Workshop 2 - Kompleksitet af algoritmer

En søgefunktion tager som input en liste (et array) af elementer $(a_1, a_2,...a_n)$ sammen med et andet input x, som er det vi søger efter. Vi vil undersøge, om x er en del af vores liste, altså om $x = a_i$ for et i mellem 1 og n. Hvis $x = a_i$ vil vi returnere placeringen, altså i. Hvis x ikke er at finde i listen, så returnerer vi 0. Vi lader A være mængden af alle lister indeholdende heltal. Da kan vi betragte følgende:

$$f : A \; x \; \mathbb{Z} \rightarrow \mathbb{N}$$

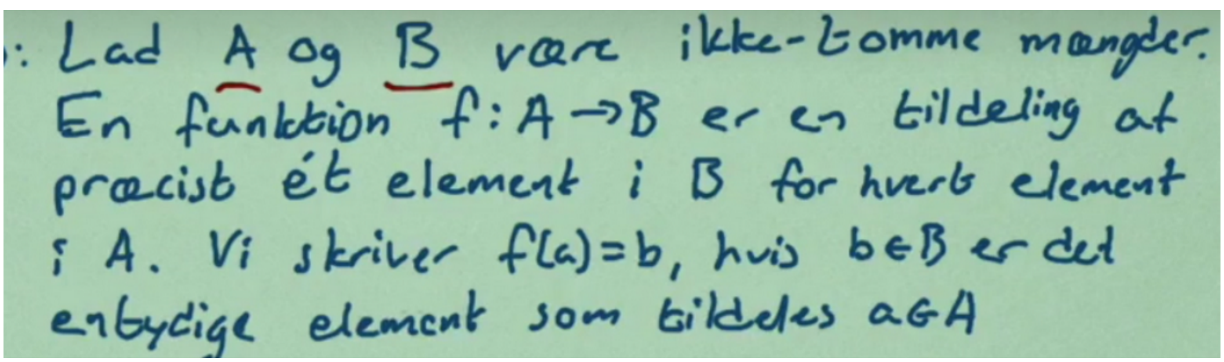
$$f\left(\left(a_1, a_2,...a_n\right), x\right) = \begin{cases} i & x = a_i \\ 0 & x \neq a_i \end{cases}$$

1.0

1.1

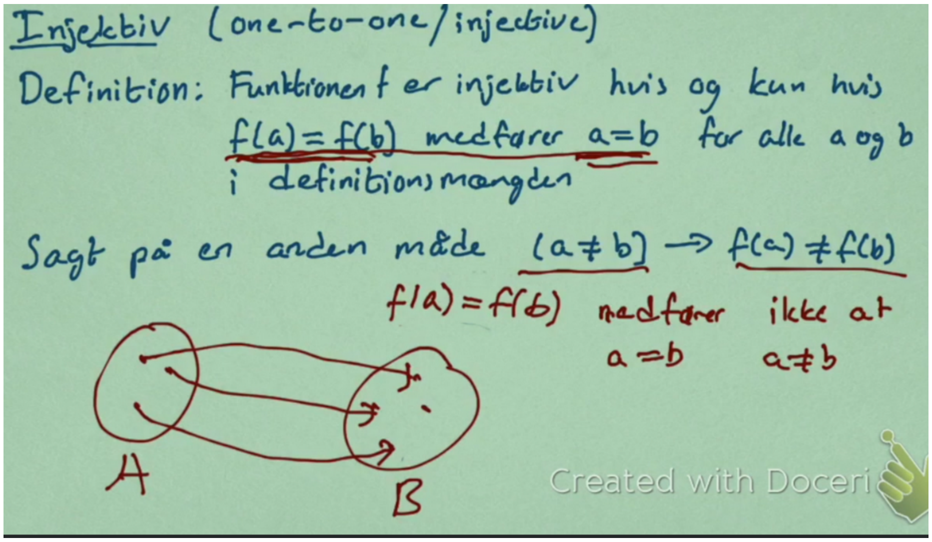
Forklar hvorfor eksempelvis listen (3, 4, 4, 5) medfører, at f reelt set ikke er en funktion i matematisk forstand.

Saetter man $x = 4$ bliver resultatet baade 2 otg 3 dette kan vi ikke have.

