Antwoorden: Algorithms & Data Structures

1. (a) Het sorteren van een array ter lengte 1 of 2 kost 1 vergelijking. Voor arrays met meer elementen vinden er 3 recursieve aanroepen plaats, ieder met een array ter lengte $\frac{2}{3}$ van de originele lengte. De niet-recursieve kosten in dit geval zijn 1, daar er 1 vergelijking van A-elementen plaatsvindt. Dit leidt tot de volgende recurrente betrekking voor $W(n)$:

$$\begin{array}{lcll} W(n) & = & 1 & \text{voor } n < 3 \\ W(n) & = & 3 \cdot W(\frac{2n}{3}) + 1 & \text{voor } n \geq 3 \end{array}$$

   Voor het bepalen van de worst-case tijdscomplexiteit passen we het Master theorema toe. Er volgt dat $a = 3$, $b = \frac{3}{2}$, $f(n) = 1$, en $E = \log a / \log b = \log 3 / \log \frac{3}{2}$. Aangezien $f(n) \in O(n^{E-\epsilon})$ voor bijvoorbeeld $\epsilon = \frac{1}{2}$ is de eerste case van het Master theorema van toepassing, en geldt dat $W(n) \in \Theta(n^E)$.

   (b) Merk op dat $E = \log 3 / \log \frac{3}{2} = {}^{1.5}\log 3 > 2$ omdat $1.5^2 = 2.25$. Dus de worst-case tijdscomplexiteit van *sort* is significant slechter is dan die van de andere sorteer-algoritmen; we geven dus nooit de voorkeur aan *sort*.

2. Stel x is de node die k bevat. Er zijn 2 mogelijkheden:

   - x heeft linkerkinderen. Ga dan 1 keer naar linksbeneden, en blijf dan vervolgens naar rechtsbeneden gaan zolang als dat kan. Je hebt dan de grootste waarde kleiner dan k.
   - x heeft geen linkerkinderen. Ga eerst net zolang rechtsomhoog tot dat niet meer kan, en ga dan linksomhoog. Kun je niet linksomhoog, dan was k het kleinste element en lever je *null* op; anders bereik je zo de grootste waarde kleiner dan k.

   Het algoritme:

```
def bstPred(x):
    if x.left != null :
        x = x.left
        while x.right != null :
            x = x.right
        return x

    else :
        y = x.parent
        while y != null and y.left == x :
            x =y
            y = y.parent
        return y
```

3. We passen dynamisch programmeren toe. Uiteindelijk moeten we het bovenste vakje vinden met het meeste aantal punten.

```
def maxpoints(p,n):

    R=[[0 for j in range(n+1)] for i in range(n+2)]

    for j in range(2,n+1):
        for i in range(1,n+1):
            R[i][j] = max(R[i-1][j-1]+p[i-1][j-1][R],
                          R[i+1][j-1]+p[i+1][j-1][R])
```

```
mx=R[1][n]
for i in range(2,n+1):
    mx=max(mx,R[i][n])
return mx
```

De complexiteit van dit algoritme is $\Theta(n^2)$.

Antwoorden: Discrete Mathematics

4. Gebruik het Euclidische Algoritme om $\gcd(1000, 444)$ te bepalen:

```
1000 = 2*444 + 112
 444 = 3*112 + 108
 112 = 1*108 +   4
 108 = 27* 4 +   0
```

Dus $\gcd(1000, 444) = 4$. We weten ook dat $\gcd(1000, 444) = \min\{1000s + 444t > 0 \mid s, t \in \mathbb{Z}\}$. Omdat $2 < 4$, kan de vergelijking $1000s + 444t = 2$ geen oplossing in $\mathbb{Z}$ hebben.

