DMA — Ugeopgave 3

Jonas Kramer<snm401>
Lauritz Andersen<>
Helga Rykov Ibsen <mcv462>

25. oktober 2021

Del 2

```
function DetectRepeat(X, n):
1.  val = FALSE
2.  for i = 0 to n-1 do
3.      target = X[i]
4.      for j = i + 1 to n - 1 do
5.          if X[j] = target then
6               val = TRUE
7.   return val
```

Vi starter med at udregne antal skridt som denne algoritme gennemløber: Vi

Linjetal	Skridt	Max gange
1	c_1	1
2 (ydre for-løkke)	c_2	n
3 (ydre for-løkke)	c_2	n
4 (indre for-løkke)	c_3	$(n*(n-1))/2$
5 (indre for-løkke)	c_3	$(n*(n-1))/2$
6 (indre for-løkke)	c_3	$(n*(n-1))/2$
7	c_4	1

beregner nu køretiden T(n):

$$T(n) = c_3\frac{n^2-n}{2} + c_2 n + c_1 + c_4$$

Vi skal nu finde et k sådan at $c_1 n^k \leq T(n) \leq c_2 n^k$. Vi ser at hvis k = 2, så vil der gælde, at vi kan gange en konstant c_1, fx 0.1, og en anden konstant c_2, fx 100, således at $c_1 n^2 \leq T(n) \leq c_2 n^2$, hvorfor vi har at T(n)=$\Theta(n^2)$.

Del 2

(a)

For at finde funktioner $f_a(n)$, $f_b(n)$ og $f_c(n)$ således at $a_n = f_a(n)$, $b_n = f_b(n)$ og $c_n = f_c(n)$ starter vi med at betragte de tre talfølger.

1. $a_0 = 2, a_1 = 200, a_2 = (a^{2-2} = a^0) = 2, a_3 = (a^{3-2} = a^1) = 200, a_4 = (a^{4-2} = a^2) = 2$

   Det ser ud til at talrækken a består af kun to tal, 2 og 200, som gentager sig selv skiftevis mellem

   (n mod 2)=0 og (n mod 2)=1.

   Hvis n er lige vil (n mod 2) give 0, og hvis n er ulige, så vil (n mod 2) give 1. sådan at:

   (4 mod 2)=0, (5 mod 2)=1, (6 mod 2)=0, (7 mod 2)=1

   2, 200, 2, 200...

   Funktionen $f_a(n)$ må derfor være $f_a(n) = 2 * 100^{(nmod2)}$, eller:

   $$f_a(n) = 2 * 100^0 = 2 \text{ for lige tal}$$

   $$f_a(n) = 2 * 100^1 = 200 \text{ for ulige tal}$$

   Funktionen $f_a(n)$ må derfor være en konstant funktion, som svinger mellem to konstanter 2 og 200.

2. Vi starter med at betragte talrækken b:

   $$b_0 = 1, b_1 = (3^{1-1}) = 3^0, b_2 = (3^{2-1}) = 3^1, b_3 = (3^{3-1}) = 3^2, b_4 = (3^{4-1}) = 3^3...$$

   Eller:

   1, 3, 9, 27, 81...

   Det ser ud til at funktionen $f_b(n)$ er en eksponentielfunktion med grundtallet 3:

   $$f_b(n) = 3^n.$$

3. Vi betragter den tredje talfølge c:

   $$c_0 = 1, c_1 = c_0 + 2^1, c_2 = c_1 + 2^2, c_3 = c_2 + 2^3...$$

   Eller:

   1, 3, 7, 15...

   Lad os betragte talfølgen for 2^n:

   1, 2, 4, 8, 16...

Hvis vi sammenligner de to talfølger, så ser det ud til, at tallene i c-rækken svarer til $c_n = 2^{n+1} - 1$. Funktionen for $f_c(n)$ må derfor være:

$$f_c(n) = 2^{n+1} - 1$$

(b)

Vi har R4 der fortæller os at logaritmer vokser hurtigere end konstanter, R5 fortæller os at polynomier vokser hurtigere end logaritmer, og til slut fortæller R6 os at eksponentielle funktioner vokser hurtigere end polynomier, ergo vil vores konstante funktion $f_a(n)$ vokse langsomst. Vi har ligeledes jf. R8 at eksponentielle funktioners vækst afhænger af deres base, hvorfor $f_b(n)$ vokser hurtigere end $f_c(n)$. Ergo har vi: $f_a(n) < f_c(n) < f_b(n)$.

