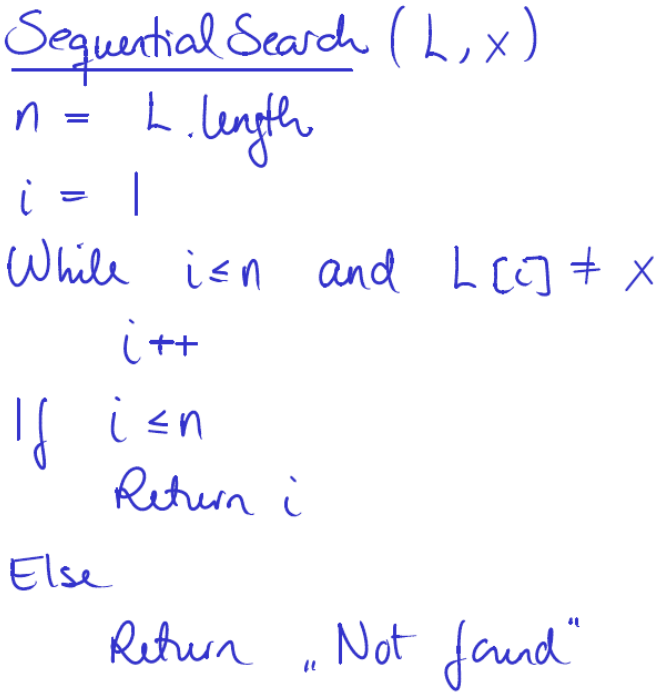
Uge 40

Opg 1

1. Opskriv algoritmen Sequential Search ved hjælp af operationerne `readNext()`, `isEndOfFile()`, `open()` og `close()` fra API'et sekventiel tilgang.

Tager udgangspunkt i denne:




```
LinSearch(L, x):
    L.open()
    i = 0
    while not L.isEndOfFile():
        current = L.readNext()
        if current == x:
            L.close()
            return i
        i ++
    L.close()
    return "not found"
```

Opg 2

2. Opskriv algoritmen for merge af to lister ved hjælp af operationerne `readNext()`, `isEndOfFile()` og `writeNext(data)` fra API'et sekventiel tilgang.

Med udgangspunkt i:

Et merge-skridt: Sammenlign nuværende forreste i A og B, og flyt mindste af disse over sidst i C.

Start med tom C
Sålænge både A og B er ikke-tomme:
    Udfør et merge-skridt
Hvis enten A eller B er ikke-tom:
    Flyt resten af dens elementer over sidst i C

```
Merge(L,R, O):
    x = L.readNext()
    y = R.readNext()
    while L.isEndOfFile() == False AND R.isEndOfFile() == False:
        if x <= y:
            O.writeNext(x)
            x = L.readNext()
        else:
            O.writeNext(y)
            y = R.readNext()
    while L. isEndofFile() == false:
        O.writeNext(L.readNext())
    while R.isEndofFile() == false:
        O.writeNext(R.readNext())
```

Opg 3

3. I denne opgaver repræsenterer vi mængder som sorterede lister uden dubletter. For eksempel vil de to mængder $A = \{5, 3, 9, 8\}$ og $B = \{3, 2, 9, 10, 27\}$ være repræsenteret som disse sorterede lister:

   $$A = [3, 5, 8, 9]$$
   $$B = [2, 3, 9, 10, 27]$$

   Beskriv en algoritme til at beregne repræsentationen af foreningsmængden $X \cup Y$ ud fra repræsentationen af to mængder X og Y.

Tager udgangspunkt i algoritme fra opg 2:

```
Merge(A,B, O):
    x = A.readNext()
    y = B.readNext()
    while A.isEndOfFile() == False AND B.isEndOfFile() == False:
        if x < y:
            O.writeNext(x)
            x = A.readNext()
        else if x > y:
            O.writeNext(y)
            y = B.readNext()
        else:
            O.writeNext(x)
            x = A.readNext()
            y = B.readNext()
    while A. isEndofFile() == False:
        O.writeNext(A.readNext())
    while B.isEndofFile() == False:
        O.writeNext(B.readNext())
```

Opg 4

4. Beskriv en algoritme til at flette (merge) indholdet af tre sorterede lister A, B og C sammen til én sorteret liste D. Hvad er køretiden for din algoritme?

find nu mindste af de tre elementer. Skriv mindste element. Avancer den liste med mindste element.

Skal lave $\Theta(|A| + |B| + |C|)$ flytninger.

Opg 5

5. Givet en algoritme til at flette indholdet af tre sorterede lister A, B og C sammen til én sorteret liste D, beskriv en variant af Mergesort baseret på denne. Hvad er køretiden for din algoritme?

Splitter nu listen i 3 dele (recursivt) og merger herefter. Vil få log_3 lag, med n arbejde i hver.

går ud fra vi stadig tæller flytninger (I/O):

$$\Theta(n \cdot \log_3(n))$$

Husk formel for omregning mellem baser.

Bemærk:

$\log_a(n) \in O(\log_b(n))$, hvis $a, b \in O(1)$

fordi:

$\log_a(n) = \frac{\log_b(n)}{\log_b(a)}$, og

$a, b \in O(1) \Rightarrow \log_b(a) \in O(1)$

Derfor skriver vi blot $O(\log n)$ i st. for $O(\log_2 n)$

Evt. Korekthedsbevis.