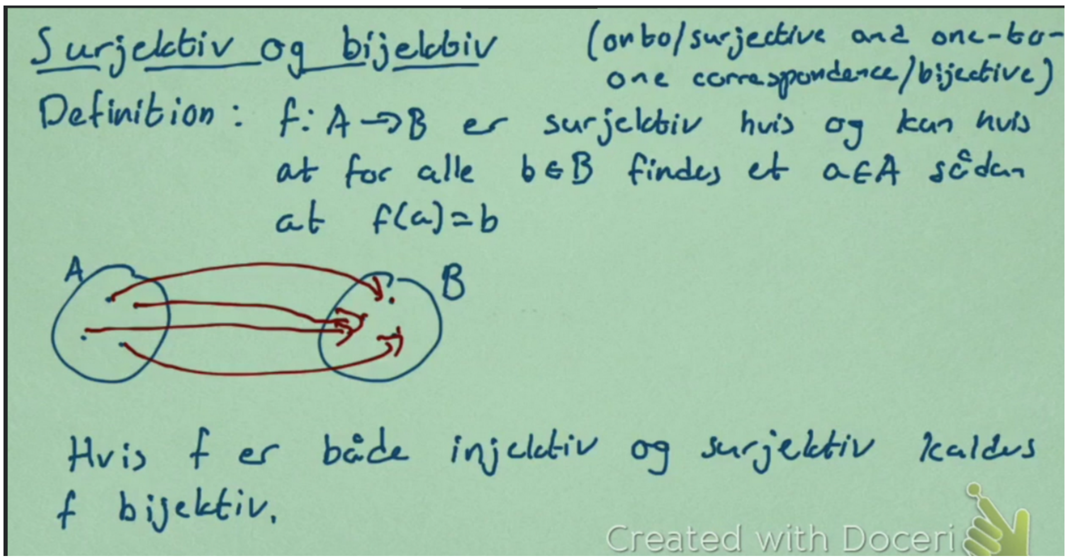


1.2

Hvis vi i stedet definerer mængden A sådan, at lister i A ikke må indeholde det samme element to gange, så er f en funktion. Er f i dette tilfælde injektiv, surjektiv, bijektiv?



tag listerne:
[1, 3, 4] og [2, 3, 4] saa vil
$f([1,3,4],2) = f([2,3,4],2) = 2,\ [1,3,4] \neq [2,3,4]$



hvis A er alle lister med heltal vil man paa et eller andet tidspunkt faa
$f(N,1) = 1$
$f(N,2) = 2$
...
$f(N,n) = n$

Da funktionen er surjektiv men ikke injektiv er den ikke bijektiv.

Pre 2.0 tekst

På Figur 1 og 2 er to søgealgoritmer, som netop returnerer i hvis $x = a_i$ og 0 ellers. Bemærk, at BinSearch kræver at listen er ordnet.

2.0

2.1

Implementer i SearchFunc.c de to søgealgoritmer fra Figur 1 og 2.1

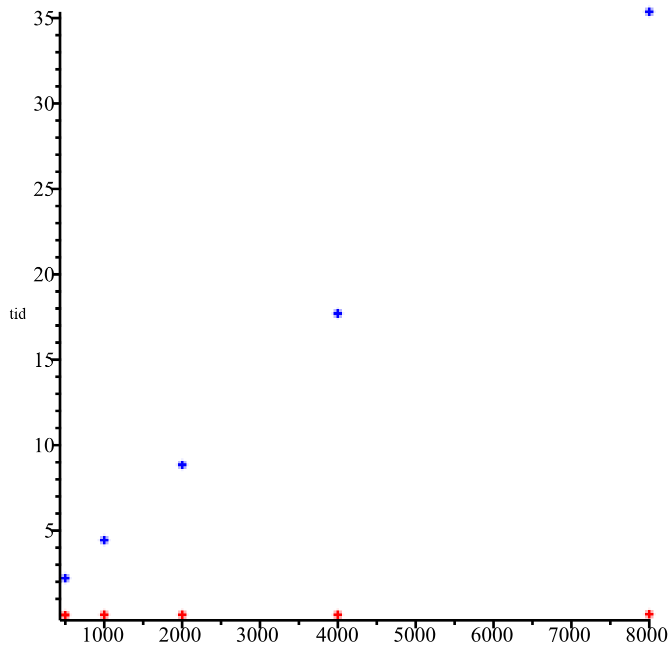
2.2

Undersøg om elementet 7000 er i listerne List500.txt, List1000.txt, List2000.txt, List4000.txt og List8000.txt. I den tilhørende kode køres begge algoritmer 1000000 gange og der tages tid på hvor lang tid hver algoritme bruger. Brug begge algoritmer og undersøg hvor lang tid det tager. Hvilken er hurtigst?
binSearch

2.3

Skriv ned hvor lang tid algoritmerne bruger på de forskellige listestørrelser og plot det i et koordinatsystem (listestørrelsen ud af x-aksen og tiden af y-aksen).

