Complexiteit

Uitwerkingen

Opgave 14.

a.

$$\begin{aligned}
T(1) &= 3 \\
T(2) &= 3 + 1 \\
T(3) &= 3 + 1 + 2 \\
T(4) &= 3 + 1 + 2 + 3 \\
&\vdots \\
T(n) &= 3 + 1 + 2 + 3 + \cdots + (n-1) \\
&= 3 + \sum_{k=1}^{n-1} k \\
&= 3 + \frac{1}{2}(n-1)n
\end{aligned}$$

b. Laten we de uitkomsten binair schrijven (e.g. $2_{\text{dec}} = 10_{\text{bin}}$). De motivatie hiervan is niet alleen om het patroon duidelijker te maken (uit patroon $1, 3, 7, 15, 31, \ldots$ kan je op zich ook wel afleiden dat de functie waarschijnlijk $T(n) = 2^n - 1$ is), maar ook omdat het op deze manier duidelijk(er) wordt *waarom* dit patroon tot stand komt. Op het moment dat je niet alleen een duidelijk patroon hebt, maar ook aannemelijk kan maken dat dit patroon zich blijft voordoen voor willekeurige n is het niet meer nodig om de gevonden functie ook nog eens met inductie te gaan bewijzen.

$$\begin{aligned}
T(1_{\text{dec}}) &= 1_{\text{bin}} \\
T(2_{\text{dec}}) &= 1_{\text{bin}} \cdot 10_{\text{bin}} + 1_{\text{bin}} = 11_{\text{bin}} \\
T(3_{\text{dec}}) &= 11_{\text{bin}} \cdot 10_{\text{bin}} + 1_{\text{bin}} = 111_{\text{bin}} \\
T(4_{\text{dec}}) &= 111_{\text{bin}} \cdot 10_{\text{bin}} + 1_{\text{bin}} = 1111_{\text{bin}} \\
T(5_{\text{dec}}) &= 1111_{\text{bin}} \cdot 10_{\text{bin}} + 1_{\text{bin}} = 11111_{\text{bin}} \\
&\vdots \\
T(n) &= 111\ldots1_{\text{bin}} \ (n \text{ 1'en}) \\
&= (2^n - 1)_{\text{dec}}
\end{aligned}$$

c. $T(n)$ is alleen gedefinieerd voor $n = 2^k$, met $k \in \mathbb{N}_{>0}$.

$$\begin{array}{rll}
T(1) = 1 & = 1 & = 1 \cdot 1 \\
T(2) = 1 \cdot 2 + 2 & = 2 + 2 & = 2 \cdot 2 \\
T(4) = 2 \cdot (2 + 2) & = 4 + 4 + 4 & = 3 \cdot 4 \\
T(8) = 2 \cdot (4 + 4 + 4) + 8 & = 8 + 8 + 8 + 8 & = 4 \cdot 8 \\
T(16) = 2 \cdot (8 + 8 + 8 + 8) + 16 & = 16 + 16 + 16 + 16 + 16 & = 5 \cdot 16 \\
\vdots & \vdots & \vdots \\
T(n) = 2 \cdot \left(\frac{2}{n} + \frac{2}{n} + \ldots \frac{2}{n}\right) + n & = n + n + n + n + \cdots + n & = \log_2(2n) \cdot n
\end{array}$$

e. Voor $n > 0$ en $T(0) = 1$:

$$\begin{aligned}
T(0) &= 1 \\
T(1) &= 1(1 + 1) \\
T(2) &= 1(1 + 1)(2 + 1) \\
&\vdots \\
T(n) &= 1(1 + 1)(2 + 1) \cdots (n + 1) \\
&= 1 \cdot 2 \cdot 3 \cdots (n + 1) \\
&= (n + 1)!
\end{aligned}$$