Opg 6

6. [Udfordrende] Beskriv en algoritme, der som input tager et tal K og to sorterede lister X og Y, hver med n tal, og finder ud af, om der findes et par at tal $x \in X$ og $y \in Y$ for hvilke $x + y = K$. Din algoritme skal køre i tid $O(n)$. Du skal argumentere for køretiden og for korrektheden af svaret (det sidste kan gøres med en invariant).

```
def E4(L, R, K):
    i = 0
    j = len(R) - 1
    while (L[i] + R[j]) != K:
        if L[i] + R[j] > K:
            j -= 1
        else:
            i += 1
        if (i >= len(L)) | (j < 0):
            print "not found"
            return
    print L[i], R[j], "at", i,j
```

Invariant: Hvis der findes et par x, y hvor $x + y = K$, findes det i listerne $X[I{:}X.len]$, $Y[0{:}J]$

før:

Indexerne er helt i enderne, kan kun være sandt.

Under:

X[i]+Y[j] > K:

Fordi listerne er sorterede ved vi at der ikke er noget i Y[j…Y.len], vi skal altså søge i venstre del af listen. derfor $j--$

X[i]+Y[j] < K:

Da listerne er sorterede ved vi X[0…i], vi skal altså søge i højre del af listen. derfor i++

Holder derfor stadig ved indgang til næste loop.

Terminering:

X[i]+Y[j] == K:

kommer ud af loop. Vi har fundet resultatet.

(i >= len(L)) | (j <= 0):

kommer ud af loop. Findes ikke i listen.

Evt historien fra n^2

Pegepind efter. Tom mængde. Krydsprodukt tomt.

Opg 7

7. Hvis en hashfunktion h er givet ved $h(x) = x \bmod 11$, på hvilke pladser i tabellen ender tallene 25, 75, 125, 175?

$$25 \,\%\, 11 = 3$$

$$75 \,\%\, 11 = 9$$

$$125 \,\%\, 11 = 4$$

$$175 \,\%\, 11 = 10$$

Opg 8

8. Hvis en hashfunktion h er givet ved $h(x) = x \bmod 11$, hvor mange pladser i hashtabellen har mere end ét element, når der indsættes elementerne 34, 65, 122 og 155?

$$34 \,\%\, 11 = 1$$

$$65 \,\%\, 11 = 10$$

$$122 \,\%\, 11 = 1$$

$$155\% \, 11 = 1$$

Vi har 1 plads som indeholder mere end et element.

Opg 9

9. Beregn med lommeregner svaret på følgende: Hvis 3 elementer indsættes tilfældigt i et array med 7 pladser, hvad er sandsynligheden for, at der ikke er to elementer som ender på samme plads?

Hvis vi kalder P(ingen med samme fødseldag blandt de n første personer) for s_n, kan vi se af ovenstående at

$$s_n = s_{n-1} \cdot \frac{365 - (n-1)}{365}$$

Da s_1 naturligvis er 1 (med kun een person i rummet er der ingen med samme fødselsdag), ser vi at:

$$\begin{aligned}
s_1 &= 1 \\
s_2 &= 1 \cdot \tfrac{364}{365} = 0.9972\ldots \\
s_3 &= 1 \cdot \tfrac{364}{365} \cdot \tfrac{363}{365} = 0.9917\ldots \\
s_4 &= 1 \cdot \tfrac{364}{365} \cdot \tfrac{363}{365} \cdot \tfrac{362}{365} = 0.9836\ldots \\
&\vdots
\end{aligned}$$

$$s_1 = 1$$

$$s_2 = 1 \cdot \frac{6}{7} \approx 0{,}85714285714$$

$$s_3 = \frac{6}{7} \cdot \frac{5}{7} \approx 0{,}61224489796$$

Opg 10

10. Beregn med lommeregner følgende svaret på følgende: Hvis 5 elementer indsættes tilfældigt i et array med 12 pladser, hvad er sandsynligheden for, at der ikke er to elementer som ender på samme plads?

$$s_1 = 1$$

$$s_5 = 1 \cdot \frac{11}{12} \cdot \frac{10}{12} \cdot \frac{9}{12} \cdot \frac{8}{12} \approx 0{,}38194444444$$

Opg 11

11. Lav et Python-program med input n og k der for situationen hvor n elementer indsættes tilfældigt i et array med k pladser finder sandsynligheden for, at der ikke er to elementer som ender på samme plads.

```
from fractions import Fraction as f

n = 5 # number of elements
k = 12 # number of spaces

res = f(1, 1)
for i in range(0, n):
    res = res * f(k - i, k)
print float(res)
```

Opg 12

12. Hvis 1000 elementer indsættes tilfældigt i et array med 1.000.000 pladser, hvad er sandsynligheden for, at der ikke er to elementer som ender på samme plads?

beregnet via python script.

0.606732971441

Opg 13

11. Hvis n elementer indsættes tilfældigt i et array med 1.000.000 pladser, hvor stor skal n være for at sandsynligheden for at der ikke er to elementer som ender på samme plads bliver mindre end 1/2?

```
n = 1
k = 1000000
res = f(1, 1)
while res > f(1, 2):
    res = res * f(k - n, k)
    n = n + 1
print float(res), n
```

n = 1178