$n := [500, 1000, 2000, 4000, 8000]:$
$linTime := [2.224, 4.427, 8.847, 17.698, 35.363]:$
$binTime := [0.059, 0.067, 0.075, 0.075, 0.089]:$
$with(plots):$
$p1 := pointplot([n, linTime], color = \text{"blue"}):$
$p2 := pointplot([n, binTime], color = \text{"red"}):$
$display(p1, p2, labels = [\text{"n"}, \text{"tid"}])$



Psuedo kode

procedure LINSEARCH(x: heltal, $(a_1, a_2, \ldots, a_n)$: liste med heltal)
    $i = 1$
    while $i \leq n \wedge x \neq a_i$ **do**
        $i = i + 1$
    if $i \leq n$ **then**
        return i
    else
        return 0

Figure 1: Line Search

procedure BINSEARCH(x: heltal, $(a_1, a_2, \ldots, a_n)$: liste med sorterede heltal)
    $i = 1$
    $j = n$
    while $i < j$ **do**
        $m = \lfloor \frac{i+j}{2} \rfloor$
        if $x > a_m$ **then**
            $i = m + 1$
        else
            $j = m$
    if $x = a_i$ **then**
        return i
    else
        return 0

Figure 2: Binary Search

Pre 3.0 tekst

Vi kigger nu nærmere på den teoretiske kompleksitet for de to algoritmer. Her gennemgås worst-case tidskompleksiteten af BinSearch. For at holde det simpelt antager vi, at længden af listen er $n = 2^k$ og tæller kun antallet af sammenligninger. Ved hvert gennemløb af while-løkken laves der 2 sammenligninger. Spørgsmålet er så, hvor mange gennemløb af while-løkken vi har. Bemærk, at hvis $n = 2^k$ bliver $m = 2^{k-1}$ i første gennemløb. Derfor har vi reelt set halveret vores liste. Ved gennemløbene af while-løkken kigger vi derfor på lister af længde $2^k, 2^{k-1}, 2^{k-2}, \ldots, 2$ og først når i = j er listestørrelsen 1. Vi har derfor $k = \log_2(n)$ gennemløb af while-løkken hvor der laves 2 sammenligninger for hver gennemløb. Læg dertil en sammenligning for while-løkken når i = j samt sammenligningen $x = a_i$. Dette giver ialt $2 \log_2(n) + 2$ sammenligninger.

3.0

3.1

Læs og forstå udledningen af kompleksiteten for BinSearch.

3.2

Forklar i hvilket tilfælde vi vil lave flest sammenligninger i LinSearch og at antallet af sammenligninger i dette tilfælde er 2n + 2.

I vaerste tilfaelde vil tallet vi leder efter ikke vaere i listen og algoritmen vil derfor gaa igennem hele listen
Man vil derfor gaa gennem while loekken n gange og man laver 2 sammenligninger hver gang altsaa 2n
naar i bliver stoerre end n laver vi endnu en sammenligning da i nu er stoerre end n sker der en short circuit
vi afslutter med at lave en sammenligning if vores if statement
altsaa faas 2n + 2

3.3

Vis, at antallet af sammenligninger for LinSearch er $\Theta(n)$, og at antallet af sammenligninger for BinSearch er $\Theta(\log(n))$ ved at finde vidner.

Jeg gaar ud fra at de hentyder til log2 naar de skriver log

Store O

find (c, k) saa $|f(x)| \leq C \cdot |g(x)|$ for alle x > k

Lin
$|2n + 2| \leq C \cdot |n|$
(4,1)

Bin
$\left|2 \cdot \log_2(x) + 2\right| \leq C \cdot \left|\log_2(x)\right|$
x stoerre end 0
$2 \cdot \log_2(x) + 2 \leq C \cdot \log_2(x)$
C=3
$2 \cdot \log_2(x) + 2 \leq 3 \cdot \log_2(x)$
ved 4
(3, 4)

