IN2010, høsten 2023

«Obligatorisk» oppgave 2

Effektive mengder

Oppgave 1: Beklager lite kreative navn, det har vært noen uker

Filer: avlnode.py, avlsucc.py, binaersucc.py, bstnode.py, set.py, sot.py

a) Eksekveringsfilen heter «binaersucc.py» og en kjøring med testdata ser ut som følger:

```
$ python binaersucc.py < inputs/input_100 | cmp - outputs/output_100
```

Den funker hele veien opp til 100 000 og benytter seg av klassene «bstnode.py» og «set.py»

b) Eksekveringsfilen heter «avlsucc.py» og en kjøring med testdata ser ut som følger:

```
$ python avlsucc.py < inputs/input_100 | cmp - outputs/output_100
```

Den funker hele veien opp til 100 000 og benytter seg av klassene «avlnode.py» og «sot.py»

Bygge balanserte søketrær

Oppgave 2: Igjen

Filer: bstutenbst.py, bstmenihvertfallikkebst.py

a) Denne gikk jaggu fort

Algoritme: Deler listen i to rundt midterste element som fjernes og skrives ut
Input: Sortert array A
Output: Printer elementene i rekkefølgen de måtte blitt lagt til i et binært søketre for å holde det balansert

```
Procedure BalancedBST(A)

      if len(A) > 1 do
            index <- len(A) // 2
            print(A.pop(index))
            SubList1 <- A[:index]
            SubList2 <- A[index:]
            BalancedBST(SubList2)
            BalancedBST(SubList1)

      else if len(A) = 1 do
            print(A.pop())
```

Kjøres med kommando

```
$ python bstutenbst.py
```

Jeg vet ikke helt hvordan jeg skulle gjort dette med `stdin` med mindre jeg skulle gjort de nødvendige endringene i input som hadde gjort det mye enklere, men håper jeg har demonstrert at jeg vet hva det er gjennom de tidligere oppgavene.

b) Denne gikk jaggu sakte

Algoritme: Deler heaps i to frem til vi har nådd midterste element som poppes
Input: Sortert heap A
Output: Printer elementene i rekkefølgen de måtte blitt lagt til i et binært søketre for å holde det balansert, bare med heaps

```
Procedure BalancedHeap(A)

      if len(A) > 1 do
            index = len(A) // 2
            SubList <- empty list
            while i <- 0 < index do:
                  pop smallest element from A to SubList
                  i = i + 1
            print(HeapPop(A))
            BalancedHeap(A)
            BalancedHeap(SubList)

      else if len(A) = 1 do
            print(HeapPop(A))
```

Kjøres med kommando

```
$ python bstmenihvertfallikkebst.py
```

Jeg skal helt ærlig vedgå at disse pseudokodene ikke kom frem i form av pseudokode, men meg med papir og fettstifter som tegnet hvordan jeg tror det ville foregått og hvordan jeg ville bedt python om å gjøre det i bakhodet.