Studieretningsprojekt 2022

Elev id: 3x 06
Elev: Balder Westergaard Holst

Fag:	Vejleder:
Matematik A	Jens Christian Larsen Tlf.: 26279734 Email: jl@soroeakademi.dk
Informatik C	Kristian Kjeldgaard Hoppe Tlf.: 28825466 Email: kh@soroeakademi.dk

Område:
Store-O notation og sorteringsalgoritmer

Opgaveformulering:
Gør rede for kompleksitetsklasserne givet ved Store-O notation, idet du giver eksempler på repræsentanter af forskellige klasser.

Beskriv sorteringsproblemer generelt, og forklar algoritmerne for insertion- og merge-sort. Vis at insertion-sort har en værste-tilfælde-udførelse på $O(n^2)$.

Implementer algoritmerne insertion- og merge-sort i Python og undersøg den faktiske udførelsestid af de to algoritmer på lister af tilfældige tal af forskellig størrelse. Vurder algoritmernes udførelsestid.

Indfør begrebet træ, og bevis relevante sætninger om træer, især højden af et træ. Bevis, at en vilkårlig sorteringsalgoritme baseret på sammenligning, vil være nedadtil begrænset af i dens værste-tilfælde-udførelsestid med $n \cdot \lg(n)$ (Stirlings formel må antages).

Insertion-sort har en bedste-tilfælde-udførelsestid på $O(n)$: Hvorfor er det ikke et modbevis til sætningen om at den nedre grænse for sorteringsalgoritmer er $n \cdot \lg(n)$?

Omfang 15-20 ns.

Sorteringsalgoritmer og O-notation

Matematik A og Informatik C

Vejledere: Jens Christian Larsen og Kristian Kjeldgaard Hoppe

Balder Westergaard Holst

25. marts 2022

Indhold

1 | Indledning

Dette er min indledning pt.

2 | Resume

Dette er hvad jeg har skrevet og fundet ud af.

3 Konklusion

Her er mine konkluderende sætninger

Litteraturliste

1. Dietzfelbinger, M. & Mehlhorn, K. *Algoritmer og datastrukturer* `https://github.com/thorehusfeldt/algoritmer-og-datastrukturer/blob/master/ad-book.pdf` ().

Bilag 1 - Algoritmer og Datageneration

Insertionsort Algoritmen

```
def insertionsort(l):
   for i in range(1,len(l)):
      element = l[i]

      if element < l[0]:
         for j in range(i,0,-1):
            l[j] = l[j-1]
         l[0] = element
      else:
         j = i
         while(l[j-1]>element):
            l[j] = l[j-1]
            j -= 1
         l[j] = element
   return(l)
```

Mergesort Algoritmen

```
def mergesort(l):
   if len(l) <= 1:
      return(l)
   else:
      return(merge(mergesort(l[:len(l)//2]),mergesort(l[len(l)//2:])))

def merge(a,b):
   c = []
   while True:
      if (len(a) == 0):
         return(c + b)
      elif (len(b) == 0):
         return(c + a)
      elif (a[0] <= b[0]):
         c.append(a[0])
         a.pop(0)
      else:
         c.append(b[0])
         b.pop(0)
```

Kode til test af algoritmerne

```
import random
import time
import pandas as pd
```

```python
import os

import sys
sys.path.insert(1, './algoritmer')
from mergesort import *
from insertionsort import *

# Denne funktion timer køretiden af en funktion med input l og returnerer funktionen køretid i milisekunder
def test(fun,l):
    start_time = time.perf_counter()

    fun(l)

    return(time.perf_counter() - start_time)

# Denne funktion returnerer en liste af tilfældige tal mellem 0 og 1000, med n elementer
def createRandomList(n):
    return([random.randint(0,1000) for i in range(n)])

# Laver en mappe i filsystemet hvis der ikke allerede er en med stien
def makeIfNeeded(dir_path):
    if(os.path.isdir(dir_path) == False):
        print(f"made dir: {dir_path}")
        os.mkdir(dir_path)
    return(dir_path)

# Finder det næste versionsnummer for til navngivning af fil på baggrund af indholdet i en folder
def newVersionNumber(dir_path,extention):
    file_names = os.listdir(dir_path)
    version = 0

    thisfilename = f"{version}{extention}"

    while(thisfilename in file_names):
        version += 1
        thisfilename = f"{version}{extention}"

    return(thisfilename)

# Dette er funktionen der tester en liste med funktioner og gemmer deres køretider
def fullTest(functions):

    # hvor mange datapunkter pr. n-værdi
    trials = 10

    data_dir = "../data/"
    version_number = newVersionNumber(data_dir,"")

    seed = time.time()
    print(f"Seed: {seed}")

    for function in functions:

        # i denne liste gemmes antallet af elementer at den liste som algoritmen sorterer for hvert datapunkt.
        ns = []
        # i denne liste gennes den tid det tager at sorterer listen med n elementer
        times = []

        # Bruger det samme seed til test at hver algoritme. på den made er det de samme pseudo-tilfældige liste som algoritmerne sorterer
        random.seed(seed)

        # Vi laver testen et antal (trials) gange pr. n-værdi
        for trial in range(0,trials):
```

```
67
68          # En lykke der køre et abitrært antal gange (jo højere en i-værdi jo højere maks antal
                elementer i listen)
69          for i in range(0,80):
70
71                # Jeg bruger en potensfunktion til at fa flere datapunker tættere på y-aksen og
                      færre lange oberationer (pga. lange liste)
72             n = round(pow(1.1,i))
73
74             print(f"function=\"{function.__name__}\": Trial: [{trial+1}/{trials}] {i=},{n=}") #
                    lidt feedback
75
76             # gennererer en tilfældig liste
77             l = createRandomList(n)
78
79             # gem størrelsen af listen der skal sorteres
80             ns.append(n)
81             # gen den tid det tager at sortere listen
82             times.append(test(function,l))
83
84
85          data = {
86             "n": ns,
87             "t": times
88             }
89
90          version_dir = makeIfNeeded(data_dir + version_number + "/")
91          algorithm_dir = makeIfNeeded(version_dir + function.__name__ + "/")
92          full_path = algorithm_dir + newVersionNumber(algorithm_dir,".csv")
93
94          print(f"\ndata saved to \"{full_path}\"\n")
95
96          pd.DataFrame(data).to_csv(full_path,index = False)
97
98
99    if __name__ == "__main__":
100      functions = [mergesort,insertionsort]
101
102      fullTest(functions)
```

