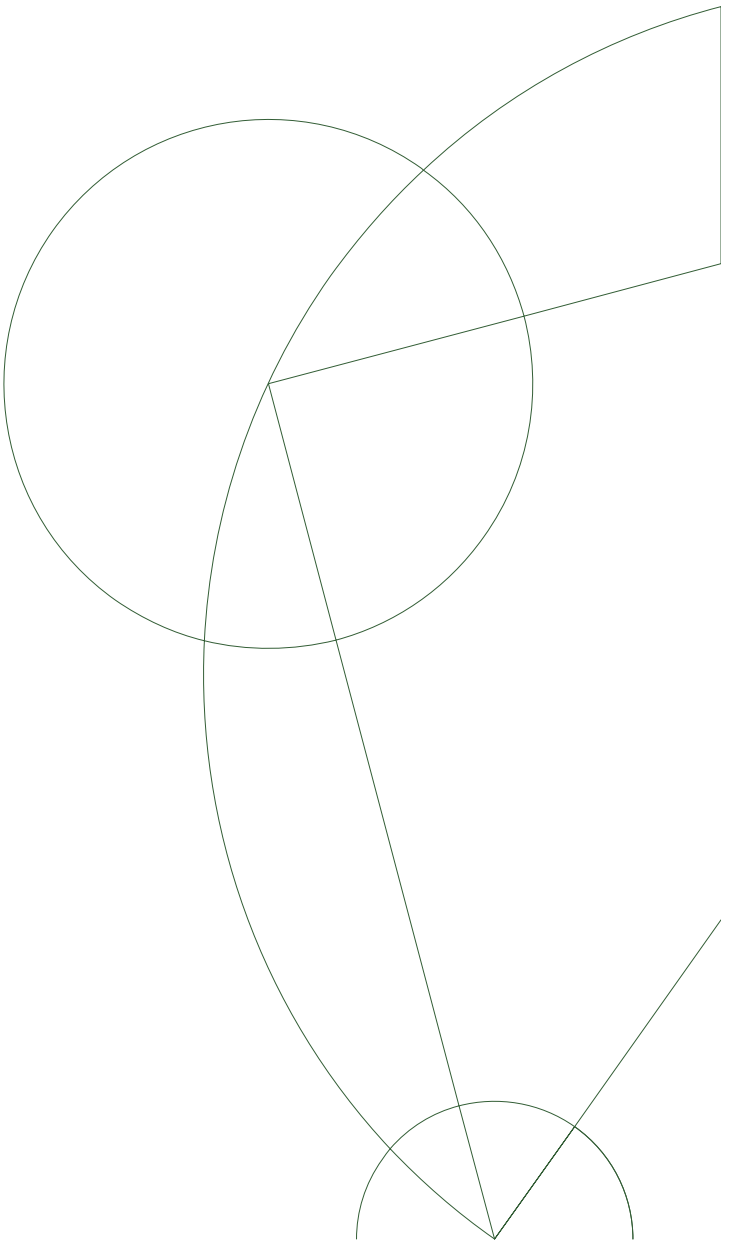
Diskret Matematik og Algoritmer

Aflevering 5g

Adam Ingwersen,
Peter Friborg,
Aske Fjellerup

Datalogisk Institut
Københavns Universitet

October 10, 2016



Del 1

(1)

Her udregnes GCD(Greatest Common Divisor) for tallene (8,5) hhv. (13,8), efter samme notation som anført på s.23 i KBR.

- **GCD(8,5):**
- $8 = 1 \cdot 5 + 3$
- $5 = 1 \cdot 3 + 2$
- $3 = 1 \cdot 2 + 1$
- $2 = 2 \cdot 1 + 0$
- $\rightarrow GCD(8, 5) = 1$
- **GCD(13,8):**
- $13 = 1 \cdot 8 + 5$
- $8 = 1 \cdot 5 + 3$
- $5 = 1 \cdot 3 + 2$
- $3 = 1 \cdot 2 + 1$
- $2 = 2 \cdot 1 + 0$
- $\rightarrow GCD(13, 8) = 1$

Tallene i figur 1 angiver antallet af nødvendige operationer for at bestemme GCD for en bestemt kombination af tal. GCD(8,5) tager 4 operationer - og derved indsættes tallet 4 i den celle, hvor 5. række og 8. kolonne krydser. For GCD(13,8) skal der stå 5. Se Bilag.