5. (a) Het karakteristieke polynoom van de bijhorende homogene recurrente betrekking is $x^2 - 10x + 21 = (x-3)(x-7)$. De wortels hiervan zijn $x_1 = 3$ en $x_2 = 7$. Dus de algemene oplossing van de homogene recurrente betrekking is

$$a_n^{(h)} = c_1 3^n + c_2 7^n \,.$$

Als particuliere oplossing kiezen wij

$$a_n^{(p)} = An3^n \,,$$

omdat $A3^n$ niet lineair onafhankelijk zou zijn. Inzetten in de inhomogene recurrente betrekking levert $An3^n - 10(n-1)3^{n-1} + 21(n-2)3^{n-2} = 60 \cdot 3^n$ voor alle n, dus $An3^n(1 - \frac{10}{3} + \frac{7}{3}) + A3^n(\frac{10}{3} - \frac{14}{3}) = 60 \cdot 3^n$, dus $A = -45$, en de algemene oplossing voor de inhomogene recurrente betrekking is

$$a_n = c_1 3^n + c_2 7^n - 45n3^n \,.$$

Nu hebben we $a_0 = 2 = c_1 + c_2$, en $a_1 = -5 = 3c_1 + 7c_2 - 135$. Hieruit volgt dat $c_1 = -29$ en $c_2 = 31$, en het antwoord is

$$a_n = -29 \cdot 3^n + 31 \cdot 7^n - 45n3^n \,.$$

(b) Laat a_n^k het aantal strings (met de genoemde eigenschappen) zijn wat eindigt op een k, dan

$$a_n = a_n^0 + a_n^1 + a_n^2 \,.$$

Verder, $a_n^0 = a_{n-1}$, $a_n^1 = a_{n-1}^1 + a_{n-1}^2$, en $a_n^2 = a_{n-1}^1 + a_{n-1}^2$. Dat geeft

$$a_n = a_{n-1} + (a_{n-1} - a_{n-1}^0) + (a_{n-1} - a_{n-1}^0) = 3a_{n-1} - 2a_{n-2} \,.$$

Tenslotte, $a_1 = 3^1 = 3$, $a_2 = 3^2 - 2$(voor 01 en 02) $= 7$, $a_3 = 3^3 - 3 \cdot 4$(voor $01X$ en $X01$ en $02X$ en $X02$) $= 15 (= 3 \cdot 7 - 2 \cdot 3)$.

6. Als we $d \geq 6$ aannemen, dan is $\sum_{v \in V} d(v) \geq 6n$, en omdat $2m = \sum_{v \in V} d(v)$, concluderen wij $m = \frac{1}{2} \sum_{v \in V} d(v) \geq 3n$. Maar voor planaire grafen geldt $m \leq 3n - 6$. Tegenspraak, dus $d < 6$.

7. (a) Neem aan dat er twee verschillende MST's bestaan, T en T'. Dan bestaat er een lijn $e = \{u, v\} \in T \setminus T'$. Laat $P_{T'}(u, v)$ de (unieke) pad tussen u en v in T' zijn, en laat $C(e)$ de cut zijn die geïnduceerd wordt door e te verwijderen uit T (wil zeggen, als $C_1 \subset V$ en $C_2 = V \setminus C_1$ de componenten zijn van $T \setminus \{e\}$, dan is $C(e)$ de verzameling van *alle* lijnen van G tussen C_1 en C_2). Omdat nu $P_{T'}(u, v)$ twee punten uit C_1 en C_2 verbindt, moet er een lijn $f \in P_{T'}(u, v)$ bestaan zodat $f \in C(e)$. Maar nu zegt de 'path condition' voor T' dat $d_f \leq d_e$, en de 'cut condition' voor T dat $d_e \leq d_f$, dus $d_e = d_f$. Tegenspraak, en daarom bestaan er geen twee verschillende MST's.

   (b) Een driehoek met lijn lengtes $1, 1$ en 2 is een tegenvoorbeeld.

8. De aantal mogelijkheden om de 24 flessen van de éne merk te verdelen over de 5 controllers is de coëfficiënt van de term x^{24} in

$$f(x) = (x^2 + x^3 + \cdots)^5 .$$

Aangezien $f(x) = x^{10}(1 - x)^{-5}$, zoeken we dus de coëfficiënt van x^{14} in $(1 - x)^{-5}$. Dat is $(-1)^{14}\binom{-5}{14} = \binom{18}{14}$. De aantal mogelijkheden om de 24 flessen van de andere merk te verdelen over de 5 controllers is de coëfficiënt van de term x^{24} in