Voor $n > 1$ en $T(1) = 1$:

$$\begin{aligned}
T(1) &= 1 \\
T(2) &= 1(2 + 1) \\
T(3) &= 1(2 + 1)(3 + 1) \\
&\vdots \\
T(n) &= 1(2 + 1) \cdots (n + 1) \\
&= 1 \cdot 3 \cdot 4 \cdots (n + 1) \\
&= \frac{(n + 1)!}{2}
\end{aligned}$$

f. We kijken eerst naar het patroon om een gok voor een functie af te leiden, en vervolgens bewijzen we die gok met een inductiebewijs.

$$\begin{array}{rll}
T(1) = T(3^0) = 0 & & \\
T(3) = T(3^1) = 3 \cdot 0 + 2 & & = 2 \\
T(9) = T(3^2) = 3 \cdot 2 + 2 & & = 8 \\
T(27) = T(3^3) = 3 \cdot 8 + 2 & & = 26 \\
\vdots & & \\
T(n) = n - 1 & &
\end{array}$$

Bewijs: Stel $T(3^k) = 3^k - 1$, dan geldt ook $T(3^\ell) = 3^\ell - 1$ voor $\ell = k + 1$.

$$\begin{aligned}
T(3^\ell) &= T\left(3^{k+1}\right) \\
&= 3 \cdot T\left(3^{k+1}/3\right) + 2 && \text{(recursieve def. van } T(n)\text{)} \\
&= 3 \cdot T(3^k) + 2 \\
&= 3 \cdot (3^k - 1) + 2 && \text{(aaname voor } T(3^k) \text{ hierboven)} \\
&= 3 \cdot 3^k - 3 + 2 \\
&= 3^{k+1} - 1 \\
&= 3^\ell - 1
\end{aligned}$$ □

g. We kijken eerst naar het patroon en geven vervolgens een iets concreter bewijs met behulp van inductie.

$$\begin{aligned}
T(4^0) &= 0 \\
T(4^1) &= 0 + 4^{1/2} && = 1 \cdot 4^{1/2} \\
T(4^2) &= 2 \cdot 4^{1/2} + 4^{2/2} && = 2 \cdot 4^{2/2} \\
T(4^3) &= 2 \cdot 2 \cdot 4^{2/2} + 4^{3/2} && = 3 \cdot 4^{3/2} \\
T(4^4) &= 2 \cdot 3 \cdot 4^{3/2} + 4^{4/2} && = 4 \cdot 4^{4/2} \\
&\vdots \\
T(n) &= \log_4(n)\sqrt{n}
\end{aligned}$$

Bewijs: Stel $T(4^k) = \log_4(4^k)\sqrt{4^k}$, dan geldt ook $T(4^\ell) = \log_4(4^\ell)\sqrt{4^\ell}$ voor $\ell = k + 1$.

$$\begin{aligned}
T(4^\ell) &= T(4^{k+1}) \\
&= 2 \cdot T(4^{k+1}/4) + \sqrt{4^{k+1}} && \text{(recursieve definitie van } T(n)\text{)} \\
&= 2 \cdot T(4^k) + \sqrt{4^{k+1}} \\
&= 2 \cdot \log_4(4^k)\sqrt{4^k} + \sqrt{4^{k+1}} && \text{(aaname voor } T(4^k) \text{ hierboven)} \\
&= 2 \cdot k \cdot \sqrt{4^k} + \sqrt{4^{k+1}} \\
&= 2 \cdot k \cdot 2^k + 2^{k+1} \\
&= k \cdot 2^{k+1} + 2^{k+1} \\
&= (k+1) \cdot 2^{k+1} \\
&= \ell \cdot 2^\ell \\
&= \log_4(4^\ell) \cdot \sqrt{4^\ell}
\end{aligned}$$ □

Opgave 15.

a. Voor $n = 2$ hebben we maar 2 elementen, en kunnen we met één vergelijking $A[1] < A[2]$ bepalen welke de grootste is. Voor $n = 2^k$ met $k \geq 2$ splitsen we de rij in tweeën en bepalen de grootste en de op een na grootste van deze twee gehalveerde rijen, wat $2G(\frac{n}{2})$ geeft. Om het uit het grootste (en een na

grootste) element van de linker en rechter helft de grootste element van de hele rij te vinden kost nog eens twee vergelijkingen (+2).
b.