Del 3

1. Vi starter med at omskrive sumformlen efter theorem 4, hvor vi omskriver summen som summen af to summer og sætter konstenten 3 udenfor den første sum:

$$\sum_{k=0}^{n}(3 * 2^k + 1) = 3 * \sum_{k=0}^{n} 2^k + \sum_{k=0}^{n} 1 \tag{1}$$

2. Vi anvender formlen for summen af kvotientrækken til (1) ved at bruge teorem 3 for k=0 og c$\neq$ 1:

$$3 * \sum_{k=0}^{n} 2^k + \sum_{k=0}^{n} 1 = 3 * \frac{2^{n+1} - 1}{2 - 1} + (n + 1) \tag{2}$$

3. Vi reducerer udtrykket i (2) og får:

$$3 * (2^{n+1} - 1) + n + 1 = 3 * 2^{n+1} + n - 2 \tag{3}$$

Del 4

(1)

Vi benytter Euklids algoritme og beregner GCD(a, b) samt antallet af trin:

Trin	GCD(3,2)	GCD(5,3)	GCD(8,5)	GCD(13,8)
1	$3 = 1 * 2 + 1$	$5 = 1 * 3 + 3$	$8 = 1 * 5 + 3$	$13 = 1 * 8 + 5$
2	$2 = 2 * 1 + 0$	$3 = 1 * 2 + 1$	$5 = 1 * 3 + 2$	$8 = 1 * 5 + 3$
3		$2 = 1 * 2 + 0$	$3 = 1 * 2 + 1$	$5 = 1 * 3 + 2$
4			$2 = 1 * 2 + 0$	$3 = 1 * 2 + 1$
5				$2 = 1 * 2 + 0$
Antal trin indsat i figur1	2	3	4	5

(2)

Ifølge figur 1, er det højeste antal trin for $t_1 = 1, t_2 = 1, t_3 = 2, t_4 = 2, t_5 = 3, t_6 = 2, t_7 = 3, t_8 = 4, t_9 = 3, t_{10} = 3, t_{11} = 4, t_{12} = 4, t_{13} = 5, t_{14} = 4, t_{15} = 4$.

(3)

Hvis vi antager at $(a_6, b_6) = (21, 13)$, så kan vi finde GCD(a_k, b_k) for k=5, 4, 3 og 2 blot ved at bruge figur 1:

$(a_5, b_5) = (13, 8)$
$(a_4, b_4) = (8, 5)$
$(a_3, b_3) = (5, 3)$
$(a_2, b_2) = (3, 2)$

(4)

Vi finder (a_7) som:

$$(a_7) = (a_6 + b_6) = (21 + 13) = 34$$

og (b_7) er nemt at indentificere ved at følge mønsret, sådan at $(b_7) = (a_6)$. Med andre ord er $(a_7, b_7) = (34, 21)$ og $(a_8, b_8) = (55, 34)$.

Følgen defineret ved $F_0 = 1, F_1 = 2, F_k$=maxa_k, b_k for k>1 genererer en generalisering af Fibonacci-tallene.

(5)

Hvis vi udtrykker n som enten a eller b i figur 1, så kan vi se at t_n stiger med 1 hver gang at a eller b antager værdien af det næste Fibonacci:

n	1	2	**3**	4	**5**	6	7	**8**	9	10	11	12	**13**	14	15
t_n	1	1	**2**	2	**3**	2	3	**4**	3	3	4	4	**5**	4	4

Eftersom det forholder sig sådan at n antager den næste Fibonacci tal før at t_n stiger med 1, må det betyde at t_n vokser langsommere end linært. I modsat fald, så ville t_n stige med 1 hver gang n stiger med 1.

Det må altså betyde at t_n er bundet af linær vækst, dvs.:

$$t_n = O(n)$$