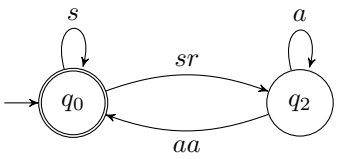
$$g(x) = (x^3 + x^3 + \cdots)^5 .$$

Nu is $g(x) = x^{15}(1-x)^{-5}$, en we zoeken de coëfficiënt van x^9 in $(1-x)^{-5}$. Dat is $(-1)^1 3\binom{-5}{9} = \binom{13}{9}$. Het antwoord is dus

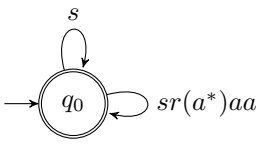
$$\binom{18}{14} \cdot \binom{13}{9} .$$

Antwoorden: Languages & Machines

9. (a) Elimineer eerst q_1 en q_3 (ik doe het tegelijk, het is onafhankelijk)



Nu q_2 elimineren:

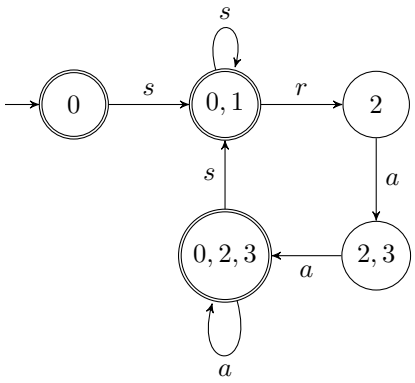


Nu kun je de reguliere expressie aflezen: $(s \cup sra^*aa)^*$.

(b)

	λ	a	s	r
0	$\{0\}$	$\emptyset$	$\{0, 1\}$	$\emptyset$
1	$\{0, 1\}$	$\emptyset$	$\{0, 1\}$	$\{2\}$
2	$\{2\}$	$\{2, 3\}$	$\emptyset$	$\emptyset$
3	$\{2, 3\}$	$\{0, 2, 3\}$	$\emptyset$	$\emptyset$

(c) De subset constructie levert de volgende onvolledige DFA op (we laten transities naar $\emptyset$ weg):



10. (a) Regulier, L_1 bevat precies alle drievouden van a's. Dit kun je met een reguliere expressie definiëren: $L_1 = \mathcal{L}\Big((aaa)^*\Big)$.

    (b) Niet-regulier, bewijs met de pompstelling: Laat $k > 0$ willekeurig gegeven zijn, kies $z = a^k b a^{2k}$, dan $|z| \geq k$ en $z \in L_2$. Laat u, v, w willekeurig gegeven zijn, zodanig dat $z = uvw$, $|uv| \leq k$ en $|v| > 0$. Dan moet $u = a^m$, $v = a^n$ en $w = a^{k-m-n}ba^{2k}$ zijn voor zekere $m \geq 0$ en $n > 0$. Kies $i = 2$, dan $uv^iw = a^{k+n}ba^{2k} \notin L_2$, omdat $n \neq 0$. We hebben laten zien dat $\forall k > 0 : \exists z : |z| \geq k \wedge z \in L_2 \wedge \forall u, v, w : |uv| \leq k \wedge |v| > 0 \wedge z = uvw \rightarrow \exists i : uv^iw \notin L_2$. Dus L_2 is niet regulier ("contrapositie" van de pompstelling).

    (c) $L_3 = L_{3a} \cap \overline{L_{3b}}$, (ofwel: $L_{3a} - L_{3b}$) met

$$L_{3a} = \mathcal{L}\Big((b+c)^*.(a(b+c+\lambda)^4)^*.(b+c)^*\Big)$$

$$L_{3b} = \mathcal{L}\Big((a+b+c)^*.(abcba).(a+b+c)^*\Big),$$

    die dus beide regulier zijn. Reguliere talen zijn gesloten onder complement en doorsnede (ofwel: verschil) en dus is L_3 ook regulier.

    (d) Stel dat $L_4 \cup L_5$ regulier is. Iedere eindige taal is regulier; L_5 is eindig, dus is ook $L_5 - L_4$ eindig, en dus ook regulier. Reguliere talen zijn gesloten onder verschil (doorsnede en complement), dus dan is ook $L_4 = (L_4 \cup L_5) - (L_5 - L_4)$ regulier. Dit is een tegenspraak met het gegeven dat L_4 niet regulier is. Dus $L_4 \cup L_5$ is ook niet regulier.