, en je hoeft vervolgens de eerste bit niet meer te vragen (zie het geval $n = 4$); als de derde bit 1 is, heb je na het vragen van de bits 1 en 2 de zaak teruggebracht (3 vragen in totaal) tot een situatie met $n - 3$ bits. Zo doorgaande kom je in dat geval bij 4 bits uit, en je hebt dan aan drie vragen voldoende. Dus dit spaart altijd (minstens) 1 vraag uit. Hieruit volgt dat voor n bits met n een veelvoud van 3 plus 1, er een algoritme is dat maximaal $n - 1$ vragen stelt. Voor andere n schijnt (?) een n-tal vragen nodig te zijn.