Bilag 2 - Databehandling og Plots

Kode til databehandling og generering af plots

```
library(ggplot2)
#library(tikzDevice)

#working dir
setwd("/home/Balder/Documents/Skole/Gym/SRP/data/5")


#import data
dataset <- read.csv("mergesort/0.csv",header=TRUE,sep=",")

#dir = "1"
algorithm_dirs = list.files()

M = NULL
for (j in 1:length(algorithm_dirs)){
   algorithm_dir = algorithm_dirs[j]

   files = list.files(algorithm_dir)

   for (i in 1:length(files)){
      file_path = paste(algorithm_dir,files[i],sep="/")
      print(file_path)
      m = read.csv(file_path,header=TRUE,sep=",")
      m$Algorithm = algorithm_dirs[j]
      M = rbind(M, m)
   }

}

M$algorithm = factor(M$Algorithm)
summary(M)

# punktm?ngder for hver algoritme
m_merge = subset(M,M$algorithm=="mergesort")
m_insertion = subset(M,M$algorithm=="insertionsort")

# laver modeller
model_merge = nls(t~a*n*log2(n), data=m_merge, start=list(a=0.000001))
model_insertion = nls(t~a*n^2 + b*n + c, data=m_insertion, start=list(a=1,b=1,c=1))


# Sætter ny path til hvor outputtet skal være
setwd("/home/Balder/Documents/Skole/Gym/SRP/img")

# gemmer r2-værdierne i to filer
writeLines(toString(round(with(m_merge,cor(t,n)),digits=3)),"r2-merge.txt")
writeLines(toString(round(with(m_insertion,cor(t,n)),digits=3)),"r2-insertion.txt")
print("r2 saved to files")

```

```
50
51 # laver modelerede v?rdier for hver n
52 m_merge$model = predict(model_merge)
53 m_insertion$model = predict(model_insertion)
54
55 m_merge$residual = resid(model_merge)
56 m_insertion$residual = resid(model_insertion)
57
58 # kombinerer de to
59 M = rbind(m_merge,m_insertion)
60
61 summary(M)
62
63
64 ggplot(M, aes(x=n, y=t, colour=Algorithm)) +
65    geom_point(size=1.5,alpha=0.1,shape=19) +
66    geom_line(aes(x=n, y=model,color=Algorithm), size=2, alpha=0.6) +
67    labs(title="To Sorteringsalgoritmer") +
68    theme(legend.position = c(.9, .9)) + # virker ikke!!
69    guides(colour = guide_legend(override.aes = list(alpha = 1))) + # lav legend alpha 1
70    theme_bw()
71
72    ggsave("toAlgoritmer.png")
73
74    ggplot(M, aes(x=log10(model), y=residual, colour=Algorithm)) +
75       geom_point(size=1.5,alpha=0.1,shape=19) +
76       labs(title="Residualer") +
77       facet_wrap(~algorithm,scales="free",ncol=1) +
78       theme_bw() +
79       theme(legend.position="none")
80    ggsave("toAlgoritmerResidual.png")
81
82    C = data.frame(
83             Algorithm = unique(M$Algorithm),
84             R2 = c(with(m_merge, cor(t,n)), with(m_insertion, cor(t,n)))
85    )
86
87    write.table(C, "r2.txt", quote=FALSE,sep="\t", row.names=FALSE)
```