Nicklas	Patrick	Viktor
		
Xmas tek Chip	Mr. Frog	Mr. Clean

programmering_algoritmer_aflevering

Aflevering i programmering som omhandler algoritmer

Opgave 1

```
import random

# opretter en liste uden indhold
list = []

# kører igennem løkken 1000 gange
for i in range(1000):
    # sætter numbers til random numre mellem 1 og 100000
    numbers = random.randint(1, 100000)
    # ligge numrene der er lige genereret ind i list
    list.append(numbers)

# Printer den usorteret liste
print("Liste - Ikke sorteret")
print(list)

# sortere listen og herefter bliver den printet
list.sort()
print("Liste - Sorteret")
print(list)
```

Opgave 2

```
# Importerer modulerne time og randome
import time
import random

# Laver en variabel ved navn numGen og et array der hidder times
numGen = 100000
times = []


def randomSort(A):
```

```
    # sætter number til at være funktionen randomSort
    number = A
    # Sorterer number
    number.sort()
    # Printer længden a number arrayet
    print(len(number))
    # Returnerer number arrayet
    return number


chooseAmount = input("Vil du selv bestemme hvor mange tal der skal sorteres\
 eller skal der køres standard mængde? selv/standard ")

# Starter en forgrening ud fra chooseAmount inputtet
if (chooseAmount == "standard"):
    # Hvis der er skrevet "standard"
    # går den i gang med at sorterer 100.000, 200.000, 300.000 og 400.000 tal
    for i in range(4):
        # Først sættes StartT variablen til at være
        # programmets daværende uptime
        StartT = time.time()
        # Så udvælger den mængden af tal angivet i numGen variablen
        # i en liste fra 0-1.000.000
        # og sætter disse tal til at være i arrayet talliste
        talliste = random.sample(range(1000000), numGen)
        # Så printer den tallisten,
        # hvor den er kørt igennem funktionen randomSort
        print(randomSort(talliste))
        # Så laver den variablen StopT som er programmets uptime
        StopT = time.time()
        # Variablen numGen forhøjes med 100.000
        numGen = numGen + 100000
        # Variablen trueTime bliver nu lavet og sat til at være StopT-StartT
        trueTime = StopT-StartT
        # Variablen trueTime bliver nu sat ind i arrayet times
        times.append(trueTime)
elif (chooseAmount == "selv"):
    # Der laves en heltals variabel ved navn amount
    amount = int(input("Hvor mange tal skal sorteres? "))
    # Sætter StartT til programmets daværende uptime
    StartT = time.time()
    # Så udvælger den mængden af tal angivet i amount variablen
    # i en liste af tal med størrelsen amount * 100
    talliste = random.sample(range(amount * 100), amount)
    # Så printer den tallisten, hvor den er kørt igennem funktionen randomSort
    print(randomSort(talliste))
    # Så laver den variablen StopT som er programmets uptime
    StopT = time.time()
    # Variablen trueTime bliver nu lavet og sat til at være StopT-StartT
    trueTime = StopT-StartT
    # Variablen trueTime bliver nu sat ind i arrayet times
    times.append(trueTime)

# Printer både arrayet times og summen af times
```

```
print("")
print(times)
print("Sammenlagt tid")
print(sum(times))
```

Opgave 3

Selection sort

Selection sort fungerer ved, at man har en usorteret liste af tal. Algoritmen fungerer ved, at man har to variable, A og B, som kører listen gennem. A starter ved at køre hele listen igennem. A leder efter det mindste tal i listen og navngiver det B. Derefter bliver B byttet rundt med det første tal i listen. Algoritmen starter forfra, dog en plads længere inde i listen end før, da det mindste tal allerede er fundet. Hvis det mindste tal allerede står på første plads, rykker A bare videre, uden at bytte noget Sådan fortsætter algoritmen indtil B har sorteret alle tal, ved hjælp af A's søgning.

Opgave 4

```
def selection_sort():

    # Der oprettes et array af tal ved navn tallist
    # og en variabel n som er længden af tallist
    tallist = [2, 4, 5, 1, 3, 6, 9, 8, 7]
    n = len(tallist)
    # printer tallisten inden sortering
    print(tallist)

    # Der oprettes en løkke i som er længden af tallisten -1
    for i in range(n-1):
        # mini variablen sættes til at være i
        mini = i

        # Der oprettes en løkke j
        # som længden mellem variablen i +1 til længden af n
        for j in range(i + 1, n):
            # Hvis tallist[j] er mindre end tallist[mini]
            # sættes mini til at være j
            if tallist[j] < tallist[mini]:
                mini = j
        # Hvis mini ikke er lig med i
        # bytter den om på tallist[mini] og tallist[i]
        if mini != i:
            tallist[mini], tallist[i] = tallist[i], tallist[mini]
    # Resultatet af den sorterede tallist skrives i terminalen
    return tallist


# Printer funktionen selection_sort
print(selection_sort())
```

Opgave 5

```
def selection_sort(A):
    # Der oprettes et array af tal ved navn tallist
    # og en variabel n som er længden af tallist
    tallist = A
    n = len(tallist)

    # Der oprettes en løkke i som er længden af tallisten -1
    for i in range(n-1):
        # mini variablen sættes til at være i
        mini = i

        # Der oprettes en løkke j
        # som længden mellem variablen i +1 til længden af n
        for j in range(i + 1, n):
            # Hvis tallist[j] er mindre en tallist[mini]
            # sættes mini til at være j
            if tallist[j] < tallist[mini]:
                mini = j
        # Hvis mini ikke er lig med i
        # bytter den om på tallist[mini] og tallist[i]
        if mini != i:
            tallist[mini], tallist[i] = tallist[i], tallist[mini]
    # Resultatet af den sorterede tallist skrives i terminalen
    return tallist
```

Opgave 6

```
# Vi har valgt at arbejde med algoritmen Gnome sort

# Gnome sort virker ved, at den først tjekker det første
# tal i tallitsten (position 0) og om der er et tal før det.
# Da der ikke er et tal før, rykker Gnome sort 1 position frem og tjekker
# om der er et tal før.
# Der er et tal før, men da det er mindre end position 1, går den videre.
# Den går så videre og tjekker for tallet i position 2,
# tallet i position 2 er så mindre end position 1.
# Gnome sort bytter så de to tal så det mindre tal nu er på position 1.
# Derefter tjekker den, om position 1 er mindre en position 0.
# Position 0 er mindre end position 1. Derfor går koden bare videre
# til næste tal i rækken, nemlig position 3.
# Sådan går den frem og tilbage hele tiden,
# indtil tallisten er blevet sorteret.


def gnome_sort(A):
    # opretter en varibel, pos, og sætter den lig med 0
    pos = 0
    # tjekker om pos er mindre end længden af A
```

```
    while pos < len(A):
        # hvis at pos er lig med 0 eller pos er større end eller lig med pos-1
        # plusser den pos med 1
        if (pos == 0 or A[pos] >= A[pos-1]):
            pos = pos + 1
        else:
            # bytter om på pos og pos-1 og rykker pos 1 tilbage
            A[pos], A[pos-1] = A[pos-1], A[pos]
            pos = pos-1
    # returnere A
    return A


# Tallisten som algoritmen skal sortere
talliste = [4, 312, 12, 312, 3, 1, 23, 1, 5, 2, 312, 31, ]
# Tallisten inden sortering
print(talliste)
# Tallisen efter sortering
print(gnome_sort(talliste))
```

Thank you for reading ❤️