(2)

Lad $T = [t_1, t_2, ..., t_{15}]$ være et array, som udfyldes med et worst-case antal operationer for t_n:

$$T = [1, 1, 2, 2, 3, 2, 3, 4, 3, 3, 4, 4, 5, 4, 4]$$

Her observeres det, at hvert "uptick" i køretid forekommer ved Fibonacci-tal(Hvor $[f_1 = 1, f_2 = 1, f_3 = 2, f_4 = 3, f_5 = 5, f_6 = 8, f_7 = 13, f_8 = 21, ...]$). Det gælder for Fibonacci-tal, at $f_k = f_{k-1} + f_{k-2}$.

(3)

Euklids algoritme vil opnå worst-case køretid, når de to tal a og b's største divisor er 1- eftersom dette implicerer, det højeste antal mulige delinger af et sæt af tal. Dette er altid tilfældet ved Fibonacci tal, der ved første operation i $GCD(f_k, f_{k-1})$ vil returnere.

$$f_k = 1 \cdot f_{k-1} + f_{k-2} = 1 \cdot f_k + f_k \mod f_{k-1} \tag{1}$$

Givet Fibonacci-tallenes konstruktion, vil denne sekvens altid resultere i, at GCD af to Fibonacci-tal er lig 1. Således, for et vilkårligt sæt (a,b) af positive naturlige tal;

$$\text{for } n \in \mathbb{N}^+ \text{ hvor } a \geq b \in n, \tag{2}$$

...til to Fibonacci-tal altid resultere i den højeste køretid for GCB(a,b), når $a = f_k, b = f_{k-1}$.

Et generelt eksempel for køretiden af Euklids GCD ved et sæt af to konsekutive Fibonacci-tal præsenteres nedenfor:

- **GCD(f_k, f_{k-1}):**
- $f_k = f_{k-1} + f_k \mod f_{k-1}$
- $f_{k-1} = f_{k-2} + f_{k-1} \mod f_{k-2}$
- $f_{k-2} = f_{k-3} + f_{k-2} \mod f_{k-3}$
- ...
- $f_{k-(k-2)} = 2 \cdot f_1 + f_{k-(k-2)} \mod f_1 = 2 \cdot 1 + f_1 \mod f_{k-(k-2)} = 2 + 0$
- $\rightarrow GCD(f_k, f_{k-1}) = 1$

Givet ovenstående sekvens er det klart, at algoritmen må terminere efter $k - 1$ operationer, da både $f_1 = f_2 = 1$. Dette vil i store-O-notation implicere en worst-case når (2) er overholdt:

$$t_n \text{ er } O(n)$$

(4)

Det fremgår af opgavebeskrivelsen, at $n \geq a \geq b > 0$. Dette sætter de nedre grænser for mulige værdier af a og b - men ikke de øvre. En vilkårlig sekvens af Fibonacci-tal der overholder $n > a > b > 1$, vil overholde ulighederne defineret, men vil resultere i en køretid, der vokser i takt med at inputs vokser langs Fibonacci-sekvensen. En konstant køretid opnås udelukkende i best-case; $n \geq a = b > 0$. Herved har vi, at:

$$t_n \text{ er } \Omega(1)$$

(5)

Grafen ser ikke ud til, at indikere en lineær vækst i antal operationer over værdier for n. Som funktion af n, ser antallet af operationer ud til at være aftagende. Dette skyldes, at ovenstående retfærdiggørelse i store-O notation er 'løs' idet $t_n er O(n)$ angiver en øvre grænse - men dette kan præciseres.
Det blev fastslået, at GCD(f_k, f_{k-1}) terminerer efter k-1 operationer. Afstanden mellem Fibonacci-tal stiger eksponentielt over k, vi kan derfor skrive, at $k \approx \gamma log(n)$, hvor gamma er en konstant. Dette kan vi netop fordi, at der for Fibonacci-tal gælder, at mængden n skal stige eksponentielt over k - og for hver ekstra operation kræves det næste tal i sekvensen, som er eksponentielt voksende over n.
Da vi nu ved, at k kan approksimeres ved $\gamma log(n)$, og at køretiden for GCD kan skrives som $O(n)$, må det da også gælde, at t_n er $O(log(n))$. Hermed er upper-bound indsnævret. Det er her klart, at med den nedre grænse fundet i (4), må t_n ligeledes være $\Omega(log(n))$, da det i selve best-case scenariet er ækvivalent med $\Omega(1)$.
Idet vi har identificeret en funktion, der tilfredsstiller:

$$t_n = O(log(n)) og t_n = \Omega(log(n))$$

...siger vi, at:

$$t_n = \Theta(log(n))$$