$$
\begin{aligned}
G(2) &= 1 \\
G(4) &= 2 \cdot 1 + 2 && = 2 + 2 \\
G(8) &= 2 \cdot (2 + 2) + 2 && = 4 + 4 + 2 \\
G(16) &= 2 \cdot (4 + 4 + 2) + 2 && = 8 + 8 + 4 + 2 \\
G(32) &= 2 \cdot (8 + 8 + 4 + 2) + 2 && = 16 + 16 + 8 + 4 + 2 \\
&\vdots \\
G(n) &= \tfrac{n}{2} + (\tfrac{n}{2} + \tfrac{n}{4} + \tfrac{n}{8} + \cdots + 2) \\
&= \tfrac{n}{2} + \sum_{k=1}^{\log_2(n)-1} 2^k \\
&= \tfrac{n}{2} + 2^{\log_2(n)} - 2 \\
&= \tfrac{n}{2} + n - 2 \\
&= \tfrac{3n}{2} - 2
\end{aligned}
$$

Bewijs met inductie: Voor $G(2)$ klopt de formule: $\frac{3 \cdot 2}{2} - 2 = 1$. Voor $n \geq 4$ willen we aantonen dat onder de aanname dat de formule klopt voor $G(\frac{n}{2})$, deze ook klopt voor $G(n)$.

$$
\begin{aligned}
G(n) &= 2 \cdot G(\tfrac{n}{2}) + 2 && \text{(recursieve definitie van } G(n)) \\
&= 2 \cdot \tfrac{3n/2}{2} + 2 && \text{(aaname voor } G(\tfrac{n}{2}) \text{ hierboven)} \\
&= 2 \cdot \tfrac{3n}{2} + 2 && \square
\end{aligned}
$$

Opgave 22.

a. Het volgende algoritme doet in het beste geval 1 vergelijking (als $A[1] = A[2]$), en in het slechtste geval $\frac{1}{2}n(n-1)$ vergelijkingen (als alle elementen verschillend zijn, of als alleen de laatste twee elementen gelijk aan elkaar zijn).

```
i := 1
while i ≤ n do
    j := i + 1
    while j ≤ n do
        if A[i] = A[j] then
            return False
        end
        j := j + 1
    end
    i := i + 1
end
return True
```

b. Er zijn in totaal $\frac{1}{2}n(n-1)$ paren $(A[i], A[j])$ om te vergelijken. We kijken niet naar de paren waar $i = j$, en tijdens het zoeken van twee elementen die gelijk aan elkaar zijn is het paar $(A[i], A[j])$ equivalent aan het paar $(A[j], A[i])$.

Met een adversary argument kan je aantonen dat je in het slechtste geval ook inderdaad al deze paren moet vergelijken.
c. Sorteer eerst de array, dit kan met $O(n \log n)$ vergelijkingen. In een gesorteerde array zullen elementen die gelijk aan elkaar zijn naast elkaar staan, dus vervolgens kunnen we de array aflopen en hoeven we alleen buren met elkaar te vergelijken, wat $O(n)$ vergelijkingen kost. In totaal is de complexiteit dus $O(n \log(n) + n) = O(n \log n)$.

Opgave 23.

a. Het volgende is min of meer bubblesort op 3 elementen.

$A[1] := a$
$A[2] := b$
$A[3] := c$
if $A[1] > A[2]$ **then**
| swap($A[1]$, $A[2]$)
end
if $A[2] > A[3]$ **then**
| swap($A[2]$, $A[3]$)
end
if $A[1] > A[2]$ **then**
| swap($A[1]$, $A[2]$)
end
return $A[2]$

b. Alle elementen gelijk aan elkaar ($a = b = c$), twee elementen gelijk aan elkaar ($a = b \neq c$), of geen elementen gelijk aan elkaar ($a \neq b \neq c$).
c. Dit algoritme doet voor alle gevallen exact 3 vergelijkingen.
d. In het algemeen zijn er in het slechtste geval 3 vergelijkingen nodig.