Storo Omega
find (c, k) saa $|f(x)| \geq C \cdot |g(x)|$ for alle x > k

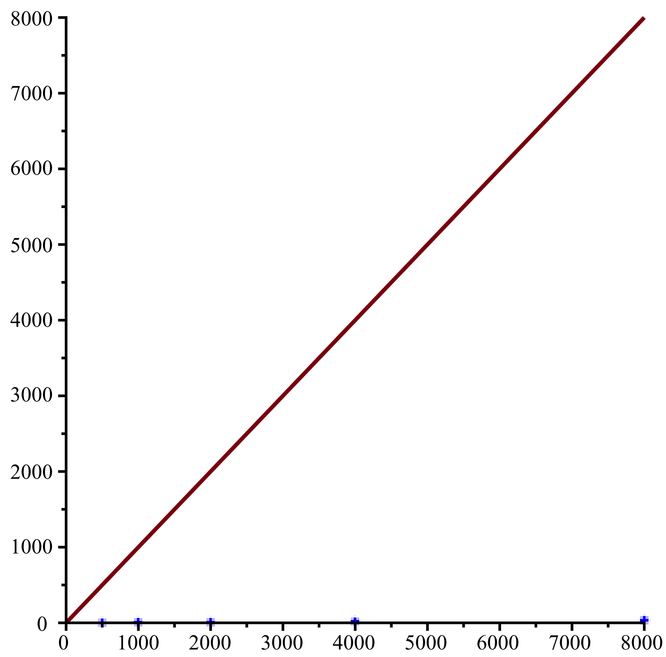
Lin
$|2\,n+2| \geq C \cdot |n|$
(1,1)

Bin
$\left|2 \cdot \log_2(x)+2\right| \geq C \cdot \left|\log_2(x)\right|$
(1,1)

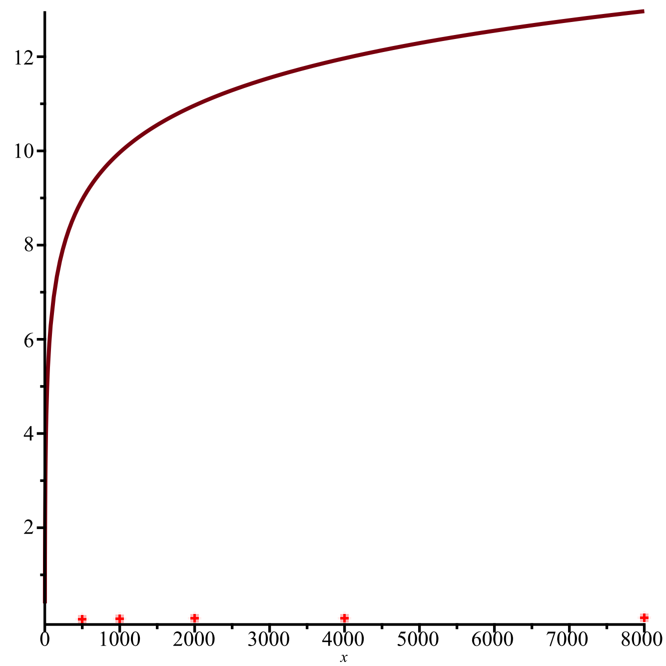
3.4

Sammenlign den teoretiske kompleksitet med det i fandt ud af i Delop-gave 2 (især koordinatsystemet).

$p3 := plot(x, x=0..8000):$
$p4 := plot(\log_2(x), x=0..8000):$
$display(p1, p3)$



$display(p2, p4)$



3.5

Det oplyses nu, at man kan sortere en liste ved at bruge $\Theta(n \cdot \log(n))$ operationer. Vi får givet en usorteret liste med 10000 elementer og vi ved, at vi skal søge efter ca 50 elementer i listen. Giv ud fra store-Theta estimaterne (vi ser altså bort fra de oplyste konstanter i søgefunktion-erne) et bud på, om det bedst kan betale sig at sortere listen først og derefter bruge BinSearch eller om det er hurtigst at bruge Lin-Search.

$n := 10000:$

$m := 50:$

$$res1 := n \cdot \log_2(n) + m \cdot \log_2(n) = \frac{40200 \ln(10)}{\ln(2)} \xrightarrow{\text{at 10 digits}} 133541.5094$$

$$res2 := m \cdot n = 500000$$

$$res1 < res2 \xrightarrow{\text{test relation}} true$$

3.6

Hvor mange søgninger skal man ca foretage for at de to tilgange er lige hurtige?

$$m \cdot n = n \cdot \log_2(n) + m \cdot \log_2(n)$$

$$m \cdot \left(n - \log_2(n)\right) = n \cdot \log_2(n)$$

$$m := \frac{n \cdot \log_2(n)}{\left(n - \log_2(n)\right)} = \frac{40000 \ln(10)}{\left(10000 - \frac{4 \ln(10)}{\ln(2)}\right) \ln(2)} \xrightarrow{\text{at 10 digits}} 13.30539220$$

14

$$solve\left(x \cdot n = n \cdot \log_2(n) + x \cdot \log_2(n)\right) = -\frac{10000 \ln(10)}{\ln(10) - 2500 \ln(2)} \xrightarrow{\text{at 10 digits}} 13.30539220$$