Opgave 27.

a. Gegeven een array A van lengte n, we zijn op zoek naar een geordend paar $(\min(A), \max(A))$. Er zijn $n(n-1)$ mogelijke paren. Een beslissingsboom voor alle mogelijke uitkomsten heeft dus (minimaal) $n(n-1)$ bladeren. De hoogte van een binaire boom met $n(n-1)$ bladeren is (minstens) $\lceil \log_2 n(n-1) \rceil \in \Theta(\log n^2) = \Theta(2 \log n) = \Theta(\log n)$.
b. Gegeven een array A van lengte n, we zijn op zoek naar een (ongeordend) 5-tuple van de 5 grootste elementen. Dit zijn in totaal

$$\binom{n}{5} = \frac{n!}{5!(n-5)!}$$

mogelijke tuples. Een binaire boom met $\binom{n}{5}$ bladeren heeft (minstens) hoogte $\lceil \log_2 \binom{n}{5} \rceil$. Dit is in ieder geval minder dan lineair ($O(n)$).

$$\binom{n}{5} \leq 2^n \text{ dus } \log_2 \binom{n}{5} \leq \log_2 2^n = n$$

c. We voegen twee gesorteerde rijen A en B (met lengtes n en m) samen tot een gesorteerde rij C (van lengte $n + m$). In de nieuwe rij C komen elementen van A en B nog steeds in dezelfde volgorde voor als ze in A en B voorkomen.

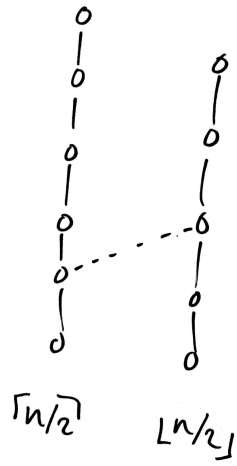
Dat wil zeggen: voor alle elementen $A[i]$ en $A[j]$, met $i < j$, zullen we voor de corresponderende elementen $C[k] = A[i]$ en $C[\ell] = A[j]$ nog steeds hebben dat $k < \ell$. Om te kijken hoeveel mogelijke rijen C er gemaakt kunnen worden onder deze conditie hoeven we alleen te kijken naar op hoeveel verschillende combinaties van plekken in C we de elementen van A kunnen zetten. Het gaat hier dus alleen om welke plekken, niet om de volgorde, want de volgorde van de elementen in A ligt al vast. Dit geeft

$$\binom{n+m}{n} = \binom{n+m}{m}$$

mogelijke rijen C. De diepte van een binaire boom met $\binom{n+m}{n}$ bladeren is (minstens) $\log_2 \binom{n+m}{n}$.

Opgave 33.

De adversary heeft een graaf G van ongeveer de volgende vorm in gedachten, met of precies één tak tussen de linker en rechter subgraaf G_L en G_R, of geen.



Om de lower bound van $\lceil n/2 \rceil \cdot \lfloor n/2 \rfloor$ aan te tonen kunnen we zelfs al (gratis) aan het algoritme laten weten dat G_L en G_R elk samenhangend zijn. Het doel van het algoritme is om er achter te komen of de hele graaf samenhangend is, wat alleen kan worden vastgesteld door een tak tussen G_L en G_R te vinden. Het algoritme vraagt voortdurend "zit er een tak tussen knoop v en w", waar v in de linker helft en w in de rechter helft. Er zijn in totaal $\lceil n/2 \rceil \cdot \lfloor n/2 \rfloor$ paren van zulke knopen. In het geval dat er precies één tak tussen G_L en G_R zit kan de adversary het zo inrichten dat dit het laatste paar is uit de $\lceil n/2 \rceil \cdot \lfloor n/2 \rfloor$ gevraagde paren. Na $\lceil n/2 \rceil \cdot \lfloor n/2 \rfloor - 1$ vragen aan de adversary weet het algoritme dus nog niet of G samenhangend is of niet.